ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240623

Shielded FOND: Planning with Safety Constraints in Pure-Past Linear Temporal Logic

Luigi Bonassi[a], Giuseppe De Giacomo[b,c], Alfonso Emilio Gerevini[a] and Enrico Scala[a]

[a]University of Brescia, Italy
[b]University of Oxford, UK
[c]Sapienza University, Italy

Abstract. In this paper, we introduce *Shielded* FOND planning (S-FOND), which is the problem of computing a strategy to reach a final-state goal while preserving a safety specification called shield. In particular, we characterize shields as Pure-Past Linear Temporal Logic formulas that must hold in every prefix of a state trace induced by a solution strategy, thus capturing the whole *safety* fragment of Linear Temporal Logic formulas over finite traces. We propose three solution encodings for handling S-FOND problems: the first, which is our baseline, simply views a shield as a temporally extended goal; the second, instead, blocks the execution of further actions when the shield gets violated, and the third prevents the execution of actions that could violate the shield by using the notion of regression. We formally prove the correctness of each encoding and experimentally prove their effectiveness over a set of benchmark shields.

1 Introduction

In this paper, we introduce *Shielded* Fully Observable Non-Deterministic planning (S-FOND), which is the task of synthesizing a strategy that reaches a final-state goal while inducing a state trace that conforms to a safety specification which, in analogy with recent work on safe reinforcement learning [1], is called *shield*. In particular, we focus on shields that capture the *safety* fragment [43] of Linear Temporal Logic (LTL) formulas [48] over finite traces [26] (LTL_f). Safety formulas are those that, when violated by a trace prefix, will remain violated by every extension of such a prefix. That is, a safety formula specifies that "Something bad" never happens. Examples of shields include simple propositional conditions that must hold in every state, such as "Always preserve a certain amount of battery", or more complex specifications such as "The cleaning robot cannot leave the room until it has been completely cleaned".

Dealing with propositional safety constraints has already been addressed in classical planning [51]. A larger subset of shields can be captured in PDDL3 [34] and handled via compilation [12, 17] or by native planners [30, 39, 11]. However, these approaches are limited to PDDL3 and are only designed for deterministic domains.

Handling shields is closely related to planning for temporally extended goals, which has an abundant literature (e.g., [3, 7, 4, 21, 22, 25, 35, 47, 9, 8, 53, 19, 18]); the objective for temporally extended goals is to satisfy a formula over the entire state trace. However, as shown in this paper, natively dealing with shields is much more efficient than viewing shields as generic temporal goals. Related are also the works on LTL_f specifications discovery and monitoring [20], reactive synthesis with safety conditions [28], and the study of control knowledge to speed up planning. Control knowledge is usually expressed in the form of temporal logic constraints [5], a hierarchy imposing partial or total ordering between actions ([46]) or by using more general constraints over the action instances ([29, 45, 13]).

Recently, Pure-Past Linear Temporal Logic (PPLTL) [41, 27], an alternative to LTL_f, has been investigated for expressing temporally extended goals [14, 15]. PPLTL is as expressive as LTL_f and looks at the trace backward by expressing properties using only past operators. Interestingly, every safety specification can be captured by imposing a PPLTL formula to hold in every prefix of a trace [43]. Moreover, handling PPLTL seems much easier than handling LTL_f, since FOND planning for PPLTL goals is EXPTIME-complete [27] , while FOND planning for LTL_f goals is 2EXPTIME-complete [24]. These properties make PPLTL a compelling formalism for expressing shields. Therefore, we focus on shields expressed in PPLTL.

Our approach for handling shields builds upon the encoding by [14, 15] for handling PPLTL goals. Specifically, this encoding allows checking the validity of a PPLTL shield as a Markovian property by exploiting the well-known [32, 42, 31, 6, 7, 52] fixpoint characterization that, similarly to the fixpoint characterization of LTL [10, 3, 4], splits a PPLTL formula into a propositional condition on the current state and a PPLTL formula to be checked at the previous instant.

The contributions of this paper are as follows. We introduce and formalize shielded FOND planning. Then, we propose three encodings named Π^{TEG}, Π^{VDT}, and Π^{RE} to reduce S-FOND to FOND planning for reachability (i.e., final-state) goals. The first encoding Π^{TEG}, which is our baseline, reformulates a shield as a temporally extended goal and then uses the technique by [15] *as-is* to obtain a FOND problem for reachability goals. Differently, Π^{VDT} and Π^{RE} modify the encoding schema by [15] to better capture the validity of a shield while planning. In particular, Π^{VDT} extends the preconditions of actions to block the execution when the shield becomes violated, while Π^{RE} uses the notion of regression to block actions from violating the shield. We show that each encoding is correct and empirically prove the effectiveness of these approaches on a set of S-FOND benchmarks. The results show that handling shields natively with Π^{VDT} and Π^{RE} is much more effective than with the baseline Π^{TEG}.

2 Background on FOND Planning

A FOND domain model is a tuple $\mathcal{D} = \langle \mathcal{F}, \mathcal{X}, A, Pre, Eff, s_0 \rangle$ where $\mathcal{F}$ is a set of fluents (atomic propositions), $\mathcal{X}$ is a set of axioms,

A is a set of action labels, Pre and Eff are two functions that define the preconditions and effects of each action $a \in A$, and $s_0 \subseteq \mathcal{F}$ is an initial state. As usual, a planning state $s \subseteq \mathcal{F}$ is a set of fluents where f holds in s if $f \in s$ and f is false in s otherwise. Axioms have the form $d \leftarrow \psi$ where d is a *derived predicate* and ψ a formula over $\mathcal{F} \cup \mathcal{F}_{der}$, where $\mathcal{F}_{der} = \{d \mid d \leftarrow \psi \in \mathcal{X}\}$ is the set of all derived predicates. Intuitively, an axiom $d \leftarrow \psi$ specifies that the truth of d in a state s is derived from the truth of ψ in s. For example, let $\mathcal{F} = \{a, b, c\}$. The two axioms $d_1 \leftarrow a \wedge b$ and $d_2 \leftarrow d_1 \wedge \neg c$ specify that "d_1 is true if a and b are true, while d_2 is true if d_1 is true and c is false". Deriving d_2 requires determining d_1 first, which instead can be derived directly from the current state. In general, assuming the set of axioms $\mathcal{X}$ to be *stratified* [37] guarantees that it is always possible to efficiently and uniquely determine the truth of all $d \in \mathcal{F}_{der}$ in any state s. The functions Pre and Eff map an action label $a \in A$ to a propositional formula over $\mathcal{F} \cup \mathcal{F}_{der}$ and a set $\{\mathsf{eff}_1, \ldots, \mathsf{eff}_n\}$ of effects, respectively. Each effect $\mathsf{eff}_i \in Eff(a)$ is a set of conditional effects each of the form $c \triangleright e$, where c is a propositional formula over $\mathcal{F} \cup \mathcal{F}_{der}$ and $e \subseteq \mathcal{F} \cup \{\neg f \mid f \in \mathcal{F}\}$ is a set of literals from $\mathcal{F}$ (effects cannot modify derived predicates). An action a can be applied in a state s only if $s \models Pre(a)$. A conditional effect $c \triangleright e$ is triggered in a state s if c is true in s. Applying a in s yields a successor state s' determined by an outcome *nondeterministically* drawn from $Eff(a)$. Let $\mathsf{eff}_i \in Eff(a)$ be the selected effect, the new state s' is such that $\forall f \in \mathcal{F}$, f holds true in s' if and only if either (*i*) f was true in s and no conditional effect $c \triangleright e \in \mathsf{eff}_i$ triggered in s deletes it ($\neg f \in e$) or (*ii*) there is a conditional effect $c \triangleright e \in \mathsf{eff}_i$ triggered in s that adds it ($f \in e$). We assume a delete-before-adding semantics [50] and use the notation $s[\mathsf{eff}_i]$ to indicate the successor state s' induced by eff_i in s and $tr(s, a)$ to denote the set of possibles successor states $\{s'_1, \ldots, s'_n\}$ obtained by executing a in s. We say that a state trace $\tau = s_0, s_1, \ldots, s_n$ is a *potential trace* of $\mathcal{D}$ iff there exists a sequence of actions $a_0, a_1, \ldots, a_{n-1}$ from $\mathcal{D}$ such that $s_{i+1} \in tr(s_i, a_i)$ for every $0 \leq i < n$. If for every $0 \leq i < n$ we also have that $s_i \models Pre(a_i)$, then we say that τ is a trace of $\mathcal{D}$. Moreover, a state s_i is *reachable* if $\tau = s_0, \ldots, s_i$ is a trace of $\mathcal{D}$.

We now review two classes of FOND problems known as FOND planning for *reachability* goals and FOND planning for *temporally extended goals* in PPLTL. A reachability goal is a property that has to hold the final state reached by a sequence of actions.

