Model-checking Counting Temporal Logics on Flat Structures

Normann Decker¹, Peter Habermehl², Martin Leucker¹, Arnaud Sangnier², and Daniel Thoma¹

¹ ISP, University of Lübeck, Lübeck, Germany
{decker, leucker, thoma}@isp.uni-luebeck.de
² IRIF, Univ. Paris Diderot, Paris, France
{habermehl, sangnier}@irif.fr

Abstract
We study several extensions of linear-time and computation-tree temporal logics with quantifiers that allow for counting how often certain properties hold. For most of these extensions, the model-checking problem is undecidable, but we show that decidability can be recovered by considering flat Kripke structures where each state belongs to at most one simple loop. Most decision procedures are based on results on (flat) counter systems where counters are used to implement the evaluation of counting operators.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Counting Temporal Logic, Model checking, Flat Kripke Structure

Introduction
Model checking [8] is a method to verify automatically the correct behaviour of systems. It takes as input a model of the system to be verified and a logical formula encoding the specification and checks whether the behaviour of the model satisfies the formula. One key aspect of this method is to find the appropriate balance between expressiveness of models and logical formalisms and efficiency of the model-checking algorithms. If the model is too expressive, e.g. Turing machines, then the model-checking problem, even with very simple logical formalisms, becomes undecidable. On the other hand, some expressive logics have been proposed in order to reason on the temporal executions of simple models such as Kripke structures. This is the case for the linear temporal logic LTL [22] and the branching-time temporal logics CTL [7] and CTL*[14], for which the model-checking problem has been shown to be PSPACE-complete, contained in P and PSPACE-complete, respectively (see, e.g., [3]).

Even though these logical formalisms allow for stating classical properties like safety or liveness over executions of Kripke structures, their expressiveness is limited. In particular they cannot describe quantitative aspects, as for instance the fact that a property has been true twice as often as another along an execution. One approach to solve this issue is to extend the logic with some ability to count positions of an execution satisfying some property and to check constraints over such numbers at some positions. Such a counting extension is proposed in [19] for CTL leading to a logic denoted here as cCTL. This formalism can state properties such as an event p will eventually occur and before that, the number of events...
q is larger than two. The authors propose further an extension called (here) \(\mathcal{CTL}_\pm\) that admits diagonal comparisons (i.e., negative and positive coefficients) to state, for instance that the number of events \(b\) is greater than the number of events \(c\). It is shown that the model-checking problem for \(\mathcal{CTL}\) is decidable in polynomial time and that the satisfiability problem for \(\mathcal{CTL}_\pm\) is undecidable. A similar extension for LTL is considered in [18] where it is proven that model checking of \(\mathcal{LTL}\) is \(\expSpace\)-complete while that of \(\mathcal{LTL}_\pm\) is undecidable.

Following the same motivation, regular availability expressions (RAE) were introduced in [10] extending regular expressions by a mechanism to express that on a (sub-)word matching an expression specific letters occur with a given relative frequency. Unfortunately, emptiness of the intersection of two such expressions was shown undecidable. Even for single expressions only a non-elementary procedure is known for verification (inclusion in regular languages) and deciding emptiness [1]. The case is similar for the logic \(\mathfrak{fLTL}\) [5], a variant of LTL that features an until operator extended by a frequency constraint. The operator is intended to relax the classical semantics where \(\varphi \mathcal{U} \psi\) requires \(\varphi\) to hold at all positions before \(\psi\). For example, the \(\mathfrak{fLTL}\) formula \(p \mathcal{U}^\frac{1}{3} q\) states that \(q\) holds eventually and before that the proportion of positions satisfying \(p\) should be at least one third. The concept of relative frequencies embeds naturally into the context of counting logics as it can be understood as a restricted form of counting. In fact, \(\mathfrak{fLTL}\) can be considered as a fragment of \(\mathcal{LTL}_\pm\) and still has an undecidable satisfiability problem [5] implying the same for model-checking Kripke structures. Moreover, most techniques employed for obtaining results on RAE as well as \(\mathfrak{fLTL}\) involve variants of counter systems.

Looking at the model-checking problem from the model point of view, recent work has shown that restrictions can be imposed on Kripke structures to obtain better complexity bounds. As a matter of fact if the structure is flat (or weak), which means every state belongs to at most one simple cycle in the graph underlying the structure, then the model-checking problem for LTL becomes \(\text{NP}\)-complete [17]. Such a restriction has as well been successfully applied to more complex classes of models. It is well known that the reachability problem for two-counter systems is undecidable [21] whereas for flat systems the problem is decidable for any number of counters [15], even more, model checking of LTL is \(\text{NP}\)-complete [11]. Flat structures are not only interesting because of their algorithmic properties, but also because they can be used as a way to under-approximate the behaviour of non-flat systems. For instance for counter systems one gets a semi-decision procedure for the reachability problem which consists in enumerating flat sub-systems and testing for reachability. In simple words, flat structures can be understood as an extension of paths typically used in bounded model checking and we expect that bounded model checking using flat structures rather than paths improves practical model checking approaches.

**Contributions.** We consider the model-checking problem for a counting logic that we call \(\mathcal{CCTL}^*\) where we use variables to mark positions on a run from where we begin to count the number of times a subformula is satisfied. Such a way of counting was also introduced in [19], see Section 2.2 for a comparison. We study as well its fragments \(\mathfrak{fCTL}\), \(\mathfrak{fLTL}\) and \(\mathfrak{fCTL}^*\) where the explicit counting mechanism is replaced by a generalized version of the until operator capable of expressing frequency constraints.

First we prove that \(\mathfrak{fCTL}\) model checking is at most exponential in the formula size and polynomial in the structure size by using an algorithm similar to the one for CTL model checking. To deal with frequency constraints a counter is employed for tracking the number of times a subformula is satisfied in a run of a Kripke structure. We then show that for flat Kripke structures the model-checking problems of \(\mathfrak{fLTL}\) and \(\mathcal{CCTL}^*\) are decidable. For the
We write \( N \) for integer counters and simple guards. A Kripke structure is denoted by \( K \), and a path is the graph of \( K \)’s runs. A \( K \)’s run is a sequence \( u = s_0 s_1 \ldots s_k \) that is equal to \( s_i \) implies \( s_i \neq s_j \) for all \( i, j \in [0, k] \) and \( (s_i, s_i+1) \in E \). A Kripke structure is called flat if for each state \( s \in S \) there is at most one simple loop \( u \) in \( K \) with \( u(0) = s \). See Fig. 1 for an example. The classes of all Kripke structures and all flat Kripke structures are denoted KS and FKS, respectively.

Counter systems. Our proofs use systems with integer counters and simple guards. A counter system is a tuple \( S = (S, s_I, C, \Delta) \) where \( S \) is a finite set of control states, \( s_I \in S \) is the initial control state, \( E \subseteq S \times S \) the set of edges and \( \Delta : S \rightarrow 2^{\text{AP}} \) the labelling function. A finite path in \( K \) is a sequence \( u = s_0 s_1 \ldots s_k \in S^+ \) with \( (s_i, s_{i+1}) \in E \) for all \( i \in [0, k-1] \). Infinite paths are defined analogously. A run \( \rho \) of \( K \) is an infinite path with \( \rho(0) = s_I \). We denote by \( \text{Runs}(K) \) the set of runs of \( K \). Due to the single initial state, we assume without loss of generality that the graph of \( K \) is connected, i.e. all states are reachable. A simple loop in \( K \) is a finite path \( u = s_0 s_1 \ldots s_k \) such that \( i \neq j \) implies \( s_i \neq s_j \) for all \( i, j \in [0, k] \) and \( (s_i, s_{i+1}) \in E \). A Kripke structure \( K \) is a counter system if for each state \( s \in S \) there is at most one simple loop \( u \) in \( K \) with \( u(0) = s \). See Fig. 1 for an example. The classes of all Kripke structures and all flat Kripke structures are denoted KS and FKS, respectively.

Table 1 Complexity characterisation of the model-checking problems of fragments of CCTL*. PH indicates polynomial reducibility to the (decidable) satisfiability problem of PH.

2 Definitions

2.1 Preliminaries

We write \( \mathbb{N} \) and \( \mathbb{Z} \) to denote the sets of natural numbers (including zero) and integers, respectively, and \([i, j]\) for \( \{k \in \mathbb{Z} \mid i \leq k \leq j\} \). We consider integers encoded with a binary representation. For a finite alphabet \( \Sigma \), \( \Sigma^* \) represents the set of finite words over \( \Sigma \), \( \Sigma^+ \) the set of finite non-empty words over \( \Sigma \) and \( \Sigma^\omega \) the set of infinite words over \( \Sigma \). For a finite set \( E \) of elements, \( |E| \) represents its cardinality. For \( (\text{finite or infinite}) \) words and general sequences \( u = a_0 a_1 \ldots a_n \ldots \) of length at least \( k+1 > 0 \) we denote by \( u(k) = a_k \) the \( (k+1) \)-th element and refer to its indices \( 0, 1, \ldots \) as positions on \( u \). If \( u \) is finite then \( |u| \) denotes its length. For arbitrary functions \( f : A \rightarrow B \) and elements \( a \in A, b \in B \) we denote by \( f[a \mapsto b] \) the function \( f' \) that is equal to \( f \) except that \( f'(a) = b \). We write \( 0 \) and \( 1 \) for the functions \( f_0 : A \rightarrow \{0\} \) and \( f_1 : A \rightarrow \{1\} \), respectively, if the domain \( A \) is understood. By \( B^A \) for sets \( A \) and \( B \) we denote the set of all functions from \( A \) to \( B \).

Kripke structures. Let \( \text{AP} \) be a finite set of atomic propositions. A Kripke structure is a tuple \( K = (S, s_I, E, \lambda) \) where \( S \) is a finite set of control states, \( s_I \in S \) the initial control state, \( E \subseteq S \times S \) the set of edges and \( \lambda : S \rightarrow 2^{\text{AP}} \) the labelling function. A finite path in \( K \) is a sequence \( u = s_0 s_1 \ldots s_k \in S^+ \) with \( (s_i, s_{i+1}) \in E \) for all \( i \in [0, k-1] \). Infinite paths are defined analogously. A run \( \rho \) of \( K \) is an infinite path with \( \rho(0) = s_I \). We denote by \( \text{Runs}(K) \) the set of runs of \( K \). Due to the single initial state, we assume without loss of generality that the graph of \( K \) is connected, i.e. all states are reachable. A simple loop in \( K \) is a finite path \( u = s_0 s_1 \ldots s_k \) such that \( i \neq j \) implies \( s_i \neq s_j \) for all \( i, j \in [0, k] \) and \( (s_i, s_{i+1}) \in E \). A Kripke structure \( K \) is a counter system if for each state \( s \in S \) there is at most one simple loop \( u \) in \( K \) with \( u(0) = s \). See Fig. 1 for an example. The classes of all Kripke structures and all flat Kripke structures are denoted KS and FKS, respectively.

Counter systems. Our proofs use systems with integer counters and simple guards. A counter system is a tuple \( S = (S, s_I, C, \Delta) \) where \( S \) is a finite set of control states, \( s_I \in S \) is
the initial state, \( C \) is a finite set of counter names and \( \Delta \subseteq S \times \mathbb{Z}^C \times 2^{\Theta(C)} \times S \) is the transition relation where \( \Theta(C) = \{(c < 0), (c \geq 0) \mid c \in C\} \). An infinite sequence \( s_0 s_1 \ldots \in \Sigma \) of states starting in \( s_0 = s_I \) is called a run of \( S \) if there is a sequence \( \theta_0 \theta_1 \ldots \in (2^C)^\omega \) of valuation functions \( \theta_i : C \to \mathbb{Z} \) with \( \theta_0 = \emptyset \) and a transition \( (s_i, u_i, G_i, s_{i+1}) \in \Delta \) for every \( i \in \mathbb{N} \) such that \( \theta_{i+1} = \theta_i + u_i \) (defined point-wise as usual), \( \theta_{i+1}(c) < 0 \) if \( (c < 0) \in G_i \) and \( \theta_{i+1}(c) \geq 0 \) if \( (c \geq 0) \in G_i \) for all \( c \in C \). Again, we denote by \( \text{Runs}(S) \) the set of all such runs and assume the graph of control states underlying \( S \) is connected.

### 2.2 Temporal Logics with Counting

We now introduce the different formalisms we use in this work as specification language. The most general one is the branching-time logic \( \text{CCTL}^* \) which extends the branching-time logic \( \text{CTL}^* \) (see e.g. [3]) with the following features: it has operators that allow for counting along a run the number of times a formula is satisfied and which stores the result into a variable. The counting starts when the associated variable is “placed” on the run. These variables may be shadowed by nested quantification, similar to the semantics of the freeze quantifier in linear temporal logic [13].

Let \( V \) be a set of variables and \( AP \) a set of atomic propositions. The syntax of \( \text{CCTL}^* \) formulae \( \varphi \) over \( V \) and \( AP \) is given by the grammar rules

\[
\varphi ::= p \mid \varphi \land \varphi \mid \neg \varphi \mid X \varphi \mid \varphi U \varphi \mid E \varphi \mid x.\varphi \mid \tau \leq \tau \mid \tau \equiv a \mid a \cdot \#(\varphi) \mid \tau + \tau
\]

for \( p \in AP \), \( x \in V \) and \( a \in \mathbb{Z} \). Common abbreviations such as \( \top \equiv p \lor \neg p \), \( \ot \equiv \neg \top \), \( F \varphi \equiv \top U \varphi \), \( G \varphi \equiv \neg F \neg \varphi \) and \( A \varphi \equiv \neg E \neg \varphi \) may also be used. The set of all subformulae of a formula \( \varphi \) (including itself) is denoted \( \text{sub}(\varphi) \) and \( |\varphi| \) denotes the length of \( \varphi \), with binary encoding of numbers.

