Promptness and Fairness in Muller LTL Formulas

Damien Busatto-Gaston
Univ Paris Est Creteil, LACL, F-94010 Creteil, France

Youssouf Oualhadj
Univ Paris Est Creteil, LACL, F-94010 Creteil, France
CNRS, ReLaX, IRL 2000, Siruseri, India

Léo Tible
Univ Paris Est Creteil, LACL, F-94010 Creteil, France

Daniele Varacca
Univ Paris Est Creteil, LACL, F-94010 Creteil, France

Abstract

In this paper we consider two different views of the model checking problem for the Linear Temporal Logic (LTL). On the one hand, we consider the *universal* model checking problem for LTL, where one asks that for a given system and a given formula all the runs of the system satisfy the formula. On the other hand, the *fair* model checking problem for LTL asks that for a given system and a given formula almost all the runs of the system satisfy the formula.

It was shown that these two problems have the same theoretical complexity, *i.e.* PSPACE-complete. A less expensive fragment was identified in a previous work, namely the *Muller fragment*, which consists of combinations of repeated reachability properties.

We consider *prompt* LTL formulas (pLTL), that extend LTL with an additional operator, *i.e.* the *prompt-eventually*. This operator ensures the existence of a bound such that reachability properties are satisfied within this bound. This extension comes at no cost since the model checking problem remains PSPACE-complete.

We show that the corresponding Muller fragment of pLTL, with prompt repeated reachability properties, enjoys similar computational improvements. Another feature of Muller formulas is that the model checking problem becomes easier under the fairness assumption. This distinction is lost in the prompt setting, as we show that the two problems are equivalent instance-wise. Subsequently, we identify a new prefix independent fragment of pLTL for which the fair model checking problem is less expensive than the universal one.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Model checking, Fairness, Temporal logics

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2024.15

Related Version *Full Version*: `https://arxiv.org/abs/2204.13215v5` [4]

1 Introduction

Linear Temporal Logic (LTL) allows system designers to easily describe behavioral properties of a system [17]. Its expressive power and convenience of use proved useful in many areas such as system design and verification [8, 20], agent planning [5, 12], knowledge representation [11], and control and synthesis [18, 19]. At the heart of these applications a fundamental formal approach is always to be found, *i.e.* the *model checking problem* [22].

Universal and fair model checking. When trying to verify a system against a specification, one usually models the system as directed graphs called *Labeled Transition Systems* (LTS). A run of the system is an infinite path in the LTS. The standard approach to model checking consists in verifying that all possible runs of the LTS comply with the specification. Some

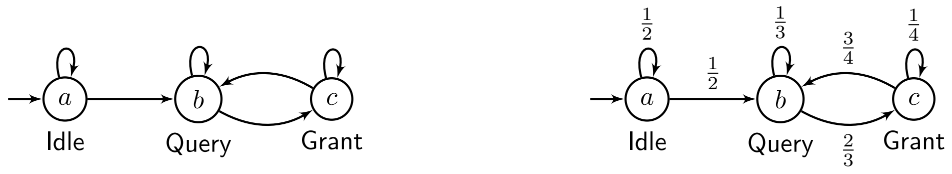

Figure 1 A protocol modeled as an LTS on the left, and as a probabilistic system on the right.

systems may not satisfy the specification because of a few *unlikely* runs. To avoid discarding these systems, the *fair* model checking approach gives a formal definition of *small sets* of executions, and then verifies that the set of executions that do not satisfy the specification is indeed small.

▶ **Example 1.** Consider the example on the left of Figure 1, this figure models a "toy protocol" that could either be in an idle state, a querying state or granting state. Assume now that the modeler wants to check that **the protocol is not always idle**. Without any fairness assumption, this system will be rejected since it could always repeat the loop on state a. Now, consider that this LTS is an abstracted view of the real protocol, which is a probabilistic system modeled as a Markov chain on the right of Figure 1. Then, the probability associated with the system remaining idle forever is 0. We can decide to ignore this "unlikely" possibility and say that the system *fairly* satisfies the specification.

In general, a set of executions of a Markov chain is considered *small* if it has probability 0. It turns out that the precise probabilities that appear in the system have no impact on which sets of executions have probability 0. Thus, we can assume without loss of generality every probability distribution to be uniform, and represent systems as LTS instead.

LTL model checking. The complexity of verifying that every run of a system satisfies an LTL formula is known to be a PSPACE-complete problem [22]. This complexity is measured with respect to the size of the LTS and its specification, an LTL formula in our case. This high complexity is due to the fact that one has to encode the specification as a *non-deterministic Büchi automaton*, that can be exponential in size. This yields an exponential blow-up and further techniques are required to keep the complexity within PSPACE, see *e.g.* [1].

In order to circumvent this blow-up, a natural idea is to identify fragments of LTL where the model checking problem becomes easier to solve. In the seminal work of Sistla and Clarke [22], they identified fragments where model checking is coNP-complete, which is more amenable to implementation than PSPACE as it opens the door to symbolic approaches using SAT-solvers. In particular they showed this complexity for the class of formulas that consists of Boolean combinations of reachability properties. We also cite the fragment identified by Muscholl et al. [16]. This fragment is obtained by prohibiting the use of reachability operators (until) and restricting the formula to exclusively using next operators indexed by a letter. They showed that the model checking problem is NP-complete for this fragment. Finally we highlight the work of Emerson et al. [10] where they studied the fragment of *Muller formulas*, *i.e.* the fragment combining repeated reachability (*i.e.* Büchi) properties. They showed that model checking becomes coNP-complete for Muller formulas.

The PSPACE-complete complexity result also holds for the *fair* model checking problem [9]. However, for the fragment of *Muller Formulas*, while the universal model checking problem becomes coNP-complete, the fair model checking problem can be solved by a polynomial time algorithm presented in [21]. In other words, this fragment allows one to take advantage of the fairness assumption to obtain tractable model checking procedures.

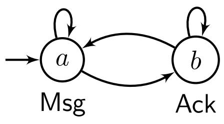

Figure 2 An LTS that satisfies a Muller formula but not its *prompt* variant, as per Example 3.

▶ **Example 2.** We go back to the example of Figure 1, and consider the following specification: **infinitely often a query is made and infinitely often a grant is granted**. This specification can be expressed as a conjunction of repeated reachability objectives, and thus as a Muller formula.

Prompt Formulas. Consider the following natural specification for messaging protocols: **All along the execution, whenever a message is sent an acknowledgment is received at a later step**. A system may satisfy this specification in an unpractical way, where the waiting time for the acknowledgment grows arbitrarily large along some executions. *Prompt LTL* (pLTL), introduced by Kupferman et al. [15], can express a variant of this specification that enforces an upper limit on waiting times: **There exists a bound k such that, along any execution, whenever a message is sent an acknowledgment is received within the next k steps**

▶ **Example 3.** Consider the LTS of Figure 2, and consider the specification asking that either Msg or Ack is seen infinitely often. Clearly enough, any infinite path in the system will satisfy this specification. Consider now the prompt variant of this specification, asking for the existence of a bound k such that either Msg is seen infinitely often in a prompt way, *i.e.* with a maximum of k steps in between successive occurrences, or Ack is seen infinitely often in a prompt way. Now, for any bound $k \geq 0$, the run $a^{k+1}b^{k+1}\cdots$ does not satisfy this specification. As such, the system does not satisfy the prompt specification.

The model checking problem for pLTL is also known to be PSPACE-complete [15]. This is achieved through an *efficient translation* into LTL. The fair model checking problem has the same complexity as well: although an explicit proof is not published, a careful inspection of the proof of [15] shows that the translation into LTL formulas holds for the fair setting, and thus it is sufficient to invoke the algorithm from [9] without further blowup.

In this paper we consider the *Muller fragment* of pLTL, that combines the prompt variants of repeated reachability properties such as the one described in Example 3.

Contributions and organisation. Our original contributions are as follows.

We first show that universal model checking for the Muller fragment of pLTL is coNP-complete. In order to show the membership, we had to depart from the already existing reduction to classical LTL and develop new technical tools. Roughly speaking, we develop combinatorial tools to represent runs and the reasons why they might not satisfy a prompt repeated reachability property of bound k. If one thinks about a faulty run as an infinite run containing a finite window of k consecutive faulty states, then *pumping* a cycle within that window leads to a longer window of faulty states. Thus, this run can be used as a generator for faulty runs of arbitrarily long bounds $k' > k$, cf. Lem. 14. However, one has to pay particular attention to the case where temporal operators are nested. In particular, "pumping" these finite sequences of faulty runs is done according to a well chosen order, cf. Section 3.1 for a formal definition of the notion of *multi-pumpings*.

Table 1 Summary of our contributions on variants of the model checking problem.

Model checking	non-prompt		prompt		
	LTL	Muller	pLTL	Muller	initialized Muller
Universal	PSPACE-c [22]	coNP-c [10]	PSPACE-c [15]	coNP-c Thm. 9, 21	coNP Thm. 26
Fair	PSPACE-c [9]	PTIME [21]	PSPACE-c [15]	coNP-c Thm. 9, 21	PTIME Thm. 28

In Section 3.2, we prove a *small witness property.* This notion, in some sense, efficiently stores data about the existence of counter-examples, and results in a coNP procedure.

Our second contribution is to show that the fair model checking problem for this fragment does not need to be studied separately as it coincides with the universal model checking problem, cf. Thm. 21. Indeed we prove that if a system is a fair model for some pLTL Muller formula, then every path must satisfy the formula.

Further, we note that prompt Muller formulas may sometimes require "too much" from a system. In particular it is possible for a system to violate a specification during an *initial* phase of the execution, but once it enters a *steady regime* the specification might be satisfied.

Our last contribution is to address this issue by introducing the *initialized Muller* fragment for pLTL. This fragment expresses the fact that a system should satisfy a specification in the long run, *i.e.* we ignore the finite initialization phase. This vision is inspired from *prefix independent* specifications. Such specification are only interested in the asymptotic behavior of a system. For instance, *parity, Rabin, Street, Büchi* are all ω-regular specifications whose satisfaction is independent of any finite prefix [1]. Not only do these specifications seem to be more suited to real life applications, they also in general enjoy nice properties, especially in the probabilistic setting, cf. [6, 13, 14]. We also mention results [3, 7] where prefix independence has been considered in a setting rather close to ours, and there again they exhibited well behaved properties [2].

In our case, we show that the fair model checking of prompt Muller formulas is more tractable on initialized formulas: the universal model checking is still in coNP, but there is a polynomial time algorithm solving the fair model checking problem for this fragment.

Due to page limit, ommitted material can be found at `https://arxiv.org/pdf/2204.13215`.

2 Preliminaries

Throughout the document we will use the following notations and conventions: AP is a finite set of atomic propositions. For an arbitrary set E, E^* is the set of finite sequences in E, and E^ω is the set of infinite sequences of E. When E is a finite set, $|E|$ will denote its size.

Labelled Transition System. An *LTS* is a tuple $\mathcal{S} = \langle \mathsf{S}, s_{\mathsf{init}}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \rangle$ such that S is a set of states, $s_{\mathsf{init}} \in \mathsf{S}$ is an initial state, $\mathsf{T} \subseteq \mathsf{S} \times \mathsf{S}$ is a set of transitions, and $\mathsf{lbl} \ : \ S \to 2^{\mathsf{AP}}$ is a labeling function mapping every state to the atomic propositions that hold on it.

