Planning for Temporally Extended Goals in Pure-Past Linear Temporal Logic (Extended Abstract)[*]

Luigi Bonassi[1] , **Giuseppe De Giacomo**[2,3] , **Marco Favorito**[4†] , **Francesco Fuggitti**[4†] ,
Alfonso Emilio Gerevini[1] and **Enrico Scala**[1]
[1]University of Brescia, Italy
[2]University of Oxford, UK
[3]Sapienza University, Italy
[4]Bank of Italy
{luigi.bonassi, alfonso.gerevini, enrico.scala}@unibs.it, giuseppe.degiacomo@cs.ox.ac.uk,
{marco.favorito, francesco.fuggitti}@gmail.com

Abstract

We study classical planning for temporally extended goals expressed in Pure-Past Linear Temporal Logic (PPLTL). PPLTL is as expressive as Linear-time Temporal Logic on finite traces (LTL_f), but as shown in this paper, it is computationally much better behaved for planning. Specifically, we show that planning for PPLTL goals can be encoded into classical planning with minimal overhead, introducing only a number of new fluents that is at most linear in the PPLTL goal and no spurious additional actions. Based on these results, we implemented a system called Plan4Past, which can be used along with state-of-the-art classical planners, such as LAMA. An empirical analysis demonstrates the practical effectiveness of Plan4Past, showing that a classical planner generally performs better with our compilation than with other existing compilations for LTL_f goals over the considered benchmarks.

1 Introduction

In AI Planning, a temporally extended goal is a (possibly complex) property that the state-trace induced by a plan has to satisfy. Planning for temporally extended goals has a long tradition in AI Planning, including pioneering work in the late '90s [Bacchus *et al.*, 1996; Bacchus and Kabanza, 1996; Bacchus *et al.*, 1997; Bacchus and Kabanza, 2000], work on planning via Model Checking [Cimatti *et al.*, 1997; De Giacomo and Vardi, 1999; Giunchiglia and Traverso, 1999], and work on declarative and procedural constraints [Baier and McIlraith, 2006; Baier *et al.*, 2008b]. Also, recent works focus on handling the subset of temporally extended goals defined by PDDL3 [Gerevini *et al.*, 2009], interpreting them as preferences [Percassi and Gerevini, 2019], state-trajectory constraints [Bonassi *et al.*, 2021; Bonassi *et al.*, 2022b; Bonassi *et al.*, 2024], or over action sequences [Bienvenu *et al.*, 2011; Bonassi *et al.*, 2022a].

In this scenario, a powerful formalism to express temporally extended goals is Linear-time Temporal Logic (LTL), which has been advocated as an excellent tool to express properties of processes in Formal Methods [Baier *et al.*, 2008a]. Given that tasks in AI planning are inherently of finite nature, a finite-trace variant of LTL, namely LTL_f, has often been employed [Bacchus and Kabanza, 1996; Baier and McIlraith, 2006; De Giacomo and Vardi, 2013]. Notably, an alternative to LTL_f is the Pure-Past Linear Temporal Logic, or PPLTL [Lichtenstein *et al.*, 1985], which has been attractive in expressing non-Markovian rewards in MDPs [Bacchus *et al.*, 1996], normative properties in multi-agent systems [Fisher and Wooldridge, 2005; Knobbout *et al.*, 2016; Alechina *et al.*, 2018], explanations in dynamical systems [Sohrabi *et al.*, 2011], synthesis specifications [Cimatti *et al.*, 2020], or in the context of plan/goal recognition [Fraga Pereira *et al.*, 2024]. PPLTL looks at the trace backward instead of forward as LTL_f and does so by expressing properties on traces using past operators only. PPLTL and LTL_f have the same expressive power, but translating a formula from one into the other (and vice versa) can be prohibitive since the best-known algorithms are 3EXPTIME [De Giacomo *et al.*, 2020].

In this paper, we study the problem of planning for PPLTL temporally extended goals in deterministic domains. Similarly to planning for LTL_f goals, which has already been studied in, e.g., [Baier and McIlraith, 2006; De Giacomo and Vardi, 2013; Torres and Baier, 2015], planning for PPLTL goals requires reaching a certain state satisfying the PPLTL goal, i.e., the state-trace produced to reach such a state satisfies the goal formula.

