Shielded Deep Reinforcement Learning for Complex Spacecraft Tasking

Robert Reed, Hanspeter Schaub, and Morteza Lahijanian

***Abstract*— Autonomous spacecraft control via Shielded Deep Reinforcement Learning (SDRL) has become a rapidly growing research area. However, the construction of shields and the definition of tasking remains informal, resulting in policies with no guarantees on safety and ambiguous goals for the RL agent. In this paper, we first explore the use of formal languages, namely Linear Temporal Logic (LTL), to formalize spacecraft tasks and safety requirements. We then define a manner in which to construct a reward function from a co-safe LTL specification *automatically* for effective training in SDRL framework. We also investigate methods for constructing a shield from a safe LTL specification for spacecraft applications and propose three designs that provide probabilistic guarantees. We show how these shields interact with different policies and the flexibility of the reward structure through several experiments.**

I. Introduction

The spacecraft task scheduling problem, which involves collecting data while adhering to system constraints, traditionally relies heavily on human intervention. This reliance is due to the inability of current autonomy modules, often based on simplistic rules and past experiences, to ensure compliance with spacecraft safety requirements. Due to recent technological advancements and economical interest [2], spacecraft autonomy has become a central research topic [3]–[5], and recent works show that computational challenges such as the large dimensionality of the state space and low on-board computation capabilities can be overcome via Reinforcement Learning (RL) [3], [4]. Nevertheless, providing correctness and safety guarantees on the decisions of the autonomy remains a major challenge. This work focuses on this challenge and aims to enable safe spacecraft autonomy by combining machine learning with formal methods.

In RL, an agent explores an unknown environment and acts to maximize a reward function that is designed to express the desired behavior of the agent. Typically, this reward function is hand designed. Deep RL (DRL) is an extension of RL that utilizes the power of Neural Networks (NNs) to learn a policy, enabling RL in high-dimensional spaces. While the optimally and data-efficiency of DRL algorithms is well understood [6], the policies returned from these algorithms have no guarantees on safety.

This led research in the direction of Shielded DRL (SDRL) [7], [8] with adaptation for spacecraft autonomy [3], [4]. In SDRL, a *shield* is designed in order to ensure *safety* of the system when it is deployed with a DRL policy [8]. The shield acts as a *minimal interference* filter on the agents action, allowing all actions that have been pre-determined to be safe and replacing unsafe actions with correct choices. SDRL had been shown to improve policy performance and reduce the necessary training time for spacecraft operations [3], [4]. Typically, shield design relies on having a description of the safety-critical aspects of the system as a Markov Decision Process (MDP), which is called a *Safety MDP*. Prior works [4] use a hand-designed safety MDP, where the transition probabilities between states are chosen from expert intuition. Such a design not only requires a domain expert with extensive knowledge, but also limits the guarantees on safety that the shield can provide.

Formal languages, such as *linear temporal logic* (LTL) [9], provide a manner to rigorously define the tasks and safety requirements needed for spacecraft deployment. Typically, the tasks that an agent should complete are composed as a *liveness specification* and the behaviors that should be avoided are written as a *safety specification*. Formal specifications have been used with great success in robotics applications [10], [11] and the effectiveness of splitting the specifications into parts (i.e., liveness and safety) has been well demonstrated [12]. This separation follows the idea of constructing a shield based on a safety specification and learning a policy that satisfies a liveness specification. When a task is defined using LTL, formal synthesis techniques [13], [14] are often used to find a policy that is correct-by-construction. However, such techniques are often limited by the dimensionality of the system, hence the need for DRL in the autonomous spacecraft scenario.

This work focuses on incorporating formal methods into the SDRL framework for the autonomous Earth imaging problem by formalizing the design of a shield for a spacecraft system and utilizing LTL formulas to define tasking and safety requirements. We propose to construct a safety MDP algorithmically using physics-simulator engines and implement three different shield designs with this MDP. Prior works in formal methods often assume the safety MDP is known a priori [7], or restrict the RL to a set of safe policies and work to optimize an unknown objective [15], [16], unlike our scenario. We also identify a manner to automatically construct a reward function from an LTL specification, removing human interpretation of a task from the process. This ensures that DRL is optimizing a reward function that has no ambiguity from the desired task, resulting in a policy that is optimized for the desired task. Our evaluations show the effectiveness of the reward and shield designs and the

This work was supported by Air Force Research Lab (AFRL) under agreement number FA9453-22-2-0050.

An extend version is available on ArXiv [1].

Authors are with the Department of Aerospace Engineering Sciences at University of Colorado Boulder, Boulder, Colorado, USA. {Robert.Reed-1, Hanspeter.Schaub, Morteza.Lahijanian}@colorado.edu

importance of training with safety specifications.

Our contributions are four-fold: 1) we improve the formalism of shield construction for spacecraft SDRL, 2) we demonstrate how to incorporate complex, formal specifications for Earth imaging tasks into a DRL framework, 3) we identify a training setup that minimizes safety violations with few shield interventions, and 4) we illustrate the efficacy of the method on several case studies and benchmarks.

II. Problem Formulation

In this work, we consider autonomous spacecraft scheduling for complex Earth observation tasks with a discrete action space. The spacecraft must select a sequence of flight modes such that a predefined Earth observing task is satisfied while remaining safe.

A. Spacecraft Model

Spacecraft dynamics are highly complex and high dimensional, with potentially thousands of states needed to accurately represent how the subsystems on a spacecraft interact. The dynamics act over continuous space and time with stochastic disturbances, which enhances the difficulty of control problems. We assume that we are able to control the switching between different modes of operation. Hence, the dynamics of the spacecraft system can be described as a continuous-state Markov decision process (MDP).