For a state $s \in \mathsf{S}$, the set of successors of s is $\mathsf{Succ}(s) = \{t \in \mathsf{S} \mid (s,t) \in \mathsf{T}\}$. A *finite path* in $\mathcal{S}$ is a finite sequence of states $\pi = s_0 s_1 \cdots s_k$ of length $k+1$ such that $\forall 0 \leq i \leq k-1, s_{i+1} \in \mathsf{Succ}(s_i)$. We denote by $|\pi|$ the length of π, *i.e.* $|\pi| = k+1$. A *run* in $\mathcal{S}$ is an infinite sequence of states $\rho = s_0 s_1 \cdots$ such that $\forall i \geq 0$, $s_{i+1} \in \mathsf{Succ}(s_i)$. Let ρ be a run and let $i \geq 0$, then $\rho[i] = s_i$, $\rho[i..] = s_i s_{i+1} \cdots$, that is the infinite suffix starting in the

$(i+1)$th letter, and $\rho[..i]$ is the prefix up to the (i)th position, that is $\rho[..i] = s_0 \cdots s_{i-1}$. For $i < j$, $\rho[i..j]$ is the finite path $s_i \cdots s_{j-1}$. We use the same notations for a finite path π, and in this case $\pi[i..]$ will be a finite suffix. The concatenation of a finite path π with a finite path (or a run) π' is denoted $\pi\pi'$. A cycle (or a loop) is a finite path $\pi = s_0 s_1 \cdots s_k$ such that $s_0 \in \mathsf{Succ}(s_k)$. In particular, the finite path π^n repeats for n iterations the cycle π.

The set of states visited infinitely often by a run ρ is denoted $\mathsf{Inf}(\rho)$, and is formally defined as $\{s \in \mathsf{S} \mid \forall i \geq 0, \exists j \geq i, \rho[j] = s\}$. Finally, let FPaths (resp. Runs) be the set of all the finite paths (resp. runs) in $\mathcal{S}$, and let $\mathsf{Runs_{init}}$ be the set of all runs starting from s_{init}, that is $\{\rho \in \mathsf{Runs} \mid \rho[0] = s_{\mathsf{init}}\}$. These notations assume that $\mathcal{S}$ is clear from context.

Linear Temporal Logic. An *LTL* formula φ is defined using the following grammar:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi\ , \text{ where } \alpha \text{ ranges over } \mathsf{AP}.$$

Runs ρ of $\mathcal{S}$ are evaluated inductively over LTL formulas as follows:

$$\rho \models \alpha \text{ iff } \alpha \in \mathsf{lbl}(\rho[0])\ , \qquad \rho \models \neg\varphi \text{ iff } \rho \not\models \varphi\ ,$$
$$\rho \models \varphi \vee \psi \text{ iff } \rho \models \varphi \text{ or } \rho \models \psi\ , \qquad \rho \models \mathsf{X}\,\varphi \text{ iff } \rho[1..] \models \varphi\ ,$$
$$\rho \models \varphi\,\mathsf{U}\,\psi \text{ iff } \exists i \geq 0,\ \rho[i..] \models \psi \text{ and } \forall j < i,\ \rho[j..] \models \varphi\ ,$$

A *state formula* is a formula that only contains Boolean operators ($\neg$ and $\vee$) and atomic propositions, and is thus entirely evaluated on the first position of ρ. The operators X and U are called *temporal operators*. We derive all standard Boolean operators from $\neg$ and $\vee$, let $\top$ and $\bot$ be atomic propositions that are respectively always true and always false, and define a few extra temporal operators as syntactic sugar: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$, $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$, $\mathsf{F}^\infty\,\varphi = \mathsf{G}\,\mathsf{F}\,\varphi$. The formula $\mathsf{F}\,\varphi$ is true for any run where φ is *eventually* true, while the formula $\mathsf{G}\,\varphi$ is true for any run where φ *always* holds. The formula $\mathsf{F}^\infty\,\varphi$ holds for any run where there exists infinitely many positions from where φ is true, and is used to encode repeated reachability properties, such as the winning condition of a Büchi automaton.

▶ **Problem 4** (Universal model checking). *Given an LTS $\mathcal{S}$ and an LTL formula φ, does $\rho \models \varphi$ hold true for every run $\rho \in \mathsf{Runs_{init}}$? In this case, we write $\mathcal{S} \models \varphi$.*

The universal model checking problem is PSPACE-complete [22]. We express the complexity of our model checking problems with respect to both the size of the formula $|\varphi|$, *i.e.* the number of operators that appear in φ, and the size of the system $|\mathcal{S}|$, *i.e.* $|\mathsf{T}| + |\mathsf{S}| \max_{s \in \mathsf{S}} |\mathsf{lbl}(s)|$.

Fairness in model checking. The fairness assumption presented in Example 1 relies on the idea that a set of runs is sometimes considered quantitatively *small.* In particular, we use a *fair coin* to view an LTS as a probabilistic system and derive a measure over paths. At each state, a fair coin is flipped and the successor state is chosen accordingly. This coin flipping procedure is assumed to be i.i.d. and it induces a natural probability measure over $\mathsf{Runs_{init}}$. In the fair model checking problem, one asks whether *almost all* the runs of an LTS satisfy a given formula φ *i.e.* if a run obtained from the fair coin process satisfies φ with probability 1.

Formally, in order to build the probability measure over $\mathsf{Runs_{init}}$, we use the classical notion of *cylinders.* For $\pi \in \mathsf{FPaths}$, the cylinder induced by π, denoted $\mathsf{Cyl}(\pi)$, is the set $\{\rho \in \mathsf{Runs} \mid \pi \text{ is a prefix of } \rho\}$. Then, the probability of a run being in a cylinder $\mathbb{P}_{\mathcal{S}}(\mathsf{Cyl}(\pi))$ is defined as the probability that π is followed under the fair coin process. This measure can be uniquely extended over the set $\mathsf{Runs_{init}}$ using Carathéodory's extension theorem.

▶ **Problem 5** (Fair model checking). *Given an LTS $\mathcal{S}$ and an LTL formula φ, does it holds that $\mathbb{P}_{\mathcal{S}}(\{\rho \in \mathsf{Runs}_{\mathsf{init}} \mid \rho \models \varphi\}) = 1$? In this case, we will $\mathcal{S} \models_{\mathsf{AS}} \varphi$, where $\models_{\mathsf{AS}}$ stands for the "almost sure" satisfaction of a formula.*

Under the fairness assumption, a model checking procedure is intuitively allowed to ignore unrealistic behaviours. However, this does not simplify the complexity of the problem, as it has been shown that the fair model checking problem is also PSPACE-complete [9].

The *Muller* fragment of LTL. In an effort to obtain lower complexity results for the model checking problem, subclasses of LTL formulas, defined by syntactic restrictions, have been considered. In particular, an LTL formula φ is in the *Muller fragment*, denoted $\varphi \in \mathcal{L}(\mathsf{F}^\infty)$, if the repeated reachability operator F^∞ is the only temporal operator that is allowed:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{F}^\infty \varphi \,.$$

The key property of Muller formulas is that their satisfaction on a run ρ is entirely determined by the initial state and $\mathsf{Inf}(\rho)$, the states visited infinitely often. This leads to lower complexity results: in [21], it was shown that the universal model checking problem for Muller formulas is coNP-complete, while the fair model checking problem can be solved in polynomial time.

Promptness in LTL. A prompt LTL formula φ is defined according to the following grammar [15]: $\varphi ::= \alpha \mid \neg\alpha \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \varphi\,\mathsf{R}\,\varphi \mid \mathsf{F_P}\,\varphi$.

The main difference with LTL lies in the addition of $\mathsf{F_P}$. This operator states that φ has to be satisfied eventually, but in a "prompt" fashion. The semantics of this operator are defined with respect to a bound $k \in \mathbb{N}$. For a given k, we write $(\rho, k) \models \mathsf{F_P}\,\varphi$ if $\exists i \leq k,\ (\rho[i..], k) \models \varphi$, in contrast, recall that $\rho \models \mathsf{F}\,\varphi$ if $\exists i \in \mathbb{N}, \rho[i..] \models \varphi$. The other operators ignore the bound k and are evaluated using the semantics of LTL defined earlier. Another difference with LTL is that Boolean negations are not allowed in pLTL, as the negation of the newly added operator $\mathsf{F_P}$ is deemed unnatural from a modelling point of view. Therefore, the grammar explicitly contains the conjunction $\wedge$, and the *release* operator R, the dual of the *until* operator U, previously expressed with negations.

▶ **Problem 6** (Universal prompt model checking). *Given an LTS $\mathcal{S}$ and a pLTL formula φ, does there exists a bound $k \in \mathbb{N}$ such that for all $\rho \in \mathsf{Runs}_{\mathsf{init}}$ in $\mathcal{S}$, $(\rho, k) \models \varphi$? In this case, we write $\mathcal{S} \models \varphi$.*

▶ **Problem 7** (Fair prompt model checking). *Given an LTS $\mathcal{S}$ and a pLTL formula φ, does there exists a bound $k \in \mathbb{N}$ such that $\mathbb{P}_{\mathcal{S}}(\{\rho \in \mathsf{Runs}_{\mathsf{init}} \mid (\rho, k) \models \varphi\}) = 1$? In this case, we write $\mathcal{S} \models_{\mathsf{AS}} \varphi$.*

The addition of the prompt eventually operator $\mathsf{F_P}$ comes at no extra cost compared with LTL, as both universal and fair model checking remain PSPACE-complete [15].

The prompt Muller fragment. In this work, we consider a subclass of pLTL formulas inspired by Muller formulas. We define a new operator, $\mathsf{F_P^\infty}\,\varphi = \mathsf{G}\,\mathsf{F_P}\,\varphi$, as a prompt variant of F^∞. Thus, $\mathsf{F_P^\infty}\,\varphi$ holds true for a pair (ρ, k) if from every position $i \in \mathbb{N}$, a position $j \in [i, i+k]$ can be found so that $(\rho[j..], k) \models \varphi$. A pLTL fromula φ is in the *prompt Muller fragment*, denoted $\varphi \in \mathcal{L}(\mathsf{F_P^\infty})$, if it is obtained by the following grammar:

$$\varphi ::= \alpha \mid \neg\alpha \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{F_P^\infty}\,\varphi \,.$$

▶ **Example 8.** Consider the specifications mentioned in Example 3, that asks that either Msg or Ack is seen infinitely often. This is expressed by the Muller formula $\mathsf{F}^\infty \mathsf{Msg} \vee \mathsf{F}^\infty \mathsf{Ack}$. The prompt variant is the formula $\mathsf{F}_\mathsf{P}^\infty \mathsf{Msg} \vee \mathsf{F}_\mathsf{P}^\infty \mathsf{Ack}$, that intuitively asks that either Msg or Ack are seen *frequently*, *i.e.* with a frequency that does not vanish along the execution.

3 Prompt Muller formulas

A remarkable property of the Muller fragment of LTL is that the satisfaction of a formula only depends on the asymptotic behavior of the system. Therefore, as shown in [21] (a corollary of a result from [10]), a system satisfies a Muller formula when every strongly connected set satisfies the formula. This property yields an easier model checking problem, namely coNP-complete. In this section we focus on the prompt Muller fragment of pLTL. We show that the complexity of the model checking problem is again coNP-complete. However, the techniques involved are different. Indeed, they do not follow from structural properties of the transition system. We introduce combinatorial tools to establish a small witness property. We further deepen our study by considering runs obtained under the fairness assumption. We show that prompt Muller formulas describe the same set of runs with or without the fairness assumption. This differs from (non-prompt) Muller formulas, where fairness allowed for more tractable approaches. As a corollary, we obtain that fair model checking is as expensive as universal model checking for the prompt Muller fragment.