From the literature, it is well-known that LTL formulas have a fixpoint characterization that allows splitting any formula into a propositional formula to be checked at the current instant and a temporal formula to be checked at the next instant [Gabbay *et al.*, 1980; Manna, 1982; Emerson, 1990]. This property has already been exploited in AI, e.g., in the MetateM approach [Barringer *et al.*, 1989], and later by Bacchus and Kabanza [1996], one of the most influential work on

[*]Extended abstract of the paper [Bonassi *et al.*, 2023b] published in ICAPS 2023.

[†]Views and opinions expressed here are of the author's own and are not representative of the Bank of Italy's official position.

planning for temporally extended goals. Analogously, when we consider PPLTL formulas, such a fixpoint characterization splits the formula into a propositional formula on the current instant and a temporal formula on the past to be checked at the *previous* instant. In this paper, similarly to what was done by [Bacchus *et al.*, 1997] in the context of (factorized) MDPs with PPLTL rewards, we exploit the fixpoint characterization of PPLTL formulas to show that planning for PPLTL goals can be polynomially encoded into classical planning and without adding any spurious actions. The use of PPLTL is crucial to obtain such nice results, as encoding LTL_f goals into classical planning problems results either in worst-case exponential encodings [Baier and McIlraith, 2006] or in encodings that include additional spurious actions significantly increasing the plan length [Torres and Baier, 2015].

Finally, we developed our approach in a tool called Plan4Past that can be used along with state-of-the-art classical planners and experimentally showed its practical effectiveness by comparing it against techniques for LTL_f goals.

2 Classical Planning with Pure-Past Linear Temporal Logic Goals

We model a classical planning domain as a tuple $\mathcal{D} = \langle \mathcal{F}, A, Pre, Eff \rangle$, where $\mathcal{F}$ is a set of fluents (i.e., a set of positive literals), A is a set of action labels, and Pre and Eff are two functions denoting the preconditions and effects of each action $a \in A$. A planning state s is a collection of atoms from $\mathcal{F}$, meaning that f is true in s if $f \in s$ and f is false in s otherwise. Both functions Pre and Eff take an action label $a \in A$ as input and return a propositional formula over $\mathcal{F}$ and a set of conditional effects, respectively. A conditional effect is a pair $c \triangleright e$, where c is a formula, and e is a set of literals. An action a can be applied in a state s only if $Pre(a)$ holds true in s, and applying a in s induces a new state s', denoted with $s[a]$, that is defined following the standard definition of conditional effects [Röger *et al.*, 2014]. Intuitively, a conditional effect $c \triangleright e$ specifies that if the condition c holds in the current state, then the effect e needs to be applied.

A classical planning problem is a tuple $\Gamma = \langle \mathcal{D}, s_0, G \rangle$, where $\mathcal{D}$ is a domain model, s_0 is the initial state, and G is a formula over $\mathcal{F}$ representing the goal to achieve. A solution to a planning problem Γ is a sequence of actions $a \in A$ called *plan* $\pi = a_0, \ldots, a_{n-1}$ such that, when executed, induces a finite *state-trace* $\tau = s_0, \ldots, s_n$, where s_i satisfies $Pre(a)$ and $s_{i+1} = s[a]$ for $i = 0, \ldots, n-1$, and s_n satisfies G.

Usually, G is a *reachability* goal as it represents a property that must hold in the final state. Many real-world scenarios require achieving more general goals than just reachability goals. In most cases, realistic goals require properties to hold over a *sequence* of planning states. These types of *temporal goals*, also called *temporally extended goals*, are often expressed using temporal logics on finite or infinite sequences of states. In particular, this paper focuses on temporally extended goals expressed in Pure-Past Linear Temporal Logic (PPLTL). PPLTL is the variant of LTL_f that talks about the past instead of the future. Given a set $\mathcal{P}$ of propositions, PPLTL is defined as:

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

where $p \in \mathcal{P}$, Y is the *yesterday* operator and S is the *since* operator. PPLTL formulas are interpreted on *finite nonempty* traces, also called *histories*, $\tau = s_0 \cdots s_n$ where s_i at instant i is a propositional interpretation over the alphabet $2^{\mathcal{P}}$. We define the satisfaction relation $\tau, i \models \varphi$, stating that φ holds at instant i, as follows:

