A Singly Exponential Transformation of $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$ into Pure Past LTL

Alessandro Artale[1], Luca Geatti[2], Nicola Gigante[1], Andrea Mazzullo[1], Angelo Montanari[2]
[1]Free University of Bozen-Bolzano
[2] University of Udine
artale@inf.unibz.it, luca.geatti@uniud.it, gigante@inf.unibz.it, mazzullo@inf.unibz.it,
angelo.montanari@uniud.it

Abstract

Confronting the past can be hard. This is true even in Linear Temporal Logic (LTL), interpreted on either infinite or finite traces, when faced with the problem of transforming a temporally future formula into an equivalent one that contains *past* temporal modalities only. To our knowledge, the best among the available *pastification* procedures for full LTL, as well as for expressive enough fragments of it (that is, containing at least one temporal modality other than *tomorrow*), are *triply exponential* in the size of the input. In this paper, we focus on the fragment of LTL that features the *tomorrow* and *eventually* modalities, and provide a *singly exponential* pastification algorithm for it. The transformation is based on a normalisation procedure that requires a non-trivial complexity analysis, and on the subsequent generation of a pure past formula from suitably-defined dependency tree structures. Moreover, leveraging its purely syntactic nature, we present an implementation of our procedure in a temporal satisfiability checking tool that deals with both future and past modalities.

1 Introduction

In this paper, we focus on the fragment of Linear Temporal Logic (LTL) (Pnueli 1977) that features tomorrow and eventually modalities, and provide a singly exponential algorithm that transforms its formulas into pure past ones. This "past rewriting" of temporal formulas proved itself to be quite useful in fundamental tasks like reactive synthesis.

LTL extends propositional logic with *future* temporal modalities to reason over *infinite* linear structures based on the order of the natural numbers, called *traces*. Together with its extension with *past* modalities (LTL+P) (Lichtenstein, Pnueli, and Zuck 1985) and its finite variant ($\mathsf{LTL_f}$) that interprets formulas over *finite traces* (De Giacomo and Vardi 2013), it proved itself to be essential in fields like automated reasoning, formal verification, and knowledge representation. A notable fragment of LTL is coSafetyLTL (Chang, Manna, and Pnueli 1992; Cimatti et al. 2022; Artale et al. 2023), syntactically defined as the fragment of LTL in negation normal form whose temporal operators are all existential, *i.e.*, *tomorrow* (X) and *until* (U).

The *pure past* fragment of LTL+P (Lichtenstein, Pnueli, and Zuck 1985; De Giacomo et al. 2021), on which we focus, is defined as the subset of formulas of LTL+P devoid of future modalities and is denoted by pLTL. Its formulas are naturally interpreted at the end of *finite traces*, *i.e.*, initial segments of the natural numbers. pLTL has received a renewed attention in the last years, due to its important theoretical and algorithmic properties.

First, it has been shown that pLTL is expressively equivalent to $\mathsf{LTL_f}$ (Lichtenstein, Pnueli, and Zuck 1985). In addition, pLTL is tightly related to *cosafety fragments* of LTL (Chang, Manna, and Pnueli 1992), that is, fragments on infinite traces for which a *finite prefix* of a trace suffices to establish whether the whole trace is a model of a formula, thus having a strong connection with finite trace semantics. Indeed, consider the logic $\mathsf{F}(\mathsf{pLTL})$, defined as the set of formulas of type $\mathsf{F}(\alpha)$ where F is the *eventually* operator and α is a pLTL formula (Chang, Manna, and Pnueli 1992). Formulas of this type have the ability to hook a future time point of the trace with F, and then to constrain the prefix up to that point by means of the pLTL formula. It turns out that such formulas are a canonical form for coSafetyLTL.

From an automata-theoretic viewpoint, pLTL formulas enjoy a compilation into *deterministic finite automata* (DFAs) of *singly exponential size* (De Giacomo et al. 2021; Cimatti et al. 2021), a result that cannot be achieved for full $\mathsf{LTL_f}$. Interestingly, this compilation for pLTL can actually be performed in a fully symbolic fashion, *i.e.*, by means of Boolean formulas only (Cimatti et al. 2021; Geatti, Montali, and Rivkin 2022). Finally, it has been recently proved that the *reactive synthesis* problem (Pnueli and Rosner 1989) of pLTL specification is EXPTIME-complete (Artale et al. 2023), in contrast to the 2EXPTIME-completeness of the same problem for LTL and $\mathsf{LTL_f}$ (Rosner 1992; De Giacomo and Vardi 2015). Similar results have been obtained for *monitoring* and *planning* problems (De Giacomo et al. 2022; De Giacomo, Favorito, and Fuggitti 2022).

Motivated by the above-summarized promising results on pure past fragments, a recent trend has focused on the study of algorithms for transforming fragments of coSafetyLTL and $\mathsf{LTL_f}$ into equivalent ones of, respectively, $\mathsf{F}(\mathsf{pLTL})$ and pLTL. This transformation is known as *pastification*. Simple planning patterns, which always correspond to $\mathsf{LTL_f}$ formulas with only *at most two* nested temporal operators, *e.g.*, the ones of DECLARE (Geatti, Montali, and Rivkin 2022), or the trajectory constraints of PDDL (De Giacomo, Favorito, and Fuggitti 2022), can be easily pastified into formulas of polynomial size with respect to the starting one. The same holds for the fragment of LTL with only *tomorrow* as temporal operator (LTL[X]) (Maler, Nickovic, and Pnueli 2005; Maler, Nickovic, and Pnueli 2007).

Nevertheless, devising an efficient algorithm for the pastification of arbitrary coSafetyLTL and $\mathsf{LTL_f}$ formulas is a challenging task. In fact, the complexity picture is radically different for the general case. The best known algorithms for the pastification of coSafetyLTL and $\mathsf{LTL_f}$ produce a pure past formula of *triply exponential* size (De Giacomo et al. 2021), where the exponential blowups derive, respectively, from: (i) the transformation into a *nondeterministic finite automaton* (NFA); (ii) its determinization into a DFA; (iii) the application of the *Krohn-Rhodes Cascaded Decomposition* (Maler and Pnueli 1990), and the consequent translation into a formula of pLTL. This transformation is impractical: after more than 60 years, to the best of our knowledge, there is only one implementation of the Krohn-Rhodes Cascaded Decomposition[1] and no implementation of the pastification algorithm for coSafetyLTL and $\mathsf{LTL_f}$. Last but not least, other than the one for LTL[X], there are no *ad hoc* pastification algorithms for natural fragments of coSafetyLTL or $\mathsf{LTL_f}$, thus forcing one to use the triply exponential algorithm for the general case.

In this paper, we study the pastification problem for LTL[X, F], that is, the fragment of LTL with X and F as temporal operators. Our main contribution is a *singly exponential* pastification algorithm for LTL[X, F]. The transformation consists of two main steps: (i) the transformation of an LTL[X, F] formula into a suitably defined *normal form*, which involves the bottom-up application of a set of equivalence-preserving rewriting rules; (ii) the construction, for any formula in normal form, of a *dependency tree*, used to represent the temporal relations between subformulas in the scope of an F operator, and from which we can extract an F(pLTL) formula equivalent to the original one. We show that the only step introducing an exponential blowup is the first one, while the other is at most quadratic. Most importantly, in constrast to the triply exponential pastification algorithm for coSafetyLTL (or $\mathsf{LTL_f}$), our algorithm is: (i) singly exponential; (ii) purely syntactic, and thus simply implementable. As a matter of fact, our current implementation of the algorithm in the temporal satisfiability checking tool BLACK (Geatti, Gigante, and Montanari 2021) takes less than 500 lines of code. Finally, given the duality between F and the *always* operator G, we also obtain a pastification procedure from LTL[X, G] (that is, the LTL fragment with X and G as sole temporal operators) to G(pLTL) (which consists of pLTL formulas prefixed by a G operator) in singly exponential space.

The paper is organized as follows. We start with Section 2 where we briefly analyze related work. Then, in Section 3, we provide the necessary background. Next, in Section 4, we present the transformation into normal form together with its complexity. The definition of dependency trees and the translation into pure past are illustrated in Section 5. We discuss the implementation of the algorithm in Section 6, while in Section 7 we point out some future directions and some open problems.

[1]https://github.com/gap-packages/sgpdec

2 Related Work

Pastification techniques have first been studied for *bounded response* MTL (MTL-B, for short), which is a fragment of *Metric Temporal Logic* (interpreted over dense linear orders) where the temporal operators have bounds representing their interval of application. As an example, $\alpha \mathsf{U}_{[a,b]} \beta$ (for some $a, b \in \mathbb{N}$) restricts β to happen at least a and at most b time units after the interpretation of the whole formula. Notice that all bounds of MTL-B (in the previous example, a and b) are represented in binary.

Maler, Nickovic, and Pnueli (2005; 2007) developed a pastification algorithm for MTL-B that produces formulas of *polynomial size* with respect to input ones. The procedure exploits the fact that, for each model of an MTL-B formula ϕ, there exists a furthermost time point d such that the subsequent states cannot be constrained by ϕ in any way. The algorithm returns a formula that (i) uses only past operators, (ii) is polynomial in $|\phi|$, and (iii) is equivalent to ϕ when interpreted at time point d instead of at the origin.