Definition 1. *A FOND problem for reachability goals is a tuple $\Gamma = \langle \mathcal{D}, G \rangle$, where $\mathcal{D}$ is a FOND domain and G a formula over $\mathcal{F} \cup \mathcal{F}_{der}$.*

Solutions to Γ are *strategies* (aka *policies*). A strategy is defined as a partial function $\pi : (2^{\mathcal{F}})^+ \rightarrow A$ mapping a sequence of *non-goal* states into an applicable action. We say that π is a strategy for Γ if, starting from the initial state s_0, π induces a set of (possibly infinite) state traces (or executions) of the form $\tau = s_0, s_1, \ldots$, where $a_i = \pi(s_0, \ldots, s_i)$, $s_{i+1} \in tr(s_i, a_i)$, and $s_i \models Pre(a)$ for $i \geq 0$. If for a certain sequence of states $\tau = s_0, \ldots, s_n$ we get $\pi(\tau)$ undefined (no action prescribed), then the generated trace τ is *finite*. A strategy can be either a *strong* or a *strong-cyclic* solution [23]. A strategy π is a *strong solution* to Γ if every generated execution is a finite trace τ such that the last state s_n of τ entails G, while π is *strong-cyclic* if every generated execution that is a *stochastic-fair trace* [2] is also a finite trace such that $s_n \models G$. Intuitively, stochastic fairness assumes that every outcome of an action can occur with a non-zero probability. So, assuming stochastic fairness, a strategy π is a strong-cyclic solution if every trace induced by π terminates satisfying the goal with probability 1. When a strategy π is a solution (either strong or strong-cyclic), we also say that π is *winning*.

Unlike reachability goals, temporally extended goals are evaluated over the state trace induced by a strategy. In particular, we consider PPLTL temporally extended goals. Given a set $\mathcal{F}$ of propositions, PPLTL formulas are defined as:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{Y}\varphi \mid \varphi \,\mathsf{S}\, \varphi$$

where $p \in \mathcal{F}$, Y is the *yesterday* operator and S is the *since* operator. PPLTL formulas are interpreted on *finite nonempty* traces $\tau = s_0, \ldots, s_n \in (2^{\mathcal{F}})^+$. We denote by $\mathsf{length}(\tau)$ the length $n+1$ of τ, and by $\mathsf{last}(\tau)$ the last element of the trace. We define the satisfaction relation $\tau, i \models \varphi$, stating that φ holds at instant i, as follows:

- $\tau, i \models p$ iff $\mathsf{length}(\tau) \geq 1$ and $p \in s_i$ (for $p \in \mathcal{P}$);
- $\tau, i \models \neg\varphi$ iff $\tau, i \not\models \varphi$;
- $\tau, i \models \varphi_1 \wedge \varphi_2$ iff $\tau, i \models \varphi_1$ and $\tau, i \models \varphi_2$;
- $\tau, i \models \mathsf{Y}\varphi$ iff $i \geq 1$ and $\tau, i-1 \models \varphi$;
- $\tau, i \models \varphi_1 \,\mathsf{S}\, \varphi_2$ iff there exists k, with $0 \leq k \leq i < \mathsf{length}(\tau)$ such that $\tau, k \models \varphi_2$ and for all j, with $k < j \leq i$, we have that $\tau, j \models \varphi_1$.

For convenience, we report the semantics of the *once* (O) and the *historically* (H) operators which can be derived by the since operator.

- $\tau, i \models \mathsf{O}\varphi$ iff there exists k with $0 \leq k \leq i$ such that $\tau, k \models \varphi$;
- $\tau, i \models \mathsf{H}\varphi$ iff for all k with $0 \leq k \leq i$ we have $\tau, k \models \varphi$;

Intuitively, $\mathsf{Y}\varphi$ specifies that φ has to hold at the previous step, $\varphi_1 \,\mathsf{S}\, \varphi_2$ specifies that φ_1 has been true since φ_2 became true, $\mathsf{O}\varphi$ specifies that φ has to hold sometime in the past, and $\mathsf{H}\varphi$ specifies that φ always held in the past. As usual, disjunctions and implications can be expressed through negations and conjunctions. A PPLTL formula φ is *true* in τ, denoted $\tau \models \varphi$, if $\tau, \mathsf{length}(\tau)-1 \models \varphi$. We denote by $\mathsf{sub}(\varphi)$ the set of all subformulas of φ [26], and say that $|\mathsf{sub}(\varphi)|$ defines the size of a PPLTL formula φ.

Definition 2. *A FOND problem for* PPLTL *goals is a tuple $\Gamma = \langle \mathcal{D}, \varphi \rangle$, where $\mathcal{D}$ is a FOND domain and φ a* PPLTL *formula over $\mathcal{F}$.*

For PPLTL goals, a strategy $\pi : (2^{\mathcal{F}})^+ \rightarrow A$ is a strong (strong-cyclic, resp.) solution for Γ if every trace τ (that is stochastic-fair, resp.) induced by π is finite and satisfies φ. Note that a strategy can be represented (with finite memory) as a finite-state transducer [24].

3 Encoding PPLTL Formulas in FOND Domains

This section briefly reviews the technique by [15] to encode a PPLTL formula φ into a FOND domain $\mathcal{D}$ to obtain a new domain $\mathcal{D}_\varphi$ where we can evaluate φ by only considering the current state. The encoding monitors the truth of a specific set of subformulas of φ by introducing a set of fresh fluents denoted with Σ_φ. These, if properly combined with the original fluents, can be used to determine the truth of φ state-by-state. This strategy requires properly setting the initial state and incrementally updating the truth of Σ_φ fluents as actions are applied. Specifically, given a FOND domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{X}, A, Pre, Eff, s_0 \rangle$ and a formula φ, we obtain a new domain $\mathcal{D}_\varphi = \langle \mathcal{F}', \mathcal{X}', A, Pre, Eff', s'_0 \rangle$ defined in the following paragraphs.

Additional Fluents. To determine the subformulas we need to monitor, we exploit the well-known fixpoint characterization of PPLTL [33, 31, 42] that allows splitting a PPLTL formula into present and past components by recursively applying the following transformation function $\mathsf{pnf}(\cdot)$:

- $\mathsf{pnf}(p) = p$;

- $\mathsf{pnf}(\mathsf{Y}\phi) = \mathsf{Y}\phi$;
- $\mathsf{pnf}(\phi_1 \,\mathsf{S}\, \phi_2) = \mathsf{pnf}(\phi_2) \vee (\mathsf{pnf}(\phi_1) \wedge \mathsf{Y}(\phi_1 \,\mathsf{S}\, \phi_2))$;
- $\mathsf{pnf}(\phi_1 \wedge \phi_2) = \mathsf{pnf}(\phi_1) \wedge \mathsf{pnf}(\phi_2)$;
- $\mathsf{pnf}(\neg\phi) = \neg\mathsf{pnf}(\phi)$.

The formula $\mathsf{pnf}(\varphi)$ can be computed in linear-time in the size of φ (i.e., $|\mathsf{sub}(\varphi)|$) and $\mathsf{pnf}(\varphi) \equiv \varphi$. Most importantly, every temporal subformula of φ appears in $\mathsf{pnf}(\varphi)$ within the scope of the Y operator. This means that we can determine the truth of φ by combining the truth of the original fluents in the current state s_i with the truth of some temporal subformulas of the form $\mathsf{Y}\phi$. Therefore, we monitor these subformulas by introducing a new set of fluents Σ_φ containing one fluent "$\mathsf{Y}\phi$" for every temporal subformula of the form $\mathsf{Y}\phi$ appearing in $\mathsf{pnf}(\varphi)$. Quotation (e.g., "$\mathsf{Y}\phi$") indicates the fluent monitoring the subformula (e.g., $\mathsf{Y}\phi$). Formally, the new set of fluents is $\mathcal{F}' = \mathcal{F} \cup \Sigma_\varphi$. For convenience, we use the notation (s_i, σ_i) with $s_i \subseteq \mathcal{F}$ and $\sigma_i \subseteq \Sigma_\varphi$ to denote the state $(s_i \cup \sigma_i) \subseteq \mathcal{F}'$.

Example 1. *As a running example, we show how to encode $\varphi = a \,\mathsf{S}\, b$ into the domain $\mathcal{D}$. We have that $\mathsf{pnf}(a \,\mathsf{S}\, b) = b \vee (a \wedge \mathsf{Y}(a \,\mathsf{S}\, b))$. Hence, $\Sigma_\varphi = \{\text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"}\}$.*

Axioms. It is easy to see that if it holds that each fluent of the form "$\mathsf{Y}(\phi)$" $\in \Sigma_\varphi$ is true iff $\mathsf{Y}\phi$ is true, then we can determine the truth of φ by simply evaluating the *propositional* formula obtained by replacing every $\mathsf{Y}\phi$ with "$\mathsf{Y}(\phi)$" in $\mathsf{pnf}(\varphi)$. This formula, in planning, can be compactly represented using axioms. In particular, for every $\phi \in \mathsf{sub}(\varphi)$ we include one axiom $\mathsf{val}_\phi \leftarrow \psi$ where ψ is defined depending on the type of ϕ as follows:

- $\mathsf{val}_p \leftarrow p$;
- $\mathsf{val}_{\mathsf{Y}\phi} \leftarrow$ "$\mathsf{Y}\phi$";
- $\mathsf{val}_{\phi_1 \,\mathsf{S}\, \phi_2} \leftarrow \mathsf{val}_{\phi_2} \vee (\mathsf{val}_{\phi_1} \wedge \text{"}\mathsf{Y}(\phi_1 \,\mathsf{S}\, \phi_2)\text{"})$;
- $\mathsf{val}_{\phi_1 \wedge \phi_2} \leftarrow \mathsf{val}_{\phi_1} \wedge \mathsf{val}_{\phi_2}$;
- $\mathsf{val}_{\neg\phi} \leftarrow \neg\mathsf{val}_\phi$.