**Semantics.** Intuitively, a variable \( x \) is used to mark some position on the concerned run. Within the scope of \( x \) a term \( \#_x(\varphi) \) refers to the number of times the formula \( \varphi \) holds between the current position and that marked by \( x \). The semantics of \( \text{CCTL}^* \) is hence defined with respect to a Kripke structure \( K = (S, s_I, E, \lambda) \), a run \( \rho \in \text{Runs}(K) \), a position \( i \in \mathbb{N} \) on \( \rho \) and a valuation function \( \theta : V \to \mathbb{N} \) assigning a position (index) on \( \rho \) to each variable. The satisfaction relation \( \models \) is defined inductively for \( p \in AP \), formulae \( \varphi, \psi \) and terms \( \tau_1, \tau_2 \) by

\[
\begin{align*}
(p, i, \theta) &\models p \quad \text{def} \quad p \in \lambda(\rho(i)), \\
(p, i, \theta) &\models X \varphi \quad \text{def} \quad (p, i + 1, \theta) \models \varphi, \\
(p, i, \theta) &\models \varphi U \psi \quad \text{def} \quad \forall k \geq i : (p, k, \theta) \models \psi \text{ and } \forall j \in [i, k - 1] : (p, j, \theta) \models \varphi, \\
(p, i, \theta) &\models E \varphi \quad \text{def} \quad \exists \rho' \in \text{Runs}(K) : \forall j \in [0, i] : \rho'(j) = \rho(j) \text{ and } (p', i, \theta) \models \varphi, \\
(p, i, \theta) &\models x.\varphi \quad \text{def} \quad (p, i, \theta[x \mapsto i]) \models \varphi, \\
(p, i, \theta) &\models \tau_1 \leq \tau_2 \quad \text{def} \quad [\tau_1](\rho, i, \theta) \leq [\tau_2](\rho, i, \theta),
\end{align*}
\]

where the Boolean cases are omitted and the semantics of terms is given, for \( a \in \mathbb{Z} \), by

\[
\begin{align*}
[a](\rho, i, \theta) &\quad \text{def} \quad a, \\
[\tau_1 + \tau_2](\rho, i, \theta) &\quad \text{def} \quad [\tau_1](\rho, i, \theta) + [\tau_2](\rho, i, \theta), \\
[a \cdot \#(\varphi)](\rho, i, \theta) &\quad \text{def} \quad a \cdot \{j \in \mathbb{N} \mid \theta(x) \leq j \leq \rho(\rho, i, \theta) \models \varphi\}.
\end{align*}
\]

We abbreviate \( (p, i, 0) \models \varphi \) by \( \models \varphi \) and \( (p, 0) \models \varphi \) by \( \models \varphi \) and say that \( \rho \) satisfies \( \varphi \) (at position \( i \)) in these cases. Moreover, we say a state \( s \in S \) satisfies \( \varphi \), denoted \( s \models \varphi \) if there are \( \rho_s \in \text{Runs}(K) \) and \( i \in \mathbb{N} \) such that \( \rho_s(i) = s \) and \( (\rho_s, i) \models \varphi \). The Kripke structure
\(K\) satisfies \(\varphi\), denoted by \(K \models \varphi\), if \(s_I \models \varphi\). Note that we choose to define the model-checking relation existentially but since the formalism is closed under negation, this does not have major consequences on our results.

**Fragments.** We define the following fragments of \(\text{CCTL}^*\) in analogy to the classical logics \(\text{LTL}\) and \(\text{CTL}\). The *linear* time fragment \(\text{CLTL}\) consists of those \(\text{CCTL}^*\) formulae that do not use the path quantifiers \(E\) and \(A\). The *branching* time logic \(\text{CCTL}\) restricts the use of temporal operators \(X\) and \(U\) such that each occurrence must be preceded immediately by either \(E\) or \(A\). Similar branching-time logics have been considered in [19].

**Frequency logics.** A major subject of our investigation are frequency constraints. This concept embeds naturally into the context of counting logics as it can be understood as a restricted form of counting. We therefore define in the following the frequency temporal logics \(\text{fCTL}^*, \text{fLTL}\) and \(\text{fCTL}\) as fragments of \(\text{CCTL}^*\). Consider the following grammar defining the syntax of formulae \(\varphi\) for natural numbers \(n, m \in \mathbb{N}\) with \(n \leq m > 0\) and \(p \in AP\).

\[
\begin{align*}
\varphi &::= p \mid \varphi \land \varphi \mid \neg \varphi \mid \alpha \\
\beta &::= X \varphi \mid \varphi U \varphi
\end{align*}
\]

With the additional rule \(\alpha ::= E \varphi \mid \beta\) it defines precisely the set of \(\text{fCTL}^*\) formulae while it defines \(\text{fCTL}\) for \(\alpha ::= E \beta \mid A \beta\) and \(\text{fLTL}\) for \(\alpha ::= \beta\). The semantics is defined by interpreting \(\text{fCTL}^*\) formulae as \(\text{CCTL}^*\) with the additional equivalence

\[
\varphi U^= \psi \overset{\text{def}}{=} \psi \lor x.F ((X \psi) \land m \cdot \#_x(\varphi) \geq n \cdot \#_x(T))
\]  

(1)

for \(\text{fCTL}^*\) formulae \(\varphi\) and \(\psi\) and a variable \(x \in V\) not being used in either \(\varphi\) or \(\psi\).

**Example 1.** Consider the Kripke structure given by Fig. 1 and the \(\text{CTL}\) formula \(\varphi_1 = z.A G (q \rightarrow (\#_z(p) \leq \#_z(E X r)))\). It basically states that on every path reaching \(s_5\) there must be a position where the states \(s_2\) and \(s_4\) (satisfying \(E X r\)) together have been visited at least as often as the state \(s_0\). A different, yet similar statement can be formulated using frequency constraints: \(\varphi_1' = A((E X r) U^= q)\) states that \(s_5\) must always be reached while visiting \(s_2\) and \(s_4\) together at least as often as \(s_0\), \(s_1\) and \(s_3\). Both \(\varphi_1\) and \(\varphi_1'\) are violated, e.g. by the path \(s_0^3s_1s_2s_4s_5s_5\). The Kripke structure however satisfies \(\varphi_2 = z.A G (\neg q \rightarrow E F \#_z(p) < \#_z(r))\) because from every state except \(s_5\) the number of positions that satisfy \(r\) can be increased arbitrary without increasing the number of those satisfying \(p\). Notice that this would not be the case, e.g., if \(s_4\) was labelled by \(p\).

While the positional variables in \(\text{CCTL}^*\) are a very flexible way of defining the scope of a constraint, frequency constraints in \(\text{fCTL}^*\) are always bound to the scope of an until operator. The same applies to the counting constraints of \(\text{LTL}\) as defined in [19]. For example, the \(\text{LTL}\) formula \(\varphi U a_1 \#_z(\varphi_1) + \cdots + a_n \#_z(\varphi_n) \geq k\) \(\psi\) is equivalent to the \(\text{CLTL}\) formula \(z.\varphi U(\psi \land a_1 \#_z(\varphi_1) \cdots + a_n \#_z(\varphi_n) \geq k)\). Admitting only natural coefficients, \(\text{LTL}\) can be encoded even in \(\text{LTL}\) making it thus strictly less expressive than \(\text{fLTL}\). On the other hand,
\[ \text{\textsc{c\textsc{CTL}}}_\pm \text{ admits arbitrary integer coefficients, which is more general than the frequency until operator of } \text{\textsc{f\textsc{LTL}}}. \text{ For example, } p \mathbb{U} \dot{=} q \text{ can be expressed as } \top \cup \{ q \# (\neg (\mathbb{T} > 0)) \} \text{ in } \text{\textsc{c\textsc{CTL}}}_\pm. \]

The relation between \( \text{\textsc{CTL}}_\pm \) and \( \text{\textsc{f\textsc{CTL}}} \), as well as \( \text{\textsc{c\textsc{CTL}}}_\pm \) and \( \text{\textsc{f\textsc{CTL}}} \) is analogous.

**Model-checking problem.** We now present the problem on which we focus our attention. The model-checking problem for a class \( \mathcal{R} \subseteq \mathcal{KS} \) of Kripke structures and a specification language \( \mathcal{L} \) (in our case all the specification languages are fragments of \( \text{\textsc{CCTL}}^* \)) is denoted by \( \text{\textsc{MC}}(\mathcal{R}, \mathcal{L}) \) and defined as the following decision problem.

**Input:** A Kripke structure \( K \in \mathcal{R} \) and a formula \( \varphi \in \mathcal{L} \).

**Decide:** Does \( K \models \varphi \) hold?

For temporal logics without counting variables, the model-checking problem over Kripke structures has been studied intensively and is known to be \( \text{\textsc{PSPACE}} \)-complete for \( \text{\textsc{LTL}} \) and \( \text{\textsc{CTL}}^* \) and in \( \text{\textsc{P}} \) for \( \text{\textsc{CTL}} \) (see e.g. [3]). It has recently been shown that when restricting to flat (or weak) structures the complexity of the model-checking problem for \( \text{\textsc{LTL}} \) is lower than in the general case [17]: it drops from \( \text{\textsc{PSPACE}} \) to \( \text{\textsc{NP}} \). As we show later, in the case of \( \text{\textsc{CCTL}}^* \), flatness of the structures allows us to regain decidability of the model-checking problem which is in general undecidable. In this paper, we propose various ways to solve the model-checking problem of fragments of \( \text{\textsc{CCTL}}^* \) over flat structures. For some of them we provide a direct algorithm, for others we reduce our problem to the satisfiability problem of a decidable extension of Presburger arithmetic.

### 3 Model-checking Frequency \text{\textsc{CTL}}

Satisfiability of \( \text{\textsc{f\textsc{LTL}}} \) is undecidable [5] implying the same for model-checking \( \text{\textsc{f\textsc{LTL}}} \), \( \text{\textsc{CTL}} \) and \( \text{\textsc{CCTL}}^* \) over Kripke structures. This applies moreover to \( \text{\textsc{CCTL}}^\omega \) in the case \( \text{\textsc{CCTL}}^* \). In contrast, we show in the following that \( \text{\textsc{MC}}(\mathcal{KS}, \mathcal{F\textsc{CTL}}) \) is decidable using an extension of the well-known labelling algorithm for \( \text{\textsc{CTL}} \) (see e.g. [3]).

Let \( K = (S, s_I, E, \lambda) \) be a Kripke structure and \( \Phi \) an \( \text{\textsc{f\textsc{CTL}}} \) formula. We compute recursively subsets \( S_\varphi \subseteq S \) of the states of \( K \) for every subformula \( \varphi \in \text{\textsc{sub}}(\Phi) \) of \( \Phi \) such that for all \( s \in S \) we have \( s \in S_\varphi \) iff \( s \models \varphi \). Checking whether the initial state \( s_I \) is contained in \( S_\varphi \) then solves the problem. Propositions \( (p \in AP) \), negation \( (\neg \varphi) \), conjunction \( (\varphi \land \psi) \) and temporal next \( (\mathbb{E} \varphi, \mathbb{A} \varphi) \) are handled as usual, e.g. \( S_p = \{ q \in S \mid p \in \lambda(q) \} \) and \( S_{\mathbb{E} \varphi} = \{ q \in S \mid \exists q' \in S_\varphi : (q, q') \in E \} \).

To compute if a state \( s \in S \) satisfies a formula of the form \( \mathbb{E} \varphi \mathbb{U} \psi \) or \( \mathbb{A} \varphi \mathbb{U} \psi \), assume that \( S_\varphi \) and \( S_\psi \) are given inductively. If \( s \in S_\varphi \) we immediately have \( s \in S_{\mathbb{E} \varphi \mathbb{U} \psi} \) and \( s \in S_{\mathbb{A} \varphi \mathbb{U} \psi} \).

For the remaining cases, the problem of deciding whether \( s \in S_{\mathbb{E} \varphi \mathbb{U} \psi} \) or \( s \in S_{\mathbb{A} \varphi \mathbb{U} \psi} \), respectively, can be reduced in linear time to the repeated control-state reachability problem in systems with one integer counter. The idea is to count the ratio along paths \( \rho \in S^\omega \) in \( K \) as follows, in direct analogy to the semantics defined in Eq. [1]. Assume \( r = \frac{n}{m} \) for \( n, m \in \mathbb{N} \) and \( n \leq m \). For any position on \( r \) we pay a fee of \( n \) and for those positions that satisfy \( \varphi \) we gain a reward of \( m \). Thus, we obtain a non-negative balance of rewards and gains at some position on \( r \) if, in average, among every \( m \) positions there are at least \( n \) positions that satisfy \( \varphi \), meaning the ratio constraint is satisfied. In \( K \), this balance along a path can be tracked using an integer counter that is increased by \( m - n \) when leaving a state \( s' \in S_\varphi \) and decreased by adding \( -n \) whenever leaving a state \( s' \notin S_\varphi \). Thus, let \( \hat{K}_s = (S, s, \{c\}, \Delta) \) be the counter system with

\[ \Delta = \{ (t, u, 0, t') \mid (t, t') \in E, t \notin S_\varphi \Rightarrow u(c) = -n, t \in S_\varphi \Rightarrow u(c) = m - n \}. \]

The state \( s \) satisfies the formula \( \mathbb{A} \varphi \mathbb{U} \psi \) if there is no path starting in state \( s \) violating the formula \( \varphi \mathbb{U} \psi \). The latter is the case if at every position where \( \psi \) holds, the balance
computed up to this position is negative. Therefore, consider an extension $R_s$ of $K_s$ where every edge leading into a state $s' \in S_\psi$ is guarded by the constraint $c < 0$. Every (infinite) run of $R_s$ is now a counter example for the property holding at $s$. To decide whether $s \in S_{\varphi \cup \psi}$ it suffices to check that in $K_s$ no state is repeatedly reachable from $s$.

A formula $\mathcal{E} \varphi \mathcal{U} \psi$ is satisfied by $s$ if there is some state $s' \in S_\psi$ reachable from $s$ with a non-negative balance. Hence, consider the counter system $U_s = (S \cup \{t\}, s, \{c\}, \Delta')$ obtained from $K_s$ featuring a new sink state $t \notin S$. The transition relation

$$\Delta' = \Delta \cup \{(s', 0, \{c \geq 0\}, t) \mid s' \in S_\psi\} \cup \{(t, 0, \emptyset, t)\}$$

extends $\Delta$ such that precisely the paths starting in $s$ and reaching a state $s' \in S_\psi$ with non-negative counter value (i.e. sufficient ratio) can be extended to reach $t$. Checking if $s$ is supposed to be contained in $S_{\varphi \cup \psi}$ then amounts to decide whether $t$ is (repeatedly) reachable from $s$ in $U_s$.

Finally, repeated reachability is easily translated to the accepting run problem of Büchi pushdown systems (BPDS) and the latter is in P [6]. A counter value $n \geq 0$ can be encoded into a stack of the form $\oplus^n$ while $\ominus^n$ encodes $-n \leq 0$ and for evaluating the guards $c \geq 0$ and $c < 0$ only the top symbol is relevant. Simulating an update of the counter by a number $a \in \mathbb{Z}$ requires to perform $|a|$ push or pop actions. The size of the system is therefore linear in the largest absolute update value and hence exponential in its binary representation. Since the updates of the constructed counter systems originate from the ratios in $\Phi$, the corresponding BPDS are of up to exponential size in $|\Phi|$. During the labelling procedure this step must be performed at most a polynomial number of times giving an exponential-time algorithm.