Definition 1 (MDP)**.** *A* Markov Decision Process *(MDP) is a tuple $M = (X, X_0, A, T, \Pi, L)$, where $X \subseteq \mathbb{R}^n$ is the state space, $X_0 \subset X$ is a set of initial states, A is a finite set of modes or actions, $T : X \times A \times \mathcal{B}(X) \to [0, 1]$ is a transition probability function, where $\mathcal{B}$ is a Borel set*[1]*, Π is a set of atomic propositions that are related to spacecraft task or safety, and $L : X \to 2^{\Pi}$ is a labeling function that assigns a state $x \in X$ to a subset of Π.*

Example 1. *Consider a spacecraft with fours modes of operation $A = \{a_i\}_{i=0}^{3}$, where a_0 is* Charging Mode*, a_1 is* Momentum Dumping Mode*, a_2 is* Imaging Mode A*, and a_3 is* Imaging Mode B*. The Earth observation tasking can be related to the two imaging modes. Momentum Dumping and Charging are then important from the perspective of safety.*

Defining the state of the system as $x \in X$, an infinite trajectory is then written as $\omega_{\mathbf{x}} = x_0 \xrightarrow{u_0} x_1 \xrightarrow{u_1} \ldots$ where each $u_i \in A$. We denote the i-th element by $\omega_{\mathbf{x}}[i]$, and the set of all finite and infinite trajectories by $\Omega_{\mathbf{x}}^{\text{fin}}$ and $\Omega_{\mathbf{x}}$, respectively. We are interested in controlling $\omega_{\mathbf{x}}$ through the choice of action taken (switching modes) via a policy.

Definition 2 (Policy)**.** *A policy $\pi : \Omega_{\mathbf{x}}^{fin} \to A$ is a function that maps a finite trajectory $\omega_{\mathbf{x}} \in \Omega_{\mathbf{x}}^{fin}$ onto the next action in A. Policy π is called stationary if it only depends on the last element of $\omega_{\mathbf{x}}^{N}$; otherwise, it is called history dependent.*

Under a policy π, a probability measure over the paths of the MDP M is well defined [17]. We denote MDP M under π as M^{π} and its sets of finite and infinite trajectories as $\Omega_{\mathbf{x}}^{\text{fin},\pi}$ and $\Omega_{\mathbf{x}}^{\pi}$, respectively.

[1]The Borel set defines an open set of states in X, hence transition probabilities can be assigned.

B. LTL for Earth Observing Tasks and Safety Requirements

We consider imaging tasks that can be completed in finite time and safety requirements that must not be violated. These tasks and requirements are related to the temporal behavior of the spacecraft system with respect to a set of state-space regions $R = \{r_1, \ldots, r_l\}$, where $r_i \subseteq X$. To enable formal description of tasks, we associate an atomic proposition p_i to each region r_i such that p_i is true iff $x \in r_i$. Then, the set of atomic proposition is $\Pi = \{\mathrm{p}_1, \ldots, \mathrm{p}_l\}$, and the labeling function $L : X \to 2^{\Pi}$ assigns each state $x \in X$ to the set of atomic propositions that are true at that state. Accordingly, we define an *observation trace* of trajectory $\omega_{\mathbf{x}}$ to be $\rho = \rho_0\rho_1 \ldots$, where $\rho_i = L(\omega_{\mathbf{x}}[i])$ for all $i \geq 0$.

To formally specify spacecraft requirements, we use *co-safe* and *safe* LTL [18], which are languages that can express the temporal behaviors of a system with a set of Boolean connectives and temporal operators. Co-safe LTL is used to describe tasks that the spacecraft should achieve.

Definition 3 (Co-safe LTL)**.** *Given a set of atomic propositions Π, a co-safe LTL formula is recursively defined as*

$$\varphi = \mathrm{p} \mid \neg\mathrm{p} \mid \varphi \wedge \varphi \mid \mathcal{X}\varphi \mid \varphi\mathcal{U}\varphi \mid \mathcal{F}\varphi$$

where $\mathrm{p} \in \Pi$, $\neg$ ("not") and $\wedge$ ("and") are Boolean connectives, and $\mathcal{X}$ ("next"), $\mathcal{U}$ ("until"), and $\mathcal{F}$ ("eventually") are temporal operators.

Safe LTL is then used to define behaviors that the spacecraft should avoid.

Definition 4 (Safe LTL)**.** *Given a set of atomic propositions Π, a safe LTL formula is inductively defined as*

$$\varphi = \mathrm{p} \mid \neg\mathrm{p} \mid \varphi \wedge \varphi \mid \mathcal{X}\varphi \mid \mathcal{G}\varphi$$

where $\mathrm{p} \in \Pi$, $\neg$, $\wedge$, and $\mathcal{X}$ are as in Definition 3 and $\mathcal{G}$ ("globally") is a temporal operator.