If interpreted over discrete linear orders, and by considering all bounds represented in unary instead of binary, MTL-B is equivalent to LTL[X], *i.e.*, the fragment of LTL whose temporal operators are restricted to X. The technique of Maler, Nickovic and Pnueli can thus be used as a blackbox to obtain a polynomial size pastification for LTL[X].

3 Background

Given a set Σ of proposition letters, an LTL+P *formula* ϕ is generated as follows:

$$
\begin{aligned}
\phi \coloneqq\ & p \mid \neg p \mid \phi \vee \phi \mid \phi \wedge \phi && \text{Boolean connectives}\\
& \mid \mathsf{X}\phi \mid \phi \,\mathsf{U}\, \phi \mid \phi \,\mathsf{R}\, \phi && \text{future modalities}\\
& \mid \mathsf{Y}\phi \mid \widetilde{\mathsf{Y}}\phi \mid \phi \,\mathsf{S}\, \phi \mid \phi \,\mathsf{T}\, \phi && \text{past modalities}
\end{aligned}
$$

where $p \in \Sigma$. We use the standard shortcuts for $\top \coloneqq p \vee \neg p$, $\bot \coloneqq p \wedge \neg p$ (for some $p \in \Sigma$) and other temporal operators: $\mathsf{F}\phi \coloneqq \top \mathsf{U} \phi$, $\mathsf{G}\phi \coloneqq \bot \mathsf{R} \phi$, $\mathsf{O}\phi \coloneqq \top \mathsf{S} \phi$, and $\mathsf{H}\phi \coloneqq \bot \mathsf{T} \phi$. In addition, for any $n \in \mathbb{N}$, we inductively define the formula $\mathsf{X}^n\phi$ as follows: $\mathsf{X}^0\phi \coloneqq \phi$ and $\mathsf{X}^{n+1}\phi \coloneqq \mathsf{X}\mathsf{X}^n\phi$. $\mathsf{Y}^n\phi$, and $\widetilde{\mathsf{Y}}^n\phi$ are defined in a similar way. Notice that, *w.l.o.g.*, in the proposed definition of LTL+P, formulas are in Negation Normal Form (NNF), that is, negation is applied to proposition letters only. The *size* of a formula $\phi \in$ LTL+P, denoted by $|\phi|$, is the number of subformulas of ϕ.

A *pure future* (resp., *past*) *formula* is an LTL+P formula devoid of occurrences of past (resp., future) modalities. We denote by LTL (resp., pLTL) the set of pure future (resp., past) formulas. We denote by LTL[X], LTL[X, F], and LTL[X, G] the set of LTL formulas with temporal modalities in {X}, {X, F}, and {X, G}, respectively. Moreover, we denote by F(pLTL) (resp., G(pLTL)) the set of LTL+P formulas of the form $\mathsf{F}\alpha$ (resp., $\mathsf{G}\alpha$), with $\alpha \in$ pLTL.

Let $\sigma \in (2^{\Sigma})^{\omega}$ be a *state sequence* (also called *trace* or *word*). The *satisfaction* of an LTL+P formula ϕ by $\sigma = \sigma_0\sigma_1 \ldots$ at time $0 \leq i < \omega$, denoted by $\sigma, i \models \phi$, is defined as follows:

- $\sigma, i \models p$ iff $p \in \sigma_i$;
- $\sigma, i \models \neg p$ iff $p \notin \sigma_i$;
- $\sigma, i \models \phi_1 \vee \phi_2$ iff $\sigma, i \models \phi_1$ or $\sigma, i \models \phi_2$;
- $\sigma, i \models \phi_1 \wedge \phi_2$ iff $\sigma, i \models \phi_1$ and $\sigma, i \models \phi_2$;
- $\sigma, i \models \mathsf{X}\phi$ iff $\sigma, i+1 \models \phi$;
- $\sigma, i \models \mathsf{Y}\phi$ iff $i > 0$ and $\sigma, i-1 \models \phi$;
- $\sigma, i \models \widetilde{\mathsf{Y}}\phi$ iff either $i = 0$ or $\sigma, i-1 \models \phi$;
- $\sigma, i \models \phi_1 \mathbin{\mathsf{U}} \phi_2$ iff there exists $j \geq i$ such that $\sigma, j \models \phi_2$, and $\sigma, k \models \phi_1$ for all k, with $i \leq k < j$;
- $\sigma, i \models \phi_1 \mathbin{\mathsf{S}} \phi_2$ iff there exists $j \leq i$ such that $\sigma, j \models \phi_2$, and $\sigma, k \models \phi_1$ for all k, with $j < k \leq i$;
- $\sigma, i \models \phi_1 \mathbin{\mathsf{R}} \phi_2$ iff either $\sigma, j \models \phi_2$ for all $j \geq i$, or there exists $k \geq i$ such that $\sigma, k \models \phi_1$ and $\sigma, j \models \phi_2$ for all $i \leq j \leq k$;
- $\sigma, i \models \phi_1 \mathbin{\mathsf{T}} \phi_2$ iff either $\sigma, j \models \phi_2$ for all $0 \leq j \leq i$, or there exists $k \leq i$ such that $\sigma, k \models \phi_1$ and $\sigma, j \models \phi_2$ for all $i \geq j \geq k$.

We say that σ is a *model* of ϕ (written as $\sigma \models \phi$) iff $\sigma, 0 \models \phi$. The *language* of ϕ, denoted by $\mathcal{L}(\phi)$, is the set of traces $\sigma \in (2^\Sigma)^\omega$ such that $\sigma \models \phi$. We say that two formulas $\phi, \psi \in \mathsf{LTL{+}P}$ are *equivalent*, written $\phi \equiv \psi$, when, for all $\sigma \in (2^\Sigma)^\omega$, it holds that σ is a model of ϕ if and only if σ is a model of ψ.

Finally, given a set $\mathbb{L}$ of formulas in $\mathsf{LTL{+}P}$, a set of formulas $\mathbb{L}'$ either in $\mathsf{F}(\mathsf{pLTL})$ or in $\mathsf{G}(\mathsf{pLTL})$, and $k \in \mathbb{N}$, a *pastification from $\mathbb{L}$ into $\mathbb{L}'$ of k-exponential size* is an algorithm that, for any $\phi \in \mathbb{L}$, returns a formula $\psi \in \mathbb{L}'$, such that $\phi \equiv \psi$ and

$$|\psi| \in \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^{\mathcal{O}(|\phi|)}}}}}}}_{k \text{ times}}$$

It is known that there exists a pastification from $\mathsf{LTL}[\mathsf{X}]$ into $\mathsf{F}(\mathsf{pLTL})$ of 0-exponential size (Cimatti et al. 2021). Moreover, it follows from the results in (De Giacomo et al. 2021) that there exists a pastification from $\mathsf{coSafetyLTL}$ into $\mathsf{F}(\mathsf{pLTL})$ of 3-exponential size.

4 Transformation into Normal Form

In this section, we illustrate the first step of our pastification procedure. We define a normal form for $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ and we give a translation, based on the bottom-up application of a set of rewriting rules, from $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ into normal form. By means of a dedicated complexity analysis, we also show that the translation produces a formula of *singly exponential* size with respect to the original one, in the worst case.

4.1 The Normal Form of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$

We start with defining the normal form of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$, denoted as $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$, and the logic $\mathsf{LTL}[\mathsf{F},\wedge]$, on which the normal form is based.

Definition 1 (The logic $\mathsf{LTL}[\mathsf{F},\wedge]$)**.** *Let ψ be a pLTL formula. The logic $\mathsf{LTL}[\mathsf{F},\wedge]$ is the set of formulas ϕ generated by the following grammar:*

$$\phi \coloneqq \psi \mid \phi_1 \wedge \phi_2 \mid \mathsf{F}\phi$$

Algorithm 1 Algorithm $\mathtt{NF}$

procedure $\mathtt{NF}(\phi)$
  if $\phi \in \mathsf{pLTL}$ **then**
    return ϕ
  else if $\phi = \mathsf{X}^h(\psi)$ **then**
    $\mathsf{X}^k \bigvee_{i=1}^c \psi_i \coloneqq \mathtt{NF}(\psi)$
    return $\mathsf{X}^{h+k} \bigvee_{i=1}^c \psi_i$
  else if $\phi = \mathsf{F}(\psi)$ **then**
    $\mathsf{X}^k \bigvee_{i=1}^c \psi_i \coloneqq \mathtt{NF}(\psi)$
    return $\mathsf{X}^k \bigvee_{i=1}^c \mathsf{F}\psi_i$ ▷ Rules R_2, R_5
  else if $\phi = \bigvee_{i=1}^c(\psi_i)$ **then**
    $\mathsf{X}^{k_i} \bigvee_{j=1}^{d_i} \psi_{i,j} \coloneqq \mathtt{NF}(\psi_i)$ for each $1 \leq i \leq c$
    return $\mathsf{X}^{k_m} \bigvee_{i=1}^c \bigvee_{j=1}^{d_i} \mathtt{PUSH\_Y}(\mathsf{Y}^{k_m-k_i}\psi_{i,j})$
      where $k_m \coloneqq \max\{k_i\}_{i=1}^c$ ▷ Rule R_1
  else if $\psi = \bigwedge_{i=1}^c(\psi_i)$ **then**
    $\mathsf{X}^{k_i} \bigvee_{j=1}^{d_i} \psi_{i,j} \coloneqq \mathtt{NF}(\psi_i)$ for each $1 \leq i \leq c$
    $\mathsf{X}^{k_m} \bigwedge_{i=1}^c \bigvee_{j=1}^{d_i} \gamma_{i,j} \coloneqq \mathsf{X}^{k_m} \bigwedge_{i=1}^c \bigvee_{j=1}^{d_i}$
      $\mathtt{PUSH\_Y}(\mathsf{Y}^{k_m-k_i}\psi_{i,j})$
    where $k_m \coloneqq \max\{k_i\}_{i=1}^c$ ▷ Rule R_1
    return $\mathsf{X}^{k_m} \bigvee_{S \in A} \bigwedge_{\gamma \in S} \gamma$ ▷ Rule R_6
  else
    unreachable code
  end if
end procedure