▶ **Theorem 2.** $\text{MC}(\mathcal{K}S, \text{fCTL})$ is in EXP.

It is worth noting that for a fixed formula (program complexity) or a unary encoding of numbers in frequency constraints, the size of the constructed Büchi pushdown systems and thus the runtime of the algorithm remains polynomial.

▶ **Corollary 3.** $\text{MC}(\mathcal{K}S, \text{fCTL})$ with unary number encoding is in P.

### 4 Model-checking Frequency LTL over Flat Kripke Structures

We show in this section that model-checking $\text{fLTL}$ is decidable over flat Kripke structures. As decision procedure we employ a guess and check approach: given a flat Kripke structure $\mathcal{K}$ and an $\text{fLTL}$ formula $\Phi$, we choose non-deterministically a set of satisfying runs to witness $\mathcal{K} \models \Phi$. As representation for such sets we introduce augmented path schemas that extend the concept of path schemas [20] [11] and provide for each of its runs a labelling by formulae. We show that if an augmented path schema features a syntactic property that we call consistency then the associated runs actually satisfy the formulae they are labelled with. Moreover, we show that every run of $\mathcal{K}$ is in fact represented by some consistent schema of size at most exponential in $|\mathcal{K}| + |\Phi|$. This gives rise to the following non-deterministic procedure.

1. **Read as input** an FKS $\mathcal{K}$ and an $\text{fLTL}$ formula $\Phi$.
2. **Guess** an augmented path schema $\mathcal{P}$ in $\mathcal{K}$ of at most exponential size.
3. **Terminate** successfully if $\mathcal{P}$ is consistent and accepts a run that is initially labelled by $\Phi$.

We fix for this section a flat Kripke structure $\mathcal{K} = (S, s_I, E, \lambda)$ and an $\text{fLTL}$ formula $\Phi$. For convenience we assume that $AP \subseteq \text{sub}(\Phi)$. Omitted technical details can be found in Appendices A and B.
Augmented Path Schemas

The set of runs of \( K \) can be represented as a finite number of so-called path schemas that consist of a sequence of paths and simple loops consecutive in \( K \). A path schema represents all runs that follow the given shape while repeating each loop arbitrarily often. For our purposes we extend this idea with additional labellings and introduce integer counters, updates and guards that can restrict the admitted runs.

**Definition 4 (Augmented Path Schema).** An augmented state of \( K \) is a tuple \( a = (s, L, G, \mathbf{u}, t) \in S \times 2^{aug}(B) \times 2^{aug}(C) \times \mathbb{Z} \times \{L, R\} \) comprised of a state \( s \) of \( K \), a set of formula labels \( L \), guards \( G \), an update \( \mathbf{u} \) over a set of counter names \( C \), and a type indicating whether the state is part of a loop (L) or a not (R). We denote by \( \text{st}(a) = s \), \( \text{lab}(a) = L \), \( g(a) = G \), \( u(a) = \mathbf{u} \) and \( t(a) = t \) the respective components of \( a \). An augmented path in \( K \) is a sequence \( u = a_0, \ldots, a_n \) of augmented states \( a_i \) such that \( \langle \text{st}(a_i), \text{st}(a_{i+1}) \rangle \in E \) for \( i \in [0, n-1] \). If \( t(a_i) = R \) for all \( i \in [0, n-1] \) then \( u \) is called a row. It is called an augmented simple loop (or simply loop) if it is non-empty and \( \langle \text{st}(a_n), \text{st}(a_1) \rangle \in E \) and \( \text{st}(a_i) \neq \text{st}(a_j) \) for \( i \neq j \) and \( t(a_i) = L \) for all \( i \in [0, n-1] \).

An augmented path schema (APS) in \( K \) is a tuple \( \mathcal{P} = (P_0, \ldots, P_n) \) where each component \( P_k \) is a row or a loop. \( P_n \) is a loop and their concatenation \( P_1 P_2 \ldots P_n \) is an augmented path.

Thanks to counters we can, for example, restrict to those runs satisfying a specific frequency constraint at some positions tracking it as discussed in Section 3. Figure 2 shows an example of an APS with edges including the possible state progressions. It features a single counter that tracks the frequency constraint of a formula \( r \mathbf{u}^2 q \) from state 1.

We denote by \( |\mathcal{P}| = |P_0 \ldots P_n| \) the size of \( \mathcal{P} \) and use global indices \( \ell \in [0, |\mathcal{P}| - 1] \) to address the \( (\ell + 1) \)-th augmented state in \( P_0 \ldots P_n \), denoted \( \mathcal{P}[\ell] \). To distinguish these global indices from positions in arbitrary sequences, we refer to them as locations of \( \mathcal{P} \). Moreover, \( \text{loc}_\mathcal{P}(k) = \{ \ell \mid |P_0 P_1 \ldots P_{k-1}| \leq \ell < |P_0 P_1 \ldots P_k| \} \) denotes for \( 0 \leq k \leq n \) the set of locations belonging to component \( P_k \) and for all locations \( \ell \in \text{loc}_\mathcal{P}(k) \) we denote the corresponding component index in \( \mathcal{P} \) by \( \text{comp}_\mathcal{P}(\ell) = k \). For example, in Fig. 2 we have \( \text{loc}_\mathcal{P}(3) = \{3, 4\} \) and \( \text{comp}_\mathcal{P}(6) = 5 \) because the seventh state of \( \mathcal{P} \) belongs to \( P_5 \). We extend the component projections for augmented states to (sequences of) locations of \( \mathcal{P} \) and write, e.g., \( \text{st}_\mathcal{P}(\ell_1, \ell_2) \) for \( \text{st}(\mathcal{P}[\ell_1]) \text{st}(\mathcal{P}[\ell_2]) \) and \( \text{up}_\mathcal{P}(\ell) \) for \( \text{up}(\mathcal{P}[\ell]) \).

An APS \( \mathcal{P} \) gives rise to a counter system \( \text{CS}(\mathcal{P}) = (Q, 0, C, \Delta) \) where \( Q = \{0, \ldots, |\mathcal{P}| - 1\} \), \( C \) are the counters used in the augmented states of \( \mathcal{P} \) and \( \Delta \) consists of those transitions \( (\ell, \text{up}_\mathcal{P}(\ell), g_{\mathcal{P}}(\ell'), \ell') \) such that \( 0 \leq \ell' = \ell + 1 < |\mathcal{P}| \) or \( \ell' < \ell \) and \( \{\ell', \ell' + 1, \ldots, \ell\} = \text{loc}_\mathcal{P}(k) \) for some loop \( P_k \). Notice that the APS in Fig. 2 is presented as its corresponding counter system. Let \( \text{succ}_\mathcal{P}(\ell) \) denote the set \( \{\ell' \in Q \mid \exists \mathbf{u}, G : (\ell, \mathbf{u}, G, \ell') \in \Delta\} \) of successors of \( \ell \) in \( \text{CS}(\mathcal{P}) \). A run of \( \mathcal{P} \) is a run of \( \text{CS}(\mathcal{P}) \) that visits each location \( \ell \in S \) at least once. The set of all runs of \( \mathcal{P} \) is denoted \( \text{Runs}(\mathcal{P}) \). As a consequence, a run visits the last loop infinitely.
often. We say that an APS \( \mathcal{P} \) is non-empty if \( \text{Runs}(\mathcal{P}) \neq \emptyset \). Since every run \( \sigma \in \text{Runs}(\mathcal{P}) \) corresponds to a path \( s_\mathcal{P}(\sigma) \in \mathcal{K} \), we define the satisfaction of an fLTL formula \( \varphi \) at position \( i \) by \( (\sigma, i) \models \varphi \iff \text{s_\mathcal{P}(\sigma), i \models \varphi} \).

Finally, notice that \( \text{CS}(\mathcal{P}) \) is in fact a flat counter system. It is shown in [11] that LTL properties can be verified over flat counter systems in non-deterministic polynomial time. Since LTL can express that each location of \( \text{CS}(\mathcal{P}) \) is visited we obtain the following result.

▶ \textbf{Lemma 5 [11].} Deciding non-emptiness of APS is in NP.

### 4.2 Labellings of Consistent APS are Correct

An APS \( \mathcal{P} \) assigns to every position \( i \) on each of its runs \( \sigma \) the labelling \( L_i = \text{lab}_\mathcal{P}(\sigma(i)) \). We are interested in this labelling being correct with respect to some fLTL formula \( \Phi \) in the sense that \( \Phi \in L_i \) if and only if \( (\sigma, i) \models \Phi \). The notion of consistency introduced in the following provides a sufficient criterion for correctness of the labelling of all runs of an APS.

An augmented path \( u = a_0 \ldots a_n \) is said to be good, neutral or bad for an fLTL formula \( \Psi = \varphi \Uparrow \psi \) if the number \( d = |\{0 \leq i < |u| \mid \varphi \in \text{lab}(u(i))\}| \) of positions labelled with \( \varphi \) is larger than \( (d > \frac{c}{\mathcal{y}} \cdot |u|) \) or smaller than \( (d < \frac{c}{\mathcal{y}} \cdot |u|) \), respectively, the fraction \( \frac{c}{\mathcal{y}} \) of all positions of \( u \). A tuple \( (P_0, \ldots, P_n) \) of runs and loops (not necessarily an APS) is called \( L \)-periodic for a set \( L \subseteq \text{sub}(\Phi) \) of labels if all augmented paths \( P_k \) share the same labelling with respect to \( L \), that is, for all \( 0 \leq k < n - 1 \) we have \( |P_k| = |P_{k+1}| \) and \( \text{lab}(P_k(i)) \cap L = \text{lab}(P_{k+1}(i)) \cap L \) for all \( 0 \leq i < |P_k| \).

▶ \textbf{Definition 6 (Consistency).} Let \( \mathcal{P} = (P_0, \ldots, P_n) \) be an APS in \( \mathcal{K} \), \( k \in [0, n] \) and \( \ell \in \text{loc}_\mathcal{P}(k) \) a location on component \( P_k \). The location \( \ell \) is consistent with respect to an fLTL formula \( \Psi \) if all locations of \( \mathcal{P} \) are consistent with respect to all strict subformulae of \( \Psi \) and one of the following conditions apply.

1. \( \Psi \in \text{AP} \) and \( \Psi \in \text{lab}_\mathcal{P}(\ell) \iff \Psi \in \lambda(s_\mathcal{P}(\ell)) \), or \( \Psi = \varphi \land \psi \) and \( \psi \in \text{lab}_\mathcal{P}(\ell) \iff \varphi \in \text{lab}_\mathcal{P}(\ell) \), or \( \Psi = \neg \varphi \) and \( \Phi \in \text{lab}_\mathcal{P}(\ell) \iff \varphi \notin \text{lab}_\mathcal{P}(\ell) \).
2. \( \Psi = \chi \varphi \) and \( \forall \ell' \in \text{succ}_\mathcal{P}(\ell) : \Psi \in \text{lab}_\mathcal{P}(\ell') \iff \varphi \in \text{lab}_\mathcal{P}(\ell') \).
3. \( \Psi = \varphi \Uparrow \psi \) and one of the following holds:
   a. \( \Psi, \psi \in \text{lab}_\mathcal{P}(\ell) \)
   b. \( \Psi \in \text{lab}_\mathcal{P}(\ell) \) and \( P_k \) is good for \( \Psi \) and \( \exists \ell' \in \text{loc}_\mathcal{P}(\ell) : \psi \in \text{lab}_\mathcal{P}(\ell') \)
   c. \( s_\mathcal{P}(\ell) = \mathcal{R} \) and there is a counter \( c \in \mathcal{C} \) such that \( \forall \ell' < \ell : u_\mathcal{P}(\ell')(c) = 0 \) and \( \forall \ell' \geq \ell : \varphi \in \text{lab}_\mathcal{P}(\ell') \Rightarrow u_\mathcal{P}(\ell')(c) = -x \) and
      - if \( \psi \notin \text{lab}_\mathcal{P}(\ell) \) then \( \psi \notin \text{lab}_\mathcal{P}(\ell') \) and \( \forall \ell' > \ell : \psi \in \text{lab}_\mathcal{P}(\ell') \Rightarrow (c < 0) \in \mathcal{g}_\ell(\ell') \) and
      - if \( \Psi \in \text{lab}_\mathcal{P}(\ell) \) then \( \exists \ell' > \ell : \psi \in \text{lab}_\mathcal{P}(\ell') \wedge (c \geq 0) \in \mathcal{g}_\ell(\ell') \).
   d. There is \( k' \in [0, n] \) such that all locations \( \ell' \in \text{loc}_\mathcal{P}(k') \) are consistent wrt. \( \Psi \) and
      - if \( k = n \) then \( k' < k \) and \( (P_k, P_{k+1}, \ldots, P_k) \) is \( \{\varphi, \psi, \Psi\} \)-periodic,
      - if \( k < n \) and \( P_k \) is good or neutral for \( \Psi \) and \( \Psi \notin \text{lab}_\mathcal{P}(\ell) \), or \( P_k \) is bad for \( \Psi \) and \( \Psi \in \text{lab}_\mathcal{P}(\ell) \) then \( k' < k < n \) and \( (P_k, P_{k+1}, \ldots, P_{k+1}) \) is \( \{\varphi, \psi, \Psi\} \)-periodic,
      - if \( k < n \) and \( P_k \) is good or neutral for \( \Psi \) and \( \Psi \in \text{lab}_\mathcal{P}(\ell) \), or \( P_k \) is bad for \( \Psi \) and \( \Psi \notin \text{lab}_\mathcal{P}(\ell) \) then \( k' < k' < n \) and \( (P_k, P_{k+1}, \ldots, P_{k+1}) \) is \( \{\varphi, \psi, \Psi\} \)-periodic.

The APS \( \mathcal{P} \) is consistent with respect to \( \Psi \) if it is the case for all its locations.