The semantics of safe and co-safe LTL are defined over infinite traces [18]. An infinite trajectory $\omega_{\mathbf{x}} \in \Omega_{\mathbf{x}}$ satisfies an LTL formula φ, denoted as $\omega_{\mathbf{x}} \models \varphi$, if its trace satisfies φ. For our problem, we consider specifications of form

$$\varphi = \varphi_L \wedge \varphi_S,$$

where φ_L is a *liveness* specification that describes the task that the spacecraft should achieve given as a co-safe LTL formula, and φ_S is a *safety* specification that identifies what the spacecraft must avoid as a safe LTL formula. While the satisfaction of co-safe LTL formula are defined over infinite trajectories, we can assess if a co-safe LTL formula is satisfied with a finite trajectory. Similarly, we can only assess satisfaction of a safe LTL formula over infinite trajectories, but the negation of a safe LTL formula ($\neg\varphi_S$) is a co-safe LTL formula, whose satisfaction can be assessed on finite trajectories. Then, if a finite trajectory satisfies $\neg\varphi_S$, it cannot satisfy φ_S, i.e., the trajectory violates φ_S. A length $N \in \mathbb{N}$

prefix of $\omega_{\mathbf{x}}$, denoted as $\omega_{\mathbf{x}}^N$, satisfies $\varphi = \varphi_L \wedge \varphi_S$ iff $\omega_{\mathbf{x}}^N$ does not violate φ_S and satisfies φ_L, i.e.,

$$\omega_{\mathbf{x}}^N \models \varphi_L \wedge \varphi_S \quad \text{iff} \quad \omega_{\mathbf{x}}^N \not\models \neg\varphi_S \wedge \exists i \leq N, \omega_{\mathbf{x}}^i \models \varphi_L.$$

The combination of safe and co-safe LTL enables the description of a wide variety of tasks.

Recall that the trajectories of M under π have an associated probability measure; hence, satisfaction of φ is probabilistic. The probability of trajectories of M satisfying formula φ under π is defined as

$$P(M^\pi \models \varphi) = P(\omega_{\mathbf{x}} \in \Omega_{\mathbf{x}}^{\text{fin},\pi} \mid \omega_{\mathbf{x}} \models \varphi).$$

Example 2. *Consider the spacecraft in Example 1 with the task "Image the target successfully, and only accept images that are taken when the attitude error ($|\sigma_{err}|$) is less than 0.008 radians and the attitude rate ($|\dot{\sigma}|$) is less than 0.002 radians per second". We define the region $r_0 = \{x \in X \mid |\sigma_{err}| < 0.008,\ |\dot{\sigma}| < 0.002,\ a \in \{a_2, a_3\},\ i = 1\}$ where $i \in \{0, 1\}$ identifies if the target is accessible for imaging. We associate the atomic proposition p_0 with r_0, i.e., p_0 is true iff $x \in r_0$. The task is then written in co-safe LTL as*

$$\varphi_{0L} = \mathcal{F}\mathrm{p}_0 \tag{1}$$

Consider an additional safety requirement as "Never allow power $\mathcal{P}$ to fall below 20% and never allow reaction wheel speeds Ω above 80% of their maximum." We define regions $r_1 : \{x \in X \mid \mathcal{P} < 0.2\}$ and $r_2 : \{x \in X \mid 0.8 < \Omega\}$ and corresponding atomic propositions $\mathrm{p}_1, \mathrm{p}_2$, resulting in a safe LTL specification

$$\varphi_S = \mathcal{G}(\neg(\mathrm{p}_1 \vee \mathrm{p}_2)). \tag{2}$$

The task is only considered satisfied if $\omega_{\mathbf{x}}^N \models \varphi_L \wedge \varphi_S$. More complex specifications can be defined similarly. [2]

C. Problem Statement

Our problem is then be defined as follows.

Problem 1 (Safe Control)**.** *Given a model of the system as an MDP M, an Earth observation task specified as a co-safe LTL formula φ_L, a safety specification as a safe LTL formula φ_S, and (safety) probability threshold p, find a policy π^* such that the probability of satisfying $\varphi_L \wedge \varphi_S$ is maximized while the probability of violating φ_S is less than or equal to p, i.e.,*

$$\pi^* = \arg\max_\pi P(M^\pi \models \varphi_L \wedge \varphi_S)$$

subject to $P(M^{\pi^*} \models \neg\varphi_S) \leq p$.

Note that there are several challenges in the above problem. First, the state space for a spacecraft can contain thousands of state parameters, resulting in a dimensionality that is too large for traditional synthesis techniques. To overcome the problem of *state explosion*, we use DRL, with the policy being captured as a NN which enables generalizability. This is possible because a high-fidelity, flight-approved spacecraft physics simulator exists, namely the Basilisk[3] astrodynamics simulation framework [19], which enables DRL for spacecraft with its computational performance and flexible framework for integration with common machine learning techniques, following the observation space setup from [3].

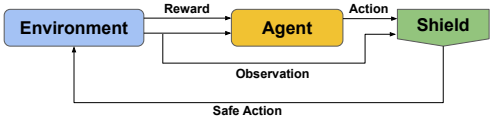

Fig. 1: Post-Posed Shielded RL architecture.

However, using a DRL policy introduces a new challenge: how can the satisfaction of the safety constraint during spacecraft operation be ensured? To address this challenge, we employ Shielded DRL (SDRL) in which a shield is constructed from φ_S and deployed with the policy to ensure the correctness of actions taken by the spacecraft. Finally, spacecraft shield design itself is a challenge due to large dimensionality. This often leads to overly conservative shields which interrupt task execution frequently and unnecessarily, significantly reducing efficiency. We propose several methods of shield design and training that mitigate this problem.

III. Reward and Shield Design for DRL

In this section, we discuss how to perform DRL with LTL specifications and describe how we construct shields for the spacecraft system. In Post-Posed SDRL, as in Figure 1, a shield acts as a monitor on the actions of the learning agent. The shield allows all safe actions and corrects unsafe actions before returning the choice to the environment.

A. Rewards for DRL with LTL Specifications

The goal of RL is to find an optimal, or nearly-optimal, policy that maximizes the expected value of a reward function. That is, given an MDP M and a reward function $R : X \times A \to \mathbb{R}$ find a policy $\pi^* = \arg\max_\pi \mathbb{E}[\sum R(s, a)]$. Traditionally, the reward function is manually constructed to define the task that the agent is learning. The manual construction of a reward function is prone to error, resulting in a disconnect between what the agent learns and the desired results. To address this issue, we write the objective in (co-safe) LTL and automatically construct a reward function from the specification. Work [20] defines a method to construct a reward function from LTL specifications, and provides a relation between the probability of satisfying the specification with the expected value of the reward. To use this method, a deterministic finite automaton (DFA) must be constructed from the co-safe LTL specification, i.e., φ_L or $\neg\varphi_S$, that accepts the same traces as φ [18].