Most of this section will be devoted to the proof of the following theorem:

▶ **Theorem 9.** *The universal model checking problem for $\mathcal{L}(\mathsf{F}_\mathsf{P}^\infty)$ is* coNP*-complete.*

3.1 Pumping a counter example

The first stepping stone to prove Thm. 9 is to define the notion of *pumping*. The idea is quite simple: for a run ρ, a pumping of ρ is a run where some cycle of ρ has been repeated. Formally, it is defined as follows.

▶ **Definition 10** (Pumping). *Given an LTS $\mathcal{S}$ and a run $\rho \in \mathsf{Runs}$, a* pumping *of ρ is a run $\rho' \in \mathsf{Runs}$ such that there exist $0 \leq i < j$ and $l > 0$, with $\rho[i..j]$ a cycle, so that $\rho' = \rho[..i]\rho[i..j]^{l-1}\rho[i..]$.*

If $i = 0$, then by convention $\rho[..0] = \varepsilon$ is the empty path. In particular, ρ' contains l copies of the cycle $\rho[i..j]$, as the last one is in $\rho[i..]$. We will say that a pumping is a 1-pumping, as one cycle is iterated. A natural extension of this notion is to allow for the pumping of several cycles in a run. Formally, we define a *multi-pumping* inductively as follows, with a run being the only 0-pumping of itself.

▶ **Definition 11** (Multi-pumping). *Given an LTS $\mathcal{S}$, a run $\rho \in \mathsf{Runs}$, and $n > 0$, an n-*pumping *of ρ is a run $\rho' \in \mathsf{Runs}$ such that there exist $0 \leq i < j$, with $\rho[i..j]$ a cycle, and there is a $(n-1)$-pumping $\tilde{\rho}$ of $\rho[i..]$ and some $l > 0$ so that $\rho' = \rho[..i]\rho[i..j]^{l-1}\tilde{\rho}$. A* multi-pumping *of ρ is a run ρ' such that ρ' is an n-pumping of ρ for some $n \geq 0$.*

The construction of an n-pumping allows for several cycles of a run to be pumped, but only consecutively in a left-to-right fashion. This keeps the n individual pumpings ordered and will allow for inductive proofs on n.

▶ **Example 12.** Consider the run $\rho = ab(abcb)aad^\omega$ represented if Figure 3, and note that $abcb$ is a cycle that can be pumped. Therefore, $\rho' = ab(abcb)^4aad^\omega$ is a 1-pumping of ρ. Now, imagine that you also want to pump the cycle bc. By definition, further pumpings can only occur starting from the last “copy” of the cycle $abcb$. In particular, $\rho'' = ab(abcb)^3a(bc)^3baad^\omega$ is a 2-pumping of ρ.

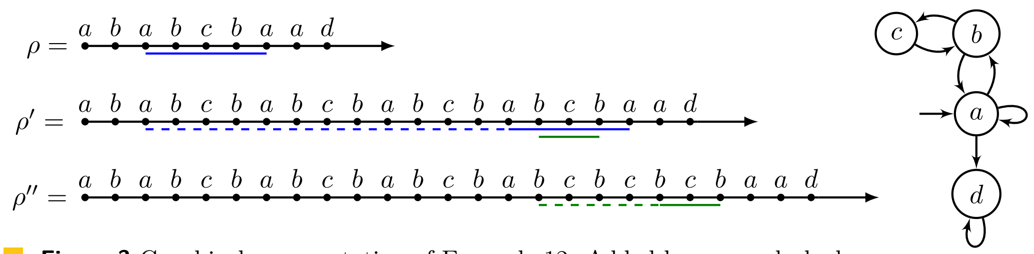
$\rho =$ a b a b c b a a d
$\rho' =$ a b a b c b a b c b a b c b a b c b a a d
$\rho'' =$ a b a b c b a b c b a b c b a b c b c b c b a a d
c
b
a
d

Figure 3 Graphical representation of Example 12. Added loops are dashed.

The main property of multi-pumpings is that they preserve counter-examples, *i.e.* if a run ρ does not satisfy a prompt Muller formula φ for some bound, then for the same bound any multi-pumping of ρ will not satisfy φ either.

▶ **Proposition 13.** *Given an lts $\mathcal{S}$, a run $\rho \in \mathsf{Runs}$, a multi-pumping ρ' of ρ, a formula $\varphi \in \mathcal{L}(\mathsf{F}_\mathsf{P}^\infty)$ and $k \geq 0$, if $(\rho, k) \not\models \varphi$, then $(\rho', k) \not\models \varphi$.*

Proof scheme. Intuitively, if there is no nesting of $\mathsf{F}_\mathsf{P}^\infty$ operator in the formula, then the formula being false on (ρ, k) only depends on "faulty" windows of length k that can be found in ρ. As a multi-pumping only extends faulty windows by duplicating cycles, any faulty window in the original run also exists somewhere in the multi-pumping. If φ has some nesting of $\mathsf{F}_\mathsf{P}^\infty$ operators, then a window being faulty or not depends on the suffix run that comes after it, which makes these considerations significantly more involved technically, as pumping changes the suffix of our runs. A full proof is detailed in Appendix A. ◀

The second core property of multi-pumpings, derived from Prop. 13, is that they can be used to generate counter-examples for arbitrarily large bounds k, as long as there is a counter example for the bound $N = |S| + 1$. This is formalised as follows:

▶ **Lemma 14.** *Let $\mathcal{S} = \langle \mathsf{S}, s_\mathsf{init}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \rangle$ be an LTS, $\varphi \in \mathcal{L}(\mathsf{F}_\mathsf{P}^\infty)$, and let $N = |S| + 1$. If there is $\rho_N \in \mathsf{Runs}_\mathsf{init}$ such that $(\rho_N, N) \not\models \varphi$ then for all $k \geq N$, there is a multi-pumping ρ_k of ρ_N such that $(\rho_k, k) \not\models \varphi$.*

Thus, in order to show that a system *does not satisfy* a prompt Muller formula for any bound k, it is enough to exhibit a single run ρ so that $(\rho, N) \not\models \varphi$. Indeed, $(\rho, k) \not\models \varphi$ is immediate for every $k \leq N$ by definition of pLTL,[1] and for any $k > N$, Lem. 14 can be used to generate another run that will not satisfy φ either.

3.2 Canonical representation for witnesses

Let ρ be a run, and assume that $(\rho, N) \not\models \varphi$. Such a run witnesses that $\mathcal{S} \not\models \varphi$, however, it is an infinite sequence. Moreover, even with a finite representation of ρ, checking that $(\rho, N) \not\models \varphi$ remains a challenging task. In this section, we introduce a new data structure, that carries a representation of ρ and enough information to efficiently check that $(\rho, N) \not\models \varphi$.

Given an LTL formula φ, let $\mathsf{SubF}(\varphi)$ be the set of all subformulas of φ.

▶ **Definition 15.** *Given an LTS $\mathcal{S} = \langle \mathsf{S}, s_\mathsf{init}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \rangle$, a run $\rho \in \mathsf{Runs}$, a bound $k \geq 0$ and a formula $\varphi \in \mathcal{L}(\mathsf{F}_\mathsf{P}^\infty)$, a* witness *for ρ with bound k is a function $\mathsf{W}_{\rho,k} : \mathsf{SubF}(\varphi) \mapsto 2^{\mathbb{N}}$ that maps subformulas to finite sets and satisfies the following conditions:*

[1] Intuitively, if a faulty window exists to falsify an $\mathsf{F}_\mathsf{P}^\infty$ operator for some bound, then every shorter bound admits the same faulty window to falsify $\mathsf{F}_\mathsf{P}^\infty$.

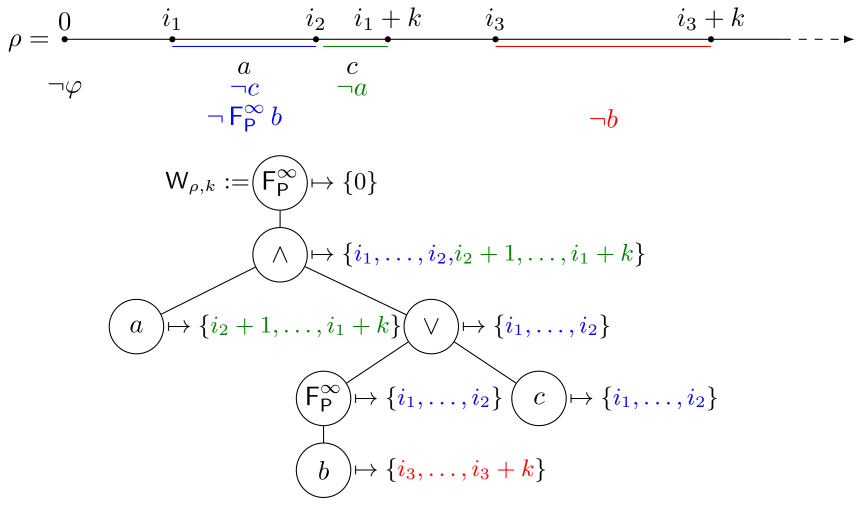
$\rho =$
0
i_1
i_2
i_1+k
i_3
i_3+k
$\neg\varphi$
a
$\neg c$
$\neg\,\mathsf{F}^\infty_\mathsf{P}\, b$
c
$\neg a$
$\neg b$
$\mathsf{W}_{\rho,k} :=$
$\mathsf{F}^\infty_\mathsf{P} \mapsto \{0\}$
$\wedge \mapsto \{i_1,\dots,i_2, i_2+1,\dots,i_1+k\}$
$a \mapsto \{i_2+1,\dots,i_1+k\}$
$\vee \mapsto \{i_1,\dots,i_2\}$
$\mathsf{F}^\infty_\mathsf{P} \mapsto \{i_1,\dots,i_2\}$
$c \mapsto \{i_1,\dots,i_2\}$
$b \mapsto \{i_3,\dots,i_3+k\}$

Figure 4 Example of a run that does not satisfy a formula, and the corresponding witness.

- $\forall i \in \mathsf{W}_{\rho,k}(\alpha), \alpha \notin \mathsf{lbl}(\rho[i])$
- $\forall i \in \mathsf{W}_{\rho,k}(\neg\alpha), \alpha \in \mathsf{lbl}(\rho[i])$
- $\forall i \in \mathsf{W}_{\rho,k}(\psi_1 \vee \psi_2), i \in \mathsf{W}_{\rho,k}(\psi_1) \cap \mathsf{W}_{\rho,k}(\psi_2)$
- $\forall i \in \mathsf{W}_{\rho,k}(\psi_1 \wedge \psi_2), i \in \mathsf{W}_{\rho,k}(\psi_1) \cup \mathsf{W}_{\rho,k}(\psi_2)$
- $\forall i \in \mathsf{W}_{\rho,k}(\mathsf{F}^\infty_\mathsf{P}\,\psi), \exists i' \geq i, \forall i' \leq j \leq i'+k, j \in \mathsf{W}_{\rho,k}(\psi)$
- $0 \in \mathsf{W}_{\rho,k}(\varphi)$

Then, we denote $\max(\mathsf{W}_{\rho,k}) = \max\{i \in \mathbb{N} \mid \psi \in \mathsf{SubF}(\varphi) \wedge i \in \mathsf{W}(\psi)\}$.