As we can see, these axioms mimic the structure of $\mathsf{pnf}(\cdot)$ except that each $\mathsf{Y}\phi$ is substituted by "$\mathsf{Y}\phi$". Moreover, these axioms allow us to conveniently evaluate any subformula ϕ of φ. Formally, $\mathcal{X}' := \mathcal{X} \cup \mathcal{X}_\varphi$ where $\mathcal{X}_\varphi = \{\mathsf{val}_\phi \leftarrow \psi \mid \phi \in \mathsf{sub}(\varphi)\}$.

Example 2. *The set of subformulas of φ is $\mathsf{sub}(\varphi) = \{a, b, a \,\mathsf{S}\, b\}$. Therefore, the new set of axioms is $\mathcal{X}_\varphi = \{\mathsf{val}_a \leftarrow a,\ \mathsf{val}_b \leftarrow b,\ \mathsf{val}_{(a \,\mathsf{S}\, b)} \leftarrow \mathsf{val}_b \vee (\mathsf{val}_a \wedge \text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"})\}$. Notice that $\mathsf{val}_{(a \,\mathsf{S}\, b)}$ is true in a state s iff s satisfies $b \vee (a \wedge \text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"})$, which is the formula obtained by replacing $\mathsf{Y}(a \,\mathsf{S}\, b)$ with "$\mathsf{Y}(a \,\mathsf{S}\, b)$" in $\mathsf{pnf}(a \,\mathsf{S}\, b)$.*

Initial state. It is crucial for the correctness of the approach that every "$\mathsf{Y}\phi$" captures the truth of $\mathsf{Y}\phi$. By the definition of PPLTL, we know that $\mathsf{Y}\phi$ is false at the start of the trace (i.e., when $i = 0$). Therefore, $s'_0 = (s_0, \sigma_0)$ where $\sigma_0 = \emptyset$.

Effects. New effects are added to keep all "$\mathsf{Y}\phi$" fluents in sync with the corresponding $\mathsf{Y}\phi$. To do so, we rely on the PPLTL semantics: if the formula ϕ holds at instant i, then $\mathsf{Y}\phi$ will hold at instant $i + 1$. Therefore, since val_ϕ encodes the truth of ϕ, we set "$\mathsf{Y}(\phi)$" to true at step $i + 1$ if val_ϕ holds at the current step i, and we falsify "$\mathsf{Y}(\phi)$" at $i + 1$ otherwise. We can do so with the pair of conditional effects $\mathsf{val}_\phi \rhd \{\text{"}\mathsf{Y}\phi\text{"}\}$ and $\neg\mathsf{val}_\phi \rhd \{\neg\text{"}\mathsf{Y}\phi\text{"}\}$. Starting from the the initial state (s_0, σ_0), where we can determine val_ϕ for every $\phi \in \mathsf{sub}(\varphi)$, these new effects iteratively keep each "$\mathsf{Y}\phi$" in sync with the corresponding $\mathsf{Y}\phi$. Formally, let $\mathsf{eff}_{\mathsf{val}} = \{\mathsf{val}_\phi \rhd \{\text{"}\mathsf{Y}\phi\text{"}\}, \neg\mathsf{val}_\phi \rhd \{\neg\text{"}\mathsf{Y}\phi\text{"}\} \mid \text{"}\mathsf{Y}\phi\text{"} \in \Sigma_\varphi\}$. The new effect function is $\mathit{Eff}'(a) := \{\mathsf{eff}_i \cup \mathsf{eff}_{\mathsf{val}} \mid \mathsf{eff}_i \in \mathit{Eff}(a)\}$ for every $a \in A$. Notice that the new conditional effects are added to *each outcome* of every action, as "$\mathsf{Y}(\phi)$" fluents are *deterministically* updated independently from the truth of the original domain fluents.

Example 3. *The set $\mathsf{eff}_{\mathsf{val}}$ of additional effects is $\mathsf{eff}_{\mathsf{val}} = \{\mathsf{val}_{a \,\mathsf{S}\, b} \rhd \{\text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"}\}, \neg\mathsf{val}_{a \,\mathsf{S}\, b} \rhd \{\neg\text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"}\}\}$.*

Definition 3. *Let $\mathcal{D} = \langle \mathcal{F}, \mathcal{X}, A, \mathit{Pre}, \mathit{Eff}, s_0 \rangle$ be a FOND domain and let φ be a* PPLTL *formula. The encoded FOND domain is $\mathcal{D}_\varphi = \langle \mathcal{F}', \mathcal{X}', A, \mathit{Eff}', \mathit{Pre}, s'_0 \rangle$.*

Theorem 1 ([15])**.** *Let $\mathcal{D}$ be a FOND domain, φ a* PPLTL *formula, and $\mathcal{D}_\varphi$ the encoded domain as by Definition 3. For every potential trace $\tau' = (s_0, \sigma_0), \ldots, (s_n, \sigma_n)$ induced by $\mathcal{D}_\varphi$, there exists a unique corresponding potential trace $\tau = s_0, \ldots, s_n$ of $\mathcal{D}$, and vice versa, such that, for every $i \leq n$, $(s_i, \sigma_i) \models \mathsf{val}_\varphi$ iff $\tau, i \models \varphi$.*

Theorem 1 implies that we can monitor the truth of φ by only considering the truth of val_φ in any state of the encoded domain $\mathcal{D}_\varphi$. A direct consequence of Theorem 1 is that we can encode FOND planning for PPLTL goals into FOND planning for reachability goals.

Corollary 2. *Let $\Gamma = \langle \mathcal{D}, \varphi \rangle$ be a FOND problem for* PPLTL *goals, and let $\Gamma' = \langle \mathcal{D}_\varphi, \mathsf{val}_\varphi \rangle$ be the corresponding encoded FOND planning problem for reachability goals. Then, Γ has a (strong or strong-cyclic) winning strategy iff Γ' has a (strong or strong-cyclic, resp.) winning strategy.*

Lastly, note that this encoding handles the H and O operators by using well-known equivalences, e.g., $\mathsf{H}\varphi \equiv \neg(\top \,\mathsf{S}\, \neg\varphi)$. For clarity, we will use $\mathsf{val}_{\mathsf{H}\varphi}$ as a shorthand for $\mathsf{val}_{\neg(\top \,\mathsf{S}\, \neg\varphi)}$.

4 Shielded FOND Planning

This section introduces a third type of FOND problem, i.e., Shielded FOND (S-FOND) problem, and provides different techniques to solve this class of problems. A S-FOND problem is formalized as a FOND problem plus a PPLTL formula φ which is called *shield*.

Definition 4. *A S-FOND problem is a tuple $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$, where $\mathcal{D}$ is a FOND domain model, G is a reachability goal, and φ is a* PPLTL *formula.*

Intuitively, the shield requires that, at every step, the trace generated while executing a strategy satisfies φ. Therefore, S-FOND planning is the task of finding a strategy that achieves the goal, and such that every prefix of every generated trace satisfies φ. Formally, a solution to a S-FOND problem is defined as follows.

Definition 5 (Shield-compliant strategy)**.** *Let $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ be a S-FOND planning problem. A strategy $\pi : (2^{\mathcal{F}})^+ \to A$ is a shield-compliant strong (strong-cyclic, resp.) solution for Γ iff*

1. *Every (stochastic-fair, resp.) trace τ induced by π is finite and such that $\mathsf{last}(\tau) \models G$;*
2. *Every prefix τ' of a trace τ induced by π is such that $\tau' \models \varphi$.*

In addition, we say that a trace τ is shield-compliant if every prefix τ' of τ satisfies φ. It is easy to see that the shield can impose any LTL_f safety constraints on the generated traces. Indeed, a safety property can be expressed by imposing that a PPLTL formula holds in every prefix of a trace [43]. Thus shields, as defined by us, can capture any LTL safety property over finite traces.

We now present three different encoding techniques to handle any S-FOND problem. The first approach works by reformulating a S-FOND problem as a FOND problem for temporally extended goals, while the other two techniques work by encoding a S-FOND problem directly into a FOND problem for reachability goals.

4.1 Handling Shields as Temporally Extended Goals

Intuitively, we can view a shield φ as a temporally extended goal φ' that enforces φ to hold in every prefix of a generated sequence of states. Specifically, we can do so with the H temporal operator.