Definition 2 (The normal form of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$)**.** *The normal form of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$, denoted with $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$, is the set of formulas of type $\mathsf{X}^k \bigvee_{i=1}^c \phi_i$, for some $k, c \in \mathbb{N}$, such that $\phi_i \in \mathsf{LTL}[\mathsf{F},\wedge]$ for any $1 \leq i \leq c$.*

In the general case, a formula of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ contains some uncertainty both on *which* eventualities have to happen and on *when* an eventuality has to be realized. Take for example the formula $\mathsf{F}(\bigwedge_{i=1}^c(p_i \rightarrow \mathsf{F}q_i))$, for some $c \in \mathbb{N}$: we don't know, *a priori*, neither which of the q_i are going to be fulfilled nor the order between the q_i. The normal form $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$ has been designed to move at top-level the uncertainty about *which* eventualities have to happen (this corresponds to the initial set of disjunctions). All formulas ϕ_i have thus only an uncertainty about *when* an eventuality is going to be fulfilled.

4.2 From $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ to $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$

We propose a transformation of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ into $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$ based on the following steps:

1. Pulling out all the *tomorrow* operators to top-level; this is done by the following rewriting rules:

   R_1. $\mathsf{X}^i\phi_1 \otimes \mathsf{X}^j\phi_2 \rightsquigarrow \begin{cases} \mathsf{X}^i(\phi_1 \otimes \mathsf{Y}^{i-j}\phi_2) & \text{if } i > j \\ \mathsf{X}^j(\mathsf{Y}^{j-i}\phi_1 \otimes \phi_2) & \text{otherwise} \end{cases}$

   R_2. $\mathsf{F}\mathsf{X}^i\phi_1 \rightsquigarrow \mathsf{X}^i\mathsf{F}\phi_1$

   for any $i, j \in \mathbb{N}$ and any $\otimes \in \{\wedge, \vee\}$.

2. Pushing in all the *yesterday* modalities in such a way that no F operator appears in the scope of a Y operator; this is done by the following rewriting rules:

Algorithm 2 Algorithm PUSH_Y

```
procedure PUSH_Y(φ)
    if φ ∈ pLTL then
        return φ
    else if φ = Y^k F(φ_1) then
        return F(PUSH_Y(Y^k φ_1))                 ▷ Rule R_4
    else if φ = Y^k ⋁_{i=1}^{c} φ_i then
        return ⋁_{i=1}^{c} PUSH_Y(Y^k φ_i)        ▷ Rule R_3
    else if φ = Y^k ⋀_{i=1}^{c} φ_i then
        return ⋀_{i=1}^{c} PUSH_Y(Y^k φ_i)        ▷ Rule R_3
    else
        unreachable code
    end if
end procedure
```

R_3: $\mathsf{Y}^i(\phi_1 \otimes \phi_2) \rightsquigarrow \mathsf{Y}^i\phi_1 \otimes \mathsf{Y}^i\phi_2$

R_4: $\mathsf{Y}^i\mathsf{F}\phi_1 \rightsquigarrow \mathsf{F}\mathsf{Y}^i\phi_1$

for any $i \in \mathbb{N}$ and any $\otimes \in \{\wedge, \vee\}$.

3. Pulling out all *disjunctions* in such a way that, for all subformulas ψ of type $\phi_1 \vee \phi_2$, *either* ψ is not in the scope of any F operator *or* both ϕ_1 and ϕ_2 contain no F modalities; this is done by the following rewriting rules:

   R_5: $\mathsf{F}(\bigvee_{i=1}^{c} \phi_i) \rightsquigarrow \bigvee_{i=1}^{c} \mathsf{F}\phi_i$

   R_6: $\bigwedge_{i=1}^{c} \bigvee_{j=1}^{d_i} \phi_{i,j} \rightsquigarrow \bigvee_{S \in A} \bigwedge_{\psi \in S} \psi$

   for some $c, d_1, \ldots, d_m \in \mathbb{N}$, where, for any $1 \leq i \leq c$, $C_i = \{\phi_{i,1}, \ldots, \phi_{i,d_i}\}$ and $A \coloneqq \{\{\psi_1, \ldots, \psi_m\} \mid \psi_i \in C_i, \forall 1 \leq i \leq d_i\}$. Rule R_6 corresponds to the transformation into *disjunctive normal form* (DNF).

The previous rewriting rules are arranged into Algorithm 1, which implements the transformation of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ into normal form: it applies rules R_1, R_2, R_5, and R_6 in a bottom-up fashion, and calls Algorithm 2 for the top-down application of rules R_3 and R_4. Since the rewriting steps lead to equivalent formulas, we obtain the following (the proof is straightforward and thus omitted).

Lemma 1. *For any formula $\phi \in \mathsf{LTL}[\mathsf{X},\mathsf{F}]$, Algorithm 1 returns a formula ϕ' such that $\phi' \equiv \phi$ and $\phi' \in \mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$.*

4.3 Analysis of the Complexity of Algorithm 1

Algorithm 1 deserves a dedicated complexity analysis of the size of the resulting formula. Clearly, at each iteration, the size of the output formula is dominated by the application of rule R_6, which corresponds to computing the DNF of a formula. It follows that the worst case for Algorithm 1 is when the input formula ϕ has the following form:

$$\mathsf{F}(\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l} \ldots \mathsf{F}(\bigwedge_{i_2=1}^{c_2} \bigvee_{j_2=1}^{d_2} \mathsf{F}(\bigwedge_{i_1=1}^{c_1} \bigvee_{j_1=1}^{d_1} \phi_{i_1,j_1,\ldots,i_l,j_l})) \ldots)$$

for some $l \in \mathbb{N}$ and some $c_1, d_1, \ldots, c_l, d_l \in \mathbb{N}$, where $\phi_{i_1,j_1,\ldots,i_l,j_l}$ is a *literal*, for each $i_1, j_1, \ldots, i_l, j_l \in \mathbb{N}$.

First, we show that l, that is the number of alternations in ϕ, is *logarithmic* in the size of ϕ. Let $n = |\phi|$ and let ε be the smallest among $c_1, \ldots, c_l, d_1, \ldots, d_l$. We have that: $n \in \Omega(c_1 \cdot d_1 \cdot \ldots \cdot c_l \cdot d_l)$, so $n \geq \varepsilon^{2\cdot l}$. Thus, $l \in \log_\varepsilon(\mathcal{O}(n))$.

Algorithm 1, applied to the formula ϕ of above, executes rules R_5-R_6 exactly l times. It is worth writing the *size* of the output formula for each of the l executions of rules R_5-R_6. In the first execution of rules R_5-R_6, the algorithm performs the following transformations, respectively (we indicate with grey boxes those portions of the formula that change at each step):

$$\mathsf{F}(\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l} \ldots \mathsf{F}(\bigwedge_{i_2=1}^{c_2} \bigvee_{j_2=1}^{d_2} \mathsf{F}(\boxed{\bigwedge_{i_1=1}^{c_1} \bigvee_{j_1=1}^{d_1}} \phi_{i_1,j_1,\ldots,i_l,j_l}) \ldots))$$

$$\mathsf{F}(\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l} \ldots \mathsf{F}(\bigwedge_{i_2=1}^{c_2} \boxed{\bigvee_{j_2=1}^{d_2} \mathsf{F}(\bigvee_{i_1=1}^{(d_1)^{c_1}}} \bigwedge_{j_1=1}^{c_1} \psi^1_{i_1,j_1,\ldots,i_l,j_l})) \ldots))$$

$$\mathsf{F}(\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l} \ldots \mathsf{F}(\bigwedge_{i_2=1}^{c_2} \boxed{\bigvee_{j_2=1}^{d_2+(d_1)^{c_1}} \mathsf{F}} (\bigwedge_{j_1=1}^{c_1} \psi^1_{j_1,\ldots,i_l,j_l})) \ldots)$$

for some *literals* $\psi^1_{i_1,j_1,\ldots,i_l,j_l}$ and $\psi^1_{j_1,\ldots,i_l,j_l}$, for any $i_1, j_1, \ldots, i_l, j_l$ in their respective range. In the second execution of rules R_5-R_6, the algorithm performs the following transformations, respectively:

$$\mathsf{F}(\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l} \ldots \mathsf{F}(\boxed{\bigwedge_{i_2=1}^{c_2} \bigvee_{j_2=1}^{d_2+(d_1)^{c_1}}} \mathsf{F}(\bigwedge_{j_1=1}^{c_1} \psi^1_{j_1,i_2,j_2,\ldots,i_l,j_l})) \ldots)$$