Definition 5 (DFA)**.** *A deterministic finite automaton (DFA) constructed from an LTL formula φ is a tuple $\mathcal{A}_\varphi = (Z, z_0, 2^\Pi, \delta, Z_f)$, where Z is a finite set of states, $z_0 \in Z$ is an initial state, 2^Π is a finite set of input symbols, $\delta : Z \times 2^\Pi \to Z$ is a transition function, and $Z_f \subseteq Z$ is the set of final (accepting) states.*

[2] For more detail, see our extended version on ArXiv [1].

[3] https://hanspeterschaub.info/basilisk

A finite run on $\mathcal{A}_\varphi$ is a sequence of states $\mathbf{z} = z_0 z_1 ... z_{n+1}$ induced by a trace $\rho = \rho_0 \rho_1 ... \rho_n$ where $\rho_i \in 2^\Pi$ and $z_{i+1} = \delta(z_i, \rho_i)$. A finite run is accepting if, for some $i \leq n$, $z_i \in Z_f$. If a run is accepting, its associated trace is accepted by $\mathcal{A}_\varphi$. The set of all traces that are accepted by $\mathcal{A}_\varphi$ is called the language of $\mathcal{A}_\varphi$. The language of $\mathcal{A}_\varphi$ is equal to the language of φ, i.e., trace ρ is accepted by $\mathcal{A}_\varphi$ iff $\rho \models \varphi$.

To perform DRL given $\mathcal{A}_\varphi$ and MDP M, the product $M_{\mathcal{A}_\varphi} = M \times \mathcal{A}_\varphi$, which captures both the dynamics of M and the constraints of φ, is necessary.

Definition 6 (Product MDP)**.** *The product MDP $M_{\mathcal{A}_\varphi} = M \times \mathcal{A}_\varphi$ is a tuple $M_{\mathcal{A}_\varphi} = (S, A, \Delta, S_f, S_v)$, where $S = X \times Z$ is the set of product states, A is as in Def. 1, $\Delta : S \times A \times (\mathcal{B}(X) \times Z) \to [0, 1]$ is the transition function such that $\Delta((x, z), a, (\mathcal{B}(x'), z')) = T(x, a, \mathcal{B}(x'))$ if $z' = \delta(z, L(x))$ and 0 otherwise, and $S_f = X \times Z_f$ is the set of final states.*

With product construction in Def. 6, the states of $M_{\mathcal{A}_\varphi}$ encode the history of trajectories of M with respect to φ. Let $Z_v \subseteq Z \setminus Z_f$ be the set of DFA sink states, i.e., $\forall z \in Z_v$ and $\forall \sigma \in 2^\Pi$, $z = \delta(z, \sigma)$. We adapt the method proposed in [20] to our environment, and define the reward $R : S \to [-1, 1]$, discount function $\Gamma \in (0, 1)$, and cumulative reward V_F as:

$$R((x,z),(x',z')) = 1-\gamma_F \text{ if } z' \in Z_f,\ -1 \text{ if } z' \in Z_v,\ 1-\gamma_T \text{ if } z \neq z',\ 0 \text{ otherwise}, \quad (3)$$

$$\Gamma((x,z),(x',z')) = \gamma_F \text{ if } z' \in Z_f,\ \gamma_T \text{ if } z \neq z',\ \gamma \text{ otherwise}, \quad (4)$$

$$V_F(\omega_\mathbf{x}, \mathbf{z}, n) = \sum_{i=0}^{n} R((\omega_\mathbf{x}[i], \mathbf{z}[i]), (\omega_\mathbf{x}[i+1], \mathbf{z}[i+1])) \prod_{j=0}^{i-1} \Gamma((\omega_\mathbf{x}[j], \mathbf{z}[j]), (\omega_\mathbf{x}[j+1], \mathbf{z}[j+1])), \quad (5)$$

where $\gamma, \gamma_T, \gamma_F \in (0, 1)$ are hyper-parameters, and $\mathbf{z}[i]$ is the i-th element of run $\mathbf{z}$. While this reward is over the product states, the product does not need to be constructed explicitly; it is sufficient to implicitly construct it for reward evaluation.

Training with this reward is not conditioned on the presence of a shield and is then applicable for generic DRL tasks. As we train with co-safe LTL specifications which can be satisfied in finite time, it would be natural to terminate a training episode once an accepting state of the DFA is reached. In order to expand the search space, we instead reset to the initial state of the DFA if we reach an accepting state and allow the agent to continue training. Intuitively, this means a trajectory is more valuable if it can satisfy a specification quickly and frequently. This can result in rewards much larger than 1 in the event a specification can be satisfied multiple times in an episode. If we reach a sink state in Z_v of the DFA, we cannot satisfy φ, and hence the episode is terminated and the reward is reduced by 1.

We note that the reward construction proposed in [20] produces a direct relation between the probability of satisfying a specification and the cumulative reward received. However, reward is only received when reaching an accepting state of the DFA which can result in extremely sparse reward as the number of states in the DFA grows. The alterations from the original design we propose in (3)-(5), in particular, the additional reward in R when $z \neq z'$, encourage transitions along the states of the DFA. However, they also limit the direct relation between reward and the probability of satisfying the specification. As noted in our case studies, a correlation between reward and satisfaction rate is retained.