Theorem 3. *Let $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ be a S-FOND problem and let $\Gamma' = \langle \mathcal{D}, G \wedge \mathsf{H}(\varphi) \rangle$ be a FOND problem for temporally extended goals. Then a strategy π is a shield-compliant strong (strong-cyclic) solution for Γ iff π is a strong (strong-cyclic, resp.) solution for Γ'.*

Proof. Given a trace $\tau = s_0, s_1, \dots s_n$, by definition of PPLTL we have that $\tau \models G \wedge \mathsf{H}(\varphi)$ iff $\mathsf{last}(\tau) \models G \wedge \forall k$ with $0 \leq k \leq n \cdot \tau, k \models \varphi$. The condition $\forall k$ with $0 \leq k \leq n \cdot \tau, k \models \varphi$ is satisfied iff every prefix $\tau' = s_0, \dots, s_k$ of τ is such that $\tau' \models \varphi$. Given these equivalences, it is easy to see that every strategy π satisfies the goal of Γ' iff π is a shield-compliant strategy for Γ. □

A direct consequence of Theorem 3 is that we can use any standard technique for handling FOND planning for PPLTL goals to find a shield-compliant strategy. The approach we propose, called Π^{TEG}, first encodes a S-FOND problem Γ into a FOND problem for temporally extended goals Γ', and then further encodes Γ' into FOND planning for reachability goals.

Definition 6 (Π^{TEG})**.** *We define Π^{TEG} as the encoding that transforms a S-FOND problem $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ into a FOND problem for reachability goals Γ'' as described in the following steps:*

1. *Encode $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ into $\Gamma' = \langle \mathcal{D}, G \wedge \mathsf{H}(\varphi) \rangle$.*
2. *Encode $G \wedge \mathsf{H}(\varphi)$ into $\mathcal{D}$ as by Definition 3 to obtain $\mathcal{D}_{G \wedge \mathsf{H}(\varphi)}$.*
3. *$\Gamma'' = \langle \mathcal{D}_{G \wedge \mathsf{H}(\varphi)}, \mathsf{val}_{G \wedge \mathsf{H}(\varphi)} \rangle$.*

Theorem 4. *Let Γ be a S-FOND problem and let Γ'' be the problem obtained by encoding Γ with Π^{TEG}. Then Γ admits a shield-compliant (strong or strong-cyclic) strategy iff Γ'' admits a winning (strong or strong-cyclic, resp.) strategy.*

Proof. Directly from Theorem 3 and Corollary 2. □

Encoding $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ into $\Gamma'' = \langle \mathcal{D}_{G \wedge \mathsf{H}(\varphi)}, \mathsf{val}_{\mathsf{H}(\varphi) \wedge G} \rangle$ via Π^{TEG} enables using any standard FOND planner for reachability goals to find a shield-compliant strategy for Γ.

We conclude with some considerations on the complexity of solving Γ'' in terms of reachable states of the domain $\mathcal{D}_{G \wedge \mathsf{H}(\varphi)}$. Observe that a trace τ of original domain $\mathcal{D}$ is shield compliant iff the corresponding trace $\tau' = (s_0, \sigma_0), \dots, (s_n, \sigma_n)$ of $\mathcal{D}_{G \wedge \mathsf{H}(\varphi)}$ is such that, for every $i \leq n$, $(s_i, \sigma_i) \models \mathsf{val}_\varphi$. If this is the case, then τ' also satisfies $\mathsf{val}_{\mathsf{H}(\varphi)}$, a necessary condition to satisfy the goal of Γ''. However, the precondition function of $\mathcal{D}_{G \wedge \mathsf{H}(\varphi)}$ is exactly the same as the original domain $\mathcal{D}$. Hence, a reachable state (s_i, σ_i) of $\mathcal{D}_{G \wedge \mathsf{H}(\varphi)}$ is not guaranteed to satisfy val_φ, and also every trace originating from such a state will never satisfy $\mathsf{val}_{\mathsf{H}(\varphi)}$. Consequently, $\mathcal{D}_{G \wedge \mathsf{H}(\varphi)}$ features many reachable dead-end states from which the shield will never be re-established.

4.2 Violation-Detection Technique

In this section, we study an approach that greatly reduces the reachable states violating val_φ. Specifically, we devise an encoding that prevents the execution of further actions when val_φ becomes falsified for the first time. Let $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ be a S-FOND problem. We start by encoding the shield φ into the domain $\mathcal{D}$, thus obtaining $\mathcal{D}_\varphi$ as by Definition 3. Then, we modify the preconditions of $\mathcal{D}_\varphi$ to block the execution of further actions when the shield becomes violated. We do so by making val_φ a precondition of every action. Remark that val_φ captures the truth of the shield φ in every state of a generated trace. To obtain a correct approach we also need to make val_φ a goal of Γ''. Otherwise, we could end the execution in a state where the last action of the strategy has violated the shield. We called this approach Violation-Detection Technique (VDT) as it blocks the executions whenever it detects that the shield has been (irreversibly) violated.

Definition 7 (Π^{VDT})**.** *We define Π^{VDT} as the encoding approach that takes in input an S-FOND problem $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ with $\mathcal{D} = \langle \mathcal{F}, \mathcal{X}, A, Pre, Eff, s_0 \rangle$ and outputs a FOND problem $\Gamma'' = \langle \mathcal{D}'', G'' \rangle$ with $\mathcal{D}'' = \langle \mathcal{F}', \mathcal{X}', A, Eff', Pre'', s_0' \rangle$ where:*

- *$\mathcal{F}'$, $\mathcal{X}'$, s_0', and Eff' are defined as in Definition 3.*
- *$Pre''(a) = Pre(a) \wedge \mathsf{val}_\varphi$*
- *$G'' = G \wedge \mathsf{val}_\varphi$*

Theorem 5. *Let Γ be a S-FOND problem and let Γ'' be the FOND problem obtained by Π^{VDT}. Then Γ admits a shield-compliant (strong or strong-cyclic) solution iff Γ'' admits a winning (strong or strong-cyclic, resp.) strategy.*

Proof sketch. ($\Rightarrow$). Let $\pi : (2^{\mathcal{F}})^+ \to A$ be a shield-compliant strategy for Γ. Observe that the initial state, effects, fluents, and axioms of D'' are defined as in Definition 3. Hence, by Theorem 1, every potential trace τ of $\mathcal{D}$ can be uniquely mapped to a potential trace τ'' of $\mathcal{D}''$. Therefore, we can build a partial function $\pi'' : (2^{\mathcal{F}'})^+ \to A$ as follows:

$$\begin{array}{lll} \pi''(\tau'') = a & \text{iff} & \pi(\tau) = a \\ \pi''(\tau'') \text{ is undefined} & \text{iff} & \pi(\tau) \text{ is undefined.} \end{array}$$

We need to prove that π'' is a strategy and is winning. By contradiction, π'' is not a strategy because some action returned by π'' cannot be executed. Suppose that, for some trace τ'', $\pi(\tau'') = a$ and $\mathsf{last}(\tau'') \not\models Pre''(a)$. Then, either $\mathsf{last}(\tau'') \not\models Pre(a)$ or $\mathsf{last}(\tau'') \not\models \mathsf{val}_\varphi$. The first case implies $\mathsf{last}(\tau) \not\models Pre(a)$, while the second case implies $\tau, \mathsf{last}(\tau) \not\models \varphi$. Both cases imply that π is not a shield-compliant strategy (contradiction). Therefore, π'' is a strategy. Moreover, since π reaches the goal G while not violating φ, then also π'' reaches the goal $G \wedge \mathsf{val}_\varphi$. Thus, π'' is winning.
($\Leftarrow$) Let $\pi'' : (2^{\mathcal{F}'})^+ \to A$ be a winning strategy for Γ''. Similarly to the previous case, we can build a partial function $\pi : (2^{\mathcal{F}})^+ \to A$ as follows:

$$\begin{array}{lll} \pi(\tau) = a & \text{iff} & \pi''(\tau'') = a \\ \pi(\tau) \text{ is undefined} & \text{iff} & \pi''(\tau'') \text{ is undefined.} \end{array}$$

It is easy to see that π is a strategy and is shield-compliant; for every $a \in A$, we have that $Pre''(a)$ subsumes $Pre(a)$. Moreover, for every finite trace $\tau'' = (s_0, \sigma_0), \dots, (s_n, \sigma_n)$ induced by π'' we have $(s_i, \sigma_i) \models \mathsf{val}_\varphi$ for every $i \leq n$ and $s_n \models G$. Therefore, the corresponding finite trace τ also reaches the goal and is shield-compliant. □

Example 4. *Consider $\varphi = a \,\mathsf{S}\, b$. This shield requires $a \,\mathsf{S}\, b$ to hold in every prefix of a generated trace. That is, b has to hold in the initial state, and every subsequent state must satisfy either a or b. The encoded problem Γ'' obtained by Π^{VDT} is $\Gamma'' = \langle \langle \mathcal{F}', \mathcal{X}', A, Eff', Pre'', s_0' \rangle, G'' \rangle$ where $\mathcal{F}'$, $\mathcal{X}'$ and Eff' are defined as in Examples 1-3, $s_0' = (s_0, \sigma_0)$, $Pre''(a) = Pre'(a) \wedge \mathsf{val}_{a \,\mathsf{S}\, b}$ for every $a \in A$, and $G'' = G \wedge \mathsf{val}_{a \,\mathsf{S}\, b}$.*