$$\mathsf{F}(\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l} \ldots \boxed{\mathsf{F}(\bigvee_{i_2=1}^{(d_2+(d_1)^{c_1})^{c_2}}} \bigwedge_{j_2=1}^{c_2} \mathsf{F}(\bigwedge_{j_1=1}^{c_1} \psi^2_{j_1,i_2,j_2,\ldots,i_l,j_l})) \ldots)$$

$$\mathsf{F}(\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l} \ldots \boxed{\bigvee_{i_3=1}^{d_3+(d_2+(d_1)^{c_1})^{c_2}}} \mathsf{F}(\bigwedge_{j_2=1}^{c_2} \mathsf{F}(\bigwedge_{j_1=1}^{c_1} \psi^2_{j_1,j_2,\ldots,i_l,j_l})) \ldots)$$

for some *literals* $\psi^1_{j_1,i_2,j_2\ldots,i_l,j_l}$ and $\psi^2_{j_1,j_2,\ldots,i_l,j_l}$, for any $j_1, i_2, j_2, \ldots, i_l, j_l$ in their respective range. In the *last* execution of rules R_5-R_6, the algorithm performs the following transformations, respectively:

$$\mathsf{F}(\boxed{\bigwedge_{i_l=1}^{c_l} \bigvee_{j_l=1}^{d_l+(d_{l-1}+(\cdots+(d_1)^{c_1})\cdots)^{c_{l-1}}}} \ldots \mathsf{F}(\bigwedge_{j_1=1}^{c_1} \psi^{l-1}_{j_1,\ldots,i_l,j_l}) \ldots))$$

$$\boxed{\mathsf{F}(\bigvee_{i_l=1}^{(d_l+(\cdots+(d_1)^{c_1})\cdots)^{c_l}}} \bigwedge_{j_l=1}^{c_l} \ldots \mathsf{F}(\bigwedge_{j_1=1}^{c_1} \psi^{l}_{j_1,\ldots,i_l,j_l}) \ldots))$$

$$\boxed{\bigvee_{i_l=1}^{(d_l+(\cdots+(d_1)^{c_1})\cdots)^{c_l}}} \mathsf{F}(\bigwedge_{j_l=1}^{c_l} \ldots \mathsf{F}(\bigwedge_{j_1=1}^{c_1} \psi^{l}_{j_1,\ldots,i_l,j_l}) \ldots))$$

for some *literals* $\psi^{l-1}_{j_1,\ldots,i_l,j_l}$ and $\psi^{l}_{j_1,\ldots,i_l,j_l}$, for any $j_1, i_2, j_2, \ldots, i_l, j_l$ in their respective range.

We now estimate the size of the last formula. Let d be the greatest among $\{d_i\}_{i=1}^{l}$. The function $d_l + (\cdots + (d_1)^{c_1})^{\cdots})^{c_l}$ is less than $(d + (\ldots + (d)^{c_1})^{\cdots})^{c_l}$, and in turn less than $(d \cdot (\ldots \cdot (d)^{c_1})^{\cdots})^{c_l}$, obtained by replacing sums with multiplications. It follows that the size of the formula resulting from the last execution of rules R_5-R_6 is less then:

$$(d)^{l \cdot \prod_{i=1}^{l} c_i} \cdot \prod_{i=1}^{l} c_i$$

Let c be the greatest among $\{c_i\}_{i=1}^{l}$. Since c is a *constant*, $l \in \log_\varepsilon(\mathcal{O}(n))$, and $\log_\varepsilon(\mathcal{O}(n)) = \frac{\log_c(\mathcal{O}(n))}{\log_c(\varepsilon)}$, we have that $l \in \log_c(\mathcal{O}(n))$, and therefore:

$$\prod_{i=1}^{l} c_i \le \prod_{i=1}^{l} c = c^l \in c^{\log_c(\mathcal{O}(n))} \in \mathcal{O}(n)$$

Since d (which is the greatest among $\{d_i\}_{i=1}^{l}$) is a *constant*, we have that:

$$(d)^{l \cdot \prod_{i=1}^{l} c_i} \cdot \prod_{i=1}^{l} c_i \le (d)^{\log_c(\mathcal{O}(n)) \cdot \mathcal{O}(n)} \cdot \mathcal{O}(n) \le (d)^{\mathcal{O}(n^2)}.$$

Therefore, in the worst case, the size of the formula resulting from the application the rules R_5-R_6 is at most *singly exponential* in the size of the starting formula.

5 Transformation into $\mathsf{F}(\mathsf{pLTL})$

In this section, we present the second main step required by our pastification technique. We first define *dependency trees*, which are labelled trees used to represent the temporal interplay between "eventualities" (*i.e.*, subformulas in the scope of an F operator) in an $\mathsf{LTL}[\mathsf{F},\wedge]$ formula. Then, combining both the translation in normal form presented in the previous section, and the machinery provided by the dependency trees, we show how to construct a translation of an $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$ formula into $\mathsf{F}(\mathsf{pLTL})$.

5.1 From Normal Form to Dependency Trees

We consider first a formula $\phi \in \mathsf{LTL}[\mathsf{F},\wedge]$, which constitutes the basic building block of the normal form $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$. By construction, we can assume without loss of generality that ϕ is of the form $\alpha \wedge \mathsf{F}(\beta_1) \wedge \cdots \wedge \mathsf{F}(\beta_n)$, for some $n \in \mathbb{N}$, where $\alpha \in \mathsf{pLTL}$ (*i.e.*, α is a pure-past formula, including $\top$) and $\beta_i \in \mathsf{LTL}[\mathsf{F},\wedge]$, for each $1 \le i \le n$.

For such a formula $\phi \in \mathsf{LTL}[\mathsf{F},\wedge]$, we define a tree-shaped structure, called *dependency tree for* ϕ, that reflects the nesting of the F operators in ϕ. In particular, each node of the tree represents what ϕ enforces at a certain instant, while an edge captures the temporal connection between what holds at a given node of the tree and the eventualities that have to be fulfilled in its future. However, whenever a conjunction of multiple eventualities has to be realised in the future of a given node, the tree branches without imposing any ordering among them.

Definition 3. *Let* $\phi \in \mathsf{LTL}[\mathsf{F},\wedge]$. *The* dependency tree of ϕ *is a tree* $T = (V, E, r, \mu, \nu)$*, with:*

- *set of* nodes V,
- *set of* edges E,
- *a distinguished* root $r \in V$,
- labelling functions $\mu : V \to \mathsf{LTL}[\mathsf{F},\wedge]$ *and* $\nu : V \to \mathsf{LTL}[\mathsf{F},\wedge]$,

and such that V and E are the minimal sets respecting the following conditions:

- $r \in V$ *and* $\mu(r) = \phi$;
- *for any node* $v \in V$*, if* $\mu(v) = \alpha \wedge \mathsf{F}(\beta_1) \wedge \cdots \wedge \mathsf{F}(\beta_n)$ *(where* $\alpha \in \mathsf{pLTL}$*,* $n \in \mathbb{N}$*, and* $\beta_i \in \mathsf{LTL}[\mathsf{F},\wedge]$*, for each* $1 \le i \le n$*), then:*
  - $\nu(v) = \alpha$;
  - v *has* n *children* $v_1, \ldots, v_n \in V$ *such that* $\mu(v_i) = \beta_i$*, for all* $1 \le i \le n$.

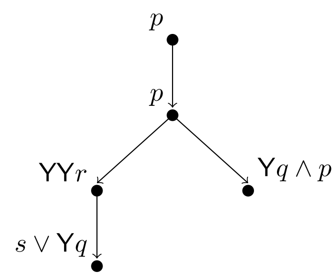


Figure 1: Example of dependency tree for the formula $p \wedge \mathsf{F}(p \wedge \mathsf{F}(\mathsf{Y}q \wedge p) \wedge \mathsf{F}(\mathsf{YY}r \wedge \mathsf{F}(s \vee \mathsf{Y}q)))$. For sake of clarity, only the ν labeling function is depicted.

Observe that such a dependency tree for ϕ, in the worst case, has size at most linear in the size of ϕ. From now on, given a dependency tree $T = (V, E, r, \mu, \nu)$, we denote with $\Pi(T)$ the set of *paths from the root to a leaf of* T, that is, sequences $\pi = \langle \pi_1, \ldots, \pi_n \rangle$, for some $n \in \mathbb{N}$, where: $r = \pi_1$; for every $1 \le i \le n-1$, $(\pi_i, \pi_{i+1}) \in E$; and $(\pi_n, v) \notin E$, for every $v \in V$.

Example 1. *Figure 1 shows the dependency tree for the formula* $\phi := p \wedge \mathsf{F}(p \wedge \mathsf{F}(p \wedge \mathsf{Y}q) \wedge \mathsf{F}(\mathsf{YY}r \wedge \mathsf{F}(s \vee \mathsf{Y}q)))$.

5.2 From Dependency Trees to $\mathsf{F}(\mathsf{pLTL})$

Consider a formula of $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$, which by definition is of the following type: $\mathsf{X}^k \bigvee_{i=1}^{c} \phi_i$, for some $k, c \in \mathbb{N}$, where ϕ_i belongs to $\mathsf{LTL}[\mathsf{F},\wedge]$, for each $1 \le i \le c$. Starting from the dependency tree of any $\phi_i \in \mathsf{LTL}[\mathsf{F},\wedge]$, our first goal is to construct a pure past formula ψ_i^k that is equivalent to ϕ_i when ϕ_i is interpreted exactly at time point k (which is the case relevant to us, due to the X^k preceding $\bigvee_{i=1}^{c} \phi_i$).