For example, the DFA constructed from φ_{0L} in Example 2 consists of only two states; an initial state and the accepting state. Here the reward formulation in [20] performs identically to our modified version. However, if the specification is more complex, the resulting DFA is likely to have more states. In a specification that requires five images with alternating imaging modes and an inclusion of safety, the DFA has seven states. In our experiments, training to satisfy this specification with the original reward results in 38.9% of trajectories ending with spacecraft failure, whereas our formulation results in only 1.4% of trajectories failing; further discussion on these benchmarks can be seen in Section IV. This difference is almost entirely due to the sparsity of reward, resulting in a sub-optimal policy when trained under the same number of epochs.

Lastly, we note that a policy $\pi : S \to A$ trained on product MDP $M_{\mathcal{A}_\varphi}$ is stationary. This policy however becomes history dependent on M, i.e., since $S = X \times \mathcal{A}_\varphi$, the history is captured by the states of $\mathcal{A}_\varphi$.

B. Shield Design

Shield design is typically based on having an MDP that describes the evolution of the safety aspects of the system in question. In our problem, the MDP that describes the evolution of the spacecraft is unknown *a priori*. This poses a major challenge for designing a shield. To that end, we abstract M to a finite MDP that fully captures the behavior of the spacecraft w.r.t. φ_S and refer to it as the Safety MDP. We obtain this abstraction by partitioning state space X such that the partition respects the regions of interest that correspond to φ_S. Let $R_S \subseteq R$ be the set of safety regions. Then, the Safety MDP is defined as follows.

Definition 7 (Safety MDP)**.** *Given a partition of X that respects the regions of interest in $R_S \subseteq R$, the* safety *MDP is a finite-state MDP $\bar{M} = (Q, A, P, \bar{\Pi}, \bar{L})$, where A is as in Def. 1, $Q = \{q_1, \ldots, q_m\}$ is a finite set of states obtained from the partition of X, i.e., $q_i \subseteq X$, $P : Q \times A \times Q \to [0, 1]$ is a transition probability function such that, for every $q, q' \in Q$ and $a \in A$, $P(q, a, q') = \mathbb{E}_{x \sim D(q)}[T(x, a, q')]$, where $D(q)$ is a probability distribution over region q, $\bar{\Pi} \subseteq \Pi$ is a set of (safety) atomic propositions that are associated with the regions in R_S, $\bar{L} : Q \to 2^{\bar{\Pi}}$ is a labeling function such that $\bar{L}(q) = L(x) \cap \bar{\Pi}$ for all $x \in q$.*

In general, the Safety MDP $\bar{M}$ does not require the full dimensionality of the MDP M. This dimensionality reduction enables rigorous safety analysis while maintaining computational tractability when constructing a shield. Specifically, for the spacecraft in Example 1, we are interested in

regulating body rates $|\dot{\sigma}|$, reaction wheel speeds Ω, and stored charge $\mathcal{P}$. As in RL, identifying an appropriate state space for the spacecraft is challenging, as the true state space may include thousands of states [4] and the problem must remain Markovian [21] in a lower dimensionality. We propose a reduction of the state space to just the values of interest: $|\dot{\sigma}|, \Omega, \mathcal{P}$. These states on their own are not Markovian. For instance, the attitude rate in the next time step cannot be predicted just from the current attitude rate as it also depends on states such as the attitude error. We approach this problem by identifying the transition probabilities between states of the safety MDP through simulation.

We first define a domain $\bar{X} \subset X$ in which the spacecraft can safely operate, e.g., $|\dot{\sigma}| \leq 0.01$, $\Omega \leq 1$, and $0 < \mathcal{P} \leq 1$. We then partition $\bar{X}$, defining the states Q of $\bar{M}$. Due to the complexity of the dynamics, we compute the transition probabilities P through simulation of M using Basilisk [19]. For each state $q \in Q$ of the MDP, we initialize the spacecraft simulation with randomized parameters (e.g. orientation) and limit the three states of interest to the values described by the discretization (e.g. $0 \leq \Omega < 0.2$) and simulate the evolution of the system N_P times, where N_P is a large number ($N_P = 10,000$ in our case studies) under each action. This allows the identified transition probabilities to no longer be conditioned on aspects such as orientation, hence the abstraction of the safety MDP remains Markovian.

Shield Algorithms. With the safety MDP constructed above, shield synthesis is enabled. Given $\bar{M}$, the goal is to find the set of *all* safe policies that guarantee no violation to φ_S with at least probability $1 - p$. As the policies can be history dependent, finding the set of all policies can lead to combinatorial problems and computational intractability. Hence, we focus on finding a set of stationary policies on the product MDP with the DFA corresponding to φ_S.

Note that the DFA $\mathcal{A}_{\neg\varphi_S}$ can be constructed that precisely accepts all the safety-violating traces. The product $\bar{M}_{\neg\varphi_S} = \bar{M} \times \mathcal{A}_{\neg\varphi_S}$ can be constructed per Def. 6 with the set of states $\bar{S} : Q \times Z$, transition probability function $\bar{\Delta} : \bar{S} \times A \times \bar{S} \to [0, 1]$, and set of final states $\bar{S}_f$. Note that the paths of $\bar{M}_{\neg\varphi_S}$ that reach $\bar{S}_f$ violate φ_S. Hence, our aim becomes to find the set of stationary policies on $\bar{M}_{\neg\varphi_S}$, under which the probability of the paths that reach $\bar{S}_f$ is at most p.