By analyzing the encoding, it is easy to see that a reachable state (s, σ) of $\mathcal{D}''$ may not satisfy val_φ. However, in every trace $\tau = (s_0, \sigma_0), \ldots, (s_i, \sigma_i)$ of $\mathcal{D}''$, only the last state (s_i, σ_i) may violate val_φ. In terms of the number of reachable states, Π^{VDT} greatly improves over Π^{TEG}. Let $\mathcal{D}_{\mathsf{TEG}}$ and $\mathcal{D}_{\mathsf{VDT}}$ the domains of the problems encoded with Π^{TEG} and Π^{VDT}, respectively. By definition, $\mathcal{D}_{\mathsf{TEG}}$ contains the same fluents of $\mathcal{D}_{\mathsf{VDT}}$ (those to monitor φ) plus one to monitor $\mathsf{H}(\varphi)$. Also, observe that if a state s of $\mathcal{D}_{\mathsf{VDT}}$ is reachable, then there exists at least one reachable state s' of $\mathcal{D}_{\mathsf{TEG}}$ such that $s \subseteq s'$. This is because the fluents shared by $\mathcal{D}_{\mathsf{TEG}}$ and $\mathcal{D}_{\mathsf{VDT}}$ are updated by the same set of conditional effects and the precondition function of $\mathcal{D}_{\mathsf{VDT}}$ subsumes that of $\mathcal{D}_{\mathsf{TEG}}$. On the contrary, it is easy to see that if a state s' of $\mathcal{D}_{\mathsf{TEG}}$ is reachable, then the subset s of s' representing a state for $\mathcal{D}_{\mathsf{VDT}}$ may not be reachable, as the preconditions of $\mathcal{D}_{\mathsf{VDT}}$ are more restrictive than those of $\mathcal{D}_{\mathsf{TEG}}$. For example, let $\tau' = s'_0, s'_1, s'_2$ be a trace of $\mathcal{D}_{\mathsf{TEG}}$ such that both s'_1 and s'_2 do not satisfy val_φ. Then, the state s_1 of $\mathcal{D}_{\mathsf{VDT}}$ obtained by s'_1 is reachable an such that $s_1 \not\models \mathsf{val}_\varphi$, but the state s_2 obtained by s'_2 is not, as no action of $\mathcal{D}_{\mathsf{VDT}}$ can be executed after s_1. Therefore, $\mathcal{D}_{\mathsf{VDT}}$ admits fewer reachable states than $\mathcal{D}_{\mathsf{TEG}}$.

4.3 Regression Technique

Π^{VDT} greatly reduces the number of reachable states violating the shield while adding negligible overhead to the encoded problem. However, we can further improve the encoding by blocking the execution of actions that could violate the shield. Not only does this optimization further reduce the number of reachable states, but it also becomes necessary in online planning scenarios where we cannot correct the behavior of an agent when the shield has been violated. Therefore, we propose a different encoding that uses the notion of (effect) regression by [40] as adapted to PDDL by [49].

Definition 8 (Effect Regression)**.** *The regression $R(\phi, \mathsf{eff})$ of a NNF formula ϕ through a set of conditional effects eff is the formula obtained from ϕ by replacing every fluent f in ϕ with $\Gamma_f(\mathsf{eff}) \vee (f \wedge \neg\Gamma_{\neg f}(\mathsf{eff}))$, where $\Gamma_l(\mathsf{eff})$ for a literal l is defined as $\Gamma_l(\mathsf{eff}) = \bigvee_{\substack{c \rhd e \in \mathsf{eff} \\ \text{with } l \in e}} c$.*

Example 5. *Consider the literal a and the set of conditional effects $\mathsf{eff} = \{c \rhd \{\neg a\}\}$. We have that $\Gamma_a(\mathsf{eff}) = \bot$, while $\Gamma_{\neg a} = c$. Hence, $R(a, \mathsf{eff}) = a \wedge \neg c$.*

Notably, the regression of a propositional formula ϕ through the outcome $\mathsf{eff} \in \mathit{Eff}(a)$ of an action a is a formula $R(\phi, \mathsf{eff})$ such that for every state s, $s \models R(\phi, \mathsf{eff})$ iff $s[\mathsf{eff}] \models \phi$ [49].

This encoding uses the effect regression to extend the precondition of actions to prune invalid extensions of a trace prefix while the search process goes on. In particular, given a state (s, σ) and an outcome eff of an action a, we want to predict if val_φ becomes false in the successor state $(s, \sigma)[\mathsf{eff}]$ obtained by applying eff in (s, σ). Similarly to Π^{VDT}, we take $\mathcal{D}_\varphi$ as a starting point. Then, starting from the set of axioms $\mathcal{X}_\varphi$ of $\mathcal{D}_\varphi$ that determine the truth of val_φ, we define a new set of axioms $\mathcal{X}^{\mathsf{eff}}_\varphi$. These axioms have the form $\mathsf{val}^{\mathsf{eff}}_\phi \leftarrow \psi'$, and are intended to be such that $(s, \sigma) \models \mathsf{val}^{\mathsf{eff}}_\phi$ iff $(s, \sigma)[\mathsf{eff}] \models \mathsf{val}_\phi$ for every $\phi \in \mathsf{sub}(\varphi)$. Intuitively, these new axioms are structured as the original ones, but with the difference that every fluent mentioned in the axiom condition ψ' is regressed through eff. Therefore, to determine ψ', we have that:

- Every fluent p of the original domain $\mathcal{D}$ gets replaced by $R(p, \mathsf{eff})$.
- Every fluent of the form "$\mathsf{Y}(\phi)$" gets replaced by val_ϕ. In this case, we rely on the semantics of PPLTL formulas to efficiently perform regression. By definition, $\mathsf{Y}(\phi)$ is true in the next state iff ϕ is true in the current state. Hence, "$\mathsf{Y}(\phi)$" is true in the next state iff val_ϕ holds in the current state.

Exploiting these observations, given a set of axioms $\mathcal{X}_\varphi = \{\mathsf{val}_\phi \leftarrow \psi \mid \phi \in \mathsf{sub}(\varphi)\}$ and a set eff of conditional effects of some action from $\mathcal{D}_\varphi$, we define the new set of axioms $\mathcal{X}^{\mathsf{eff}}_\varphi = \{\mathsf{val}^{\mathsf{eff}}_\phi \leftarrow \psi' \mid \phi \in \mathsf{sub}(\varphi)\}$ where every $\mathsf{val}^{\mathsf{eff}}_\phi \leftarrow \psi'$ is defined depending on ϕ by the following rules:

- $\mathsf{val}^{\mathsf{eff}}_p \leftarrow R(p, \mathsf{eff})$
- $\mathsf{val}^{\mathsf{eff}}_{\mathsf{Y}\phi} \leftarrow \mathsf{val}_\phi$
- $\mathsf{val}^{\mathsf{eff}}_{\phi_1 \,\mathsf{S}\, \phi_2} \leftarrow (\mathsf{val}^{\mathsf{eff}}_{\phi_2} \vee (\mathsf{val}^{\mathsf{eff}}_{\phi_1} \wedge \mathsf{val}_{\phi_1 \,\mathsf{S}\, \phi_2}))$
- $\mathsf{val}^{\mathsf{eff}}_{\phi_1 \wedge \phi_2} \leftarrow (\mathsf{val}^{\mathsf{eff}}_{\phi_1} \wedge \mathsf{val}^{\mathsf{eff}}_{\phi_2})$
- $\mathsf{val}^{\mathsf{eff}}_{\neg\phi} \leftarrow \neg\mathsf{val}^{\mathsf{eff}}_\phi$

Following the definition of these new axioms, for any formula $\phi \in \mathsf{sub}(\varphi)$, and for any outcome $\mathsf{eff} \in \mathit{Eff}'(a)$ of some action a, where $\mathit{Eff}'(a)$ is defined as in Definition 3, we have that $(s, \sigma) \models \mathsf{val}^{\mathsf{eff}}_\phi$ iff $(s, \sigma)[\mathsf{eff}] \models \mathsf{val}_\phi$.