To illustrate the idea behind this transformation, let ϕ be an $\mathsf{LTL}[\mathsf{F},\wedge]$ formula with dependency tree T. By considering separately each path of T that goes from the root to a leaf, we can "rewrite" each branch upside-down (*i.e.*, going from the leaf to the root), by means of a formula that uses only the past modalities *once* (O), *weak yesterday* ($\widetilde{\mathsf{Y}}$), and *yesterday* (Y), as temporal operators, and that appropriately reverses the nesting of eventualities on that branch. Such formulas (one for each reversed path) will coincide in the description of the "common past", which is the portion of the tree shared by all the branches up to the root of T. Moreover, their conjunction guarantees that *all and only* those orders imposed by the original $\mathsf{LTL}[\mathsf{F},\wedge]$ formula are captured.

The transformation, defined below, is parameterized with respect to a number $k \in \mathbb{N}$, which is supposed to represent the number of nested *tomorrow* operators in the original $\mathsf{nfLTL}[\mathsf{X},\mathsf{F}]$ formula $\mathsf{X}^k \bigvee_{i=1}^{c} \phi_i$. As an auxiliary notion, for any $k \in \mathbb{N}$, let at_k be the formula $\widetilde{\mathsf{Y}}^{k+1}\bot \wedge \mathsf{Y}^k\top$. Clearly, for any state sequence σ and any $i \in \mathbb{N}$, it holds that $\sigma, i \models \mathsf{at}_k$ iff $i = k$.

Definition 4. *Let ϕ be a formula of $\mathsf{LTL}[\mathsf{F}, \wedge]$, let $T = (V, E, r, \mu, \nu)$ be its dependency tree, and let $k \in \mathbb{N}$. Given a path $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ from the root to a leaf of T, we inductively define the formula ${}^k\langle\!\langle \phi \rangle\!\rangle^i_\pi$ as follows, for each $1 \leq i \leq n$:*

$$ {}^k\langle\!\langle \phi \rangle\!\rangle^i_\pi = \begin{cases} \mathsf{O}(\nu(\pi_1) \wedge \mathsf{at}_k) & \text{if } i = 1 \\ \mathsf{O}(\nu(\pi_i) \wedge {}^k\langle\!\langle \phi \rangle\!\rangle^{i-1}_\pi) & \text{otherwise} \end{cases} $$

In particular, the formula ${}^k\langle\!\langle \phi \rangle\!\rangle^n_\pi$ corresponds to the formula of the whole branch $\pi = \langle \pi_1, \ldots, \pi_n \rangle$.

For any $k \in \mathbb{N}$, we define $\mathsf{past}(\phi, k)$ as the $\mathsf{F}(\mathsf{pLTL})$ formula equivalent to the $\mathsf{LTL}[\mathsf{F}, \wedge]$ formula ϕ, when ϕ is interpreted at time point k. It is obtained by conjuctively relating the pure past formulas corresponding to each branch (from root to a leaf) of the dependency tree T of ϕ, as follows:

$$ \mathsf{past}(\phi, k) = \mathsf{F}\Big(\bigwedge_{\substack{\pi \in \Pi(T). \\ \pi = \langle \pi_1, \ldots, \pi_n \rangle}} {}^k\langle\!\langle \phi \rangle\!\rangle^n_\pi \Big) $$

Example 2. *Given ϕ as in Example 1, for any $k \in \mathbb{N}$, $\mathsf{past}(\phi, k)$ is the following formula:*

$$ \mathsf{F}\big(\mathsf{O}((s \vee \mathsf{Y}q) \wedge \mathsf{O}(\mathsf{YY}r \wedge \mathsf{O}(p \wedge \mathsf{O}(p \wedge \mathsf{at}_k)))) \wedge \\ \mathsf{O}((p \wedge \mathsf{Y}q) \wedge \mathsf{O}(p \wedge \mathsf{O}(p \wedge \mathsf{at}_k)))\big) $$

Example 3. *Consider the $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$ formula $\phi := \mathsf{F}(p_0 \wedge (\mathsf{F}q_1 \wedge \mathsf{F}q_2))$. In this case, ϕ requires a time point where p_0 holds followed by a time point in which q_1 holds and by a time point in which q_2 holds. The formula $\mathsf{past}(\phi, 0)$ is $\mathsf{F}(\mathsf{O}(q_1 \wedge \mathsf{O}p_0) \wedge \mathsf{O}(q_2 \wedge \mathsf{O}p_0))$ and it captures all and only the models of ϕ. For example, the trace $\langle \{p_0\}, \{q_1\}, \{p_0\}, \{q_2\} \rangle$ is a model of $\mathsf{past}(\phi, 0)$ as well as of ϕ, since the* first *time point where ϕ holds (in this case 0) suffices to satisfy ϕ.*

The next lemma shows that, for any $\phi \in \mathsf{LTL}[\mathsf{F}, \wedge]$ and for any $k \in \mathbb{N}$, the formula ϕ is equivalent to $\mathsf{past}(\phi, k)$ when ϕ is interpreted at time point k.

Lemma 2. *For any $k \in \mathbb{N}$, and for any $\phi \in \mathsf{LTL}[\mathsf{F}, \wedge]$, it holds that $\mathsf{X}^k\phi \equiv \mathsf{past}(\phi, k)$.*

Proof. Let $k \in \mathbb{N}$. We prove that $\mathsf{X}^k\phi \equiv \mathsf{past}(\phi, k)$ by induction on the number f of nested *eventually* (F) operators in ϕ (which corresponds also to the height of the dependency tree of ϕ).

Base case. If $f = 0$, then $\phi := \mathsf{X}^k\alpha$ with $\alpha \in \mathsf{pLTL}$ (α is a pure-past LTL+P formula). By definition, the dependency tree of α is made of only its root r and it is such that $\nu(r) = \mu(r) = \alpha$. For all state sequences σ, it holds that:

$$ \begin{aligned} \sigma, 0 \models \mathsf{X}^k\alpha &\Leftrightarrow \sigma, k \models \alpha \\ &\Leftrightarrow \exists j \geq 0 \,.\, \exists i \leq j \,.\, (\sigma, i \models \alpha \wedge i - k = 0) \\ &\Leftrightarrow \sigma, 0 \models \mathsf{F}(\mathsf{O}(\alpha \wedge \mathsf{at}_k)) \\ &\Leftrightarrow \mathsf{past}(\phi, k) \end{aligned} $$

Therefore, $\mathsf{X}^k\phi \equiv \mathsf{past}(\phi, k)$.

Inductive step. If $f > 0$, then $\phi = \alpha \wedge \mathsf{F}(\beta_1) \wedge \cdots \wedge \mathsf{F}(\beta_m)$ for some $m \in \mathbb{N}$. We prove that $\sigma \models \mathsf{X}^k\phi$ iff $\sigma \models \mathsf{past}(\phi, k)$, for all state sequence σ. We have that $\sigma, 0 \models \mathsf{X}^k(\alpha \wedge \mathsf{F}(\beta_1) \wedge \cdots \wedge \mathsf{F}(\beta_m))$ iff:

$$ (\sigma, k \models \alpha) \wedge (\exists j_1 \geq k \,.\, \sigma, j_1 \models \beta_1) \wedge \ldots \\ \wedge (\exists j_m \geq k \,.\, \sigma, j_m \models \beta_m) $$

which in turn is equivalent to:

$$ (\sigma, k \models \alpha) \wedge (\exists j_1 \geq k \,.\, \sigma^{j_1}, 0 \models \beta_1) \wedge \ldots \\ \wedge (\exists j_m \geq k \,.\, \sigma^{j_m}, 0 \models \beta_m) $$

where σ^{j_i} is the state sequence corresponding to the suffix of σ starting at j_i, for all $1 \leq i \leq m$. Since the number of nested F in β_i (for each $1 \leq i \leq m$) is strictly smaller than the number of nested F in ϕ, by inductive hypothesis we have that $\beta_i \equiv \mathsf{past}(\beta_i, 0)$, for all $1 \leq i \leq m$. Therefore:

$$ (\sigma, k \models \alpha) \wedge (\exists j_1 \geq k \,.\, \sigma^{j_1}, 0 \models \mathsf{past}(\beta_1, 0)) \wedge \ldots \\ \wedge (\exists j_m \geq k \,.\, \sigma^{j_m}, 0 \models \mathsf{past}(\beta_m, 0)) $$

We now focus on a generic conjunct of the formula above, say, the one with index i, for some $1 \leq i \leq m$. Let T_{β_i} be the dependency tree of β_i. The height of T_{β_i} is at most $f - 1$, *i.e.*, all paths in $\Pi(T_{\beta_i})$ are long at most $f - 1$. Without loss of generality, we suppose that are all of length $f - 1$. By definition of the formula $\mathsf{past}(\beta_i, 0)$, we have that:

$$ \sigma^{j_i}, 0 \models \mathsf{F}\Big(\bigwedge_{\substack{\langle \pi_1, \ldots, \pi_{f-1} \rangle \\ \in \Pi(T_{\beta_i})}} \mathsf{O}(\nu(\pi_{f-1}) \wedge \mathsf{O}(\cdots \wedge \mathsf{O}(\nu(\pi_1) \wedge \mathsf{at}_0)))\Big) $$