Intuitively, such a witness justifies that $(\rho,k) \not\models \varphi$ by describing for each subformula a relevant set of positions from where they are falsified. In particular, $0 \in \mathsf{W}_{\rho,k}(\varphi)$ ensures $(\rho,k) \not\models \varphi$. Then, the other conditions ensure that inductively, the positions falsify subformulas up to reaching the atomic propositions.

▶ **Example 16.** Consider the run ρ of Figure 4, and the formula $\varphi = \mathsf{F}^\infty_\mathsf{P}(a \wedge (\mathsf{F}^\infty_\mathsf{P}\, b \vee c))$. Some of the labels of states in ρ are represented, so that a holds between positions i_1 and i_2 for example. The syntactic tree of the formula is also represented in Figure 4, where every subtree corresponds to a subformula. Here, for a given bound k, it is assumed that $(\rho,k) \not\models \varphi$, and a potential witness for ρ with bound k is described on the syntactic tree of φ: every subformula is mapped to a set of positions. Then, one can check that every inductive rule of Def. 15 holds. For example, for the subformula a, no state $\rho[i]$ such that $i \in \mathsf{W}_{\rho,k}$ has label a, and for the subformula $a \wedge (\mathsf{F}^\infty_\mathsf{P}\, b \vee c)$, it holds that $\mathsf{W}_{\rho,k}(a \wedge (\mathsf{F}^\infty_\mathsf{P}\, b \vee c)) = \mathsf{W}_{\rho,k}(a) \cup \mathsf{W}_{\rho,k}(\mathsf{F}^\infty_\mathsf{P}\, b \vee c)$. Note how the witness describes which positions along ρ are relevant to prove where each subformula is falsified, up to the position 0 at the root $\mathsf{W}_{\rho,k}(\varphi)$, so that $(\rho,k) \not\models \varphi$.

Note that Def. 15 does not require a witness W to be small in size, as it could:
- map subformulas ψ_1 and ψ_2 to positions that are lost by intersection in $\mathsf{W}(\psi_1 \vee \psi_2)$,
- map ψ to more than the k positions needed to falsify $\mathsf{W}(\mathsf{F}^\infty_\mathsf{P}\,\psi)$, and
- use positions that are needlessly far away, so that $\max(\mathsf{W})$ is too large.

We address all of these concerns by proving a *small witness property*, *i.e.*, there is a path ρ with $(\rho,k) \not\models \varphi$ iff there exists a *small* witness for some path ρ' with bound k. Here, small is meant as a polynomial upper bound on the size needed to represent W, *i.e.* $|\varphi| \max(\mathsf{W})$.

▶ **Proposition 17.** *Given a formula $\varphi \in \mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$, an LTS $\mathcal{S} = \langle \mathsf{S}, s_{\mathsf{init}}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \rangle$ and a bound $k \geq 0$, there exists a run $\rho \in$ Runs such that $(\rho, k) \not\models \varphi$ iff there exists a finite path π and a function* W *such that for all runs $\pi\rho'$,* W *is a witness for $\pi\rho'$ with bound k and $|\pi| \leq (k+1)|\varphi|(|\mathsf{S}|+1)$ and $\max(\mathsf{W}) < |\pi|$.*

Proof scheme. First, we show that if $\mathcal{S} \not\models \varphi$ then a witness W can be obtained from any counter-example run based on the semantics of pLTL. Then, we show that if there exists a witness then there is one of size polynomial in k and φ. This intuitively comes from the fact that for the case $\mathsf{F}^{\infty}_{\mathsf{P}}\,\psi$, one window of length k falsifying ψ is sufficient, which prevents the sets of indexes from needing to be large. Third, we show that given a witness, we can construct a finite path π of length polynomial in the size of the witness and the size of the system such that for any run ρ', we have $(\pi\rho', k) \not\models \varphi$. This holds because the states of ρ that appear in the witness are enough to guarantee that the formula is falsified, and in-between those states, we can always find a short path. This construction yields a polynomial bound for max(W), and the proposition follows. ◀

Therefore, in order to show that a system does not satisfy φ, it is now enough to search for a finite path π and a witness W of polynomial size, as described in Prop. 17.

3.3 Universal model checking

We are now ready to prove coNP membership, as a corollary of Prop. 17:

▶ **Lemma 18.** *The universal model checking problem for $\mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$ is in* coNP*.*

Proof. Given a formula φ and a system $\mathcal{S} = \langle \mathsf{S}, s_{\mathsf{init}}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \rangle$, one can guess a witness of polynomial size for the bound $N = |\mathsf{S}| + 1$, and check that it is indeed a witness. The check can be done bottom up on the formula and is clearly polynomial, as it consists of label checks and standard set operations. Prop. 17 guarantees that the algorithm is correct, while Lem. 14 ensures that checking the bound k equal to N is equivalent to checking every bound in $\mathbb{N}$. This non-deterministic procedure is detailed in Algorithm 1. ◀

In order to finish the proof of Thm. 9, we show the following lower complexity bound.

▶ **Lemma 19.** *The universal model checking problem for $\mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$ is* coNP*-hard.*

This result is obtained as a reduction from the Boolean satisfiability problem to model checking, based on Figure 5. Somewhat classically, the reduction draws parallels between executions in this system and valuations over the Boolean variables $x_1, \ldots, x_n$, based on which states x_i are visited. The main novelty resides in the $\mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$ formula built in the reduction: as we cannot use reachability properties directly, we need to carefully encode “reaching x_i” in a roundabout way, that uses $\mathsf{F}^{\infty}_{\mathsf{P}}$ operators and the self-loops on each x_i.

Proof. Let $\bigwedge_{i=1}^{l} \bigvee_{j=1}^{3} l_{i,j}$ be an instance of 3-SAT over variables $x_1, \ldots, x_n$, where every literal $l_{i,j}$ is either a variable x or its negation $\bar{x}$. Consider the system $\mathcal{S}$ in Figure 5, and the $\mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$ formula $\varphi = \varphi_1 \vee \varphi_2$, obtained from $\varphi_1 = \bigvee_{i=1}^{n} (\mathsf{F}^{\infty}_{\mathsf{P}} \neg x_i \wedge \mathsf{F}^{\infty}_{\mathsf{P}} \neg \bar{x_i})$ and $\varphi_2 = \bigvee_{i=1}^{l} \bigwedge_{j=1}^{3} \mathsf{F}^{\infty}_{\mathsf{P}} \neg l_{i,j}$. We show that the 3-SAT formula is *unsatisfiable* (a coNP-hard problem) iff $\mathcal{S} \models \varphi$.

Algorithm 1 coNP algorithm for the universal model checking of $\mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$.

```
Data: An lts $\mathcal{S} = \langle \mathsf{S}, s_{\mathsf{init}}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \rangle$ and a formula $\varphi \in \mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$
Result: whether $\mathcal{S} \not\models \varphi$

GuessAndCheck $(\mathcal{S}, \varphi)$
    guess a finite path $\pi$ such that $|\pi| \leq (|\mathsf{S}| + 2)|\varphi|(|\mathsf{S}| + 1)$;
    guess a function $\mathsf{W} : \mathsf{SubF}(\varphi) \mapsto 2^{\mathbb{N}}$ such that $\max(\mathsf{W}) < |\pi|$;
    return $(0 \in \mathsf{W}(\varphi)) \wedge$ CheckW$(\mathcal{S}, \mathsf{W}, \varphi, \pi)$;
CheckW$(\mathcal{S}, \mathsf{W}, \varphi, \pi)$
    if $\varphi = \alpha$ then
        return $\forall i \in \mathsf{W}(\varphi), a \notin \mathsf{lbl}(\pi[i])$;
    else if $\varphi = \neg\alpha$ then
        return $\forall i \in \mathsf{W}(\varphi), a \in \mathsf{lbl}(\pi[i])$;
    else if $\varphi = \psi_1 \vee \psi_2$ then
        return $(\mathsf{W}(\varphi) = \mathsf{W}(\psi_1) \cap \mathsf{W}(\psi_2)) \wedge$ CheckW$(\mathcal{S}, \mathsf{W}, \psi_1, \pi)$ $\wedge$
          CheckW$(\mathcal{S}, \mathsf{W}, \psi_2, \pi)$;
    else if $\varphi = \psi_1 \wedge \psi_2$ then
        return $(\mathsf{W}(\varphi) = \mathsf{W}(\psi_1) \cup \mathsf{W}(\psi_2)) \wedge$ CheckW$(\mathcal{S}, \mathsf{W}, \psi_1, \pi)$ $\wedge$
          CheckW$(\mathcal{S}, \mathsf{W}, \psi_2, \pi)$;
    else if $\varphi = \mathsf{F}^{\infty}_{\mathsf{P}}\, \psi$ then
        return $(\exists 0 \leq i \leq |\pi|, \forall i \leq j \leq i + |\mathsf{S}| + 1, j \in \mathsf{W}(\psi)) \wedge$ CheckW$(\mathcal{S}, \mathsf{W}, \psi, \pi)$;
```

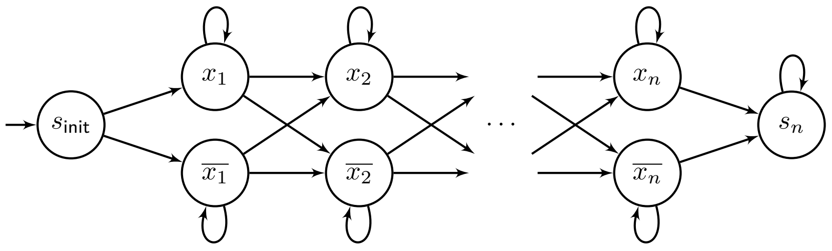

Figure 5 System used in Lem. 19. States are labeled by their name, *e.g.* $x_1 \in \mathsf{lbl}(x_1)$.

Assume that for every valuation ν, $\nu \not\models \bigwedge_{i=1}^{l} \bigvee_{j=1}^{3} l_{i,j}$, and thus $\nu \models \bigvee_{i=1}^{l} \bigwedge_{j=1}^{3} \bar{l}_{i,j}$. Every path from s_{init} to s_n can be seen as encoding a valuation ν, based on the visited variables. Then, for every run ρ in $\mathcal{S}$ that reaches s_n, we have $\rho \models \bigvee_{i=1}^{l} \bigwedge_{j=1}^{3} \mathsf{F}\, \bar{l}_{i,j}$. Note that a run that does not reach s_n (and gets stuck in one of the loops) also satisfies this formula, as removing F terms from conjunctions can only help. Moreover, we note that $\rho \models \mathsf{F}\, \bar{l}_{i,j}$ implies $\rho \models \mathsf{G}\, \neg l_{i,j}$, as a run of $\mathcal{S}$ cannot visit both a variable and its negation. Then, by definition of $\mathsf{F}^{\infty}_{\mathsf{P}}$ as $\mathsf{G}\,\mathsf{F}_{\mathsf{P}}$, we have that $\rho \models \mathsf{G}\, \neg l_{i,j}$ is equivalent with $(\rho, 0) \models \mathsf{F}^{\infty}_{\mathsf{P}}\, \neg l_{i,j}$. Therefore, for every run ρ, $(\rho, 0) \models \varphi_2$. This implies that there exists a k such that every run satisfies $\varphi_1 \vee \varphi_2$, *i.e.* $\mathcal{S} \models \varphi$.