- $\tau, i \models p$ iff p holds in s_i (for p proposition);
- $\tau, i \models \neg\varphi$ iff τ, i does not satisfy φ;
- $\tau, i \models \varphi_1 \wedge \varphi_2$ iff both φ_1 and φ_2 hold at instant i;
- $\tau, i \models \mathsf{Y}\varphi$ iff φ held at the previous instant $i-1$ with $i \geq 1$. If $i = 1$, then $\mathsf{Y}\varphi$ is false.
- $\tau, i \models \varphi_1 \,\mathsf{S}\, \varphi_2$ iff φ_2 held at some previous instant k, and φ_1 held in every subsequent instant j with $k < j \leq i$.

The entire trace τ satisfies φ, denoted with $\tau \models \varphi$, when φ holds in the last instant of τ. From the primitive PPLTL operators, one can easily derive other operators, such as $\varphi_1 \vee \varphi_2 \equiv \neg(\varphi_1 \wedge \varphi_2)$, the *once* operator $\mathsf{O}\varphi \equiv \mathit{true}\,\mathsf{S}\,\varphi$ and the *historically* operator $\mathsf{H}\varphi \equiv \neg\mathsf{O}\neg\varphi$. Intuitively, $\mathsf{O}\varphi$ specifies that φ held true sometimes in the past, while $\mathsf{H}\varphi$ requires φ to always hold in each past state. We denote by $\mathsf{sub}(\varphi)$ the set of all subformulas of φ obtained from the syntax tree of φ [De Giacomo and Vardi, 2013].

If we specify planning goals using PPLTL, a planning problem becomes a tuple $\Gamma = \langle \mathcal{D}, s_0, \varphi \rangle$, where φ is a PPLTL goal formula. In the case of planning with a temporally extended goal φ, a plan π is a solution to Γ if the sequence of states $\tau = s_0, \ldots, s_n$ induced by π satisfies φ. The following example shows two properties that are naturally expressed in PPLTL. Further examples can be found in Fuggitti [2023].

Example 1. *The goal "We are now at location l_1 and have passed through location l_2" in* PPLTL *is $l_1 \wedge \mathsf{O}(l_2)$. Another interesting property is "Every time I took the bus, I bought a new ticket beforehand", which translates as $\mathsf{H}(Bus \implies \mathsf{Y}(\neg Bus \,\mathsf{S}\, Ticket))$ [De Giacomo* et al.*, 2020].*

3 Handling PPLTL Goals

Our approach exploits three main observations: (*i*) to evaluate the PPLTL goal formula we only need the truth value of its subformulas; (*ii*) every PPLTL formula can be put in a form where its evaluation depends only on the current state and the evaluation of a key set of PPLTL subformulas at the previous instant; (*iii*) one can recursively compute the value of such a small set of formulas and keep track of them with additional propositional variables in the state of the planning domain.

The well-known fixpoint characterization of LTL and variants [Gabbay *et al.*, 1980; Manna, 1982; Emerson, 1990] is key to our approach. For PPLTL, such a fixpoint characterization splits the formula into a propositional formula on the current instant and a temporal formula on the past to be checked at the *previous* instant. In particular, PPLTL formulas can be decomposed into present and past components, given the fixpoint characterization of the *since* operator: $\phi_1 \,\mathsf{S}\, \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge \mathsf{Y}(\phi_1 \,\mathsf{S}\, \phi_2))$. Exploiting this equivalence, the formula decomposition can be computed by recursively applying the following transformation function $\mathsf{pnf}(\cdot)$:

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

A formula resulting from the application of $\mathsf{pnf}(\cdot)$ is in Previous Normal Form (PNF). Note that formulas in PNF have proper temporal subformulas (i.e., subformulas whose main construct is a temporal operator) appearing only in the scope of the Y operator. Also, observe that the formulas of the form $\mathsf{Y}\phi$ in $\mathsf{pnf}(\varphi)$ are such that $\phi \in \mathsf{sub}(\varphi)$. It is easy to see that every PPLTL formula φ can be converted to its PNF form $\mathsf{pnf}(\varphi)$ in linear-time in the size of the formula (i.e., $|\mathsf{sub}(\varphi)|$) and that $\mathsf{pnf}(\varphi)$ is equivalent to φ.