Thus, we can say that $\sigma^{j_i}, 0 \models \mathsf{past}(\beta_i, 0)$ iff:

$$ \exists j'_i \geq 0. \bigwedge_{\substack{\langle \pi_1, \ldots, \pi_{f-1} \rangle \\ \in \Pi(T_{\beta_i})}} \exists h_{f-1} \leq j'_i . \exists h_{f-2} \leq h_{f-1} . \ldots . \exists h_1 \leq h_2 . \\ (\sigma^{j_i}, h_{f-1} \models \nu(\pi_{f-1}) \wedge \cdots \wedge \sigma^{j_i}, h_1 \models \nu(\pi_1) \wedge \\ h_1 = 0) $$

It follows that, for any $1 \leq i \leq m$, $(\sigma, k \models \alpha) \wedge (\exists j_i \geq k \,.\, \sigma^{j_i}, 0 \models \mathsf{past}(\beta_i, 0))$ is true iff:

$$ \exists j_i \geq 0 \,.\, \bigwedge_{\substack{\langle \pi_1, \ldots, \pi_{f-1} \rangle \\ \in \Pi(T_{\beta_i})}} \exists h_{f-1} \leq j_i \,.\, \exists h_{f-2} \leq h_{f-1} \ldots \exists h_0 \leq h_1 . \\ (\sigma, h_{f-1} \models \nu(\pi_{f-1}) \wedge \cdots \wedge \sigma, h_1 \models \nu(\pi_1) \wedge \\ \wedge\, \sigma, h_0 \models \alpha \wedge h_0 = k) $$

Then $(\sigma, k \models \alpha) \wedge (\exists j_1 \geq k \,.\, \sigma^{j_1}, 0 \models \beta_1) \wedge \cdots \wedge (\exists j_m \geq k \,.\, \sigma^{j_m}, 0 \models \beta_m)$ is true iff:

$$ \exists j_1 \geq 0. \bigwedge_{\substack{\langle \pi_1, \ldots, \pi_{f-1} \rangle \\ \in \Pi(T_{\beta_1})}} \exists h_{f-1} \leq j_1 . \exists h_{f-2} \leq h_{f-1} \ldots \exists h_0 \leq h_1 . \\ (\sigma, h_{f-1} \models \nu(\pi_{f-1}) \wedge \cdots \wedge \sigma, h_1 \models \nu(\pi_1) \wedge \\ \wedge \sigma, h_0 \models \alpha \wedge h_0 = k) \wedge \\ \wedge \cdots \wedge \\ \exists j_m \geq 0. \bigwedge_{\substack{\langle \pi_1, \ldots, \pi_{f-1} \rangle \\ \in \Pi(T_{\beta_m})}} \exists h_{f-1} \leq j_m . \exists h_{f-2} \leq h_{f-1} . \ldots . \exists h_0 \leq h_1 . \\ (\sigma, h_{f-1} \models \nu(\pi_{f-1}) \wedge \ldots \wedge \sigma, h_1 \models \nu(\pi_1) \wedge \\ \wedge \sigma, h_0 \models \alpha \wedge h_0 = k) $$

Since the variables $j_1, \ldots, j_m$ appear only as guards for the quantified variable h_{f-1}, the existential quantification on $j_1, \ldots, j_m$ can be factorized, obtaining the following:

$$
\begin{aligned}
&\exists j \geq 0 \ . \ (\\
&\quad \bigwedge_{\substack{\langle \pi_1,\ldots,\pi_{f-1}\rangle \\ \in \Pi(T_{\beta_1})}} \exists h_{f-1} \leq j . \exists h_{f-2} \leq h_{f-1} \ldots . \exists h_0 \leq h_1 . \\
&\qquad\qquad (\sigma, h_{f-1} \models \nu(\pi_{f-1}) \wedge \cdots \wedge \sigma, h_1 \models \nu(\pi_1) \wedge \\
&\qquad\qquad\quad \wedge \sigma, h_0 \models \alpha \wedge h_0 = k) \wedge \\
&\quad \wedge \cdots \wedge \\
&\quad \bigwedge_{\substack{\langle \pi_1,\ldots,\pi_{f-1}\rangle \\ \in \Pi(T_{\beta_m})}} \exists h_{f-1} \leq j . \exists h_{f-2} \leq h_{f-1} \ldots . \exists h_0 \leq h_1 . \\
&\qquad\qquad (\sigma, h_{f-1} \models \nu(\pi_{f-1}) \wedge \cdots \wedge \sigma, h_1 \models \nu(\pi_1) \wedge \\
&\qquad\qquad\quad \wedge \sigma, h_0 \models \alpha \wedge h_0 = k) \\
&)
\end{aligned}
$$

Since, by definition, the dependency tree T_ϕ of ϕ is made of its root node whose children are the root nodes of the dependency tree of $\beta_1, \ldots, \beta_m$, we have that for each path $\langle \pi_1, \ldots, \pi_f \rangle$ in $\Pi(T_\phi)$ there exists an $1 \leq i \leq m$ and a path $\langle \pi'_1, \ldots, \pi'_{f-1} \rangle$ in $\Pi(T_{\beta_i})$ such that $\nu(\pi_1) = \alpha$, and $\nu(\pi_l) = \nu(\pi'_{l-1})$, for each $1 < l \leq f$ (the *vice versa* holds as well). In other words, the paths in $\Pi(T_\phi)$ are the paths in $\Pi(T_{\beta_i})$ prefixed by the root of T_ϕ. Therefore, up to a renaming of the quantified variables, we have:

$$
\begin{aligned}
\exists j \geq 0 . \bigwedge_{\substack{\langle \pi_1,\ldots,\pi_f\rangle \\ \in \Pi(T_\phi)}} & \exists h_f \leq j . \exists h_{f-1} \leq h_f \ldots . \exists h_1 \leq h_2 . \\
& (\sigma, h_f \models \nu(\pi_f) \wedge \cdots \wedge \\
& \wedge \sigma, h_1 \models \alpha \wedge h_1 = k)
\end{aligned}
$$

Thus, we showed that, if $\sigma \models \mathsf{X}^k \phi$ iff:

$$
\sigma, 0 \models \mathsf{F}(\bigwedge_{\substack{\langle \pi_1,\ldots,\pi_f\rangle \\ \in \Pi(T_\phi)}} \mathsf{O}(\nu(\pi_f) \wedge \mathsf{O}(\cdots \wedge \mathsf{O}(\nu(\pi_1) \wedge \mathsf{at}_k))))
$$

Then, by definition of $\mathsf{past}(\cdot, \cdot)$, $\sigma, 0 \models \mathsf{past}(\phi, k)$. $\square$

5.3 Analysis of the Size of $\mathsf{past}(\phi, k)$

For any $\phi \in \mathsf{LTL}[\mathsf{F}, \wedge]$ and any $k \in \mathbb{N}$, we show that the size of $\mathsf{past}(\phi, k)$ is at most *quadratic* in the size of ϕ and linear in k. In particular, we will show that $|\mathsf{past}(\phi, k)| \in \mathcal{O}(n^2 + k \cdot n)$, where $n = |\phi|$. Let T_ϕ be the dependency tree of ϕ. By definition of $\mathsf{past}(\cdot, \cdot)$, we have that:

$$
|\mathsf{past}(\phi, k)| = \sum_{\substack{\pi = \langle \pi_1,\ldots,\pi_f\rangle \\ \in \Pi(T_\phi)}} \Big(\mathcal{O}(k) + \sum_{1 \leq j \leq f} (|\nu(\pi_j)| + \mathcal{O}(1)) \Big)
$$

The task now is to estimate the number of paths π in T_ϕ from the root to a leaf, as well as the dimension of the labels $\nu(v)$ of the nodes v in such paths. In order to do that, we notice that, since any node in T_ϕ is labeled by ν with a different occurrence of a subformula of ϕ with respect to any other node in T_ϕ, this implies that:

- the number of nodes of T_ϕ is less than the number of subformulas of ϕ, and thus also less than the number of characters of ϕ, *i.e.*, n;
- $\sum_{v \in V_\phi} |\nu(v)| \leq n$, where V_ϕ is the set of nodes of T_ϕ.