The cases 1 and 2 reflect the semantics syntactically. For instance, location 0 in Fig. 2 can be labelled consistently with \( \chi p \) since all its successor (0 and 1) are labelled with \( p \). Case 3 concerning the (frequency) until operator, is more involved.
Assume that \( \Phi = \varphi \mathcal{U} \psi \) is an until formula and that the labelling of \( K \) by \( \varphi \) and \( \psi \) is consistent. In some cases, it is obvious that \( \Phi \) holds, namely at positions labelled by \( \psi \) (case 3a) or if the final loop already guarantees that \( \Phi \) always holds (case 3b). If neither is the case we can apply the idea discussed in Section 3 and use a counter to check explicitly if at some point the formula \( \Phi \) holds (case 3c). Recall that to validate (or invalidate) the labelling of a location by the formula \( \Phi \) a specific counter tracks the frequency constraint in terms of the balance between fees and rewards along a run. For the starting point to be unique this case only applies to locations that are not part of a loop. For those labelled with \( \Phi \) there should exist a location in the future where \( \psi \) holds and the balance counter is non-negative. For those not labelled with \( \Phi \) all locations in the future where \( \psi \) holds must be entered with negative balance. Finally, case 3d can apply (not only) to loops and is based on the following reasoning: if a loop is good (bad) and \( \Phi \) is supposed to hold at some of its locations then it suffices to verify that this is the case during any of its future (past) iterations, e.g. the last (first) and vice versa if \( \Phi \) is supposed not to hold. This is the reason why this case allows for delegating consistency along a periodic pattern.

For instance, consider the formula \( \Psi = r \mathcal{U} q \) and the APS shown in Fig. 2. It is consistent to not label location 1 by \( \Psi \) because the counter \( c \) tracks the balance and locations 7 and 8 are guarded as required. If a run takes, e.g., the loop \( P_5 \) seven times, it has to take \( P_3 \) at least twice to satisfy all guards. This ensures that the ratio for the proposition \( r \) is strictly less than \( \frac{2}{3} \) upon reaching the first (and thus any) occurrence of \( q \). Note that to also make location 2 consistent, an additional counter needs to be added. Consistency with respect to \( \Psi \) is then inherited by location 0 from location 1 according to case 3d of the definition. Intuitively, additional iterations of the bad loop \( P_0 \) can only diminish the ratio.

The definition of consistency guarantees that if an APS is consistent with respect to \( \Phi \) then for every run of the APS, each time the formula \( \Phi \) is encountered, it holds at the current position (see Appendix A for complete details). Hence we obtain the following lemma that guarantees correctness of our decision procedure.

\[ \textbf{Lemma 7 (Correctness).} \text{ If there is an APS } \mathcal{P} \text{ in } K \text{ such that } \mathcal{P} \text{ is consistent wrt. } \Phi \text{ and } \Phi \in \text{lab}_\mathcal{P}(0) \text{ and } \text{Runs}(\mathcal{P}) \neq \emptyset \text{ then } K \models \Phi. \]

### 4.3 Constructing Consistent APS

Assuming that our flat Kripke structure \( K \) admits a run \( \rho \) such that \( \rho \models \Phi \), we show how to construct a non-empty APS that is initially labelled by \( \Phi \) and consistent with respect to \( \Phi \). It will be of at most exponential size in \( |K| + |\Phi| \) and is built recursively over the structure of \( \Phi \).

Concerning the base case where \( \Phi \in \text{AP} \), all paths in a flat structure can be represented by a path schema of linear size \( [20, 11] \). Intuitively, since \( K \) is flat, every subpath \( s_is_{i+1}...s_{j}...s_{j'} \) of \( \rho \) where a state \( s_i = s_j = s_{j'} \) occurs more than twice is equal to \( (s_is_{i+1}...s_{i+k-1})^k s_{j'} \) for some \( k \in \mathbb{N} \). Hence, there are simple subpaths \( u_0, ..., u_m \in S^+ \) of \( \rho \) and positive numbers of iterations \( n_0, ..., n_m-1 \in \mathbb{N} \) such that \( \rho = u_0^{n_0}u_1^{n_1}...u_{m-1}^{n_{m-1}}u_m \) and \( |u_0u_1...u_m| \leq 2|S| \). From this decomposition, we build an APS being consistent with respect to all propositions. Henceforth, we assume by induction an APS \( \mathcal{P} \) being consistent with respect to all strict subformulae of \( \Phi \) and a run \( \sigma \in \text{Runs}(\mathcal{P}) \) with \( \text{stp}(\sigma) = \rho \). If \( \Phi = \varphi \land \psi \) or \( \Phi = \neg \varphi \), Definition 6 determines for each augmented state of \( \mathcal{P} \) whether it is supposed to be labelled by \( \Phi \) or not. It remains hence to deal with the next and frequency until operators.

**Labelling \( \mathcal{P} \) by \( X \varphi \).** If \( \Phi = X \varphi \) the labelling at some location \( \ell \) is extended according to the labelling of its successors. These may disagree upon \( \varphi \) (only) if \( \ell \) has more than one
successor, i.e., being the last location on a loop $P_k$ of $\mathcal{P} = (P_0, \ldots, P_m)$. In that case we consult the run $\sigma$: if it takes $P_k$ only once, this loop can be cut and replaced by $P'_k$ that we define to be an exact copy except that all augmented states have type $R$ instead of $L$. If otherwise $\sigma$ takes $P_k$ at least twice, the loop can be unfolded by inserting $P'_k$ between $P_k$ and $P_{k+1}$, i.e. letting $\mathcal{P}' = (P_0, \ldots, P_k, P'_k, P_{k+1}, \ldots, P_m)$. Either way, $\sigma$ remains a run of the obtained APS, up to shifting the locations $\ell' > \ell$ if the extra component was inserted (recall that locations are indices). Importantly, cutting or unfolding any loop, even any number of times, in $\mathcal{P}$ preserves consistency.

**Labelling $\mathcal{P}$ by $\varphi U^r \psi$.** The most involved case is to label a location $\ell$ by $\Phi = \varphi U^r \psi$. First, assume that $\ell$ is part of a row. Whether it must be labelled by $\Phi$ is uniquely determined by $\sigma$. This is consistent if case $\text{5a}$ or $\text{3b}$ of Definition $\text{6}$ applies. The conditions of case $\text{3c}$ are also realised easily in most situations. Only, if $\Phi$ holds at $\ell$ but every location $\ell'$ witnessing this (by being reachable with sufficient frequency and labelled by $\psi$) is part of some loop $P'$. Adding the required guard directly to $P'$ may be too strict if $\sigma$ traverses $P'$ more than once. However, the first iteration (if $P'$ is bad for $\Phi$) or the last iteration (if $P'$ is good) on $\sigma$ contains a position (labelled with $\psi$) witnessing that $\Phi$ holds if any iteration does. Thus it suffices to unfold the loop once in the respective direction. For example, consider in Fig. 2 location 5 and a formula $\varphi = r U^2 q$. Location 8 could witness that $\varphi$ holds but a corresponding guard would be violated eventually since $P_7$ is bad for $\varphi$. The first iteration is thus the optimal choice. The unfolding $P_0$ separates it such that location 7 can be guarded instead without imposing unnecessary constraints.

Now assume that location $\ell$, to be labelled or not with $\Phi$, is part of a loop $P$ which is stable in the sense that $\Phi$ holds either at all positions $i$ with $\sigma(i) = \ell$ or at none of them. With two unfoldings of $P$, made consistent as above, case $\text{3d}$ applies. However, $\sigma$ may go through $\ell$ several, say $n > 1$, times where $\Phi$ holds at some but not all of the corresponding positions. If $n$ is small we can replace $P$ by precisely $n$ unfoldings, thus reducing to the previous case without increasing the size of the structure too much. We can moreover show that if $n$ is not small then it is possible to decompose such a problematic loop into a constant number of unfoldings and two stable copies based on the following observation.

**Lemma 8 (Decomposition).** Let $P = \mathcal{P}[\ell_0]\ldots\mathcal{P}[\ell_{|P|-1}]$ be a non-terminal loop in $\mathcal{P}$ with corresponding location sequence $v = \ell_0\ldots\ell_{|P|-1}$ and $\hat{n} = |P| \cdot y$ for some $y > 0$. For every run $\sigma = uv^n w \in \text{Runs}(\mathcal{P})$ where $n \geq \hat{n} + 2$ there are $n_1$ and $n_2$ such that $\sigma = u v^{n_1} v^n v^{n_2} w$ and for all positions $i$ on $\sigma$ with $|u| \leq i < |uv^{n_1-1}|$ or $|uv^{n_1} w| \leq i < |uv^{n_1} v^n v^{n_2-2}|$ we have $(\sigma, i) \models P \Phi$ iff $(\sigma, i + |P|) \models P \Phi$.

**Example 9.** Consider again the APS $\mathcal{P}$ in Fig. 3 a run $\sigma \in \text{Runs}(\mathcal{P})$ and the location 3. Whether or not $\varphi = r U^2 q$ holds at some position $i$ with $\sigma(i) = 3$ depends on how often $\sigma$ traverses the good loop $P_3$ (the more the better) and how often it repeats $P_3$ after position $i$. 

![Figure 3](image.png)
(the more the worse). Assume \( \sigma \) traverses \( P_3 \) exactly five times and \( P_3 \) sufficiently often, say 10 times. Then, during the last three iterations of \( P_3 \), \( \varphi \) holds when visiting location 3, and also location 4. In the two iterations before, the formula holds exclusively at location 4 and in any preceding iteration, it does not hold at all. Thus any labelling of \( P_3 \) would necessarily be incorrect. However, we can replace \( P_3 \) by four copies of it that are labelled as indicated in Fig. \( 3 \) and \( \sigma \) can easily be mapped onto this modified structure.

The presented procedure for constructing an APS from the run \( \rho \) in \( \mathcal{K} \) performs only linearly many steps in \(|\Phi|\), namely one step for each subformula. It starts with a structure of size at most \( 2|\mathcal{K}| \) and all modifications required to label an APS increase its size by a constant factor. Hence, we obtain an APS \( P_\Phi \) of size at most exponential in the length of \( \Phi \) and polynomial in the number of states of \( \mathcal{K} \). This consistent APS still contains a run corresponding to \( \rho \) and hence its first location must be labelled by \( \Phi \) because \( (\rho, 0) \models \Phi \) and we have seen that consistency implies correctness.

\[ \text{\textbf{Lemma 10 (Completeness).}} \quad \text{If } \mathcal{K} \models \Phi \text{ then there is a consistent APS } P \text{ in } \mathcal{K} \text{ of at most exponential size in } |\Phi| \text{ and } \Phi \text{ where } \Phi \in \text{lab}(P(0)) \text{ and } P \text{ is non-empty.} \]

We have seen in this section that the decision procedure presented in the beginning is sound and complete due to Lemma \( 7 \) and \( 10 \), respectively. The guessed APS is of exponential size in \(|\Phi|\) and of polynomial size in \(|\mathcal{K}|\). Since both checking consistency and non-emptiness (cf. Lemma \( 5 \)) require polynomial time (in the size of the APS) the procedure requires at most exponential time.

\[ \text{\textbf{Theorem 11.}} \quad \text{MC}(\text{FKS, } \mathcal{F}\text{LTL}) \text{ is in NEXP.} \]

This result immediately extends to \( \mathcal{F}\text{CTL}^* \). For an arbitrary \( \mathcal{F}\text{LTL} \) formula \( \varphi \), the procedure allows us to decide in \( \text{NEXP} \) whether \( q \models \text{E} \varphi \) holds. It allows us further to decide if \( q \models \text{K} \varphi \) holds in \( \text{ExpSpace} \) by the dual formulation \( q \not\models \text{E} \neg \varphi \) and Savitch’s theorem. Following otherwise the standard labeling procedure for \( \text{CTL} \) (cf. Section \( 3 \)) requires to invoke the procedure a polynomial number of times in \(|\mathcal{K}| + |\Phi|\).

\[ \text{\textbf{Theorem 12.}} \quad \text{MC}(\text{FKS, } \mathcal{F}\text{CTL}^*) \text{ is in ExpSpace.} \]

\section{On model-checking CCTL* over flat Kripke structures}

In this section, we prove decidability of \( \text{MC}(\text{FKS, CCTL}^*) \). We provide a polynomial encoding into the satisfiability problem of a decidable extension of Presburger arithmetic featuring a quantifier for counting the solutions of a formula. For the reverse direction an exponential reduction provides a corresponding hardness result for \( \text{CLTL, CCTL and CCTL}^* \).

**Presburger arithmetic with Härtig quantifier.** First-order logic over the natural numbers with addition was shown to be decidable by M. Presburger \[ 23 \]. It has been extended with the so-called Härtig quantifier \[ 2, 24, 25 \] that allows for referring to the number of values for a specific variable that satisfy a formula. We denote this extension by \( \text{PH} \). The syntax of \( \text{PH} \) formulae \( \varphi \) and \( \text{PH} \) terms \( \tau \) over a set of variables \( V \) is defined by the grammar

\[
\varphi ::= \tau \leq \tau \mid \neg \varphi \mid \varphi \land \varphi \mid \exists x.\varphi \mid \exists^{=n} y.\varphi \\
\tau ::= a \mid a \cdot x \mid \tau + \tau
\]

for natural constants \( a \in \mathbb{N} \) and variables \( x, y \in V \). Since the structure \( (\mathbb{N}, +) \) is fixed, the semantics is defined over valuations \( \eta : V \rightarrow \mathbb{N} \) that are extended to terms \( t \) as expected, e.g., \( \eta(3 \cdot x + 1) = 3 \cdot \eta(x) + 1 \). We define the satisfaction relation \( \models_{\text{PH}} \) as usual for first-order
Both predicates are definable by Presburger arithmetic formulae of polynomial size and used to encode their mapping on the positions (third parameter). Finally, we can equivalently add path quantifiers to all temporal operators in \( \Phi \) and obtain, syntactically, a CCTL formula.

\section*{Theorem 13.} The satisfiability problem of PH is reducible in exponential time to both MC(FKS, CCTL) and MC(FKS, CTL).

\section*{Deciding MC(FKS, CCTL).} We provide a polynomial reduction to the satisfiability problem of PH. Given a flat Kripke structure \( K \), we can represent each run \( \rho \) by a fixed number of natural numbers. We use a predicate \( Conf \) that allows for accessing the \( i \)-th state on \( \rho \) given its encoding and a predicate \( Run \) characterising all (encodings of) runs in \( Run(K) \). Such predicates were shown to be definable by Presburger arithmetic formulae of polynomial size and used to encode MC(FKS, CCTL) \cite{12, 10}. We adopt this idea for MC(FKS, CCTL) and PH. Let \( K = (S, s_I, E, \Lambda) \) and assume \( S \subseteq N \) without loss of generality. For \( N \in N \) let \( V_N = \{ r_1, \ldots, r_N, i, s \} \) be a set of variables that we use to encode a run, a position and a state, respectively.