Example 2. *As a running example, we use the formula $\varphi = t \wedge \neg r \,\mathsf{S}\, m$, which enforces scenarios where an agent finishes the task t without using the resource r since activating the machine m. The* pnf *of φ is:*

$$\mathsf{pnf}(t \wedge \neg r \,\mathsf{S}\, m) = t \wedge (m \vee (\neg r \wedge \mathsf{Y}(\neg r \,\mathsf{S}\, m))).$$

Notably, the PNF transformation allows us to determine the truth value of a PPLTL formula by knowing the truth of some atomic propositions at the current instant and the truth of some key subformulas at the previous instant. In particular, the key subformulas that we need to consider are those appearing within the Y-scope in the PNF. Therefore, we interpret these specific subformulas as *atomic propositions*, denoting them with quotes, and collecting them in a set denoted as Σ_φ. These propositions are:

1. "$\mathsf{Y}\phi$" for each subformula of φ of the form $\mathsf{Y}\phi$;
2. "$\mathsf{Y}(\phi_1 \,\mathsf{S}\, \phi_2)$" for each subformula of φ of the form $\phi_1 \,\mathsf{S}\, \phi_2$;

Example 3. *For $\varphi = t \wedge \neg r \,\mathsf{S}\, m$ we have $\Sigma_\varphi = \{$"$\mathsf{Y}(\neg r \,\mathsf{S}\, m)$"$\}$. Therefore, to evaluate φ, we only need to keep track of the subformula $\mathsf{Y}(\neg r \,\mathsf{S}\, m)$ using the propositional variable "$\mathsf{Y}(\neg r \,\mathsf{S}\, m)$".*

To evaluate the truth value of propositions in Σ_φ, we introduce the interpretation $\sigma : \Sigma_\varphi \to \{\top, \bot\}$, and characterize the evaluation of subformulas at a certain instant $i \leq n$ with σ_i. Intuitively, given an instant i, σ_i tells us which propositions related to the previous instant (i.e., in Σ_φ) are true at the instant i. Therefore, a PPLTL formula φ can be simply evaluated considering the propositional interpretation in the current instant i and the truth value assigned by σ_i to propositions related to the previous instant. We can easily do so by introducing an evaluation procedure denoted by $\mathsf{val}(\varphi, \sigma_i, s_i)$. Essentially, $\mathsf{val}(\cdot)$ takes as input a PPLTL formula φ, the current interpretation s_i of the propositional symbols, and the current interpretation σ_i of the "$\mathsf{Y}(\phi)$" atoms, and evaluates the (propositional) formula obtained by swapping every $\mathsf{Y}(\phi)$ with "$\mathsf{Y}(\phi)$" in the recursive computation of $\mathsf{pnf}(\varphi)$. An example of this procedure is shown below.

Example 4. *We have that $\mathsf{val}(t \wedge \neg r \,\mathsf{S}\, m, s_i, \sigma_i)$ evaluates whether the formula:*

$$t \wedge (m \vee (\neg r \wedge \text{“}\mathsf{Y}(\neg r \,\mathsf{S}\, m)\text{”}))$$

holds true given the interpretations s_i and σ_i. For instance, $\mathsf{val}(t \wedge \neg r \,\mathsf{S}\, m, s_i, \sigma_i)$ is true when $\sigma_i($"$\mathsf{Y}(\neg r \,\mathsf{S}\, m)$"$) = \top$ and s_i satisfy both t and $\neg r$.