This implies that the *number of paths* from the root to each leaf is less than n and that $\Sigma_{1 \leq i \leq f} |\nu(\pi_i)| \leq n$, for any path $\langle \pi_1, \ldots, \pi_{f-1} \rangle \in \Pi(T_\phi)$, having that:

$$
\begin{aligned}
|\mathsf{past}(\phi, k)| &= \sum_{\substack{\pi = \langle \pi_1,\ldots,\pi_f\rangle \\ \in \Pi(T_\phi)}} \Big(k + \sum_{1 \leq j \leq f} (|\nu(\pi_j)| + \mathcal{O}(1)) \Big) \\
&\in \mathcal{O}(n^2 + k \cdot n)
\end{aligned}
$$

5.4 Putting the Results Together

Given a formula $\mathsf{X}^k \bigvee_{i=1}^c \phi_i$ of $\mathsf{nfLTL}[\mathsf{X}, \mathsf{F}]$, the last remaining bit is to combine the translation of each $\phi_i \in \mathsf{LTL}[\mathsf{F}, \wedge]$, shown in the previous part, to obtain an equivalent formula in $\mathsf{F}(\mathsf{pLTL})$. By the distributive property of the *tomorrow* and *eventually* modalities and by Lemma 2, for every $1 \leq i \leq c$, we have the following equivalences:

$$
\mathsf{X}^k \bigvee_{i=1}^c \phi_i \ \equiv \ \bigvee_{i=1}^c \mathsf{X}^k \phi_i \ \equiv \ \bigvee_{i=1}^c \mathsf{past}(\phi_i, k) \ \equiv \ \mathsf{F} \bigvee_{i=1}^c \psi_i
$$

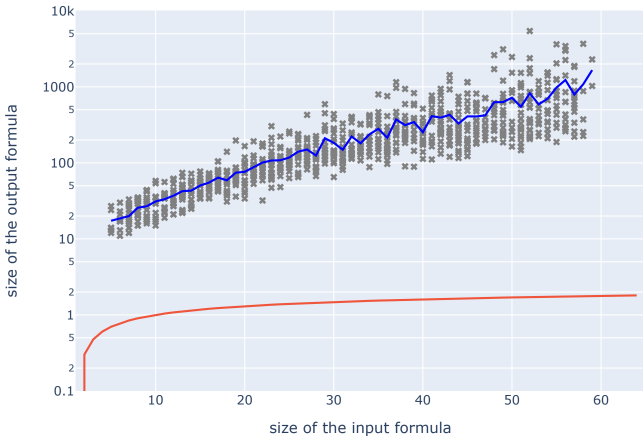
where $\mathsf{past}(\phi_i, k) = \mathsf{F}\psi_i$, for some $\psi_i \in \mathsf{pLTL}$. The formula $\mathsf{F} \bigvee_{i=1}^c \psi_i$ is the output of our entire pastification procedure, and it clearly belongs to $\mathsf{F}(\mathsf{pLTL})$.

We now estimate the size of $\mathsf{F} \bigvee_{i=1}^c \psi_i$. Let $n = |\mathsf{X}^k \bigvee_{i=1}^c \phi_i|$, and let $n_i = |\phi_i|$, for each $1 \leq i \leq c$. By Section 5.3, $|\psi_i| \in \mathcal{O}((n_i)^2 + k \cdot n_i)$. We have that:

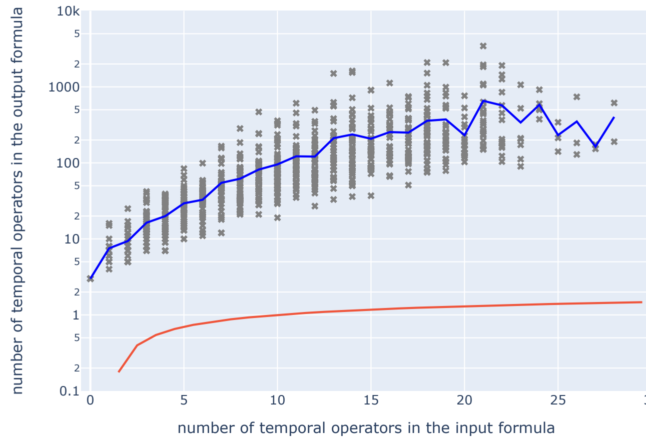
$$
\begin{aligned}
|\mathsf{F} \bigvee_{i=1}^c \psi_i| &= \sum_{i=1}^c |\psi_i| + \mathcal{O}(1) \\
&= \sum_{i=1}^c \mathcal{O}((n_i)^2 + k \cdot n_i) \\
&= \sum_{i=1}^c \mathcal{O}((n_i)^2) + \sum_{i=1}^c \mathcal{O}(k \cdot n_i) \\
&\leq \mathcal{O}((\sum_{i=1}^c n_i)^2) + \mathcal{O}(k) \cdot \sum_{i=1}^c \mathcal{O}(n_i) \\
&\leq \mathcal{O}(n^2) \qquad \text{since } k \leq n \text{ and } \sum_{i=1}^c n_i \leq n
\end{aligned}
$$

The transformation of any $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$ formula ϕ into an equivalent one in $\mathsf{F}(\mathsf{pLTL})$ works as follows: first, it applies the transformation decribed in Section 4 to translate ϕ into normal form, and then it goes from normal form to pure past LTL as described in this section. From Lemmas 1 and 2 and from the previous complexity analysis of all the steps, this theorem follows.

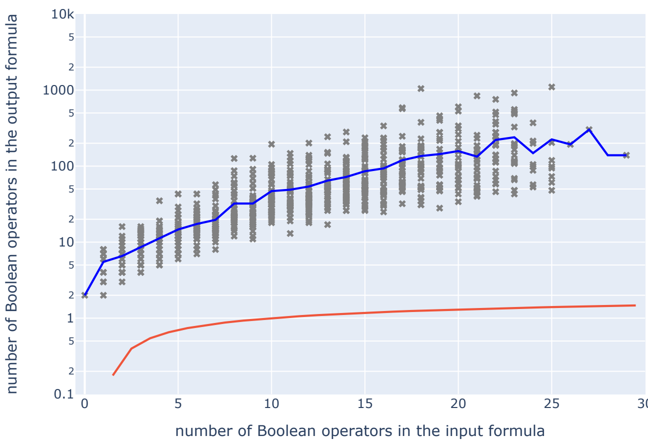
Theorem 1. *There exists a pastification from* $\mathsf{LTL}[\mathsf{X}, \mathsf{F}]$ *into* $\mathsf{F}(\mathsf{pLTL})$ *of* 1*-exponential size.*



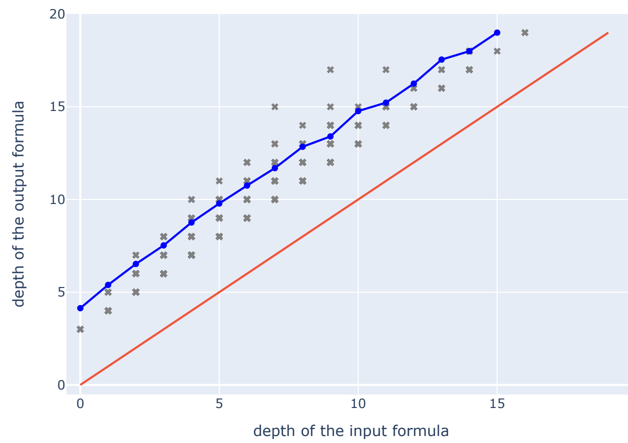
(a) Plot for the size.



(b) Plot for the number of occurrences of temporal operators.



(c) Plot for the number of occurrences of Boolean operators.



(d) Plot for the depth.

Figure 2: Plots for Section 6. The x-axis is for the input formula while the y-axis is for the output formula. The orange line is the diagonal, and the blue line interpolates the mean value among the output formulas correspoding to an input of a given size.

By dualization of the *eventually* operator F into the *always* operator G, and given the equivalence between the $\mathsf{X}\phi$ and $\neg\mathsf{X}\neg\phi$, we obtain a similar result for the *safety* fragment $\mathsf{LTL}[\mathsf{X},\mathsf{G}]$.

Corollary 1. *There exists a pastification from* $\mathsf{LTL}[\mathsf{X},\mathsf{G}]$ *into* $\mathsf{G}(\mathsf{pLTL})$ *of* 1*-exponential size.*

We point out that the translation from dependency trees to $\mathsf{F}(\mathsf{pLTL})$ produces formulas of type $\mathsf{F}(\alpha)$ where α is a pLTL formula in negation normal form devoid of the *since* (S) operator. By duality, it follows that the pastification of $\mathsf{LTL}[\mathsf{X},\mathsf{G}]$ into $\mathsf{G}(\mathsf{pLTL})$ is such that the inner pLTL formula is in negation normal form and it does not contain any *trigger* (T) operator.

6 Implementation

We implemented the algorithm described in the previous sections in a tool called Pastello,[2] which uses the APIs of the BLACK tool (Geatti, Gigante, and Montanari 2021) for all kinds of manipulations of formulas. We use BLACK also to check the equivalence between input and output formulas. The code for the implementation took less than 500 lines of C++ code.

We evaluated Pastello on the following set of benchmarks. For each $i \in \{5,\ldots,50\}$, we randomly generated 20 formulas of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ of size $i \pm 10$. From now on, with *input formula* we refer to one of the benchmark formulas, and we refer to *its corresponding output formula* to the for-

[2]http://users.dimi.uniud.it/~luca.geatti/tools/pastello.html

mula produced by giving to Pastello the input formula.

For each input formula, we measured four different metrics with respects to its corresponding output formula: (i) *size*, see Fig. 2a; (ii) *number of occurrences of temporal operators*, see Fig. 2b; (iii) *number of occurrences of Boolean operators*, see Fig. 2c; (iv) *depth*, defined as the maximum number of nested temporal operators, see Fig. 2d.

Before discussing the results of the experimental evaluation, we describe how the plots are organized. The x-axis (resp., y-axis) refers to the input formula (resp., output formula). In all the four plots, the x-axis is in linear scale, while the y-axis is in logarithmic scale, except for Fig. 2d in which both axis are in linear scale. The orange lines represent the diagonals of the plots. The blue lines, instead, are piecewise functions that interpolate the mean value for the output formulas corresponding to an input formula of a given size. The more the blue line converges (resp., diverges) from the orange line, the more the trend of the output is polynomial (resp., exponential) with respect to the input.