Example 6. *Consider the shield $\varphi = a \,\mathsf{S}\, b$ and the set of conditional effects $\mathsf{eff} = \{c \rhd \{\neg a\}, \mathsf{val}_{a \,\mathsf{S}\, b} \rhd \{\text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"}\}, \neg\mathsf{val}_{a \,\mathsf{S}\, b} \rhd \{\neg\text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"}\}\}$. Recall from Example 2 that $\mathcal{X}_\varphi = \{\mathsf{val}_a \leftarrow a, \mathsf{val}_b \leftarrow b, \mathsf{val}_{(a \,\mathsf{S}\, b)} \leftarrow \mathsf{val}_b \vee (\mathsf{val}_a \wedge \text{"}\mathsf{Y}(a \,\mathsf{S}\, b)\text{"}\}$. We have that $R(a, \mathsf{eff}) = a \wedge \neg c$ and $R(b, \mathsf{eff}) = b$. Therefore, the new set of axioms is $\mathcal{X}^{\mathsf{eff}}_\varphi = \{\mathsf{val}^{\mathsf{eff}}_a \leftarrow a \wedge \neg c, \mathsf{val}^{\mathsf{eff}}_b \leftarrow b, \mathsf{val}^{\mathsf{eff}}_{(a \,\mathsf{S}\, b)} \leftarrow \mathsf{val}^{\mathsf{eff}}_b \vee (\mathsf{val}^{\mathsf{eff}}_a \wedge \mathsf{val}_{(a \,\mathsf{S}\, b)})\}$*

Lemma 6. *Let Eff' be the effect function of a FOND domain $\mathcal{D}_\varphi$ obtained by encoding φ into a FOND domain $\mathcal{D}$ following Definition 3, let $\mathsf{eff} \in \mathit{Eff}'(a)$ be a set of conditional effects of some action a of $\mathcal{D}_\varphi$ and let (s, σ) be a state from $\mathcal{D}_\varphi$. Then, using the set of axioms $\mathcal{X}^{\mathsf{eff}}_\varphi$, for every $\phi \in \mathsf{sub}(\varphi)$, it holds that $(s, \sigma) \models \mathsf{val}^{\mathsf{eff}}_\phi$ iff $(s, \sigma)[\mathsf{eff}] \models \mathsf{val}_\phi$.*

Proof sketch. By induction on the structure of the formula φ and by definition of regression. □

Consider an action a with effects $\mathit{Eff}'(a) = \{\mathsf{eff}_0, \ldots, \mathsf{eff}_n\}$. The axiom $\mathsf{val}^{\mathsf{eff}_i}_\varphi \in \mathcal{X}^{\mathsf{eff}_i}_\varphi$ captures whether or not the outcome eff_i will violate φ in the successor state. To prevent φ from being violated by *any* nondeterministic outcome, we set $\mathsf{val}^{\mathsf{eff}_i}_\varphi$ as a precondition of a for every outcome $\mathsf{eff}_i \in \mathit{Eff}'(a)$.

Definition 9 (Π^{RE})**.** *We define Π^{RE} as the encoding approach that takes in input a S-FOND problem $\Gamma = \langle \mathcal{D}, G, \varphi \rangle$ with $\mathcal{D} = \langle \mathcal{F}, \mathcal{X}, A, \mathit{Pre}, \mathit{Eff}, s_0 \rangle$ and outputs a FOND problem $\Gamma'' = \langle \mathcal{D}'', G \rangle$ with $\mathcal{D}'' = \langle \mathcal{F}', \mathcal{X}'', A, \mathit{Eff}', \mathit{Pre}'', s'_0 \rangle$ where:*

- $\mathcal{X}'' = \mathcal{X}' \cup \bigcup_{a \in A} \bigcup_{\mathsf{eff} \in \mathit{Eff}'(a)} \mathcal{X}^{\mathsf{eff}}_\varphi$
- $\mathit{Pre}''(a) = \mathit{Pre}(a) \wedge \bigwedge_{\mathsf{eff} \in \mathit{Eff}'(a)} \mathsf{val}^{\mathsf{eff}}_\varphi$, *for every $a \in A$.*
- *$\mathcal{F}'$, $\mathcal{X}'$, Eff' and s'_0 are defined as in Definition 3.*

Moreover, if $s_0 \not\models \mathsf{val}_\varphi$, then Π^{RE} detects that Γ admits no solution.

Theorem 7. *Let Γ be a S-FOND problem and let Γ'' be the FOND problem obtained by Π^{RE}. Then Γ admits a shield-compliant (strong or strong-cyclic) solution iff Γ'' admits a winning (strong or strong-cyclic, resp.) strategy.*

Proof sketch. ($\Rightarrow$). Let $\pi : (2^{\mathcal{F}})^+ \to A$ be a shield-compliant strategy for Γ. Similarly to the proof of Theorem 5, Theorem 1 implies that every potential trace τ of $\mathcal{D}$ can be uniquely mapped to a potential trace τ'' of $\mathcal{D}''$. Therefore, we can build a partial function $\pi'' : (2^{\mathcal{F}'})^+ \to A$ defined as $\pi''(\tau'') = a$ iff $\pi(\tau) = a$ and $\pi''(\tau'')$ is undefined iff $\pi(\tau)$ is undefined. By contradiction, π'' is not a strategy because some action returned by π'' cannot be executed. Suppose that, for some trace τ'', $\pi''(\tau'') = a$ and $\mathsf{last}(\tau'') \not\models Pre''(a)$. If $\mathsf{last}(\tau'') \not\models Pre(a)$, then also $\pi(\tau) \not\models Pre(a)$, and therefore π is not a strategy (contradiction). Therefore, there exists a derived predicate $\mathsf{val}^{\mathsf{eff}}_{\varphi}$ in $Pre''(a)$ such that $\mathsf{last}(\tau'') \not\models \mathsf{val}^{\mathsf{eff}}_{\varphi}$. This implies that π'' can induce the trace $\tau'' \cdot \mathsf{last}(\tau'')[\mathsf{eff}]$ where $\mathsf{last}(\tau'')[\mathsf{eff}] \not\models \mathsf{val}_{\varphi}$ (Lemma 6). Therefore, the corresponding trace $\tau \cdot \mathsf{last}(\tau)[\mathsf{eff}]$ with $\mathsf{eff} \in \mathit{Eff}(a)$ can be induced by π, and is such that $\tau \cdot \mathsf{last}(\tau)[\mathsf{eff}] \not\models \varphi$ (Theorem 1). Therefore, π'' is a strategy. Moreover, since every trace τ induced by π achieves G, then so does every trace τ'' induced by π''. Hence π'' is winning.
($\Leftarrow$). Let $\pi'' : (2^{\mathcal{F}'})^+ \to A$ be a winning strategy for Γ''. By Theorem 1, we can build a partial function $\pi : (2^{\mathcal{F}})^+ \to A$ defined as $\pi(\tau) = a$ iff $\pi''(\tau'') = a$ and $\pi(\tau)$ is undefined iff $\pi''(\tau'')$ is undefined. It is easy to see that π is a strategy; for every $a \in A$, we have that $Pre''(a)$ subsumes $Pre(a)$. Moreover, (1) every trace induced by π is shield-compliant, as all states induced by π'' satisfy val_{φ} by Lemma 6, and (2) since every trace τ'' induced by π'' achieves G, then so does every trace τ induced by π. Hence π is also shield-compliant.

□

We conclude by relating Π^{VDT} and Π^{RE}. Let $\mathcal{D}_{\mathsf{VDT}}$ and $\mathcal{D}_{\mathsf{RE}}$ be the domains resulting from problems compiled by Π^{VDT} and Π^{RE}, respectively. By analyzing the encodings, $\mathcal{D}_{\mathsf{VDT}}$ and $\mathcal{D}_{\mathsf{RE}}$ share the same fluents. Moreover, it is easy to see that Lemma 6 plus Definition 9 implies that no outcome of any action from $\mathcal{D}_{\mathsf{RE}}$ can induce a state (s_i, σ_i) that violates val_{φ}. On the contrary, $\mathcal{D}_{\mathsf{VDT}}$ admits reachable states that violate val_{φ}. Therefore, if a state (s_i, σ_i) is reachable for $\mathcal{D}_{\mathsf{RE}}$ then so does for $\mathcal{D}_{\mathsf{VDT}}$, but not vice versa.

5 Experiments

Our experiments aim to understand Π^{TEG}, Π^{VDT}, and Π^{RE} practically over a set of S-FOND benchmarks with different shield specifications. For Π^{TEG}, we simply reformulate the S-FOND problems as FOND problems for PPLTL goals (following Theorem 3) and then encode such problems with the Plan4Past tool [15]. Instead, Π^{VDT} and Π^{RE} have been implemented in Python and are available at [16].

Theoretically, the advantage of one encoding over the others seems clear. In terms of the number of reachable states, Π^{TEG} is dominated by Π^{VDT}, which in turn is dominated by Π^{RE}. However, the practical effectiveness of these encodings is not clear. Therefore, we tested Π^{TEG}, Π^{VDT}, and Π^{RE} over a set of S-FOND problems with different shields. As a planner, we considered myND [44], which is a state-of-the-art FOND planner supporting conditional effects and derived predicates, which are necessary for our evaluation. We configured myND to search for strong-cyclic solutions with a LAO* [36] search guided by the h^{FF} [38] heuristic. We compared all encodings in terms of coverage (number of solved instances), the total time required to find a solution, the number of nodes expanded by the planner during the search, and the size of the solution policies. We ran all experiments on a Xeon-Gold 6140M 2.3 GHz with 8GB of memory and a time limit of 900s.

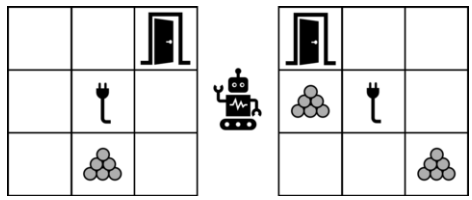

Figure 1. A CLEANING instance with two rooms and three dirty tiles.