Note that, the same procedure can be applied to determine $\mathsf{val}(\phi, \sigma_i, s_i)$ for every subformula $\phi \in \mathsf{sub}(\varphi)$. With this approach, we can use $\mathsf{val}(\varphi, \sigma_i, s_i)$ to capture the truth of φ at any instant i (i.e., $\mathsf{val}(\varphi, \sigma_i, s_i)$ is true iff $\tau, i \models \varphi$). Clearly, computing $\mathsf{val}(\cdot)$ for some instant i requires the interpretation σ_i. Luckily, given a trace $\tau = s_0, s_1, \dots s_n$, we can easily construct σ_i for every $i \in \{0, 1, \dots n\}$ as follows. At the beginning of the trace, we know that, by definition of PPLTL, every formula of the form $\mathsf{Y}(\phi)$ is false. Hence, we set $\sigma_0($"$\mathsf{Y}\phi$"$) \doteq \bot$ for each "$\mathsf{Y}\phi$" $\in \Sigma_\varphi$, enabling the computation of $\mathsf{val}(\phi, s_0, \sigma_0)$ for every $\phi \in \mathsf{sub}(\varphi)$. Then, since $\mathsf{Y}(\phi)$ holds at instant i iff ϕ holds at the previous instant $i-1$, we can iteratively set $\sigma_i($"$\mathsf{Y}\phi$"$) \doteq \mathsf{val}(\phi, \sigma_{i-1}, s_{i-1})$ for every $i = \{1, \dots n\}$. By doing so, we can evaluate any PPLTL formula at any instant i without considering the entire history produced so far.

4 Encoding PPLTL Goals in Planning

Intuitively, given a goal formula φ, we introduce a set of fresh atoms to keep track of the interpretation σ. Then, we simulate the construction of σ_i and the computation of $\mathsf{val}(\varphi, \sigma_i, s_i)$ for every instant i of the state trace as actions are applied. We do so by modifying the effect function and by employing *axioms* and *derived predicates* [Thiébaux *et al.*, 2005]. Given a problem $\Gamma = \langle \mathcal{D}, s_0, \varphi \rangle$, in the following we describe how to encode Γ into $\Gamma' = \langle \mathcal{D}', s'_0, G' \rangle$, where G' is a reachability goal, and $\mathcal{D}' = \langle \mathcal{F}', \mathcal{X}', A, Pre, Eff' \rangle$ is the new planning domain extended with a set of axioms $\mathcal{X}'$.

Fluents. $\mathcal{F}'$ contains the fluents of $\mathcal{F}$, as well as one fluent for each proposition "$\mathsf{Y}\phi$" in Σ_φ to keep track of propositional interpretations σ_i, and the set of predicates P_{der} defined below. Formally, $\mathcal{F}' = \mathcal{F} \cup \Sigma_\varphi \cup P_{der}$.

Axioms. We employ axioms to elegantly determine $\mathsf{val}(\phi, \sigma_i, s_i)$ in every state s_i. Axioms have the form $d \leftarrow \psi$, where $d \in P_{der}$ is a positive literal called *derived predicate*, and ψ is a propositional formula over a set of predicates. Let s be a state, axiom $d \leftarrow \psi$ determines that the derived predicate d holds in s if and only if $s \models \psi$.

We include an axiom $\mathsf{val}_\phi \leftarrow \psi$ for every subformula $\phi \in \mathsf{sub}(\varphi)$, and P_{der} is defined as the set of all derived predicates val_ϕ, i.e., $P_{der} = \{\mathsf{val}_\phi \mid \phi \in \mathsf{sub}(\varphi)\}$. By mimicking the recursive computation of $\mathsf{pnf}(\cdot)$, we get the following axioms:

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

Clearly, we have that val_ϕ captures the evaluation of $\mathsf{val}(\phi, \sigma_i, s_i)$. We add to $\mathcal{D}'$ a set of axioms for every subformula ϕ in $\mathsf{sub}(\varphi)$, i.e., $\mathcal{X}' = \{\mathsf{val}_\phi \leftarrow \phi \mid \phi \in \mathsf{sub}(\varphi)\}$.

Initial State. The initial state assigns each fluent "$\mathsf{Y}\phi$" $\in \Sigma_\varphi$ to the truth value given by σ_0. That is, $s'_0 = (\sigma_0, s_0)$.