Consider Fig. 2a, which plots the size of the output formula (on the y-axis, in logarithmic scale) with respect to the size of the input formula (on the x-axis, in linear scale). The exponential growth is quite evident from the trend of the blue line, which diverges from the orange line (the diagonal). This allows us to observe that the exponential blowup of the pastification algorithm from $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ into $\mathsf{F}(\mathsf{pLTL})$ is not only a matter of worst-case scenario, but, instead, is the common trend for the majority of the cases.

We refined the previous analysis of the size of the output formula by plotting the number of *temporal operators* (Fig. 2b) and *Boolean operators* (Fig. 2c) in the output formulas with respect to the input ones. Both plots show the same exponential growth and they indicate that the exponential trend in Fig. 2a is due to the growth of the number of temporal operators as much as to the growth of the number of Boolean operators.

Finally, the only one of the four metric that do not grow exponentially is the *depth*. In fact, the depth of the output formulas, plotted in Fig. 2d, follows a linear trend with respect to the input ones. This comes with no surprise, since: (i) during the transformation into normal form, all rewriting rules increase the depth of the formula only by a constant factor; (ii) for any formula ϕ in normal form, the *height* of the dependency tree T_ϕ is exactly the depth of ϕ; (iii) the *depth* of the formula built starting from a dependency tree T is exactly the *height* of T.

7 Conclusions and Open Problems

We presented a purely syntactic pastification algorithm for $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ formulas that produces $\mathsf{F}(\mathsf{pLTL})$ formulas of size singly exponential with respect to the size of the input. The algorithm is purely syntactical and thus simple to implement. As a matter of fact, we implemented it by using the APIs of the tool BLACK, and the whole source code took less than 500 lines.

We conclude the paper by discussing some limits of our technique. Consider the logic coSafetyLTL, that is, LTL in NNF with temporal modalities restricted to X, F, and U. To

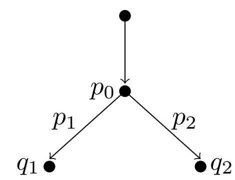


Figure 3: Example of a natural generalization of dependency tree for the coSafetyLTL formula $\mathsf{F}(p_0 \wedge ((p_1 \mathsf{U} q_1) \wedge (p_2 \mathsf{U} q_2)))$.

our knowledge, the best algorithms for coSafetyLTL produce a 3-exponential size pastification (De Giacomo et al. 2021). The proposed technique, based on dependency trees, is hardly applicable to formulas of coSafetyLTL. For instance, consider the coSafetyLTL formula $\mathsf{F}(p_0 \wedge ((p_1 \mathsf{U} q_1) \wedge (p_2 \mathsf{U} q_2)))$. One may think of a straightforward adaptation of the techniques that enriches dependency trees with the possibility of labeling both nodes and edges, corresponding to the universal part (*i.e.*, leftmost argument) of the until operators. Figure 3 shows a possible (generalized) dependency tree for the above formula. By a corresponding generalization of the translation from dependency trees to $\mathsf{F}(\mathsf{pLTL})$ formulas, one would obtain the formula:

$$\mathsf{F}(\mathsf{O}(q_1 \wedge \mathsf{Y}(p_1 \mathsf{S}\ (p_1 \wedge p_0))) \wedge \mathsf{O}(q_2 \wedge \mathsf{Y}(p_2 \mathsf{S}\ (p_2 \wedge p_0))))$$

Unfortunately, such a formula is not equivalent to the original one, since *e.g.*, the trace $\langle \{p_0, p_1\}, \{q_1\}, \{p_0, p_2\}, \{q_2\} \rangle$ is a model of the $\mathsf{F}(\mathsf{pLTL})$ formula but not of the original one. In particular, while for the $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ case the existence of two points where p_0 holds is not a problem (see Example 3), in case of coSafetyLTL formulas we must take into account also the universal condition of the *until* modalities. In the above trace, *e.g.*, the universal condition p_2 is violated at the first and the second time points.

There are also other meaningful open questions. Finding a lower bound for the complexity of the pastification of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ is still an open problem. We conjecture that the algorithm that we presented for the 1-exponential size pastification is *optimal*, and in particular that there exists a family of formulas $\{\phi_i\}_{i=1}^{\omega}$ in $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ such that, for all $i \in \mathbb{N}$, any formula in $\mathsf{F}(\mathsf{pLTL})$ that is equivalent to ϕ_i is of size at least exponential in $|\phi_i|$, that is, $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ *can be exponentially more succinct than* $\mathsf{F}(\mathsf{pLTL})$. As a matter of fact, we conjecture that the family $\mathsf{F}(\bigwedge_{i=1}^{n}(p_i \vee \mathsf{F} q_i))$ may be a witness of such a lower bound.

We also conjecture that there is no pastification algorithm for coSafetyLTL that produces formulas of size less than $\mathcal{O}(n!)$, where n is the size of the input formula. As a matter of fact, we see no way to produce a pastification for the family of formulas $\mathsf{F}(p_0 \wedge \bigwedge_{i=1}^{n} p_i \mathsf{U}\, q_i)$ that does not enumerate all possible orders over $q_1, \ldots, q_n$. This also suggests us that a singly exponential pastification procedure, as the one that we proposed in this paper, is very unlikely for coSafetyLTL.

In view of the above remarks, we believe that the study of the optimality of the proposed algorithm, and the related succinctness properties of $\mathsf{LTL}[\mathsf{X},\mathsf{F}]$ and $\mathsf{F}(\mathsf{pLTL})$, are definitely interesting lines of research.

Last but not least, devising a syntactic pastification algorithm for coSafetyLTL is an important research direction.

Acknowledgments

The authors want to thank Dario Della Monica for the helpful discussions about the complexity analysis of Section 4.3. Luca Geatti and Angelo Montanari acknowledge the support from the 2022 Italian INdAM-GNCS project *"Elaborazione del Linguaggio Naturale e Logica Temporale per la Formalizzazione di Testi"*, ref. no. `CUP_E55F22000270001`. Andrea Mazzullo and Angelo Montanari acknowledge the support of the MUR PNRR project FAIR - Future AI Research (`PE00000013`) funded by the NextGenerationEU. Nicola Gigante acknowledges the support of the PURPLE project, in the context of the AIPlan4EU project's First Open Call for Innovators.

References

Artale, A.; Geatti, L.; Gigante, N.; Mazzullo, A.; and Montanari, A. 2023. Complexity of safety and cosafety fragments of Linear Temporal Logic. In *Proc. of the 36th AAAI Conf. on Artificial Intelligence (AAAI-22)*. AAAI Press.

Chang, E. Y.; Manna, Z.; and Pnueli, A. 1992. Characterization of temporal property classes. In Kuich, W., ed., *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, 474–486. Springer.

Cimatti, A.; Geatti, L.; Gigante, N.; Montanari, A.; and Tonetta, S. 2021. Extended bounded response LTL: a new safety fragment for efficient reactive synthesis. *Formal Methods in System Design* 1–49 (published online on November 18, 2021, doi: 10.1007/s10703–021–00383–3).

Cimatti, A.; Geatti, L.; Gigante, N.; Montanari, A.; and Tonetta, S. 2022. A first-order logic characterisation of safety and co-safety languages. In Bouyer, P., and Schröder, L., eds., *Foundations of Software Science and Computation Structures - 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings*, volume 13242 of *Lecture Notes in Computer Science*, 244–263. Springer.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 854–860. IJCAI/AAAI.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In Yang, Q., and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 1558–1564. AAAI Press.

De Giacomo, G.; Di Stasio, A.; Fuggitti, F.; and Rubin, S. 2021. Pure-past linear temporal and dynamic logic on finite traces. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 4959–4965.

De Giacomo, G.; De Masellis, R.; Maggi, F. M.; and Montali, M. 2022. Monitoring constraints and metaconstraints with temporal logics on finite traces. *ACM Trans. Softw. Eng. Methodol.* 31(4):68:1–68:44.

De Giacomo, G.; Favorito, M.; and Fuggitti, F. 2022. Planning for temporally extended goals in pure-past linear temporal logic: A polynomial reduction to standard planning. *arXiv preprint arXiv:2204.09960*.

Geatti, L.; Gigante, N.; and Montanari, A. 2021. BLACK: A fast, flexible and reliable LTL satisfiability checker. In Monica, D. D.; Pozzato, G. L.; and Scala, E., eds., *Proceedings of the 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis*, volume 2987 of *CEUR Workshop Proceedings*, 7–12. CEUR-WS.org.

Geatti, L.; Montali, M.; and Rivkin, A. 2022. Reactive synthesis for DECLARE via symbolic automata. *arXiv preprint arXiv:2212.10875*.

Lichtenstein, O.; Pnueli, A.; and Zuck, L. 1985. The glory of the past. In *Workshop on Logic of Programs*, 196–218. Springer.

Maler, O., and Pnueli, A. 1990. Tight bounds on the complexity of cascaded decomposition of automata. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, 672–682. IEEE.

Maler, O.; Nickovic, D.; and Pnueli, A. 2005. Real time temporal logic: Past, present, future. In *International Conference on Formal Modeling and Analysis of Timed Systems*, 2–16. Springer.

Maler, O.; Nickovic, D.; and Pnueli, A. 2007. On synthesizing controllers from bounded-response properties. In *International Conference on Computer Aided Verification*, 95–107. Springer.

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *Proceedings of POPL'89*, 179–190. ACM Press.

Pnueli, A. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 46–57. IEEE.

Rosner, R. 1992. *Modular synthesis of reactive systems*. Ph.D. Dissertation, PhD thesis, Weizmann Institute of Science.