5.1 Benchmarks

We designed a new FOND domain called CLEANING to test the shielding techniques. We generated 6 instances of increasing dimensions, and for each instance, we defined four different classes of shields that we call *Simple*, *Precendence*, *Precondition*, and *Interval*. In addition, to demonstrate that our approaches can generalize to different domains, we also considered the Robot-coffee (ROBOT) and Blocksworld (BLOCKS) FOND domains. For BLOCKS, we have 8 instances and shields of types *Simple* and *Precendence*, while for ROBOT we have 8 instances and shields of type *Precondition* and *Interval*. In total, our benchmark features 56 instances: 24 for CLEANING, 16 for BLOCKS, and 16 for ROBOT. Domains and shields are described in the following paragraphs and are available at [16].

Cleaning. In this domain, a robot can move between rooms to clean dirty floor tiles. Each room is encoded as a 3x3 grid and has an entry point. Moreover, the robot has a battery that can be charged on certain floor tiles. The cleaning action is non-deterministic and may leave the floor dirty. However, this action consumes a level of battery independently from the outcome. An example of an instance is shown in Figure 1. The shield of type *Simple* specifies that the robot must preserve at least two levels of battery with the formula: $\neg \mathit{charge}(l_1)$. *Precedence* shields specify an order of achievement of the problem's subgoals. In our instances, the goal is to achieve $\mathit{clean}(r_x, t_x)$ for every dirty tile t_x inside room r_x. To specify that $\mathit{clean}(r_y, t_y)$ should be achieved (strictly) before $\mathit{clean}(r_x, t_x)$, we use the shield:

$$\mathit{clean}(r_x, t_x) \rightarrow \mathsf{Y}(\mathsf{O}(\mathit{clean}(r_y, t_y)))$$

We then repeat the same shield for every pair of subgoals to force a total order of achievement of the subgoals. Shields of type *Precondition* require that an event can occur only if a certain condition is met. In our case, we want to enforce the property "The robot cannot go outside until the room has been completely cleaned". Let t_x be the access point of room r_x. We can do so with the formula:

$$(\mathit{outside} \wedge \mathsf{Y}(\mathit{pos}(r_x, t_x))) \rightarrow \bigwedge_{\{t | \mathit{dirty}(r_x, t)\}} \mathit{clean}(r_x, t).$$

Essentially, $\mathit{outside} \wedge \mathsf{Y}(\mathit{pos}(r_x, t_x))$ becomes true only when the robot leaves room r_x, while $\bigwedge_{\{t | \mathit{dirty}(r_x, t)\}} \mathit{clean}(r_x, t)$ becomes true when all dirty cells of r_x become cleaned. Therefore, the shield specifies that the robot cannot leave a room before cleaning all dirty tiles. Lastly, *Interval* is a shield requiring that there cannot be more than k consecutive states that do not satisfy a certain condition. In our case, we want the robot to be fully charged at least once every four states. This property is specified by the formula:

$$\mathit{charge}(l_4) \vee \mathsf{Y}\,\mathit{charge}(l_4) \vee \mathsf{YY}\,\mathit{charge}(l_4) \vee \mathsf{YYY}\,\mathit{charge}(l_4)$$

Blocksworld. The goal of BLOCKS is to arrange blocks in a particular configuration. In this version of the domain, an arm can pick up either one or two blocks at the same time. In our instances, all blocks

Table 1. Coverage, average runtime (Time), average expanded nodes (Expanded), and average size of solution strategies (Policy) achieved by all encodings across all domains. The best performers are in bold.

Domain	Shield	Coverage			Time			Expanded			Policy		
		Π^{TEG}	Π^{VDT}	Π^{RE}	Π^{TEG}	Π^{VDT}	Π^{RE}	Π^{TEG}	Π^{VDT}	Π^{RE}	Π^{TEG}	Π^{VDT}	Π^{RE}
CLEANING	*Simple*	**6**	**6**	5	100.64	**23.51**	105.68	3407.40	746.20	**560.00**	76.20	**70.40**	71.60
	Precedence	4	**6**	**6**	52.19	**14.25**	36.83	2851.00	461.75	**359.25**	84.75	**81.25**	83.00
	Precondition	4	**6**	4	93.42	**19.10**	156.09	2953.50	673.00	**643.00**	95.00	**88.25**	**88.25**
	Interval	4	**6**	**6**	234.62	**13.32**	82.17	6228.00	899.50	**667.50**	60.75	**60.50**	76.50
BLOCKS	*Simple*	2	**8**	**8**	3.26	**2.00**	24.49	298.00	**6.00**	**6.00**	**6.00**	**6.00**	**6.00**
	Precedence	0	4	**5**	–	**4.08**	48.46	–	11.50	**11.25**	–	11.50	**11.25**
ROBOT	*Precondition*	5	**8**	6	43.27	**6.91**	88.57	3416.20	**662.60**	697.60	**61.00**	**61.00**	**61.00**
	Interval	6	**8**	7	39.21	**9.17**	60.71	2667.83	1054.83	**681.50**	**62.83**	63.50	63.17
	Total	31	**52**	47									

are initially stacked to form a single tower, and the objective is to put all blocks on the table. In this domain, we have *Simple* and *Precedence* shields. With *Simple* we prevent the planner from picking up a tower of two blocks with the formula $\neg \mathit{holding\text{-}two}$. Instead, *Precedence* specifies that block b_{i+1} must be placed on the table before block b_i. For example, with four blocks b_1, b_2, b_3, b_4 the shield is:

$$(\mathit{table}(b_1) \rightarrow \mathsf{YO}(\mathit{table}(b_2))) \wedge (\mathit{table}(b_3) \rightarrow \mathsf{YO}(\mathit{table}(b_4)))$$

Robot-Coffee. A robot has to prepare coffee in a kitchen and deliver it to different offices. The robot can move between offices, and could non-deterministically spill the coffee while making a delivery. In this domain, we have *Precondition* and *Interval* shields. In particular, *Precondition* requires that the robot may enter office o_x only if the coffee has not yet been delivered to o_x. Specifically, the shield is:

$$(\mathit{at}(o_x) \wedge \mathsf{Y}(\neg \mathit{at}(o_x))) \rightarrow \neg \mathit{coffee}(o_x)$$

Instead, *Interval* specifies that the robot must hold a mug of coffee at least every four states with the formula:

$$\mathit{has\text{-}coffee} \vee \mathsf{Y}\mathit{has\text{-}coffee} \vee \mathsf{YY}\mathit{has\text{-}coffee} \vee \mathsf{YYY}\mathit{has\text{-}coffee}.$$

5.2 Results

Table 1 reports the overall results for all encodings. Π^{VDT} achieves the highest overall coverage, followed by Π^{RE} and by Π^{TEG}. The comparison of Π^{VDT} versus Π^{TEG} confirms our expectations; Π^{VDT} yields encoded problems with fewer reachable states than the problem encoded by Π^{TEG}, and as a consequence, myND has to do much less search when combined with Π^{VDT}. This behavior can be observed by looking a the number of expanded nodes: on average, Π^{VDT} expands from one to two orders of magnitude fewer nodes than Π^{TEG}. Figure 3 (left) reports the instance-by-instance comparison between Π^{VDT} and Π^{TEG} in terms of expanded nodes. We observe that every instance solved by Π^{TEG} is also solved by Π^{VDT} and that Π^{VDT} expands fewer nodes in all but 1 instance.

The comparison between Π^{VDT} and Π^{RE} shows a different picture. On average, Π^{RE} expands fewer nodes than Π^{VDT}, but Π^{VDT} is both faster and achieves a higher coverage than Π^{RE}. Looking at the raw output logs of myND, across all instances, we observed that on average Π^{RE} introduces up to 17332 axioms. As a reference, across all instances, Π^{VDT} adds on average 41 axioms. Clearly, Π^{RE} significantly increases both the preprocessing and search times of myND. Looking at the pairwise comparison reported in Figure 3 (right), we observe that the number of nodes expanded by Π^{RE} and Π^{VDT} is very similar. Still, myND is much faster with Π^{VDT}. This indicates that

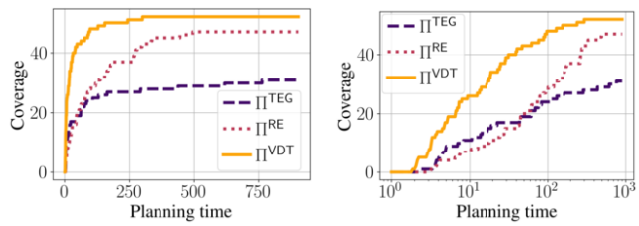

Figure 2. Survival plot in linear (lhs) and logarithmic (rhs) scale.

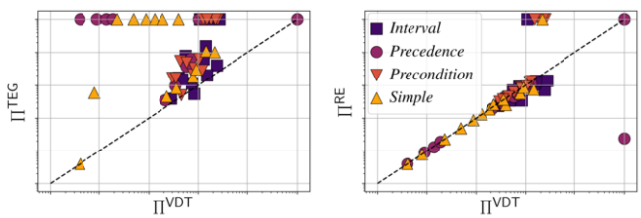

Figure 3. Instance-by-instance comparison of Π^{VDT} (x-axis) versus Π^{TEG} (left, y-axis) and Π^{RE} (right, y-axis) in terms of number of expanded nodes.

myND has to spend time evaluating the truth of the axioms resulting from Π^{RE}, and overall takes more time to expand a node. In terms of the average size of solution policies, we can see that all systems perform about the same.