Effects. The effect of every action is modified by adding a way to update the assignments of propositions in Σ_φ. For each "$\mathsf{Y}\phi$" $\in \Sigma_\varphi$, we model the assignment $\sigma_i(\text{"}\mathsf{Y}\phi\text{"}) \doteq \mathsf{val}(\phi, \sigma_{i-1}, s_{i-1})$ using two conditional effects of the form:

$$\mathsf{val}_\phi \rhd \ \text{"}\mathsf{Y}\phi\text{"}$$
$$\neg\mathsf{val}_\phi \rhd \neg\text{"}\mathsf{Y}\phi\text{"}$$

These effects specify that the fluent "$\mathsf{Y}\phi$" is set to true (false, resp.) in the state s_i iff val_ϕ is true (false, resp.) in s_{i-1}. The effect of each action is extended with the same set of conditional effects. Formally, for all $a \in A$, $\mathit{Eff}'(a) = \mathit{Eff}(a) \cup \{\mathsf{val}_\phi \rhd \text{"}\mathsf{Y}\phi\text{"}, \neg\mathsf{val}_\phi \rhd \neg\text{"}\mathsf{Y}\phi\text{"} \mid \text{"}\mathsf{Y}\phi\text{"} \in \Sigma_\varphi\}$.

Goal. The new goal is $G' = \mathsf{val}_\varphi$. That is, we require the derived predicate associated with the original PPLTL goal formula φ to hold in the last instant.

The encoding is *polynomially* related to the original problem. In addition, every plan π for Γ is also a plan for Γ' and vice versa, making the encoding sound and complete.

Theorem 1 (Bonassi *et al.* 2023b)**.** *Let Γ be a planning problem with a* PPLTL *goal φ, and Γ' be the corresponding encoded planning problem. The size of Γ' is polynomial in the size of Γ. In particular, the additional fluents introduced are linear in the size of the* PPLTL *goal φ of Γ. Moreover, every plan π is a plan for Γ iff π is a plan for Γ'.*

5 Experiments

This section briefly summarizes the experimental analysis presented in Bonassi *et al.* [2023b]. The novel approach has been implemented in a tool called Plan4Past (P4P), which is available at https://github.com/whitemech/Plan4Past.

The experimental analysis compares P4P with the two state-of-the-art compilations Exp [Baier and McIlraith, 2006] and Poly [Torres and Baier, 2015] for LTL$_f$ temporally extended goals. In particular, Exp explicitly uses NFAs to represent the LTL$_f$ formula, whereas Poly implicitly constructs the NFA for the goal formula. As a result, the Exp encoding is worst-case exponential in the size of the LTL$_f$ formula, while the Poly encoding remains polynomial in the size of the LTL$_f$ goal at the cost of a polynomial increase in the size of solution plans. On the other hand, the P4P encoding is polynomial in the size of the PPLTL goal and preserves the plan length.

We evaluated all compilation systems over a set of semantically equivalent LTL$_f$ and PPLTL goals. In total, our benchmark suite features 152 instances across the planning domains BLOCKSWORLD, ELEVATOR, ROVERS, and OPENSTACKS. The instances resulting from each compilation system were solved using the LAMA [Richter and Westphal, 2010] classical planner with runtime and memory limits of 1800s and 8GB, respectively.

Table 1 reports on the coverage (number of instances solved) of all compilations across all domains. We observe that P4P performs equally to or better than both Poly and Exp in all domains. In many cases, the Poly encoding forces LAMA to schedule many spurious actions after each original planning action taken. For example, the biggest instance of BLOCKSWORLD solved by Poly requires 6402 actions, of which 104 are actions of the original domain. Instead, the

Domain	P4P	Poly	Exp
ROVERS (47)	**40**	13	28
BLOCKSWORLD (36)	**36**	16	9
OPENSTACKS (40)	**17**	15	14
ELEVATOR (29)	**29**	4	**29**
Total	**122**	48	80

Table 1: Number of instances solved by P4P, Poly, and Exp across all domains. In bold the best performers.

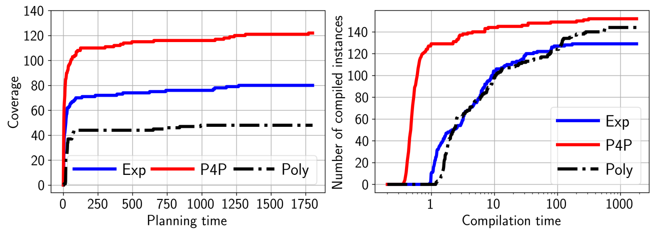

Figure 1: Number of solved instances (left) and compiled instances (right) versus computation time.