Assume now that $\mathcal{S} \models \varphi$, *i.e.* there is a k so that every run satisfies $\varphi_1 \vee \varphi_2$. Let R be the set of runs that go through $\mathcal{S}$ by iterating every self-loop on the states x_i or $\bar{x}_i$ exactly $k+1$ times before continuing, until s_n is reached. Then, for every run $\rho \in R$ and every variable x, we have that either $(\rho, k) \models \mathsf{F}^{\infty}_{\mathsf{P}}\, \neg x$ holds (if x is not visited), or $(\rho, k) \models \mathsf{F}^{\infty}_{\mathsf{P}}\, \neg \bar{x}$ holds (if $\bar{x}$ is not visited). This is an exclusive either/or because every state that is seen is iterated $k+1$ times, so that $(\rho, k) \not\models \varphi_1$. It follows that the runs in R must all satisfy φ_2. Since a run in R satisfies $\mathsf{F}^\infty_\mathsf{P} \neg x$ iff it satisfies $\mathsf{F}\, \bar{x}$, every run in R satisfies $\bigvee_{i=1}^{l} \bigwedge_{j=1}^{3} \mathsf{F}\, \bar{l}_{i,j}$. Then, if we interpret the runs in R as valuations ν based on which variables are visited, we have $\nu \models \bigvee_{i=1}^{l} \bigwedge_{j=1}^{3} \bar{l}_{i,j}$ for every valuations ν, so that the 3-SAT formula $\bigwedge_{i=1}^{l} \bigvee_{j=1}^{3} l_{i,j}$ is unsatisfiable. ◀

3.4 Expressiveness under the fairness assumption

Let us focus our attention on the expressiveness of the prompt Muller fragment under fairness. It turns out that for this fragment, assuming fairness does not change the interpretation of a formula. In particular, the complexity of the model checking problem remains the same.

Indeed, we note the following property of $\mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$, obtained as a corollary of Prop. 17:

▶ **Corollary 20.** *Given a formula $\varphi \in \mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$, an LTS $\mathcal{S}$ and a bound $k \geq 0$, if there exists a run ρ so that $(\rho, k) \not\models \varphi$, then there is a finite path π such that for all $\rho' \in \mathsf{Cyl}(\pi)$, $(\rho', k) \not\models \varphi$.*

Thus, $\mathcal{S} \not\models \varphi$ implies for any choice of k the existence of an entire cylinder $\mathsf{Cyl}(\pi)$ of counterexamples for the bound k. The cylinder of a finite path must always have non-zero measure under $\mathbb{P}_\mathcal{S}$, by definition of our coin-toss process. As such, $\mathcal{S} \not\models \varphi$ implies $\mathbb{P}_\mathcal{S}(\{\rho) \in \mathsf{Runs}_\mathsf{init} \mid (\rho, k) \models \varphi\}) < 1$ for every k, so that $\mathcal{S} \not\models_\mathsf{AS} \varphi$. This proves the following theorem, as the reverse implication holds by definition.

▶ **Theorem 21.** *For all LTS $\mathcal{S}$ and every formula $\varphi \in \mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$, $\mathcal{S} \models \varphi$ iff $\mathcal{S} \models_\mathsf{AS} \varphi$.*

Let us give some alternative intuition as to why the complexity drop between the universal and fair model checking problems for Muller formulas does not extend under promptness:

- The study of an $\mathcal{L}(\mathsf{F}^\infty)$ formula in a system under the fairness assumption is based on the core principle that the only thing that matters is the "bottom SCC" the run ends up in, as $\mathsf{Inf}(\rho)$ is almost always equal to such a component. A PTIME algorithm is derived from this principle [21]. Crucially, this approach exploits the fact that a formula of the shape $\mathsf{F}^\infty \varphi$ is *prefix independent*, in the sense that any deviation made within a finite prefix of a run ρ will not change $\mathsf{Inf}(\rho)$, and thus will not change its satisfaction of $\mathsf{F}^\infty \varphi$.
- In the prompt setting however, a formula of the shape $\mathsf{F}^\infty_\mathsf{P} \varphi$ is clearly *not prefix independent*, as its satisfaction for a bound k is impacted by whether finite prefixes contain faulty windows of length k or not. Thus, the fairness assumption does not allow us to restrict the analysis of prompt Muller formulas to bottom SCCs in the same way.

4 Initialized systems

We now reflect on the expressiveness of the prompt Muller fragment. The motivation to replace $\mathsf{F}^\infty \varphi$ by $\mathsf{F}^\infty_\mathsf{P} \varphi$ was to reinforce the guarantees obtained by executions of the system: By enforcing a strong notion of regularity in the occurrences of φ, we prevent a good event φ from being seen infinitely often but with a vanishing frequency. Indeed, requiring $\mathsf{F}^\infty_\mathsf{P} \varphi$ is sufficient to imply a frequency for φ of at least $1/k$ for some k.

We now argue that this requirement may be "too strong", as some systems may be rejected despite enforcing this kind of non-zero frequency guarantee on the occurrences of φ.

▶ **Example 22.** Consider the system presented on the left in Figure 6, of atomic propositions a and b, and the prompt Muller formula $\varphi = \mathsf{F}^\infty_\mathsf{P}\, A \vee \mathsf{F}^\infty_\mathsf{P}\, B$. Despite satisfying the Muller formula $\mathsf{F}^\infty\, A \vee \mathsf{F}^\infty\, B$ (every run will either stay in a forever or jump to b and stay there forever), this system does not satisfy the prompt variant φ. However, in every run either a or b happen with a long-term average frequency of 1. the key difference here, once again, is that the long-term average frequency of an event is a prefix independent notion, unlike φ.

4.1 Towards prefix independence

We argue that being unaffected by the addition of a prefix is a desirable property for a specification: in practice, a system might require a "guarantee-less" initialization period before reaching a steady regime where stronger conditions can be enforced. We introduce a *new fragment* of pLTL, capturing specifications that allow the system to pass the initialization period while enforcing prompt Muller guarantees on the regularity of good events eventually. The intuition for this construction is based on the following reasoning:

- Coro. 20 means that the satisfaction of a formula in $\mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$ can always be proven false because of some finite path. From that point of view, prompt Muller formulas behave like safety conditions $\mathsf{G}\,\alpha$, as opposed to Muller formulas that behave like repeated reachability objectives. This is the crux of what prevents prefix independence.
- In order to allow a safety formula $\mathsf{G}\,\alpha$ to accommodate for an initialisation period in the system, a natural idea is to replace it with the formula $\mathsf{F}\,\mathsf{G}\,\alpha$, that is prefix independent.
- Following this intuition, we introduce the *initialized* variant of $\mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$, that contains formulas of the shape $\mathsf{F}\,\varphi$, with $\varphi \in \mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$.

▶ **Example 23.** Consider again the system on the left of Figure 6, and the prompt Muller formula $\varphi = \mathsf{F}^\infty_\mathsf{P}\, A \vee \mathsf{F}^\infty_\mathsf{P}\, B$. Despite not satisfying φ, this system does satisfy $\mathsf{F}\,\varphi$, as in every run ρ either $\mathsf{G}\,a$ or $\mathsf{G}\,b$ eventually holds, and thus $(\rho, 0) \models \mathsf{F}(\mathsf{F}^\infty_\mathsf{P}\, A \vee \mathsf{F}^\infty_\mathsf{P}\, B)$. Consider now the system on the right of Figure 6. There, the run $abaabbaaabbb\ldots$ witnesses that neither φ nor $\mathsf{F}\,\varphi$ are satisfied.

We must finally address a last point of detail. Even without promptness, a Muller formula in $\mathcal{L}(\mathsf{F}^\infty)$ may not be prefix-independent: indeed, a state formula with no F^∞ operators solely depends on the initial state. In more general terms, the presence of atomic propositions outside of the scope of an F^∞ prevents prefix independence, and must be forbidden. This is without loss of generality for $\mathcal{L}(\mathsf{F}^\infty)$ formulas, as replacing such atomic propositions by true or false depending on the initial state of the system can be done as a pre-processing step [23].

Formally, we introduce $\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})$ as formulas where every atomic proposition is in the scope of an $\mathsf{F}^\infty_\mathsf{P}$ operator. They are generated by the nonterminal φ in the following grammar:

$$\begin{aligned} \varphi &::= \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{F}^\infty_\mathsf{P}\, \psi \\ \psi &::= \alpha \mid \neg\alpha \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathsf{F}^\infty_\mathsf{P}\, \psi\ . \end{aligned}$$

For example, $\mathsf{F}^\infty_\mathsf{P}\,\mathsf{A}$ belongs to $\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})$, but $\mathsf{B} \vee \mathsf{F}^\infty_\mathsf{P}\,\mathsf{A}$ does not.

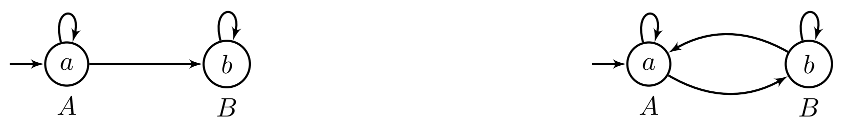

Figure 6 Two systems that satisfy $\mathsf{F}^\infty\, A \vee \mathsf{F}^\infty\, B$. The one on the left does not satisfy $\mathsf{F}^\infty_\mathsf{P}\, A \vee \mathsf{F}^\infty_\mathsf{P}\, B$ unless its "initialization period" is ignored. The rightmost one does not satisfy it either ways.

▶ **Definition 24.** *The* initialized *fragment* $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$ *is defined as* $\{\mathsf{F}\,\varphi \mid \varphi \in \mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\}$.

This fragment enjoys the property of prefix independence:

▶ **Proposition 25.** *If ρ is a run and $\mathsf{F}\,\varphi$ is a formula in $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$, then for any $k \geq 0$ and $i \geq 0$, $(\rho, k) \models \mathsf{F}\,\varphi$ iff $(\rho[i..], k) \models \mathsf{F}\,\varphi$.*

4.2 Universal model checking

In this section, we show that the universal model checking problem remains in coNP for $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$, by adapting the techniques we developed for $\mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$. However, we note that the lower bound is lost in the process, as the reduction detailed in Lemma 19 does not play well with $\mathsf{F}\,\varphi$ formulas.

▶ **Theorem 26.** *The universal model checking problem for $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$ is in* coNP.

The idea is to make use of the same pumpings and witness structures as Section 3 to build a short witness for $\mathcal{S} \not\models \mathsf{F}\,\varphi$, while taking into account the fact that only long-term behaviours of the system should matter. This means that a "faulty window" where a subformula ψ does not hold can only serve as counter-example to $\mathsf{F}(\mathsf{F}^\infty_\mathsf{P}\,\psi)$ if it can be reached after an arbitrarily long prefix. Intuitively, faulty windows that are reached infinitely often along a run fit that description, as any suffix of the run will eventually reach them. We show that these are the only witnesses that we need, and that they remain small in size:

▶ **Proposition 27.** *Given an LTS $\mathcal{S} = \left\langle \mathsf{S}, s_{\mathsf{init}}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \right\rangle$, a formula $\mathsf{F}\,\varphi \in \mathsf{F}(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P}))$ and a bound $k \geq 0$, there exists a run $\rho \in \mathsf{Runs}$ such that $(\rho, k) \not\models \mathsf{F}\,\varphi$ iff there exists a finite path π and a function W such that for all run $\pi\rho'$, W is a witness for $\pi\rho'$ and k, $|\pi| \leq (k+1)|\varphi|(|\mathsf{S}|+1)$, $\max(\mathsf{W}) < |\pi|$ and $\pi[0]$ is reachable from $\rho[0]$ and $\pi[|\pi|-1]$.*

This statement is very close to Prop. 17, the only difference is the last condition, that is $\pi[0]$ is reachable from $\rho[0]$ and $\pi[|\pi|-1]$, so that the finite path π can be repeated infinitely often, as part of a lasso $\pi_0(\pi\pi_1)^\omega$ for some finite paths π_0 and π_1.