\section*{Lemma 14 \cite{10}.} There is a number \( N \in N \), a mapping \( enc : N^N \to S^N \) and predicates \( Conf(r_1, \ldots, r_N, i, s) \) and \( Run(r_1, \ldots, r_N) \) such that for all valuations \( \eta : V_N \to N \) we have 1. \( \eta \models_{PH} Run(r_1, \ldots, r_N) \iff enc(\eta(r_1), \ldots, \eta(r_N)) \in Run(K) \) and 2. if \( \eta \models_{PH} Run(r_1, \ldots, r_N) \) then \( \eta \models_{PH} Conf(r_1, \ldots, r_N, i, s) \iff enc(\eta(r_1), \ldots, \eta(r_N), \eta(i)) = \eta(s) \). Both predicates are definable by PH formulae over variables \( V \supseteq V_N \) of polynomial size in \( |K| \).

Now, let \( \Phi \) be a CCTL formula to be verified on \( K \). Without loss of generality we assume that all comparisons \( \varphi \leq \in \text{sub}(\Phi) \) of the form \( \tau_1 \leq \tau_2 \) have the shape \( \varphi \leq =
Model-checking Counting Temporal Logics on Flat Structures

In this paper, we have seen that model checking flat Kripke structures with some expressive counting temporal logics is possible whereas this is not the case for general, finite Kripke structures. However, our results provide an under-approximation approach to this latter problem that consists in constructing flat sub-systems of the considered Kripke structure. We furthermore believe our method works as well for flat counter systems. We left as open problem the precise complexity for model checking \( f\text{CTL}, f\text{LTL} \) and \( f\text{CTL}^* \) over flat Kripke structures. It follows from [17] that the latter two problems are NP-hard while we obtain exponential upper bounds. However, we believe that if we fix the nesting depth of the frequency until operator in the logic, the complexity could be improved.

This work has shown, as one could have expected, a strong connection between \( \text{CLTL} \) and counter systems and as future work we plan to study automata-based formalisms inspired by \( f\text{LTL} \) where we will equip our automata with some counters whose role will be to evaluate the relative frequency of particular events.

Theorem 15. \( \text{MC}(\text{FKS}, C\text{CTL}^*) \) is reducible to \( \text{PH} \) satisfiability in polynomial time.

6 Conclusion

In this paper, we have seen that model checking flat Kripke structures with some expressive counting temporal logics is possible whereas this is not the case for general, finite Kripke structures. However, our results provide an under-approximation approach to this latter problem that consists in constructing flat sub-systems of the considered Kripke structure. We furthermore believe our method works as well for flat counter systems. We left as open problem the precise complexity for model checking \( f\text{CTL}, f\text{LTL} \) and \( f\text{CTL}^* \) over flat Kripke structures. It follows from [17] that the latter two problems are NP-hard while we obtain exponential upper bounds. However, we believe that if we fix the nesting depth of the frequency until operator in the logic, the complexity could be improved.

This work has shown, as one could have expected, a strong connection between \( \text{CLTL} \) and counter systems and as future work we plan to study automata-based formalisms inspired by \( f\text{LTL} \) where we will equip our automata with some counters whose role will be to evaluate the relative frequency of particular events.

\[
\sum_{k=1}^k a_k \cdot \# x_i(\varphi) + b \leq \sum_{k=k+1}^m a_k \cdot \# x_i(\varphi) + c
\]
References

1. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Roland Meyer, and Mehdi Seyed Salehi. What’s decidable about availability languages? In Prahladh Harsha and G. Ramalingam, editors, 35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India, volume 45 of LIPIcs, pages 192–205. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. [Pdoi:10.4230/LIPIcs.FSTTCS.2015.192]

2. H. Apelt. Axiomatische Untersuchungen über einige mit der Presburgerschen Arithmetik verwandten Systeme. Z. Math. Logik Grundlagen Math., 12:131–168, 1966.

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

4. Leonard Berman. The complexity of logical theories. Theor. Comput. Sci., 11:71–77, 1980. [Pdoi:10.1016/0304-3975(80)90037-7]

5. Benedikt Bollig, Normann Decker, and Martin Leucker. Frequency linear-time temporal logic. In Tiziana Margaria, Zongyan Qiu, and Hongli Yang, editors, Sixth International Symposium on Theoretical Aspects of Software Engineering, TASE 2012, 4-6 July 2012, Beijing, China, pages 85-92. IEEE Computer Society, 2012. [Pdoi:10.1109/TASE.2012.43]

6. Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In Antoni W. Mazurkiewicz and Józef Winkowski, editors, CONCUR ’97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings, volume 1243 of Lecture Notes in Computer Science, pages 135–150. Springer, 1997. [Pdoi:10.1007/3-540-63141-0_10]

7. Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, Logics of Programs, Workshop, Yorktown Heights, New York, May 1981, volume 131 of Lecture Notes in Computer Science, pages 52–71. Springer, 1981. [Pdoi:10.1007/BFb0025774]

8. Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. Commun. ACM, 52(11):74–84, 2009. [Pdoi:10.1145/1592761.1592781]

9. Normann Decker, Peter Habermehl, Martin Leucker, Arnaud Sangnier, and Daniel Thoma. Model-checking counting temporal logics on flat structures. In Roland Meyer and Uwe Nestmann, editors, 28th International Conference on Concurrency Theory, CONCUR 2017, volume 85 of LIPIcs, pages 25:1–25:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. [Pdoi:10.4230/LIPIcs.CONCUR.2017.25]

10. Stéphane Demri, Amit Kumar Dhar, and Arnaud Sangnier. Equivalence between model-checking flat counter systems and Presburger arithmetic. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings, volume 8762 of Lecture Notes in Computer Science, pages 85–97. Springer, 2014. [Pdoi:10.1007/978-3-319-11439-2_7]

11. Stéphane Demri, Amit Kumar Dhar, and Arnaud Sangnier. Taming past LTL and flat counter systems. Inf. Comput., 242:306–339, 2015. [Pdoi:10.1016/j.ic.2015.03.007]

12. Stéphane Demri, Alain Finkel, Valentin Goranko, and Gerton van Drimmelen. Model-checking CTL* over flat Presburger counter systems. Journal of Applied Non-Classical Logics, 20(4):313–344, 2010. [Pdoi:10.3109/10586208.2010.534129]

13. Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log., 10(3):16:1-16:30, 2009. [Pdoi:10.1145/1507244.1507246]

14. E. Allen Emerson and Joseph Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time. In John R. Wright, Larry Landweber, Alan J. Demers, and Tim Teitelbaum, editors, Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 1983, pages 127–140. ACM Press, 1983. [Pdoi:10.1145/567067.567081]
Model-checking Counting Temporal Logics on Flat Structures

15 Alain Finkel and Jérôme Leroux. How to compose Presburger-accelerations: Applications to broadcast protocols. In Manindra Agrawal and Anil Seth, editors, FST TCS 2002: Foundations of Software Technology and Theoretical Computer Science, 22nd Conference Kanpur, India, December 12-14, 2002, Proceedings, volume 2556 of Lecture Notes in Computer Science, pages 145–156. Springer, 2002. Pdoi:10.1007/3-540-36206-1_14

16 Jochen Hoenicke, Roland Meyer, and Ernst-Rüdiger Olderog. Kleene, rabin, and scott are available. In Paul Gastin and François Laroussinie, editors, CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings, volume 6269 of Lecture Notes in Computer Science, pages 402–477. Springer, 2010. Pdoi:10.1007/978-3-642-15375-4_32

17 Lars Kuhtz and Bernd Finkbeiner. Weak Kripke structures and LTL. In Joost-Pieter Katoen and Barbara König, editors, CONCUR 2011 - Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011. Proceedings, volume 6901 of Lecture Notes in Computer Science, pages 419–433. Springer, 2011. Pdoi:10.1007/978-3-642-23217-6_28

18 François Laroussinie, Antoine Meyer, and Eudes Petonnet. Counting LTL. In Nicolas Markey and Jef Wijsen, editors, TIME 2010 - 17th International Symposium on Temporal Representation and Reasoning, Paris, France, 6-8 September 2010, pages 51–58. IEEE Computer Society, 2010. Pdoi:10.1109/TIME.2010.20

19 François Laroussinie, Antoine Meyer, and Eudes Petonnet. Counting CTL. Logical Methods in Computer Science, 9(1), 2012. Pdoi:10.2168/LMCS-9(1:3)2013

20 Jérôme Leroux and Grégoire Sutre. Flat counter automata almost everywhere! In Doron A. Peled and Yih-Kuen Tsay, editors, Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005, Proceedings, volume 3707 of Lecture Notes in Computer Science, pages 489–503. Springer, 2005. Pdoi:10.1007/11562948_36

21 M. Minsky. Computation, Finite and Infinite Machines. Prentice Hall, 1967.

22 Amir Pnueli. The temporal logic of programs. In 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977, pages 46–57. IEEE Computer Society, 1977. Pdoi:10.1109/SFCS.1977.32

23 M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa, pages 92–101, 1929.

24 William Pugh. Counting solutions to Presburger formulas: How and why. In Vivek Sarkar, Barbara G. Ryder, and Mary Lou Soffa, editors, Proceedings of the ACM SIGPLAN'94 Conference on Programming Language Design and Implementation (PLDI), Orlando, Florida, USA, June 20-24, 1994, pages 121–134. ACM, 1994. Pdoi:10.1145/178243.178254

25 Nicole Schweikardt. Arithmetic, first-order logic, and counting quantifiers. ACM Trans. Comput. Log., 6(3):634–671, 2005. Pdoi:10.1145/1071596.1071602
A Consistency Implies Correctness

This section is dedicated to proving Lemma 7.

Lemma 7 (Correctness). If there is an APS $\mathcal{P}$ in $\mathcal{K}$ such that $\mathcal{P}$ is consistent wrt. $\Phi$ and $\Phi \not\in \operatorname{lab}(\mathcal{P})$ and $\operatorname{Runs}(\mathcal{P}) \neq \emptyset$ then $\mathcal{K} \models \Phi$.

Recall that $\mathcal{K} = (S, s_1, E, \lambda)$ is a Kripke structure and $\Phi$ an $\mathcal{fLTL}$ formula. Let us first formally define the notion of correctness.

Definition 16. A location $\ell$ of $\mathcal{P}$ is correct wrt. a formula $\xi$ if and only if

$$\forall \sigma \in \operatorname{Runs}(\mathcal{P}) : \forall i \in \mathbb{N} : \sigma(i) = \ell \Rightarrow (\xi \in \operatorname{lab}(\ell) \Leftrightarrow (\sigma, i) \models \xi).$$

An APS $\mathcal{P}$ is correct wrt. $\xi$ if that is the case for all locations of $\mathcal{P}$.

Notice that $\ell$ can only be consistent if $\xi$ holds at all positions where $\ell$ occurs or at none of them.

If $\mathcal{K}$ now contains an APS $\mathcal{P}$ that is correct wrt. $\Phi$ and that path schema contains a run $\sigma \in \operatorname{Runs}(\mathcal{P})$ then the correct labelling of the initial location 0 by $\Phi$ implies that $(\sigma, 0) \models \Phi$ and thus $\mathcal{K} \models \Phi$. Therefore, Lemma 7 is implied by the following, that we prove in the remainder of this section.

Lemma 17. If an APS $\mathcal{P}$ consistent wrt. $\Phi$ then it is correct wrt. $\Phi$.

Let in the following $\mathcal{P} = (P_0, P_1, \ldots, P_m)$ be an APS fixed and consistent wrt. $\Phi$. Let further $\sigma \in \operatorname{Runs}(\mathcal{P})$ be any run of $\mathcal{P}$. We use an induction over the structure of $\Phi$ to show that for all locations $\ell$ of $\mathcal{P}$ if $\Phi \not\in \operatorname{lab}(\ell)$ then $\Phi$ holds at every occurrence of $\ell$ on $\sigma$ and if $\Phi \not\in \operatorname{lab}(\ell)$ then $\Phi$ does not hold at any position where $\ell$ occurs on $\sigma$.

For easier reading, we use some abbreviations in the following. Let $L_i = \operatorname{lab}(\sigma(i))$ be the labelling of the location at position $i$ on $\sigma$ for $i \in \mathbb{N}$. We also denote the set of occurrences of a location $\ell$ on $\sigma$ by $\sigma^{-1}(\ell) = \{ i \in \mathbb{N} | \sigma(i) = \ell \}$.

A.1 Propositions, Boolean Combinations and Temporal Next

Let $i \in \mathbb{N}$ be any position on $\sigma$ and $\ell_i = \sigma(i)$ be the corresponding location in $\mathcal{P}$ with labelling $L_i$. Consider the following cases for the structure of $\Phi$, the first being the induction base case.

$(\Phi = p \in AP)$ By consistency $p \in L_i \iff p \in \lambda(\mathcal{s}(\ell_i))$ and by semantics $p \in \lambda(\mathcal{s}(\ell_i)) \iff (\sigma, i) \models p$.

$(\Phi = \neg \varphi)$

$$\neg \varphi \in L_i \iff \neg \varphi \not\in L_i \iff \neg \varphi \not\in \text{induct.} \iff (\sigma, i) \not\models \varphi \iff (\sigma, i) \models \neg \varphi$$

$(\Phi = \varphi \land \psi)$

$$\varphi \land \psi \in L_i \iff (\sigma, i) \models \varphi \text{ and } (\sigma, i) \models \psi \iff (\sigma, i) \models \varphi \land \psi$$

$(\Phi = X \varphi)$ By the definition of a run we have $\sigma(i + 1) \in \text{succ}(\sigma(i))$ and thus

$$X \varphi \in L_i \iff (\sigma, i + 1) \models \varphi \iff (\sigma, i) \models X \varphi.$$
A.2 Temporal Until

Assume finally $\Phi = \varphi U^* \psi$. By consistency and induction $P$ is correct wrt. $\varphi$ and $\psi$, thus $(\sigma, i) \models \varphi \Leftrightarrow \varphi \in L_i$ and $(\sigma, i) \models \psi \Leftrightarrow \psi \in L_i$ for all $i \in \mathbb{N}$. Let $k = \text{comp}_P(\ell_i)$. Hence $P_k$ is the component that $\ell_i = \sigma(i)$ belongs to. Further, for finite augmented paths $a_0 \ldots a_n$ let

$$\text{bal}(a_0 \ldots a_n) = y \cdot |\{j \in [0, n] \mid \varphi \in \text{lab}(a_j)\}| - x \cdot (n + 1)$$

denote the balance between those positions that are labelled by $\varphi$ and those that are not, weighted according to the ratio required by $\Phi$. That is, as discussed earlier, a “good” position contributes a reward of $y - x$ while a “bad” position causes a fee of $-x$. Then, $\text{bal}(a_0 \ldots a_n) \geq 0$ is equivalent to the ratio condition $|\{j \in [0, n] \mid \varphi \in \text{lab}(a_j)\}| \geq \frac{x}{y} \cdot (n + 1)$ specified by $\Phi$ but allows us to reason on discrete integer numbers. For convenience we apply this notation likewise for sequences $v = v_1 \ldots v_k$ of locations of $P$ and write $\text{bal}(v) := \text{bal}(P(v_1) \ldots P(v_k))$.