We note that the traditional shield construction methods as in [7] are based on game formulation. Those approaches strictly require all the paths of the MDP to remain safe all the time, which is analogous to requiring a violation probability threshold of $p = 0$, which result in very conservative designs. For example, in our case studies, these approaches never produce a safe action. Therefore, we relax such strong requirement and focus on probabilistic shields, allowing a small violation probability up to a threshold p, and propose three different shield designs that result in safe actions for our MDP, each design with a different guarantee on safety.

1) One-Step Safety: The first shield we investigate is a one-time step safety shield, i.e., we find actions that enable transition to the safe set $\bar{S} \setminus \bar{S}_f$ with high probability. This shield is the simplest to implement, as actions that are safe on the product $\bar{M} \times \mathcal{A}_{\neg\varphi_{S1}}$ can be directly assessed from the transition probabilities of the MDP $\bar{M}$ given a state and action. This design provides a guarantee that the system remains safe with probability $1 - p$ for at least one time step; however, there are no guarantees on long term safety. We allow any action that transitions to a safe state, even if the next states have no safe actions themselves. Then the shield is defined as, for every $s \in \bar{S}$,

$$\pi^1_{\textit{shield}}(s) = \{a \in A \mid \sum_{s' \in \bar{S}_f} \bar{\Delta}(s, a, s') < p\}.$$

2) Two-Step Safety: As the first design has no long term safety guarantees, we design a shield that only allows actions that have a high probability of transitioning to safe states, where a safe state is recursively defined as a state where there is a safe action. This guarantees the system will remain safe for two consecutive time steps with probability $\geq 1 - p$ at each step. Two steps are considered sufficient as transition probabilities in MDPs are not history dependent; hence, this guarantee holds for each time step the system begins in a safe state. We call the set of unsafe states U and initialize it with $U = \bar{S}_f$. Then, states are recursively added to U when no safe action is available. The process repeats until we reach a fixed point (U gains no more states). The algorithm is as follows.

$U = \bar{S}_f$
`While` $U \neq U'$
  $U' = U$
  $\forall s \in \bar{S},\ \pi^2_{\textit{shield}}(s) = \{a \in A \mid \Sigma_{s' \in U} \bar{\Delta}(s, a, s') < p\}$
  $U = U \cup \{s \in \bar{S} \mid \pi^2_{\textit{shield}}(s) = \emptyset\}$
`Return` $\bar{S} \setminus U,\ \pi^2_{\textit{shield}}$

In the event that the system transitions into the unsafe states (U for $\pi^2_{\textit{shield}}$ or $\bar{S}_f$ for $\pi^1_{\textit{shield}}$), we select the action with the highest probability of returning to the safe set.

3) Q-optimal Safety: The final shield design we assess is based on dynamic programming, which results in the strongest guarantees for safety. We consider safety horizon $N \in \mathbb{N} \cup \{\infty\}$ and use a dynamic programming approach to compute the policy that minimizes probability of reaching unsafe set $\bar{S}_f$ in N steps, resulting in optimal policy π^*_S and value function V^*_S, i.e.,

$$V^\pi_S(s) = P(\omega^N_{\mathbf{x}} \in \Omega^{\text{fin},\pi}_{\mathbf{x}} \mid \omega^N_{\mathbf{x}}[0] \in s, \\ \omega^N_{\mathbf{x}}[i] \in \bar{S}_f \text{ for some } i \in \mathbb{N}),$$

and $\pi^*_S = \arg\min_\pi V^\pi_S$ and $V^*_S = V^{\pi^*_S}_S$. When the shield is used, we assess the Q-value of each action and only allow actions that have a (unsafety probability) value below threshold p. Then, the shield is defined as

$$\pi^{\text{Q}}_{\textit{shield}}(s) = \{a \in A \mid \sum_{s' \in \bar{S}} \bar{\Delta}(s, a, s') V^*_S(s') < p\}.$$

In the event no action satisfies the bound p, the shield uses the optimal policy π^*_S, i.e., if for a $s \in \bar{S}$, $\pi^{\text{Q}}_{\textit{shield}}(s) = \emptyset$, then $\pi^{\text{Q}}_{\textit{shield}}(s) = \pi^*_S(s)$.

Remark 1. *Following convention from [8], there is no penalty (reward) associated with the post-posed shield changing the action during learning.*

IV. Case Studies

We evaluate the efficacy of our LTL-SDRL framework on the spacecraft in Example 1. Here, we present a summary of our results; for more results and discussions, see our extended version [1]. We first show the importance of training with safety specification on a simple task scenario. Then, we consider a complex scenario for benchmarking (comparing) the three shield designs. The shields are each designed to prevent the spacecraft from having less than 20% power and wheel speeds above 80%. We also assess training with the shield and training without the shield.

In all the case studies, the Basilisk [19] simulator is used to create an environment for implementation of PPO2 [22]. The agent is trained with a learning rate $\alpha = 3 \times 10^{-4}$ with a network composed of two hidden layers of width 10 with a hyperbolic tangent activation function. The network is trained for 4.6×10^5 time steps on an Intel Core i7-12700K CPU at 3.60GHz with 32 GB of RAM limited to 8 threads. The input space of the network contains information on 19 states, composed as 13 states with 12 as in [3] and the addition of the DFA state. The Safety MDP consists of 100 discrete states. All validations are computed on 1000 simulation runs.

A. Simple Task: Importance of φ_S in Training

We first demonstrate the effect of learning with and without a safety specification in (unshielded) DRL. We use the formula φ_{0L} and φ_S in (1)-(2) from Example 2 and train for a fixed orbit and fixed target location. We compare the frequency of satisfying the specifications and the frequency of spacecraft failures over 300 minutes (100 time steps), which is longer than the time needed for one orbit (271 minutes). Here, we expect to see a high satisfaction rate of φ_{0L} and some violations of φ_S. Results are shown in Table I.