Proof scheme. If $(\rho, k) \not\models \mathsf{F}\,\varphi$, then for every suffix of ρ we can apply Prop. 17 to get a short witness of $(\rho[i..], k) \not\models \varphi$. Note that these paths π and mappings W are all bounded in size by the same bounds on k and $|\varphi|$, so that there are finitely many of them. Eventually, we must visit the same state at two positions i and j far enough apart to enforce that the witness path π for $(\rho[i..], k) \not\models \varphi$ ends before j. Thus, we get $\pi[0]$ is reachable from $\rho[0]$ and $\pi[|\pi|-1]$.

The converse direction of the proof is straight-forward, as we can deduce from such a lasso-shaped witness run that $(\rho[i..], k) \not\models \varphi$ for infinitely many positions i. This implies $(\rho, k) \not\models \mathsf{F}\,\varphi$ by prefix-independence. ◀

The proof of Thm. 26 is then obtained from Prop. 27. More precisely, we use a variation of Algorithm 1, that guesses π and W and adds the extra reachability checks of Prop. 27 over the end-points of π. Notice that this extra step is polynomial and therefore is not detrimental to membership in coNP.

4.3 Fair model checking

Going back to the bigger picture, we recall that in the non-prompt setting a polynomial procedure is obtained for the Muller fragment under fairness. This is based on exploiting the prefix-independence property on the one side, and the propensity of almost all runs to maximise the strongly connected sets of states they visit infinitely often on the other side. As such, only runs visiting bottom SCCs (maximal strongly connected sets of states almost always reached by runs in probabilistic systems) are relevant under fairness.

The intuition is then that every such SCC is either entirely winning for a prefix-independent $\mathcal{L}(\mathsf{F}^\infty)$ objective φ, regardless of what initial state is picked, or entirely losing. Moreover, checking this fact for a given bottom SCC and a given formula is straightforward, as φ satisfaction is entirely determined by $\mathsf{Inf}(\rho)$ on a given run ρ, and $\mathsf{Inf}(\rho)$ is almost always equal to the full bottom SCC by fairness.

For a $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$ formula, we follow the same recipe: bottom SCCs will also be eventually reached, and will either be entirely winning for a given φ, or entirely losing for φ. The main difference brought by promptness is that we can no longer simply rely on fairness to make sure that every state of the SCC is visited infinitely often: in order to guarantee *prompt* visits, we must assume the fair coin to be adversarial and make sure that every state of the SCC *will be visited shortly* no matter what. This results in a kind of attractor computation to check if a bottom SCC is winning or not, that can then be paired with the algorithm from $\mathcal{L}(\mathsf{F}^\infty)$. This is the key to proving that indeed, prefix independence makes the fair model checking problem for $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$ polynomial.

▶ **Theorem 28.** *The fair model checking problem for* $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$ *is in* PTIME.

5 Discussion and conclusion

In this section we discuss our results, some immediate corollaries and future work. The first point we address might seem technical at a first glance, but we believe that it is worth mentioning; it concerns the quantification of the bound k used in the semantics of pLTL.

Strong against weak semantics. Recall that pLTL formulas are evaluated with respect to a uniform bound k, *i.e.* for a system to be a model to some formula, all its runs are evaluated using the same bound k. Let us refer to this semantics as *strong*. One could want to relax this semantics and define the following semantics: A system $\mathcal{S}$ satisfies a formula φ if for every run ρ there exists a bound k such that $(\rho, k) \models \varphi$. Let us call this semantics *weak*. Indeed one could argue that the strong semantics is too conservative and that a system designer might be interested in a more permissive behavior. This raises the following questions: *are both semantics equivalent?* Obviously, the strong semantics always implies the weak one. However, the converse does not hold. To separate these two semantics, consider the system depicted on the left of Figure 6, and the formula $\varphi = \mathsf{F}^\infty_\mathsf{P}\,\mathsf{A} \vee \mathsf{F}^\infty_\mathsf{P}\,\mathsf{B}$. This system satisfies the formula φ with respect to the weak semantics but not the strong one. This raises in turn another natural question: *Are these two semantics equivalent in some fragment?* We answer positively to this question by noticing that they collapse for formulas in $\mathsf{F}(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P}))$. This follows from the following observation:

The weak and strong semantics are equivalent for formulas in $\varphi \in \mathcal{L}(\mathsf{F}^\infty_\mathsf{P})$ *if they are evaluated over a strongly connected LTS.*

Indeed, if for every k there is a counter-example run ρ_k so that $(\rho_k, k) \not\models \varphi$, then by our small witness property (Prop. 17) we can get paths $\pi_1, \pi_2, \dots$ that are sufficient to falsify φ for $k = 1, 2, \dots$ respectively. It is then sufficient to chain them one after the other – by exploiting the strongly connected assumption – to construct a single path ρ that falsifies every bound k, as required by the weak semantics. As a consequence, both semantics are equivalent in our prefix independent fragment $\mathsf{F}\left(\mathcal{L}^+(\mathsf{F}^\infty_\mathsf{P})\right)$, where satisfaction for a run ρ is determined by the long-term behavior of ρ, *i.e.* the suffixes that stay confined to the strongly connected set $\mathsf{Inf}(\rho)$.

Probabilistic model checking. The last point we want to discuss is the complexity of quantitative verification for the fragment $\mathsf{F}(\mathcal{L}^+(\mathsf{F}_\mathsf{P}^\infty))$. Once again, our fragment behaves well. We argue that one can compute the *satisfaction probability* of a system with respect to a formula in $\mathsf{F}(\mathcal{L}^+(\mathsf{F}_\mathsf{P}^\infty))$ in polynomial time. This nice property follows from the following *zero-one-law* for the bottom SCCs of an LTS $\mathcal{S}$:

Let B be a bottom SCC and φ be a formula in $\mathcal{L}^+(\mathsf{F}_\mathsf{P}^\infty)$, then either almost all the runs of B satisfy φ or almost no run of B does.

This is a consequence of Coro. 20, as any finite path that witnesses the falsification of φ in a bottom SCC will eventually be visited with probability 1. Now to conclude, one has to notice that computing the probability of reaching B can be done in PTIME, and use the prefix independence of $\mathsf{F}(\mathcal{L}^+(\mathsf{F}_\mathsf{P}^\infty))$.

Future work. Finally, we hint at some future directions. The fragment $\mathsf{F}(\mathcal{L}^+(\mathsf{F}_\mathsf{P}^\infty))$ turned out to behave well in the presence of fairness, leading to a tractable model checking procedure. While, this is an improvement over the coNP procedure for the universal problem, we are still missing a matching lower bound to separate the two formally. A more ambitious perspective is the study of the controller synthesis problem induced by these fragments in 2-player games.

References

1 Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

2 Thomas Brihaye, Florent Delgrange, Youssouf Oualhadj, and Mickael Randour. Life is random, time is not: Markov decision processes with window objectives. *Log. Methods Comput. Sci.*, 16(4), 2020. URL: https://lmcs.episciences.org/6975.

3 Véronique Bruyère, Quentin Hautem, and Mickael Randour. Window parity games: an alternative approach toward parity games with time bounds. In Domenico Cantone and Giorgio Delzanno, editors, *Proceedings of the Seventh International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2016, Catania, Italy, 14-16 September 2016*, volume 226 of *EPTCS*, pages 135–148, 2016. doi:10.4204/EPTCS.226.10.

4 Damien Busatto-Gaston, Youssouf Oualhadj, Léo Tible, and Daniele Varacca. Fairness and promptness in muller formulas, 2024. arXiv:2204.13215.

5 Diego Calvanese, Giuseppe De Giacomo, and Moshe Y. Vardi. Reasoning about actions and planning in LTL action theories. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 593–602. Morgan Kaufmann, 2002.

6 Krishnendu Chatterjee. Concurrent games with tail objectives. *Theor. Comput. Sci.*, 388(1-3):181–198, 2007. doi:10.1016/j.tcs.2007.07.047.

7 Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. *Inf. Comput.*, 242:25–52, 2015. doi:10.1016/j.ic.2015.03.010.

8 Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986. doi:10.1145/5397.5399.

9 Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995. doi:10.1145/210332.210339.

10 E. Allen Emerson and Chin-Laung Lei. Modalities for model checking: Branching time logic strikes back. *Sci. Comput. Program.*, 8(3):275–306, 1987. doi:10.1016/0167-6423(87)90036-0.

11 Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge.* MIT Press, Cambridge, MA, USA, 2003.

12 Giuseppe De Giacomo and Moshe Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In Susanne Biundo and Maria Fox, editors, *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, pages 226–238. Springer, 1999. doi:10.1007/10720246_18.

13 Hugo Gimbert and Florian Horn. Solving simple stochastic tail games. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 847–862. SIAM, 2010. doi:10.1137/1.9781611973075.69.

14 Hugo Gimbert and Edon Kelmendi. Two-player perfect-information shift-invariant submixing stochastic games are half-positional. *CoRR*, abs/1401.6575, 2014. arXiv:1401.6575.

15 Orna Kupferman, Nir Piterman, and Moshe Vardi. From liveness to promptness. *Formal Methods in System Design*, 34, April 2009. doi:10.1007/s10703-009-0067-z.

16 Anca Muscholl and Igor Walukiewicz. An np-complete fragment of LTL. *Int. J. Found. Comput. Sci.*, 16(4):743–753, 2005. doi:10.1142/S0129054105003261.

17 Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.

18 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989*, pages 179–190. ACM Press, 1989. doi:10.1145/75277.75293.

19 Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, July 11-15, 1989, Proceedings*, volume 372 of *Lecture Notes in Computer Science*, pages 652–671. Springer, 1989. doi:10.1007/BFb0035790.

20 Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *International Symposium on Programming, 5th Colloquium, Torino, Italy, April 6-8, 1982, Proceedings*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 1982. doi:10.1007/3-540-11494-7_22.

21 Matthias Schmalz, Hagen Völzer, and Daniele Varacca. Model checking almost all paths can be less expensive than checking all paths. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New Delhi, India, December 12-14, 2007, Proceedings*, volume 4855 of *Lecture Notes in Computer Science*, pages 532–543. Springer, 2007. doi:10.1007/978-3-540-77050-3_44.

22 A. Sistla and Edmund Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32:733–749, July 1985. doi:10.1145/800070.802189.

23 Hagen Völzer and Daniele Varacca. Defining fairness in reactive and concurrent systems. *J. ACM*, 59(3):13:1–13:37, 2012. doi:10.1145/2220357.2220360.

A Multi-pumpings and their properties

▶ **Proposition 13.** *Given an lts $\mathcal{S}$, a run $\rho \in \mathsf{Runs}$, a multi-pumping ρ' of ρ, a formula $\varphi \in \mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$ and $k \geq 0$, if $(\rho, k) \not\models \varphi$, then $(\rho', k) \not\models \varphi$.*

In order to prove Prop. 13. we establish some structural properties of multi-pumpings.

A multi-pumping of a run can be seen as pumping multiple loops of the same run, but with an ordering of the loops first. Indeed, loops are pumped in order, one after the other, and not one inside another. This implies that merging two multi-pumpings is quite easy. It suffices to respect the order of the loops, as illustrated in Figure 7. This is formalise by the following lemma.