Figure 2 reports how each system increases its cover over time. We can see that Π^{VDT} dominates the other encodings right from the start. Instead, Π^{TEG} is initially faster than Π^{RE}, which manages to catch up and surpass Π^{TEG} after about 46 seconds. This indicates that Π^{TEG} might be preferable over Π^{RE} on small instances. However, compared to Π^{TEG}, Π^{RE} scales to much larger problems.

6 Conclusions

We studied the problem of FOND planning with PPLTL shields. We propose three techniques to encode S-FOND into FOND planning for reachability goals. The first encoding handles a shield as a temporally extended goal, the second encoding modifies the preconditions of actions to block the execution upon shield violation, while the last approach uses the notion of regression to prevent the violation of the shield. Our results show that it is much better to handle shields natively rather than using a technique designed for temporally extended goals. Future work includes investigating shield-aware heuristics and encoding approaches for shields interpreted over action executions.

Acknowledgements

This work has been partially supported by ERC Advanced Grant WhiteMech (No. 834228), PRIN project RIPER (No. 20203FFYLK), EU ICT-48 2020 project TAILOR (GA 952215), PNRR MUR project FAIR (No. PE0000013), and PNRR MUR project SERICS (PE00000014).

References

[1] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. In *AAAI*, pages 2669–2678. AAAI Press, 2018.

[2] B. Aminof, G. De Giacomo, and S. Rubin. Stochastic fairness and language-theoretic fairness in planning in nondeterministic domains. In *ICAPS*, pages 20–28. AAAI Press, 2020.

[3] F. Bacchus and F. Kabanza. Planning for temporally extended goals. In *AAAI*, pages 1215–1222. AAAI Press, 1996.

[4] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Ann. Math. Artif. Intell.*, 22(1-2):5–27, 1998.

[5] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *AIJ*, 116(1-2):123–191, 2000.

[6] F. Bacchus, C. Boutilier, and A. Grove. Rewarding behaviors. In *AAAI*, pages 1160–1167, 1996.

[7] F. Bacchus, C. Boutilier, and A. Grove. Structured solution methods for non-markovian decision processes. In *AAAI*, pages 112–117, 1997.

[8] J. A. Baier and S. A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *AAAI*, pages 788–795. AAAI, 2006.

[9] J. A. Baier and S. A. McIlraith. Planning with temporally extended goals using heuristic search. In *ICAPS*, pages 342–345. AAAI, 2006.

[10] H. Barringer, M. Fisher, D. M. Gabbay, G. Gough, and R. Owens. METATEM: A framework for programming in temporal logic. In *REX Workshop*, volume 430 of *LNCS*, pages 94–129. Springer, 1989.

[11] J. Benton, A. J. Coles, and A. Coles. Temporal planning with preferences and time-dependent continuous costs. In *ICAPS*. AAAI, 2012.

[12] L. Bonassi, A. E. Gerevini, F. Percassi, and E. Scala. On planning with qualitative state-trajectory constraints in PDDL3 by compiling them away. In *ICAPS*, pages 46–50. AAAI Press, 2021.

[13] L. Bonassi, A. E. Gerevini, and E. Scala. Planning with qualitative action-trajectory constraints in PDDL. In *IJCAI*, pages 4606–4613. ijcai.org, 2022.

[14] L. Bonassi, G. D. Giacomo, M. Favorito, F. Fuggitti, A. E. Gerevini, and E. Scala. Planning for temporally extended goals in pure-past linear temporal logic. In *ICAPS*, pages 61–69. AAAI Press, 2023.

[15] L. Bonassi, G. D. Giacomo, M. Favorito, F. Fuggitti, A. E. Gerevini, and E. Scala. FOND planning for pure-past linear temporal logic goals. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 279–286. IOS Press, 2023.

[16] L. Bonassi, G. De Giacomo, A. E. Gerevini, and E. Scala. Code and data for "Shielded FOND: Planning with safety constraints in pure-past linear temporal logic". https://github.com/LBonassi95/ppltl4shields, 2024.

[17] L. Bonassi, A. E. Gerevini, and E. Scala. Dealing with numeric and metric time constraints in PDDL3 via compilation to numeric planning. In *AAAI*, pages 20036–20043. AAAI Press, 2024.

[18] A. Camacho and S. A. McIlraith. Strong fully observable nondeterministic planning with ltl and ltl-f goals. In *IJCAI*, pages 5523–5531, 2019.

[19] A. Camacho, E. Triantafillou, C. J. Muise, J. A. Baier, and S. A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, pages 3716–3724. AAAI Press, 2017.

[20] A. Cecconi, G. D. Giacomo, C. D. Ciccio, F. M. Maggi, and J. Mendling. Measuring the interestingness of temporal logic behavioral specifications in process mining. *Inf. Syst.*, 107:101920, 2022.

[21] A. Cimatti, F. Giunchiglia, E. Giunchiglia, and P. Traverso. Planning via model checking: A decision procedure for *AR*. In *ECP*, pages 130–142. Springer, 1997.

[22] A. Cimatti, M. Roveri, and P. Traverso. Strong planning in non-deterministic domains via model checking. In *AIPS*, pages 36–43. AAAI, 1998.

[23] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *AIJ*, 147(1-2): 35–84, 2003.

[24] G. De Giacomo and S. Rubin. Automata-theoretic foundations of FOND planning for ltlf and ldlf goals. In *IJCAI*, pages 4729–4735, 2018.

[25] G. De Giacomo and M. Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *ECP*, pages 226–238. Springer, 1999.

[26] G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860. IJCAI/AAAI, 2013.

[27] G. De Giacomo, A. Di Stasio, F. Fuggitti, and S. Rubin. Pure-past linear temporal and dynamic logic on finite traces. In *IJCAI*, pages 4959–4965. ijcai.org, 2020.

[28] G. De Giacomo, A. Di Stasio, L. M. Tabajara, M. Y. Vardi, and S. Zhu. Finite-trace and generalized-reactivity specifications in temporal synthesis. In *IJCAI*, pages 1852–1858. ijcai.org, 2021.

[29] P. Doherty and J. Kvarnström. Temporal action logics. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 709–757. Elsevier, 2008.

[30] S. Edelkamp. On the compilation of plan constraints and preferences. In *ICAPS*, pages 374–377. AAAI, 2006.

[31] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Chapter 16*, 1990.

[32] D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *POPL*, pages 163–173. ACM Press, 1980.

[33] D. M. Gabbay, I. Hodkinson, and M. Reynolds. Temporal logic: mathematical foundations and computational aspects, 1994.

[34] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *AIJ*, 173(5-6): 619–668, 2009.

[35] F. Giunchiglia and P. Traverso. Planning as model checking. In *ECP*, pages 1–20. Springer, 1999.

[36] E. A. Hansen and S. Zilberstein. Lao*: A heuristic search algorithm that finds solutions with loops. *Artif. Intell.*, 129(1-2):35–62, 2001.

[37] J. Hoffmann and S. Edelkamp. The deterministic part of IPC-4: An overview. *JAIR*, 24:519–579, 2005.

[38] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253–302, 2001.

[39] C. Hsu, B. W. Wah, R. Huang, and Y. Chen. Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In *IJCAI*, pages 1924–1929, 2007.

[40] H. J. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Electron. Trans. Artif. Intell.*, 2:159–178, 1998.

[41] O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In *Logic of Programs*, pages 196–218. Springer, 1985.

[42] Z. Manna. *Verification of Sequential Programs: Temporal Axiomatization*, pages 53–102. Springer Netherlands, 1982.

[43] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *PODC*, pages 377–410. ACM, 1990.

[44] R. Mattmüller, M. Ortlieb, M. Helmert, and P. Bercher. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, pages 105–112. AAAI, 2010.

[45] A. Micheli and E. Scala. Temporal planning with temporal metric trajectory constraints. In *AAAI*, pages 7675–7682. AAAI Press, 2019.

[46] D. S. Nau, T. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: an HTN planning system. *J. Artif. Intell. Res.*, 20: 379–404, 2003.

[47] M. Pistore and P. Traverso. Planning as model checking for extended goals in non-deterministic domains. In *IJCAI*, pages 479–486. Morgan Kaufmann, 2001.

[48] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977.

[49] J. Rintanen. Regression for classical and nondeterministic planning. In *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 568–572. IOS Press, 2008.

[50] G. Röger, F. Pommerening, and M. Helmert. Optimal planning in the presence of conditional effects: Extending lm-cut with context splitting. In *ECAI*, pages 765–770, 2014.

[51] M. Steinmetz, J. Hoffmann, A. Kovtunova, and S. Borgwardt. Classical planning with avoid conditions. In *AAAI*, pages 9944–9952. AAAI Press, 2022.

[52] S. Thiébaux, C. Gretton, J. K. Slaney, D. Price, and F. Kabanza. Decision-theoretic planning with non-markovian rewards. *J. Artif. Intell. Res.*, 25:17–74, 2006.

[53] J. Torres and J. A. Baier. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *IJCAI*, pages 1696–1703. AAAI Press, 2015.