We first train on a DFA constructed from φ_{0L} and refer to the policy returned from DRL as π_0. When we deploy with π_0, we see a high satisfaction rate of φ_0, frequent violations of φ_S and even some spacecraft failures. This is expected, as without shielding and with no information in the reward about safety the policy never learns to avoid unsafe behavior.

When we train on a DFA constructed from $\varphi = \varphi_{0L} \wedge \varphi_S$, which implicitly incorporates the safety requirements into the reward structure, we find a policy π_1. We see a similarly high satisfaction rate of φ_{0L}; however, very few runs violate φ_S and no runs end with spacecraft failure. This demonstrates the power of incorporating safety into the training specification. We show sample trajectories Figure 2.

TABLE I: Training results for a simple task. Reported values are average value V_F during training, rates of satisfaction of φ_{0L} and violation of φ_S, and spacecraft (SC) failure rate.

Spec.	Avg. V_F	% Sat. φ_{0L}	% Violate φ_S	SC Failure
φ_{0L}	3.295	99.6	27.1	3.4
$\varphi_{0L} \wedge \varphi_S$	3.1	99.0	3.5	0

B. Complex Tasks and Shielding

We consider a more complex imaging task, to highlight the flexibility provided by training with LTL specifications and demonstrate the effects of shielding. Here we train to satisfy a task for a random LEO and random target locations. Our task is "Image targets at least five times, alternating imaging modes starting from Imaging mode A. Only accept images that are taken when the attitude error ($|\sigma_{err}|$) is less than 0.008 radians and the attitude rate ($|\dot{\sigma}|$) is less than 0.002 radians per second.", translated into the co-safe LTL formula

$$\varphi_{1L} = \mathcal{F}(\mathrm{p}_3 \wedge \mathcal{X}\mathcal{F}(\mathrm{p}_4 \wedge \mathcal{X}\mathcal{F}(\mathrm{p}_3 \wedge \mathcal{X}\mathcal{F}(\mathrm{p}_4 \wedge \mathcal{X}\mathcal{F}\mathrm{p}_3))))$$

where p_3 and p_4 are similar to p_0 from Example 2, but the corresponding regions only include one imaging mode. Note that in traditional reward design a finite state machine would need to be constructed to account for mode switching; however, in our formulation the switching is automatically captured through the specification. The safety specification is the same φ_S in (2) as above with $p = 0.05$.

For this scenario, each episode consists of 90 actions which corresponds to three orbits. We assess the average value V_F during training, the satisfaction rate of specification with and without the shield after the agent has been trained, the average number of shield interventions, and the frequency of unsafe behavior. Results are shown in Table II.

1) Trained and Deployed without Shielding: Similar to the simple specification, training with safety included results in a lower reward, fewer spacecraft failures, and fewer safety violations. Due to the more stressing environment, we see more safety violations during training which results in the large gap in value V_F between the two policies. However there is still a correlation between V_F and the probability of satisfying the liveness specification.

2) Trained without Shielding and Deployed with Shielding: We first note that in each case a shield is deployed, there are no instances of spacecraft failure and the frequency of violations of φ_S fall below the threshold of 5%. When we train with just φ_{1L}, we see a significant reduction in satisfaction rate and a high frequency of shield interventions. This follows intuition, as over 50% of trajectories violate φ_S without shielding. Contrarily, when we train on $\varphi_{1L} \wedge \varphi_S$, we see far fewer shield interventions and a higher rate of satisfaction of φ_{1L}, showing that the policy learned from φ_S acts more flexibly when shielded. In all cases, we see that far more shield interventions occur when a trajectory does not satisfy φ_{1L} as actions must be repeatedly corrected to maintain guarantees.

3) Trained and Deployed with Shielding: Minor differences are seen between the shields here. As noted in [8], the policy learns to rely on the shield interventions which results in far fewer trajectories satisfying φ_{1L}. The improvements seen when training with φ_S are more noticeable, despite the same number of shield interventions occurring showing the correlation between V_F and the satisfaction rate.

TABLE II: Results for φ_{1L}. We report average value V_F during training, liveness satisfaction rate (% Sat. φ_{1L}), safety violation rate (% Violate φ_S), spacecraft failure rate (SC Fail, % of tests), and the average number of shield interventions (over 90 time steps) when φ_{1L} was satisfied and not satisfied. $*1$ and $*2$ denote that the same policy was used for these experiments, i.e., they were trained without a shield.

Shield Type	Trained w/ Shield	Spec.	Avg. V_F	% Sat. φ_{1L}	% Violate φ_S	SC Failure	Avg. # of Interventions	
							Sat. φ_{1L}	Not Sat. φ_{1L}
No Shield	No^{*1}	φ_{1L}	3.39^{*1}	92.7	55.3	10.9	–	–
	No^{*2}	$\varphi_{1L} \wedge \varphi_S$	2.47^{*2}	90.9	11.7	1.4	–	–
One Step (π^1_{shield})	No^{*1}	φ_{1L}	3.39^{*1}	79.2	0.4	0	13.6	29.2
	No^{*2}	$\varphi_{1L} \wedge \varphi_S$	2.47^{*2}	**85.9**	0.6	**0**	**3.7**	**11.1**
	Yes	φ_{1L}	1.88	64.8	0.8	0	44.0	60.6
	Yes	$\varphi_{1L} \wedge \varphi_S$	1.93	74.9	0.6	0	48.6	66.8
Two Step (π^2_{shield})	No^{*1}	φ_{1L}	3.39^{*1}	82.0	0.6	0	13.2	28.5
	No^{*2}	$\varphi_{1L} \wedge \varphi_S$	2.47^{*2}	**86.2**	0.9	**0**	**3.9**	**10.9**
	Yes	φ_{1L}	1.84	61.6	1.0	0	41.4	59.9
	Yes	$\varphi_{1L} \wedge \varphi_S$	1.86	72.6	1.2	0	47.1	64.0
Q-optimal (π^Q_{shield})	No^{*1}	φ_{1L}	3.39^{*1}	79.9	0.2	0	14.2	28.9
	No^{*2}	$\varphi_{1L} \wedge \varphi_S$	2.47^{*2}	**85.2**	0.4	**0**	**3.8**	**11.2**
	Yes	φ_{1L}	1.70	63.2	0.5	0	44.2	61.0
	Yes	$\varphi_{1L} \wedge \varphi_S$	2.23	73.4	0.2	0	49.1	66.9