We will now treat the different case of the notion of consistency.

Case 3a. Assume $\Phi, \psi \in L_i$ then, by induction, $(\sigma, i) \models \psi$ which implies that $(\sigma, i) \models \Phi$.

Case 3b. $\ell_i$ is part of the final loop $P_m$ of $P$. Hence there is a smallest position $j > i$ where $\sigma(j)$ is part of the final loop $P$ of $P$ and $\psi \in L_j$ (and thus holds there). Hence, also $\psi \in L_{j+n[P]}$ for every number $n > 0$. Since $P$ is good for $\Phi$ the balance $\text{bal}(P) > 0$ is positive and thus

$$\text{bal}(\ell_i \ldots \ell_j) < \text{bal}(\ell_i \ldots \ell_{j+n[P]}) < \ldots < \text{bal}(\ell_i \ldots \ell_{j+n[P]})$$

For sufficiently large $n$ (i.e., sufficiently many iterations of $P$) we obtain necessarily a non-negative balance $\text{bal}(\ell_i \ldots \ell_{j+n[P]}) \geq 0$ on the corresponding subpath of $\sigma$ and thus $(\sigma, i) \models \Phi$.

Case 3c. This is the case where we have a counter tracking the balance. For this case to apply, $\ell_i$ must be part of a row and thus $i$ is the only position where $\ell_i$ occurs. The condition requires that there is a counter $c$ of $P$ that tracks the balance wrt. $\Phi$, starting at the occurrence of $\ell_i$. Since $\sigma$ is a run, there is a corresponding sequence of valuations $\theta_0 \theta_1 \ldots$ and for all $j \geq i$

$$\theta_j(c) = \text{bal}(\sigma(i) \sigma(i+1) \ldots \sigma(j-1))$$

If $\Phi \in \text{lab}_P(\sigma(i))$, the definition provides that there is a location $\ell' > \ell_i$ such that $\psi \in \text{lab}_P(\ell')$ and guarded by $(c \geq 0) \in \text{gr}_P(\ell')$. Thus, there is a position $j > i$ on $\sigma$ such that $\sigma(j) = \ell'$ and $\text{bal}(\sigma(i) \ldots \sigma(j-1)) = \theta_j \geq 0$. It follows that $(\sigma, i) \models \Phi$. Similarly, if $\Phi \not\in \text{lab}_P(\sigma(i))$, then there is no position $j$ on $\sigma$ where $\psi$ holds and the balance between $i$ and $j$ is non-negative. This is guaranteed because every such position carries a location $\ell' > \ell_i$ (since $\ell_i$ is not on a loop) and either $\psi \not\in \text{lab}_P(\ell')$ or $(c < 0) \in \text{gr}_P(\ell')$ and thus $\text{bal}(\sigma(i) \ldots \sigma(j-1)) = \theta_j(c) < 0$.

Case 3d. It remains to consider the cases requiring a periodic sequence. Let us start by establishing a lemma that provides a convenient argument for correctness and motivates the periodicity requirement imposed by the definition.

Recall that $P = (P_0, P_1, \ldots, P_m)$ and thus $\sigma$ has the form

$$\sigma = v_0^{n_0}v_1^{n_1} \ldots v_{m-1}^{n_{m-1}}v_m^{n_m}$$

where $v_j$, for $0 \leq j \leq m$, is the sequence of locations corresponding to the $j$-th component, i.e. $P[v_j] = P_j$, $n_j = 1$ if $P_j$ is a row and $n_j = 1$ if $P_j$ is a loop.
Lemma 18. Let \( (P_j, P_{j+1}, \ldots, P_j) \) be a \( \{\varphi, \psi\} \)-periodic sequence of components of \( \mathcal{P} \) with \( 0 \leq j < j < m \) and \( \sigma = uvw \) for \( v = v_1^n v_2^{n_1+1} \ldots v_j^n \). Let further \( |u| < i_1 \leq i_2 \) be positions on \( \sigma \) and \( n \in \mathbb{N} \) such that \( i_2 = i_1 + n|P_j| < |uv| \).

1. If \( P_j \) is good or neutral for \( \Phi \) then \( (\sigma, i_2) \models \Phi \Rightarrow (\sigma, i_1) \models \Phi \).
2. If \( P_j \) is bad or neutral for \( \Phi \) and \( i_2 < |uv| - |P_j| \) then \( (\sigma, i_1) \models \Phi \Rightarrow (\sigma, i_2) \models \Phi \).

Proof. 1. Assuming \( (\sigma, i_2) \models \Phi \) there is a position \( i_3 \geq i_2 \) such that \( (\sigma, i_3) \models \psi \) and \( \text{bal}(\sigma(i_2) \ldots \sigma(i_3 - 1)) \geq 0 \). Due to \( \varphi \)-periodicity we have also

\[
\text{bal}(\sigma(i_1) \ldots \sigma(i_2 - 1) \ldots \sigma(i_3)) = \text{bal}(\sigma(i_1) \ldots \sigma(i_1 + n|P_j| - 1) \ldots \sigma(i_3 - 1))
\]

\[
= n \cdot \text{bal}(P_j) + \text{bal}(\sigma(i_2) \ldots \sigma(i_3 - 1))
\]

due to \( \varphi \)-periodicity and \( \text{bal}(P_j) \leq 0 \).

If \( i_3 < i_2 \) we can assume w.l.o.g. that \( i_3 < i_1 + |P_j| \) because otherwise we can also choose \( i_3 - |P_j| \) as witness instead of \( i_3 \): \( \psi \in \text{lab}_\varphi(\sigma(i_3 - |P_j|)) \) due to \( \psi \)-periodicity and since \( \text{bal}(P_j) \leq 0 \) we would have

\[
0 \leq \text{bal}(\sigma(i_1) \ldots \sigma(i_1 + |P_j| - 1) \ldots \sigma(i_3 - 1)) = \text{bal}(\sigma(i_1) \ldots \sigma(i_3 - |P_j| - 1)) + \text{bal}(P_j) \leq \text{bal}(\sigma(i_1) \ldots \sigma(i_3 - |P_j| - 1)).
\]

Repeating this argument eventually provides a witness \( i_3 < i_1 + |P_j| \).

Then,

\[
0 \leq \text{bal}(\sigma(i_1) \ldots \sigma(i_3)) = \text{bal}(\sigma(i_1 + n|P_j|) \ldots \sigma(i_3 - 1 + n|P_j|)) = \text{bal}(\sigma(i_2) \ldots \sigma(i_3 - 1 + n|P_j|))
\]

because position \( i_3 + n|P_j| < i_1 + (n + 1)|P_j| = i_2 + |P_j| < |uv| \) on \( \sigma \) still carries a location from the periodic part \( (P_k, \ldots, P_j) \) of \( \mathcal{P} \). For the same reason we have \( \psi \in \text{lab}_\varphi(\sigma(i_3 + n|P_j|)) \) and thus \( \Phi \) holds at position \( i_2 \).

Based on Lemma 18 correctness can easily be established. The definition demands a component \( P_{k'} \) where each location is consistent and as shown earlier we can assume that it is thus correct not only wrt. \( \varphi \) and \( \psi \) but also wrt. \( \Phi \). We do then the following case analysis:

1. Considering the final loop \( P_m \) we have a preceding correct component \( P_k \) for \( k' < m \).

Periodicity wrt. \( \Phi \) provides that for some \( n \in \mathbb{N} \) the position \( i' = i - n|P_k| \) on \( \sigma \) carries a location \( \sigma(i') \in \text{loc}_\varphi(k') \) from \( P_{k'} \) and \( \Phi \in \text{lab}_\varphi(\sigma(i')) \Leftrightarrow \Phi \in \text{lab}_\varphi(\sigma(i)). \)

Due to periodicity (and correctness) wrt. \( \varphi \) and \( \psi \), the formula \( \Phi \) cannot distinguish any of the positions \( i' + n'k|P_k| \), i.e., \( (\sigma, i') \models \Phi \Leftrightarrow (\sigma, i' + n'|P_k|) \models \Phi \) for any \( n' \in \mathbb{N} \) since the infinite suffix \( \sigma(i') \sigma(i' + 1) \ldots \) is equivalent to every suffix \( \sigma(i' + n'k|P_k|) \sigma(i' + n'|P_k| + 1) \ldots \) regarding the positions where \( \varphi \) and \( \psi \) hold. Hence,

\[
\Phi \in \text{lab}_\varphi(\sigma(i)) \Leftrightarrow \Phi \in \text{lab}_\varphi(\sigma(i')) \Leftrightarrow (\sigma, i') \models \Phi \Leftrightarrow (\sigma, i) \models \Phi.
\]
Model-checking Counting Temporal Logics on Flat Structures

$P_k$ is good or neutral for $\Phi$ and $\Phi \in \texttt{lab}(\sigma(i))$: $k' > k$ and there is $i' = i + n|P_k|$ for some (unique) $n$ such that $\sigma(i') \in 10cP(k')$. Due to $\Phi$-periodicity, we have that $\Phi \in \texttt{lab}(\sigma(i'))$ and thus by correctness of that labelling and Lemma 18 we have $(\sigma, i) \models \Phi$.

$P_k$ is good or neutral for $\Phi$ and $\Phi \not\in \texttt{lab}(\sigma(i))$: $k' < k$ and there is $i' = i - n|P_k|$ for the unique $n$ such that $\sigma(i') \in 10cP(k')$. Due to $\Phi$-periodicity, we have that $\Phi \not\in \texttt{lab}(\sigma(i'))$ and thus $(\sigma, i') \not\models \Phi$ which implies by Lemma 18 that $(\sigma, i) \not\models \Phi$.

$P_k$ is bad for $\Phi$ and $\Phi \in \texttt{lab}(\sigma(i))$: $k' < k$ and there is $i' = i - n|P_k|$ for some (unique) $n$ such that $\sigma(i') \in 10cP(k')$. Since $i$ is a position in an iteration of $P_k$ at least one iteration of $P_{k+1}$ follows this position on $\sigma$, which still belongs to the periodic sequence. Therefore we can apply Lemma 18 and conclude from $(\sigma, i') \models \Phi$ that $(\sigma, i) \not\models \Phi$.

$P_k$ is bad for $\Phi$ and $\Phi \not\in \texttt{lab}(\sigma(i))$: $k' > k$ and there is $i' = i + n|P_k|$ for some (unique) $n$ such that $\sigma(i') \in 10cP(k')$. Again, periodicity and the guaranteed additional iteration of $P_{k+1}$ after $i'$ on $\sigma$ allows for applying Lemma 18 and to conclude from $(\sigma, i') \not\models \Phi$ that $(\sigma, i) \not\models \Phi$.

B Constructing Path Schemas from Satisfying Runs

This section is dedicated to proving Lemma 10. We will use the notation $\texttt{bal}$ and $\sigma^{-1}$ for a run $\sigma$ of an APS introduced in Appendix A. Recall that $\mathcal{K}$ is a flat Kripke structure that admits a run $\rho \in \texttt{runs}(\mathcal{K})$ and $\Phi$ is an fLTL formula. Assume for this section that $\rho \models \Phi$. From $\rho$ we construct a non-empty APS $\mathcal{P}_\rho$ that is consistent wrt. $\Phi$ and of which the first location $\mathcal{P}[0]$ is labelled by $\Phi$. In fact, it admits a run $\sigma$ representing $\rho$, i.e. such that $\texttt{st}_\mathcal{P}(\sigma) = \rho$. The construction provides an exponential bound on the size of $\mathcal{P}_\rho$, and thereby proves the lemma.

Lemma 10 (Completeness). If $\mathcal{K} \models \Phi$ then there is a consistent APS $\mathcal{P}$ in $\mathcal{K}$ of at most exponential size in $\mathcal{K}$ and $\Phi$ where $\Phi \in \texttt{lab}(\mathcal{P}(0))$ and $\mathcal{P}$ is non-empty.

Every run $\rho \in \texttt{runs}(\mathcal{K})$ can be represented by a small path schema in $\mathcal{K}$, labelled only by propositions. The labelling is then extended stepwise to include larger and larger subformulae of $\Phi$ until all subformulae and finally $\Phi$ itself are consistently annotated. Every step needs to ensure that the new annotation is consistent, which may require the modification of the structure, namely unfolding and duplication of loops. Thus, we also need to argue that after each modification there is still a valid run that represents $\rho$ and the obtained schema grew only linearly in size.

We use induction over the structure of $\Phi$ starting by its base case of $\Phi \in AP$ being atomic.

Base case. Since $\mathcal{K}$ is flat, any subpath $\rho(i)\rho(i+1)\ldots\rho(i')\ldots\rho(i'')$ of $\rho$ where a state $\rho(i) = \rho(i') = \rho(i'')$ occurs more than twice is equal to $(\rho(i)(i + 1)\ldots\rho(i' - 1))^2\rho(i'')$.

Hence, there are simple subpaths $u_0, \ldots, u_m \in S^+$ of $\rho$ and positive numbers of iterations $n_0, \ldots, n_{m-1} \in \mathbb{N}$ such that

$$\rho = u_0^{n_0}u_1^{n_1}\ldots u_{m-1}^{n_{m-1}}u_m$$

and $|u_0u_1\ldots u_m| \leq 2|S|$. They naturally induce the augmented path schema $\mathcal{P}_\Phi = (P_0, \ldots, P_m)$ where the augmented paths $P_k$ correspond directly to the paths $u_k$. Formally, for $u_k = s_0 \ldots s_n$ we let

$$P_k = (s_0, \lambda(s_0), \emptyset, 0, t_0) \ldots (s_k, \lambda(s_k), \emptyset, t_n)$$
where, for $0 \leq i \leq n$, the type of each augmented state is $t_i = \mathbb{R}$ if it is iterated $n_i = 1$ times and $t_i = \mathbb{L}$ if it is iterated $n_i > 1$ times on $\rho$. By construction $\mathcal{P}_\Phi$ is consistent wrt. any proposition from $AP$ and we have a run $\sigma \in \text{Runs}(\mathcal{P})$ such that $st_{\mathcal{P}}(\sigma) = \rho$.

The path schema does not use any counter so we can consider the set of counters $C$ to be empty. During the following constructions we may introduce new counters. Technically that means we would have to adjust all augmented states, simply because the signature of updates changes. For convenience, we therefore implicitly extend update functions and assigning zero to counter names if not explicitly stated otherwise.