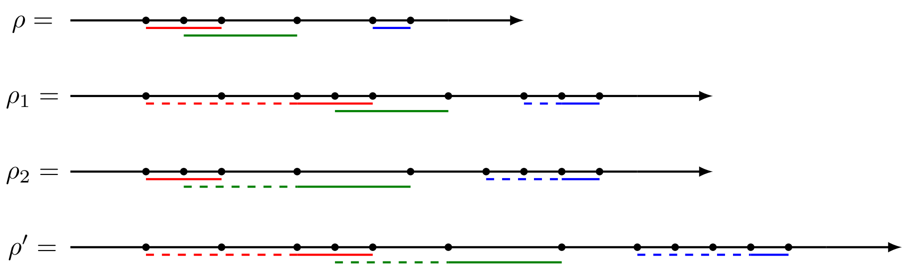

Figure 7 Illustration of the merging of two multi-pumpings, as per the construction of Lem. 29.

▶ **Lemma 29.** *Given an LTS $\mathcal{S}$ and a run $\rho \in$ Runs, let ρ_1 and ρ_2 be two multi-pumpings of ρ. Then, there exists a multi-pumping ρ' of ρ such that ρ' is also a multi-pumping of ρ_1 and of ρ_2.*

The idea behind the proof is to do all the pumpings of ρ_1 and ρ_2 in the right order in order to merge the two pumpings.

Proof. As ρ_1 and ρ_2 are two multi-pumping of ρ, there are n_1 and n_2 such that ρ_1 is a n_1-pumping of ρ and ρ_2 is a n_2-pumping of ρ. Let us prove the result by induction over $n_1 + n_2$.

- $n_1 + n_2 = 0$: In that case $n_1 = n_2 = 0$ and $\rho_1 = \rho_2 = \rho$ are trivial pumpings. Therefore the result trivially holds.
- Assume now that $n_1 + n_2 \geq 1$. If $n_1 = 0$, then $\rho_1 = \rho$ and the result trivially holds. The same is true if $n_2 = 0$. Now assume that $n_1 > 0$ and $n_2 > 0$. Then by definition $\exists i_1 < j_1, \rho[i_1] = \rho[j_1]$ such that there exists a $(n_1 - 1)$-pumping $\tilde{\rho}_1$ of $\rho[i_1..]$ and $\rho_1 = \rho[..i_1]\rho[i_1..j_1]^{l_1-1}\tilde{\rho}_1$, where $l_1 > 0$. In the same way, $\exists i_2 < j_2, \rho[i_2] = \rho[j_2]$ such that there exists a $(n_2 - 1)$-pumping $\tilde{\rho}_2$ of $\rho[i_2..]$ and $\rho_2 = \rho[..i_2]\rho[i_2..j_2]^{l_2-1}\tilde{\rho}_2$, where $l_2 > 0$. Without loss of generality assume that $i_1 \leq i_2$. By construction, as $i_1 \leq i_2$, $\rho_2[i_1..]$ is a n_2-pumping of $\rho[i_1..]$, and $\tilde{\rho}_1$ is by definition a $(n_1 - 1)$-pumping of $\rho[i_1..]$. By induction hypothesis, there exists a multi-pumping $\tilde{\rho}'$ of $\rho[i_1..]$ that is also a multi-pumping of $\rho_2[i_1..]$ and $\tilde{\rho}_1$. Then, by construction, the run $\rho' = \rho[..i_1]\rho[i_1..j_1]^{l_1-1}\tilde{\rho}'$ is a multi-pumping of both ρ_1 and ρ_2. ◀

Note that the merging ρ' is not unique. For example, if $i_1 = i_2$, one can do the pumping in any order and the resulting pumping will work, inducing two different multi-pumping. In the following, when we refer to the merging of two multi-pumping, we fix an arbitrary one.

▶ **Lemma 30.** *Given an LTS $\mathcal{S}$, a run $\rho \in$ Runs and ρ' a multi-pumping of ρ, let i_0 be the first index of a repeating state in ρ, that is, there exists a $0 \leq i' < i_0$ such that $\rho[i'] = \rho[i_0]$ and for all $0 \leq j < j' < i_0$, $\rho[j] \neq \rho[j']$. Then, $\rho'[0..i_0 + 1] = \rho[0..i_0 + 1]$.*

The intuitive idea behind this lemma is that a pumping can not modify the run before the first loop.

▶ **Lemma 31.** *Given an LTS $\mathcal{S}$, a finite path π, and two runs $\pi\rho \in$ Runs, and $\pi\rho' \in$ Runs such that ρ' is a multi-pumping of ρ, then $\pi\rho'$ is a multi-pumping of $\pi\rho$.*

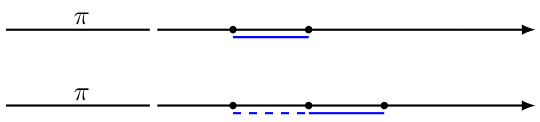

Figure 8 Graphical representation of Lem. 31.

▶ **Lemma 32.** *Given an lts $\mathcal{S}$, a run $\rho \in$ Runs, and a pumping ρ' of ρ such that $\rho' = \rho[0..i_1]\rho[i_1..i_2]^l\rho[i_2..]$, then one of the following holds for any position $j \geq 0$:*

- *$\rho'[j..]$ is a pumping of $\rho[j..]$.*
- *There is $i_1 \leq j' \leq j$ such that $\rho'[j..] = \rho[j'..]$.*
- *There is a cycle π and $i_1 \leq j' \leq j$ such that $\rho'[j..] = \pi\rho[j'..]$.*

The idea of this lemma is that every position of a pumping can be matched to a position of the original run. Depending on whether the position is before the added loop, in the middle of it or afterwards, the structure is a bit different, but the matching still holds. Mainly, this lemma is a tool used in other proofs. As the proof is rather technical, we will explain the idea of this lemma with some figures.

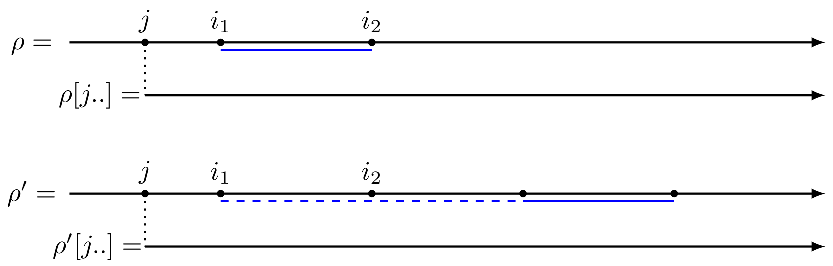

Figure 9 First case : $j < i_1$.

Consider the run ρ and the position j describe in Figure 9, and the pumping ρ' of ρ where two loop $i_1 \cdots i_2$ are added. Clearly, $\rho[..j] = \rho'[..j]$ and thus $\rho'[j..]$ is a pumping of $\rho[j..]$.

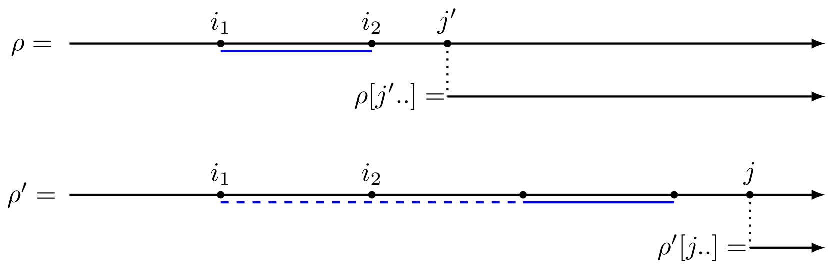

Figure 10 Second case : $j > i_2 + (i_2 - i_1)(l - 1)$.

Now, consider the same run and same pumping, but consider that $j > i_2 + (i_2 - i_1)(l - 1)$, as represented in Figure 10. Here, two loops are added, therefore $l = 2$. As j is after every added loops in ρ', one can find a j' such that $\rho'[j..] = \rho[j'..]$. Note that if j is in the last loop $i_1 \cdots i_2$, then $j > i_2 + (i_2 - i_1)(l - 1)$ and this idea still works.

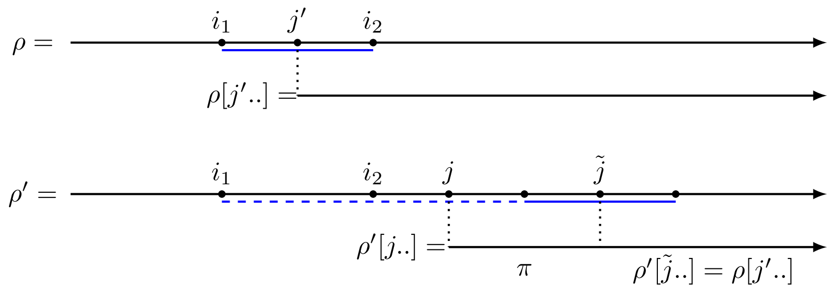

Figure 11 Third case : $i_1 \leq j \leq i_2 + (i_2 - i_1)(l - 1)$.

The last case is the most technical. Consider the same run ρ and same pumping ρ', but with $i_1 \leq j \leq i_2 + (i_2 - i_1)(l - 1)$, that is, j is somewhere inside an added loop, as described in Figure 11. Then, $\rho'[j..]$ may not be a pumping of any $\rho[j'..]$, as $\rho'[j..]$ starts with a bit of the loop, then has some complete occurrences of the loop, and then continues in the same way as ρ. Yet, one can consider the last occurrence of $\rho'[j]$, that is, the same vertex but in the last loop, let say $\rho'[\tilde{j}]$. Then, $\rho'[j..]$ can be described as a loop $\pi = \rho'[j..\tilde{j}]$ followed by $\rho'[\tilde{j}..]$. Then, by the same argument as in the second case, there exists a j' such that $\rho[j'..] = \rho'[\tilde{j}..]$.

Those ideas are formalised in the following proof.

Proof of Lem. 32. Let us look at every possibility.

Firstly, assume that $j < i_1$. Then, $\rho'[j..] = \rho[j..i_1]\rho[i_1..i_2]^l\rho[i_2..]$. By definition, $\rho[j..]$ is a pumping of $\rho[j..]$.

Secondly, assume that $j > i_2 + (i_2 - i_1)(l - 1)$. Then, let $j' = j - (i_2 - i_1)(l - 1)$. We have $j' > i_2$, and thus by construction $\rho'[j..] = \rho[j'..]$.

Thirdly, we have $i_1 \leq j \leq i_2 + (i_2 - i_1)(l - 1)$. There is a $0 \leq l' \leq l - 2$ such that

$$i_1 + (i_2 - i_1)l' \leq j \leq i_1 + (i_2 - i_1)(l' + 1).$$

Let $j' = j - (i_2 - i_1)l'$, and let us write

$$\rho' = \rho[0..i_1]\rho[i_1..i_2]^{l'}\rho[i_1..j']\rho[j'..i_2]\rho[i_1..i_2]^{l-l'-2}\rho[i_1..j']\rho[j'..i_2]\rho[i_2..]\,.$$

Note that $\rho'[j..] = \rho[j'..i_2]\rho[i_1..i_2]^{l-l'-2}\rho[i_1..j']\rho[j'..i_2]\rho[i_2..]$.
Let $\pi = \rho[j'..i_2]\rho[i_1..i_2]^{l-l'-2}\rho[i_1..j']$. Note that $\pi[0] = \rho[j'] \in \mathsf{Succ}(\pi[|\pi| - 1])$ and π is a cycle. Therefore, $\rho'[j..] = \pi\rho[j'..]$. ◀

Now, we show some properties of pumpings in regard to the satisfaction of formulas. Those properties are necessary in order to use pumpings and still keep satisfaction, or invalidation, of the formula.