Exp encoding blows up during compilation or produces problems that cannot be handled by LAMA. The compilation time seems to be an issue for both LTL$_f$ compilations. Indeed, if we look at Figure 1 (right), P4P compiles 94.7% instances within 10s, while both Poly and Exp converge much more slowly. Figure 1 (left) displays the number of benchmark instances solved with a given timeout. All systems achieve their maximum coverage quite quickly, with P4P leaving the others well behind right after the start.

6 Conclusion

This paper summarizes the results presented in Bonassi *et al.* [2023b] on classical planning for PPLTL goals. In particular, our work shows that planning for PPLTL goals can be encoded into classical planning for reachability goals with minimal overhead and without increasing the plan length. Handling PPLTL goals is remarkably simple and elegant, given the direct mapping between the theoretical formulation and the encoding compilation without sacrificing efficiency. Moreover, we practically show that the novel encoding is able to solve a broader range of problems compared to the state-of-the-art encodings for LTL$_f$ goals. Interestingly, the proposed encoding can be seamlessly extended to Fully Observable Non-Deterministic (FOND) planning domains [Bonassi *et al.*, 2023a]. Also, our findings have already paved the way for the use of PPLTL in other applications, such as plan selection through natural language [Fuggitti and Chakraborti, 2023; Chakraborti *et al.*, 2024], suggesting that PPLTL may definitely become a promising candidate to be the mainstream language to express temporal goals in planning. Future work concerns handling more expressive formalisms such as Pure-Past Linear Dynamic Logic (PPLDL) [De Giacomo *et al.*, 2020], developing planners that can *natively* handle PPLTL goals, and exploring PPLTL-aware heuristics.

Acknowledgments

This work has been partially supported by the EU H2020 project AIPlan4EU (No. 101016442), the ERC-ADG White-Mech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No. 20203FFYLK), and the PNRR MUR project FAIR (No. PE0000013).

References

[Alechina *et al.*, 2018] Natasha Alechina, Brian Logan, and Mehdi Dastani. Modeling norm specification and verification in multiagent systems. *FLAP*, 5(2):457–490, 2018.

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

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

[Bacchus *et al.*, 1996] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Rewarding behaviors. In *AAAI*, pages 1160–1167, 1996.

[Bacchus *et al.*, 1997] Fahiem Bacchus, Craig Boutilier, and Adam Grove. Structured solution methods for non-markovian decision processes. In *AAAI*, pages 112–117, 1997.

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

[Baier *et al.*, 2008a] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of Model Checking*. MIT, 2008.

[Baier *et al.*, 2008b] Jorge A. Baier, Christian Fritz, Meghyn Bienvenu, and Sheila A. McIlraith. Beyond classical planning: Procedural control knowledge and preferences in state-of-the-art planners. In *AAAI*, pages 1509–1512. AAAI, 2008.

[Barringer *et al.*, 1989] Howard Barringer, Michael Fisher, Dov M. Gabbay, Graham Gough, and Richard Owens. METATEM: A framework for programming in temporal logic. In *REX Workshop*, volume 430 of *LNCS*, pages 94–129. Springer, 1989.

[Bienvenu *et al.*, 2011] Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith. Specifying and computing preferred plans. *Artif. Intell.*, 175(7-8):1308–1345, 2011.

[Bonassi *et al.*, 2021] Luigi Bonassi, Alfonso Emilio Gerevini, Francesco Percassi, and Enrico Scala. On planning with qualitative state-trajectory constraints in PDDL3 by compiling them away. In *ICAPS*, pages 46–50. AAAI Press, 2021.

[Bonassi *et al.*, 2022a] Luigi Bonassi, Alfonso Emilio Gerevini, and Enrico Scala. Planning with qualitative action-trajectory constraints in PDDL. In *IJCAI*, pages 4606–4613. ijcai.org, 2022.

[Bonassi *et al.*, 2022b] Luigi Bonassi, Enrico Scala, and Alfonso Emilio Gerevini. Planning with PDDL3 qualitative constraints for cost-optimal solutions through compilation (short paper). In *IPS/RiCeRcA/SPIRIT@AI*IA*, volume 3345 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

[Bonassi *et al.*, 2023a] Luigi Bonassi, Giuseppe De Giacomo, Marco Favorito, Francesco Fuggitti, Alfonso Emilio Gerevini, and Enrico 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.