Now, building on this base case, we show how to construct $\mathcal{P}_\Phi$ assuming by induction that there is an APS $\mathcal{P}$ that contains a run $\sigma \in \text{Runs}(\mathcal{P})$ with $st_{\mathcal{P}}(\sigma) = \rho$ and is consistent with respect to all strict subformulae of $\Phi$.

**Boolean combinations.** If $\Phi$ is a boolean combination then every augmented state $a = (s, L, G, u, t) \in \mathcal{P}$ is easily adjusted to obey Definition 6. For $\Phi = \neg \varphi$ we add $\Phi$ to $L$ if and only if $\varphi \not\in L$. For $\Phi = \varphi \land \psi$ we add $\Phi$ to $L$ if and only if $\varphi, \psi \in L$. These changes do not modify the set of runs and so $\sigma$ remains a run in the obtained structure $\mathcal{P}_\Phi$.

**B.1 Temporal Next**

For $\Phi = X\varphi$ the labelling at some location $\ell$ is extended according to the labelling of its successors. If $\varphi \in \text{lab}_P(\ell')$ for all $\ell' \in \text{succ}_P(\ell)$ then we modify $\text{lab}_P(\ell)$ such that it contains $\Phi$ and if $\varphi \not\in \text{lab}_P(\ell')$ for all successors $\ell'$ then the labelling remains untouched, not including $\Phi$.

The labellings, however, may disagree upon $\varphi$ if $\ell$ is the last location in a loop $P_k$ of $\mathcal{P}$. In that case the loop $P_k$ needs to be removed or unfolded, in order to make $\mathcal{P}$ consistent. For augmented states let $\text{row}(s, L, G, u, t) := (s, L, G, u, \mathbb{R})$ denote the same state but with type $\mathbb{R}$ and for sequences let $\text{row}(a_0 \ldots a_n) := \text{row}(a_0) \ldots \text{row}(a_n)$.

Now, if the run $\sigma$ takes $P_k$ only once it can be cut by replacing it with $P_k' = \text{row}(P_k)$. This eliminates runs that take $P_k$ more than once but $\sigma$ remains. If otherwise $\sigma$ takes $P_k$ at least twice, the loop can be unfolded by inserting $P_k'$ between $P_k$ and $P_{k+1}$, i.e. letting

$$\mathcal{P}' = (P_0, \ldots, P_k, P'_k, P_{k+1}, \ldots, P_m).$$

The run $\sigma$ representing $\rho$ persists, up to adjusting it according to the new shifted indices due to the insertion of $P_k'$. Formally, $\sigma$ has the form

$$\sigma = v_0^{n_0} \ldots v_{m-1}^{n_{m-1}} v_m,$$

where $v_i = (\ell_i, \ell_i + 1), \ldots, (\ell_i + |P_i| - 1)$ for $0 \leq i \leq m$ and $\ell_i = |P_0 \ldots P_{i-1}|$. Hence, there is a run $\sigma' \in \text{Runs}(\mathcal{P}')$ with

$$\sigma'(i) = \begin{cases} \sigma(i) & \text{if } i < |v_0^{n_0} \ldots v_k^{n_k-1}| \\ \sigma(i) + |P_k| & \text{otherwise.} \end{cases}$$

and thus $st_{\mathcal{P}'}(\sigma') = st_{\mathcal{P}}(\sigma) = \rho$.

Importantly, cutting or unfolding any loop, even any number of times, in $\mathcal{P}$ preserves consistency.

**Lemma 19.** Let $\mathcal{P} = (P_0, \ldots, P_k, \ldots, P_m)$ be an APS, $P_k$ a loop in $\mathcal{P}$ that is consistent with respect to an fLTL formula $\xi$ and $P_k' = \text{row}(P_k)$ an unfolding. The component $P_k'$ and all components $P_h$ that are consistent with respect to $\xi$ in $\mathcal{P}$ are also consistent with respect to $\xi$ in all of the following APS:
Assume now that

\[ \Phi = \varphi \Upsilon \psi \]

is an until formula. In order to construct \( \mathcal{P}_\Phi \) given \( \mathcal{P} \) and \( \sigma \) we iterate through the components of \( \mathcal{P} \), beginning at the last and transforming them one by one until the first. The invariant is that the number of components that are yet to be considered becomes smaller by one in each step (although the overall number of components may increase) and that there is always a run representing \( \rho \). The following lemma formalises one such step.

**Lemma 20.** Let \( \mathcal{P} = (P_0, \ldots, P_m) \) be an augmented path schema, \( k \in [0, m] \) and \( \ell = |P_0 \ldots P_{k-1}| \) such that

\[ \begin{align*}
\mathcal{P} & \text{ is consistent wrt } \varphi \text{ and } \psi, \\
\text{ every location } \ell' \in [\ell + |P_k|, |\mathcal{P}| - 1] & \text{ is consistent wrt. } \Phi \text{ and } \\
\text{ there is a run } \sigma \in \text{Runs}(\mathcal{P}) & \text{ with } \text{st}_\mathcal{P}(\sigma) = \rho. 
\end{align*} \]

There is an augmented path schema \( \mathcal{P}' = (P_0, \ldots, P_{k-1}, P'_k, \ldots, P'_m) \) such that

\[ \begin{align*}
\mathcal{P}' & \text{ is consistent wrt } \varphi \text{ and } \psi, \\
\text{ every location } \ell' \in [\ell, |\mathcal{P}'| - 1] & \text{ is consistent wrt. } \Phi, \\
\text{ there is a run } \sigma' \in \text{Runs}(\mathcal{P}') & \text{ with } \text{st}_{\mathcal{P}'}(\sigma') = \rho \text{ and } \\
|\mathcal{P}'| & \leq |\mathcal{P}| + 17|\gamma||K|^3. 
\end{align*} \]

For \( 0 \leq k \leq m \) let \( \ell_k = |P_0 \ldots P_{k-1}| \) be the first location in \( \mathcal{P} \) corresponding to component \( P_k \).

**Proof.** We proceed by a case analysis.

**Final loop.** Assume that \( k = m \), thus \( P_k = P_m \) is the final loop in \( \mathcal{P} \). If Case 3b of Definition 4 applies, all \( P_m \) is to be entirely labelled by \( \Phi \). Otherwise, we consider the first iteration of \( P_m \), starting at position \( i_m := \min \sigma^{-1}(\ell_m) \) (where \( \ell_m \) is the first location of \( P_m \)) and have \( P_m(j) \) labelled by \( \Phi \) if and only if \( (\sigma, i_m + j) \models \Phi \) for \( 0 \leq j < |P_m| \).

\[ \text{Notecase}\]

Notice that for proving the statement it is not necessary to be constructive. It suffices to observe that such a labelling exists.
Hence, those states labelled by $\psi$ are labelled by $\Phi$ which is consistent. If there are others states that we label by $\Phi$ and that are not labelled by $\psi$, we unfold $P_m$ twice and hence let $P' = (P_m, P_{m+1})$ and $P'' = (P_m, P_{m+1}, P_{m+2})$ for $P'_m = P_{m+1} = \text{row}(P_m)$ and $P''_{m+2} = P_m$.

The locations $\ell_0, \ldots, \ell_m + |P_m| - 1$, now associated with $P'_m$, can be made consistent (case 3c). For every location $\ell \in [\ell_0, \ell_m + |P_m| + 1]$ we introduce a fresh counter $c$ that is updated on the locations succeeding $\ell$ as required by the definition. If $\Phi \not\in \text{lab}_{P'}(\ell)$ we only need to add the guard $c < 0$ to the states at those locations $\ell < \ell' < |P'|$ that are labelled by $\psi$. If $\Phi \in \text{lab}_{P'}(\ell)$ then because $\Phi$ holds at its first occurrence $j$ on $\sigma$. In that case, if $\psi \not\in \text{lab}_{P'}(\ell)$, there must be a position $j' > j$ on $\sigma$ where $\psi$ holds and that is reached with positive balance. Second, $P_m$ must be bad for $\Phi$, because otherwise the case above applied already, and thus we can assume without loss of generality that $j' < j + |P_m|$ and hence location $\ell' = \sigma(j')$ carries a state from $P'_m$ or $P''_m$. They are both rows and therefore $\ell'$ can serve as witness location to be guarded by $(c \geq 0)$.

The locations $\ell \in [\ell_m, \ell_m + |P_m| - 1]$ of $P''_m$ where $\Phi \not\in \text{lab}_{P'}(\ell)$ can also be made consistent by adding a fresh counter that is updated an guarded as required. Again, the case that $\Phi \in \text{lab}_{P'}(\ell)$ only occurs if $P_m$ is bad. In that case $\ell$ is consistent already because $P''_m$ is part of the $(\sigma, \psi, \Phi)$-repeating sequence $(P'_m, P''_m, P_m)$ where $P'_m$ is consistent (case 3d). The final loop $P_m$ is consistent for the same reason.

The size of the final loop is bounded by $|P_m| \leq |K|$ and at most two new copies of it are added to obtain $P''$.

**Rows.** Assume $k \in \{0, m - 1\}$ and $P_k = a_0 \ldots a_n$ is a row in $\mathcal{P}$ starting at location $\ell_k = |P_0 \ldots P_{k-1}|$. Let $h \in N$ be the position on $\sigma$ with $\sigma(h) = \ell_k$.

We first adjust the labelling of $P_k$ such that $\Phi \in \text{lab}(a_i)$ if and only if $(\sigma, h + i) \models \Phi$ for $i \in [0, n]$. Now, with every location $\ell_i \in [\ell_k, \ell_k + n]$ that is not already consistent with respect to $\Phi$ (because of case 3a in Definition 3.1) we proceed as follows. A fresh counter $c$ is introduced and updated at all locations $\ell \in [\ell_i + 1, |P'| - 1]$ to count the balance as required in the definition. If $\Phi \not\in \text{lab}(a_i)$ then the additional guard $c < 0$ is added to the augmented states at those locations $\ell \in [\ell_i + 1, |P'| - 1]$ that are labelled by $\Psi$. This makes $\ell_i$ consistent and moreover, since $\Phi$ does not hold at position $h + i$ on $\sigma$, these constraint are not violated by the run.

If $\Phi \in \text{lab}(a_i)$ (although $\psi \not\in \text{lab}(a_i)$) there is a position $h' > h + i$ such that $(\sigma, h') \models \psi$ and thus $\psi \in \text{lab}_{P'}(\ell')$ for $\ell' = \sigma(h')$. If $\ell'$ is on a row we add the constraint $c \geq 0$ to the augmented state at $\ell'$. If $\ell'$ is on a loop $P_{k'}$ of $\mathcal{P}$ but $\sigma$ takes it only once we can replace $P_{k'}$ by $\text{row}(P_{k'})$ in $\mathcal{P}$ and then add the constraint. In case $\sigma$ takes $P_{k'}$ at least twice either the last or first iteration of $P_{k'}$ can serve as a witness: If $P_{k'}$ is good for $\Phi$ more iterations of it between $h$ and $h'$ can only improve the balance so that the ratio between $h$ and the last position $h'' = \max \sigma^{-1}(\ell')$ where $\ell'$ is sufficient for $\Phi$. Hence, we unfold $P_{k'}$ by adding a copy $\text{row}(P_{k'})$ right after $P_k$ in $\mathcal{P}$. Notice that we can assume that $k' < m$ because if $P_{k'}$ were the final loop and good for $\Phi$ the case above had already applied and we would not need to unfold the loop. Similarly, if $P_{k'}$ is bad (or neutral) for $\Phi$ then we let $h'' = \min \sigma^{-1}(\ell')$ be the first position where $\ell'$ occurs. Since $\text{bal}(\sigma(h) \ldots \sigma(h')) \leq \text{bal}(\sigma(h) \ldots \sigma(h''))$ in this case $h''$ also can serve as witness and we unfold the loop by inserting $P_{k'}'$ immediately before $P_{k'}$ in $\mathcal{P}$.

As argued earlier, these transformations do not make any consistent location inconsistent with respect to any formula and there is still a run $\sigma'$ representing $\text{st}_{\mathcal{P}}(\sigma) = \rho$. However, the location $\ell''$ (at position $h''$ on $\sigma'$) is not part of a loop and can safely be guarded by $(c \geq 0)$ while preserving the run.

During this procedure we introduce at most one unfolding of some loop for each position
on $P_k$ and the size of $\mathcal{P}$ increases thus by at most $|\mathcal{K}|^2$ because $|\mathcal{K}|$ bounds the length of each loop.

**Non-final Loops.** It remains to consider the case that $P_k$ is a non-final loop. The run $\sigma$ has the form $\sigma = w^nu$ where $v = \ell_k (\ell_k + 1) \ldots (\ell_k + |P_k| - 1))$ is the sequence of locations corresponding to $P_k$ in $\mathcal{P}$ and $n \in \mathbb{N}$ is maximal, that is $u$ and $w$ do not intersect with $v$. We assume in the following that $n$ is not small as otherwise we may simply replace $P_k$ by $n$ copies of $\text{row}(P_k)$ and proceed as above. More precisely, let $\hat{n} = y \cdot |P_k|$ and assume that $n \geq \hat{n} + 2$. This constant $\hat{n}$ essentially bounds the effect of frequency variations within a single loop iteration. Its specific choice will become apparent in the later construction. For now it suffices to observe that if $n \leq \hat{n} + 1 = y|P_k| + 1 \leq y|\mathcal{K}| + 1$ and we replace $P_k$ by $n$ unfoldings the size of $\mathcal{P}$ increases by $(n - 1) \cdot |P_k|$. Applying the procedure for rows above to each component may force us to unfold other loops. As a (rough) estimate, we will have to introduce no more than one further unfolding of some loop for each new location originating from the unfoldings of $P_k$. Hence, after making all $n$ copies of $P_k$ consistent the size of $\mathcal{P}$ did not grow by more than

$$(n - 1) \cdot |P_k| + n \cdot |P_k| \cdot |\mathcal{K}| \leq (y|\mathcal{K}| + 1 - 1) \cdot |\mathcal{K}| + (y|\mathcal{K}| + 1) \cdot |\mathcal{K}| \cdot |\mathcal{K}| \leq 3y|\mathcal{K}|^3.$$

Given that $n \geq \hat{n} + 2$ we distinguish two situations of $\sigma$ determining a labelling for $P_k$.

- Either, for all position $|u| i < |w^{n-1}|$ we have $(\sigma, i) \models \Phi \Leftrightarrow (\sigma, i + |v|) \models \Phi$, meaning that the labelling of the augmented state $\mathcal{P}(\ell)$ at location $\ell$ on $v$ is unambiguously determined by $\sigma$ (we say that the loop is **stable**), or there is a location on $v$ such that at some of its occurrence on $\sigma$ the formula $\Phi$ holds while at another it does not (in that case the loop is **unstable**). We consider first the former case and how it can be made consistent. Afterwards we show that in the latter case it is possible to modify $\mathcal{P}$ such that the former case applies.