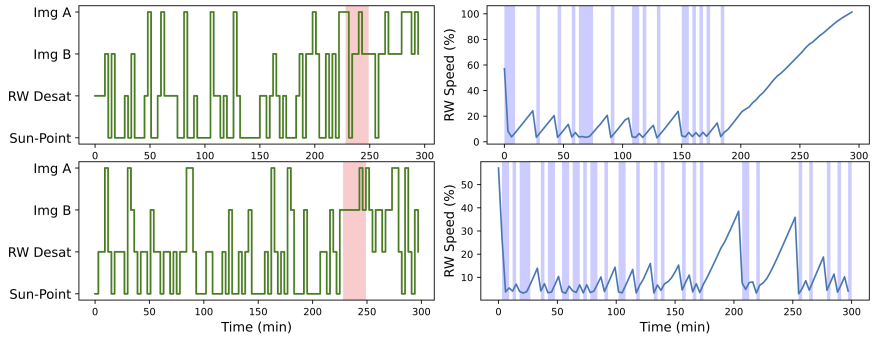

Fig. 2: Action history and reaction wheel speeds when deploying under policy π_0 (top) and π_1 (bottom) from a fixed initial condition. The red highlight shows access to the target, the blue highlights show when Momentum Dumping (RW Desat) occurs. Note, policy π_1 maintains safety after imaging the target whereas policy π_0 prioritizes imaging over survival.

V. Conclusion

In this work, we provide a method to enable autonomous decision making for the satisfaction of complex Earth imaging tasks through SDRL. Safety is ensured through the designs of three shield algorithms, which are based on a simulated safety MDP. We identify that training on composed liveness and safety specifications without a shield results in a high rate of satisfaction of liveness, few violations of safety, and few shield interventions. While the shields are less restrictive than prior designs, the safety guarantees hold for the safety MDP which was generated through empirical evaluation (physics simulator). Numerical simulations show the shields provide stronger guarantees than necessary, suggesting that the safety MDP contains overly conservative transition probabilities, resulting in similar policies among the shields. The states chosen to represent safety likely increase conservatism as these states have complex transitions which require the shields to select actions for worst case orientations (e.g., vector norms have no information about direction). We hope to further formalize the safety MDP in future work and reduce the conservatism seen by shielding.

References

[1] R. Reed, H. Schaub, and M. Lahijanian, "Shielded deep reinforcement learning for complex spacecraft tasking," *arXiv: 2403.05693*, 2024. [Online]. Available: https://arxiv.org/pdf/2403.05693.pdf

[2] C. Frost, A. Butt, and D. Silva, "Challenges and opportunities for autonomous systems in space," in *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2010 Symposium*, 2010.

[3] A. T. Harris and H. Schaub, "Spacecraft command and control with safety guarantees using shielded deep reinforcement learning," in *AIAA Scitech 2020 Forum*, 2020.

[4] I. Nazmy, A. Harris, M. Lahijanian, and H. Schaub, "Shielded deep reinforcement learning for multi-sensor spacecraft imaging," in *ACC*. IEEE, 2022.

[5] C. Adams, B. Kempa, M. Iatauro, J. Frank, and W. Vaughan, "An overview of distributed spacecraft autonomy at nasa ames," in *Small Satellite Conference*, no. 37, 2023.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, 2017.

[7] R. Bloem, B. Könighofer, R. Könighofer, and C. Wang, "Shield synthesis: Runtime enforcement for reactive systems," in *TACAS*. Springer, 2015.

[8] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[9] A. Pnueli, "The temporal logic of programs," in *SFCS*. IEEE, 1977.

[10] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010.

[11] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE RA Magazine*, vol. 18, 2011.

[12] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 32, no. 3, 2016.

[13] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, 2018.

[14] M. Lahijanian, S. B. Andersson, and C. Belta, "Formal verification and synthesis for discrete-time stochastic systems," *IEEE TAC*, 2015.

[15] M. Wen, R. Ehlers, and U. Topcu, "Correct-by-synthesis reinforcement learning with temporal logic constraints," in *IROS*. IEEE, 2015.

[16] S. Junges, N. Jansen, C. Dehnert, U. Topcu, and J.-P. Katoen, "Safety-constrained reinforcement learning for mdps," in *TACAS*, 2016.

[17] M. Lahijanian, S. Andersson, and C. Belta, "Control of markov decision processes from pctl specifications," in *ACC*. IEEE, 2011.

[18] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal methods in system design*, vol. 19, pp. 291–314, 2001.

[19] P. W. Kenneally, S. Piggott, and H. Schaub, "Basilisk: A flexible, scalable and modular astrodynamics simulation framework," *Journal of aerospace information systems*, vol. 17, no. 9, pp. 496–507, 2020.

[20] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Mungojerrie: Linear-time objectives in model-free reinforcement learning," in *TACAS*. Springer, 2023.

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

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.