[Bonassi *et al.*, 2023b] Luigi Bonassi, Giuseppe De Giacomo, Marco Favorito, Francesco Fuggitti, Alfonso Emilio Gerevini, and Enrico Scala. Planning for temporally extended goals in pure-past linear temporal logic. In *ICAPS*, pages 61–69. AAAI Press, 2023.

[Bonassi *et al.*, 2024] Luigi Bonassi, Alfonso Emilio Gerevini, and Enrico Scala. Dealing with numeric and metric time constraints in PDDL3 via compilation to numeric planning. In *AAAI*, pages 20036–20043. AAAI Press, 2024.

[Chakraborti *et al.*, 2024] Tathagata Chakraborti, Jungkoo Kang, Francesco Fuggitti, Michael Katz, and Shirin Sohrabi. Interactive plan selection using linear temporal logic, disjunctive action landmarks, and natural language instruction. In *AAAI*, 2024.

[Cimatti *et al.*, 1997] Alessandro Cimatti, Fausto Giunchiglia, Enrico Giunchiglia, and Paolo Traverso. Planning via model checking: A decision procedure for *AR*. In *ECP*, pages 130–142. Springer, 1997.

[Cimatti *et al.*, 2020] Alessandro Cimatti, Luca Geatti, Nicola Gigante, Angelo Montanari, and Stefano Tonetta. Reactive synthesis from extended bounded response LTL specifications. In *FMCAD*, pages 83–92. IEEE, 2020.

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

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

[De Giacomo *et al.*, 2020] Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin. Pure-past linear temporal and dynamic logic on finite traces. In *IJCAI*, pages 4959–4965. ijcai.org, 2020.

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

[Fisher and Wooldridge, 2005] Michael Fisher and Michael Wooldridge. Temporal reasoning in agent-based systems. In *FAI*, pages 469–495. Elsevier, 2005.

[Fraga Pereira *et al.*, 2024] Ramon Fraga Pereira, Francesco Fuggitti, Felipe Meneguzzi, and Giuseppe De Giacomo.

Temporally extended goal recognition in fully observable non-deterministic domain models. *Appl. Intell.*, 54(1):470–489, 2024.

[Fuggitti and Chakraborti, 2023] Francesco Fuggitti and Tathagata Chakraborti. NL2LTL - a python package for converting natural language (NL) instructions to linear temporal logic (LTL) formulas. In *AAAI*, pages 16428–16430. AAAI Press, 2023.

[Fuggitti, 2023] Francesco Fuggitti. *Efficient Techniques for Automated Planning for Goals in Linear Temporal Logics on Finite Traces*. PhD Dissertation, Sapienza University & York University, September 2023.

[Gabbay *et al.*, 1980] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *POPL*, pages 163–173. ACM Press, 1980.

[Gerevini *et al.*, 2009] Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *AIJ*, 173(5-6):619–668, 2009.

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

[Knobbout *et al.*, 2016] Max Knobbout, Mehdi Dastani, and John-Jules Ch. Meyer. A dynamic logic of norm change. In *ECAI*, pages 886–894, 2016.

[Lichtenstein *et al.*, 1985] Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In *Logic of Programs*, pages 196–218. Springer, 1985.

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

[Percassi and Gerevini, 2019] Francesco Percassi and Alfonso Emilio Gerevini. On compiling away PDDL3 soft trajectory constraints without using automata. In *ICAPS*, pages 320–328. AAAI Press, 2019.

[Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR*, 39:127–177, 2010.

[Röger *et al.*, 2014] Gabriele Röger, Florian Pommerening, and Malte Helmert. Optimal planning in the presence of conditional effects: Extending LM-Cut with context splitting. In *ECAI*, pages 765–770, 2014.

[Sohrabi *et al.*, 2011] Shirin Sohrabi, Jorge Baier, and Sheila McIlraith. Preferred explanations: Theory and generation via planning. In *AAAI*, volume 25, pages 261–267, 2011.

[Thiébaux *et al.*, 2005] Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. In defense of PDDL axioms. *Artif. Intell.*, 168(1-2):38–69, 2005.

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