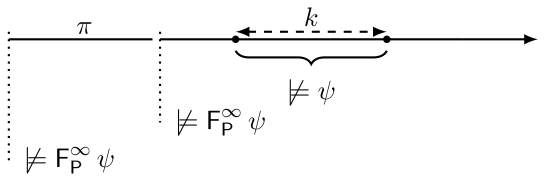
π
k
$\not\models \psi$
$\not\models \mathsf{F}^{\infty}_{\mathsf{P}}\,\psi$
$\not\models \mathsf{F}^{\infty}_{\mathsf{P}}\,\psi$

Figure 12 Visual representation of Lem. 33.

▶ **Lemma 33.** *Given an LTS $\mathcal{S}$, a finite path π, a run $\rho = \pi\rho' \in \mathsf{Runs}$, a bound $k \geq 0$, and a formula $\varphi \in \mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$, if $\rho'[0] = \pi[0]$ and $(\rho', k) \not\models \varphi$, then $(\rho, k) \not\models \varphi$.*

Intuitively, the idea is that adding a finite prefix to a run will not remove any faulty window, just postpone them, as can be seen in Figure 12. For state formulas, we have to check that the initial state stays the same.

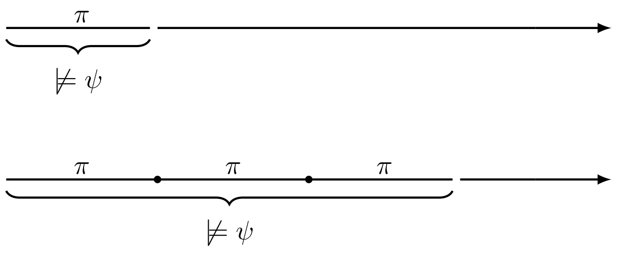

Figure 13 Visual representation of Lem. 34.

▶ **Lemma 34.** *Given an LTS $\mathcal{S}$, a finite path π, a run $\rho = \pi\rho' \in \mathsf{Runs}$ such that $\pi[0] = \rho'[0]$, a bound $k \geq 0$, a formula φ, and the run $\rho_l = \pi^l\rho'$ for $l > 0$, if for all $0 \leq j \leq |\pi|$, $(\rho[j..], k) \not\models \varphi$, then for all $0 \leq j' \leq l \times |\pi|$, $(\rho_l[j..], k) \not\models \varphi$.*

This lemma is quite technical and used only as a tool in other proofs. The idea is that if a loop does not satisfy a formula, no iteration of that loop can satisfy the formula, as represented in Figure 13. It is a direct corollary of Lem. 33, as formally shown in the following proof.

Proof of Lem. 34. First, notice that $\pi[0] = \rho'[0]$ ensures that $\pi^l\rho'$ is indeed a valid run in the system. Now, consider $0 \leq j' \leq l \times |\pi|$, and $j = j'$ modulo $|\pi|$. As $\rho_l[((l-1) \times |\pi|)..] = \rho$ by definition of ρ_l, we have that $\rho_l[((l-1) \times |\pi| + j)..] = \rho[j..]$. Moreover, by definition, $0 \leq j \leq |\pi|$, and $\rho_l[j'] = \rho[j]$. By noting that $\rho_l[j'..] = \rho_l[j'..((l-1) \times |\pi| + j)]\rho[j..]$ and $(\rho[j..], k) \not\models \varphi$, Lem. 33 is enough to conclude. ◀

We have now presented every tool needed to prove Prop. 13

Proof of Prop. 13. By definition, a multi-pumping is a n-pumping for some $n \geq 0$, that is, a multi-pumping is a succession of a finite number of pumpings. Therefore, to show the result for ρ' a pumping of ρ is enough to conclude with a structural induction.

Let us prove by structural induction over φ that if ρ' is a pumping of ρ, then $(\rho, k) \not\models \varphi$ implies that $(\rho', k) \not\models \varphi$.

- $\varphi = \alpha$ or $\varphi = \neg\alpha$: Then the result trivially holds as $\rho[0] = \rho'[0]$ by construction.
- $\varphi = \psi_1 \vee \psi_2$: By definition, $(\rho, k) \not\models \varphi$ if $(\rho, k) \not\models \psi_1$ and $(\rho, k) \not\models \psi_2$. By induction hypothesis, this implies that $(\rho', k) \not\models \psi_1$ and $(\rho', k) \not\models \psi_2$, and therefore $(\rho', k) \not\models \varphi$.
- $\varphi = \psi_1 \wedge \psi_2$: By definition, $(\rho, k) \not\models \varphi$ if $(\rho, k) \not\models \psi_1$ or $(\rho, k) \not\models \psi_2$. Without loss of generality assume that $(\rho, k) \not\models \psi_1$. By induction hypothesis, this implies that $(\rho', k) \not\models \psi_1$ and therefore $(\rho', k) \not\models \varphi$.
- $\varphi = \mathsf{F}^{\infty}_{\mathsf{P}}\,\psi$: By definition, there exists a position $i \geq 0$ such that for all $i \leq j \leq i + k$, $\rho[j..] \not\models \psi$. Write $\rho' = \rho[0..i_1]\rho[i_1..i_2]^l\rho[i_2..]$ with $l > 0$ and $i_1 < i_2$. Firstly, if $i_1 < i$ then for all $i \leq j \leq i+k$, $\rho'[(j+(i_2-i_1)(l-1))..] = \rho[j..]$. Therefore for all $i+(i_2-i_1)(l-1) \leq j \leq i + (i_2 - i_1)(l-1) + k$, $\rho'[j..] \not\models \psi$. Secondly, we have $i \leq i_1$. Let us show that $\forall i \leq j \leq i + k$, $\rho'[j..] \not\models \psi$. By Lem. 32, there are three cases.

- If $\rho'[j..]$ is a pumping of $\rho[j..]$, then by induction hypothesis $\rho'[j..] \not\models \psi$.
- If there exists $i_1 \leq j' \leq j$ such that $\rho'[j..] = \rho[j'..]$ then, as $i \leq i_1$, we have that $i \leq j' \leq i + k$, so that $\rho[j'..] \not\models \psi$, and thus $\rho'[j..] \not\models \psi$.
- If there exists a cycle π and $i_1 \leq j' \leq j$ such that $\rho'[j..] = \pi\rho[j'..]$, then once again $i \leq j' \leq i + k$, so that $\rho[j'..] \not\models \psi$, and by Lem. 33 $\rho'[j..] \not\models \psi$. ◀

▶ **Lemma 14.** *Let $\mathcal{S} = \langle \mathsf{S}, s_{\mathsf{init}}, \mathsf{T}, \mathsf{lbl} \colon \mathsf{S} \to 2^{\mathsf{AP}} \rangle$ be an LTS, $\varphi \in \mathcal{L}(\mathsf{F}^{\infty}_{\mathsf{P}})$, and let $N = |S| + 1$. If there is $\rho_N \in \mathsf{Runs}_{\mathsf{init}}$ such that $(\rho_N, N) \not\models \varphi$ then for all $k \geq N$, there is a multi-pumping ρ_k of ρ_N such that $(\rho_k, k) \not\models \varphi$.*

Proof. By structural induction over φ.

- $\varphi = \alpha$ or $\varphi = \neg\alpha$: The result trivially holds as the bound is irrelevant to the satisfaction of φ, and ρ_N is a multi-pumping of itself.
- $\varphi = \psi_1 \vee \psi_2$: By definition, $(\rho_N, N) \not\models \varphi$ if $(\rho_N, N) \not\models \psi_1$ and $(\rho_N, N) \not\models \psi_2$. By induction hypothesis, for all $k \geq N$, there are two multi-pumpings ρ^1_k and ρ^2_k of ρ_N such that $(\rho^1_k, k) \not\models \psi_1$ and $(\rho^2_k, k) \not\models \psi_2$. By lemma 29, there exists a multi-pumping ρ_k of ρ_N such that ρ_k is also a multi-pumping of ρ^1_k and ρ^2_k. Then, by lemma 13, $(\rho_k, k) \not\models \psi_1$ and $(\rho_k, k) \not\models \psi_2$. Therefore, $(\rho_k, k) \not\models \psi_1 \vee \psi_2$.
- $\varphi = \psi_1 \wedge \psi_2$: By definition, $(\rho_N, N) \not\models \varphi$ if $(\rho_N, N) \not\models \psi_1$ or $(\rho_N, N) \not\models \psi_2$. Without loss of generality, suppose that $(\rho_N, N) \not\models \psi_1$. By induction hypothesis, for all $k \geq N$, there exists a multi-pumping ρ_k of ρ_N such that $(\rho_k, k) \not\models \psi_1$. Then, by definition, $(\rho_k, k) \not\models \psi_1 \wedge \psi_2$.
- $\varphi = \mathsf{F}^{\infty}_{\mathsf{P}}\,\psi$: By definition, if $(\rho_N, N) \not\models \varphi$ then $\exists i, \forall 0 \leq j \leq N, (\rho_N[(i+j)..], N) \not\models \psi$. As $N > |\mathsf{S}|$, there are two positions i_1, i_2 with $i \leq i_1 < i_2 \leq i + N$ such that $\rho_N[i_1] = \rho_N[i_2]$. Moreover, one can assume w.l.o.g. that i_1 and i_2 are chosen such that the finite path $\rho_N[i_1..i_2]$ is such that for each $i_1 \leq j_1 < j_2 \leq i_2 - 1$, $\rho_N[j_1] \neq \rho_N[j_2]$. For each $i_1 \leq j \leq i_2$, we have $(\rho_N[j..], N) \not\models \psi$. For each $i_1 \leq j \leq i_2$, we can apply the induction hypothesis to obtain a multi-pumping ρ^j_k of $\rho_N[j..]$ such that $(\rho^j_k, k) \not\models \psi$. By Lem. 31, for each $i_1 \leq j \leq i_2$, we have that $\rho_N[i_1..j]\rho^j_k$ is a multi-pumping of $\rho_N[i_1..]$. By applying Lem. 29 multiple times, there exists a run $\tilde{\rho}_k$ such that for each $i_1 \leq j \leq i_2$, $\tilde{\rho}_k$ is a multi-pumping of $\rho_N[i_1..j]\rho^j_k$. Moreover, observe that it is not possible to have a loop between i_1 and i_2 in ρ. Therefore, by construction, for each $i_1 \leq j \leq i_2$, we have that $\tilde{\rho}_k[(j - i_1)..]$ is a multi-pumping of ρ^j_k and therefore, by Prop. 13, $(\tilde{\rho}_k[(j - i_1)..], k) \not\models \psi$. Consider the run $\tilde{\rho}'_k = \tilde{\rho}_k[0..(i_2 - i_1)]^k \tilde{\rho}_k[(i_2 - i_1)..]$. By Lem. 34, for all $0 \leq j \leq k \times (i_2 - i_1)$, $(\tilde{\rho}'_k[j..], k) \not\models \psi$. Consider the run $\rho_k = \rho_N[0..i_1]]\tilde{\rho}_k$. Then, as $i_2 > i_1$, $k \times (i_2 - i_1) > k$, for each $0 \leq j \leq k$, $\rho_k[(i_1 + j)..] = \tilde{\rho}'_k[j..]$, and therefore $(\rho_k[(i_1 + j)..], k) \not\models \psi$. By definition, $(\rho_k, k) \not\models \mathsf{F}^{\infty}_{\mathsf{P}}\,\psi$. ◀