**Stable loops.** If the pattern of positions where $\Phi$ holds is stable along the iterations of $P_k$ on $\sigma$ we apply it to the labelling of $P_\hat{n}$. That is, we adjust $P_k$ such that $\Phi \in \text{lab}(P_k(i))$ if and only $(\sigma, |u| i + |v|) \models \Phi$. Likely, at least some of the locations $\ell \in [\ell_k, \ell_k + |P_k| - 1]$ are still not consistent with respect to $\Phi$. If $n \leq 4$ we replace $P_k$ in $\mathcal{P}$ by $n$ unfoldings $P'_k = \text{row}(P_k)$ that can be made consistent as above. Otherwise, let $R_1 = R_2 = R_3 = R_4 = P'_k$ and insert $(R_1, R_2)$ before and $R_3, R_4$ after $P_k$ in $\mathcal{P}$. The two last unfoldings $R_3$ and $R_4$ can be made consistent as above. For $R_1$ we proceed the same way except that if $P_k$ is to be unfolded again (for instance to find a location labelled with $\psi$) $R_2$ or $R_3$ are considered instead. Now, $R_2$ and $P_k$ are also consistent because the surrounding components $R_1, P_k, R_3, R_4$ cover every possible case. Overall no more than $4$ additional copies of $P'_k$ are added and for the locations of at most three of them other loops needed to be unfolded giving a total of no more than

$$4 \cdot |P_k| + 3 \cdot |P_k| \cdot |\mathcal{K}| \leq 7|\mathcal{K}|^2$$

new locations being added to $\mathcal{P}$.

**Unstable loops.** In general, $\sigma$ does not uniquely determine whether the state at some location $\ell \in [\ell_k, \ell_k + |P_k| - 1]$ in $\mathcal{P}$ is supposed to be labelled by $\Phi$ because that may vary between corresponding position on $\sigma$, that is, the iterations of $P_k$. However, we observe that along any run the validity of $\Phi$ at some specific location can change at most once. We have argued earlier that as soon as $\Phi$ holds somewhere, more iterations of a good loop inserted between the position in question and a witness position does not affect validity. Similarly, introducing additional iterations of a bad loop do not change the fact that $\Phi$ does not hold at some specific position.
It follows, for example, that if $\Phi$ does hold in the last iteration of a bad loop but not in the first, there is a unique iteration for each location on the loop where validity swaps. The diagram presented in Fig. 4 shows an example of how the balance between a position $i$ and a witness position $i'$ may evolve on $\sigma$. Observe that there are three parts of the run iterating through the first loop. In part one $\Phi$ holds nowhere because the balance (and hence the ratio) on the path to the (only) witness is insufficient. It covers too many iterations of the bad loop. In the last part, $\Phi$ holds everywhere because the ratio condition is satisfied or not. In between it depends on local differences whether the ratio condition is satisfied or not. The first and last part can be uniformly labelled and thus represented each by a copy of the original loop. On the other hand, the intermediate part is short: its length depends only on the length of the loop and the ratio, more precisely, on the size of the denominator (3 in the example) as measure of how sensitive the property is to changes in the frequency on an arbitrarily long path.

Lemma 8 formalises this observation.

**Lemma 8 (Decomposition).** Let $P = P[\ell_0] \ldots P[\ell_{|P|-1}]$ be a non-terminal loop in $P$ with corresponding location sequence $v = \ell_0 \ldots \ell_{|P|-1}$ and $\hat{n} = |P| \cdot y > 0$. For every run $\sigma = w^n w \in \text{Run}(P)$ where $n \geq \hat{n} + 2$ there are $n_1$ and $n_2$ such that $\sigma = w^{n_1}v^h v^{n_2}w$ and for all positions $i$ on $\sigma$ with $|u| \leq i < |w^{n_1-1}|$ or $|w^{n_1}v^h| \leq i < |w^{n_1}v^h v^{n_2-2}|$ we have $(\sigma, i) \models \Phi$ iff $(\sigma, i + |P|) \models \Phi$.

**Proof.** Assume that $P$ is good for $\Phi$ and thus $\text{bal}(P) > 0$. Consider the first (smallest) position $i \geq |u|$ on $\sigma = w^n w$ where $(\sigma, i) \not\models \Phi$. If $i$ does not exist or $i \geq |w^{n-\hat{n}}|$ we can choose $n_1 = n - \hat{n} - 1$ and $n_2 = n - \hat{n} - n_1 = 1$.

Otherwise let $n_1$ be the last iteration of $P$ entirely satisfying $\Phi$, that is such that $|w^{n_1}| \leq i < w^{n_1+1}$, and $\hat{n} = |w^{n_1}|$. Consequently we let $n_2 = n - \hat{n} - n_1$. Consider now any position $i' \in [h, h + |P| - 1]$ in the $(n_1 + 1)$-th iteration where $\Phi$ still holds. If there is none, then $\Phi$ does not hold in later iterations either and the statement of the lemma holds.

Since $(\sigma, i') \models \Phi$ there is some position $j > i'$ with $(\sigma, j) \models \psi$ and $\text{bal}(\sigma(i')\sigma(i' + 1) \ldots \sigma(j - 1)) \geq 0$ Observe that we can assume that $j \geq |w^{n-1}|$ because otherwise $j + |P|
would serve as witness since in that case
\[
\text{bal}(\sigma(i')\sigma(i' + 1)\ldots\sigma(j - 1 + |P|)) = \text{bal}(\sigma(i')\sigma(i' + 1)\ldots\sigma(j - 1)) + \text{bal}(P)
\]
\[
\geq \text{bal}(\sigma(i')\sigma(i' + 1)\ldots\sigma(j - 1))
\]
\[
\geq 0
\]
while \( \sigma(j) = \sigma(j + |P|) \) and thus \( \psi \in \text{lab}_P(\sigma(j + |P|)) \).

However, the balance cannot be too large, more precisely, \( \text{bal}(\sigma(i')\sigma(i' + 1)\ldots\sigma(j - 1)) \leq |P| \cdot y \). Depending on whether \( i' < i \) or \( i < i' \) we have
\[
\text{bal}(\sigma(i')\ldots\sigma(j - 1)) = \begin{cases} 
\text{bal}(\sigma(i)\ldots\sigma(j - 1)) - \text{bal}(\sigma(i)\ldots\sigma(i' - 1)) & \text{if } i < i' \\
\text{bal}(\sigma(i)\ldots\sigma(j - 1)) + \text{bal}(\sigma(i')\ldots\sigma(i - 1)) & \text{if } i' < i
\end{cases}
\]
Considering the first case, we can bound the difference by the maximal gain
\[
\text{bal}(\sigma(i)\ldots\sigma(i' - 1)) \leq (|P| - 1) \cdot (y - x) \leq y|P|
\]
on a path of length at most \(|P| - 1\). In the second case, the lower bound on the balance
\[
\text{bal}(\sigma(i')\ldots\sigma(i - 1)) \geq |P| - 1 \cdot (-x) \geq -y|P|
\]
is of interest because we conclude that in any case
\[
\text{bal}(\sigma(i')\ldots\sigma(j - 1)) \leq \text{bal}(\sigma(i)\ldots\sigma(j - 1)) + y|P| < y|P|
\]
Since \( \text{bal}(P) \geq 1 \) we have that
\[
\text{bal}(\sigma(i' + \hat{n}|P|)\ldots\sigma(j - 1)) = \text{bal}(\sigma(i')\ldots\sigma(j - 1)) - \hat{n} \cdot \text{bal}(P)
\]
\[
< y|P| - y|P| \cdot \text{bal}(P)
\]
\[
< 0
\]
meaning that after at most \( \hat{n} \) further iteration \( \Phi \) can not hold any more.

Assuming now that \( P \) is bad for \( \Phi \) allows for similar reasoning. Consider \( i \geq |uv| \) to be the first position on \( \sigma \) where \( (\sigma, i) \models \Phi \) while \( (\sigma, i - |P|) \) if \( i \) does not exist or \( i \geq |uv^n| - \hat{n} \) we can again choose \( n_1 = n - \hat{n} - 1 \) and \( n_2 = n - \hat{n} - n_1 = 1 \). Otherwise we choose \( n_1 \) such that \( |uv^n| \leq i < |uv^{n+1}| \) and let \( h = |uv^n| \).

There is a position \( j > i \) such that \( \psi \in \text{lab}_P(\sigma(j)) \) and \( \text{bal}(\sigma(i)\ldots\sigma(j - 1)) \geq 0 \). Observe that \( j \geq |uv^n| \) because otherwise \( \text{bal}(\sigma(i - |P|)\ldots\sigma(j - 1 - |P|)) \geq 0 \) and \( \psi \in \text{lab}_P(\sigma(i - |P|)) \) contradicting that \( (\sigma, i - |P|) \not\models \Phi \).

Consider now any position \( i' \in [h, h + |P| - 1] \) where \( (\sigma, i') \not\models \Phi \), if any. We have
\[
\text{bal}(\sigma(i')\ldots\sigma(j - 1)) = \begin{cases} 
\text{bal}(\sigma(i)\ldots\sigma(j - 1)) - \text{bal}(\sigma(i)\ldots\sigma(i' - 1)) & \text{if } i < i' \\
\text{bal}(\sigma(i)\ldots\sigma(j - 1)) + \text{bal}(\sigma(i')\ldots\sigma(i - 1)) & \text{if } i' < i
\end{cases}
\]
and obtain the bounds
\[
\text{bal}(\sigma(i)\ldots\sigma(i' - 1)) \leq |P| - 1 \cdot (y - x) \leq y|P| \quad \text{(if } i < i' \text{)}
\]
\[
\text{bal}(\sigma(i')\ldots\sigma(i - 1)) \geq |P| - 1 \cdot (-x) \geq -y|P| \quad \text{(if } i' < i \text{)}
\]
Hence
\[
\text{bal}(\sigma(i')\ldots\sigma(j - 1)) \geq \text{bal}(\sigma(i)\ldots\sigma(j - 1)) - y|P| \geq -y|P|
\]
Now, since \( \text{bal}(P) < 0 \) we have that

\[
\text{bal}(\sigma(i' + \hat{n}|P|) \ldots \sigma(j - 1)) = \text{bal}(\sigma(i') \ldots \sigma(j - 1)) - \hat{n} \cdot \text{bal}(P)
\]

\[
\geq -y|P| - (y|P| \cdot \text{bal}(P))
\]

\[
\geq 0
\]

providing that after \( \hat{n} = y|P| \) more iterations, \( \Phi \) holds at every position on the loop.

If \( P \) is neutral for \( \Phi \) then an iteration of \( P \) more or less does not change if there is a witness or not and \( (\sigma, i) \models \Phi \) if and only if \( (\sigma, i + |P|) \models \Phi \) for all \( |u| \leq i < |uv^{n-1}| \).

Lemma 8 provides a bound on how often we need to unfold \( P_k \) at most in order to guarantee that \( \sigma \) determines a unique labelling. Recall we assumed that \( \sigma \) repeats \( P_k \) for \( n \geq \hat{n} + 2 \) times. In \( \mathcal{P} \), we may hence replace \( P_k \) by \( \langle P_k, P_k', \ldots, P_k', P_k \rangle \) introducing two copies and a sequence of exactly \( \hat{n} \) unfoldings of it. The decomposition \( \sigma = uv^{n_1}v^{n_2}w \) given by Lemma 8 provides a corresponding run \( \sigma' \) of the obtained path schema and a unique labelling for all of the new components. Now, we are only left with cases discussed earlier: two stable loops and \( \hat{n} \) rows. For each of the stable loops, we can estimate that establishing consistency requires no more than \( 7|\mathcal{K}|^2 \) additional locations. For each of the \( \hat{n} \) new rows it no more than \( |\mathcal{K}|^2 \) additional locations. We can conclude that \( \mathcal{P}' \) can be constructed with in total no more than

\[
|P_k| + \hat{n}|P_k| + 2 \cdot 7|\mathcal{K}|^2 + \hat{n} \cdot |\mathcal{K}|^2 \leq 17y|\mathcal{K}|^3
\]

additional locations.

\[\blacksquare\]

### B.3 The Size of \( \mathcal{P}_\Phi \)

The induction provides the construction of \( \mathcal{P}_\Phi \) from \( \mathcal{K} \) requiring (at most) one step for each subformula of \( \Phi \). Let \( \mathcal{P}_0 \) be the APS provided by the base case that covers all propositions occurring in \( \Phi \). As argued earlier, its size is bounded by \( 2|\mathcal{K}| \) and the length of every loop is bounded by \( |\mathcal{K}| \). Applying the induction step now recursively for \( \Phi \), i.e., augmenting \( \mathcal{P}_0 \) consistently with more and more subformulas of \( \Phi \) we obtain a sequence of possibly growing path schemas until \( \mathcal{P}_\Phi \) is obtained after at most \( |\text{sub}(\Phi)| \leq |\Phi| \) steps.

We have seen that in the case of a next formula, constructing the consistent schema \( \mathcal{P}' \) from \( \mathcal{P} \) requires at most one unfolding of some loop for each location in \( \mathcal{P} \) and thus \( |\mathcal{P}'| \leq |\mathcal{P}| \cdot |\mathcal{K}| \). In the case of an until formula Lemma 20 provides that for each component of \( \mathcal{P} \) no more than \( 17y|\mathcal{K}|^3 \) locations are added and thus \( |\mathcal{P}'| \leq |\mathcal{P}| \cdot 17y|\mathcal{K}|^3 \). Counting the bits for representing \( y \) to the length of \( \Phi \) and hence estimating \( y \leq 2^{|\Phi|} \) it follows that after \( |\Phi| \) steps, the resulting path schema is of size

\[
|\mathcal{P}_\Phi| \leq |\mathcal{P}_0| \cdot (17 \cdot 2^{|\Phi|} |\mathcal{K}|^3)^{|\Phi|} \in O(2^{|\Phi|(|\Phi|+|\mathcal{K}|)})
\]

for some polynomial \( f \) and thus at most exponential in the size of the input.

By construction \( \mathcal{P}_\Phi \) is correct and there is a run \( \sigma \in \text{Runs}(\mathcal{P}_\Phi) \) with \( \text{st}_{\mathcal{P}_\Phi}(\sigma) = \rho \models \Phi \) and hence \( \text{lab}(\mathcal{P}(0)) = \text{lab}(\mathcal{P}(\sigma(0))) \models \Phi \). This completes the proof for Lemma 10.