Model-Checking Counting Temporal Logics on Flat Structures

Normann Decker¹, Peter Habermehl², Martin Leucker³, Arnaud Sangnier⁴, and Daniel Thoma⁵

₁ ISP, University of Lübeck, Lübeck, Germany
decker@isp.uni-luebeck.de

₂ IRIF, Univ. Paris Diderot, Paris, France
haberm@irif.fr

³ ISP, University of Lübeck, Lübeck, Germany
leucker@isp.uni-luebeck.de

⁴ IRIF, Univ. Paris Diderot, Paris, France
sangnier@irif.fr

⁵ ISP, University of Lübeck, Lübeck, Germany
thoma@isp.uni-luebeck.de

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

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.29

1 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 $\mathcal{C}TL$. 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) $\mathcal{C}TL^\pm$ 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 $\mathcal{C}TL$ is decidable in polynomial time and that the satisfiability problem for $\mathcal{C}TL^\pm$ is undecidable. A similar extension for LTL is considered in [18] where it is proven that model checking of $\mathcal{C}LTL$ is ExpSpace-complete while that of $\mathcal{C}LTL^\pm$ is undecidable.

Following the same motivation, regular availability expressions (RAE) were introduced in [16] 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 $\mathfrak{f}LTL$ [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 $\varphi U \psi$ requires $\varphi$ to hold at all positions before $\psi$. For example, the $\mathfrak{f}LTL$ formula $p U^{\frac{1}{3}} q$ 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, $\mathfrak{f}LTL$ can be considered as a fragment of $\mathcal{C}LTL$ 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 $\mathfrak{f}LTL$ 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 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 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 $\mathcal{C}CTL^\ast$ where we use variables to mark positions on a run from where we begin to count the
Table 1 Complexity characterisation of the model-checking problems of fragments of CCTL*. PH indicates polynomial reducibility to the (decidable) satisfiability problem of PH.

<table>
<thead>
<tr>
<th></th>
<th>CTL</th>
<th>LTL</th>
<th>CTL*</th>
<th>fLTL</th>
<th>fCTL</th>
<th>fCTL*</th>
<th>CLTL</th>
<th>CCTL</th>
<th>CCTL*</th>
</tr>
</thead>
<tbody>
<tr>
<td>FKS</td>
<td>P</td>
<td>NP-c.</td>
<td>PSPACE</td>
<td>NExp</td>
<td>Exp</td>
<td>ExpSpace</td>
<td>PH</td>
<td>PH</td>
<td>PH</td>
</tr>
</tbody>
</table>

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 fCTL, fLTL and 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 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 fLTL and CCTL* are decidable. For the former, our method is a guess and check procedure based on the existence of a flat counter system as witness of a run of the Kripke structure satisfying the fLTL formula. For the latter, we use a technique which consists in encoding the run of a flat Kripke structure into a Presburger arithmetic formula and then we show that model checking of CCTL* can be translated into the satisfiability problem of a decidable extension of Presburger arithmetic, called PH, featuring a counting quantifier known as Härtig quantifier. We hence provide new decidability results for CCTL* which in practice could be used as an under-approximation approach to the general model-checking problem. We furthermore relate an extension of Presburger arithmetic, for which the complexity of the satisfiability problem is open, to a concrete model-checking problem. In summary, for model checking different fragments of CCTL* on Kripke structures (KS) or flat Kripke structures (FKS) we obtain the picture shown in Table 1 where bold entries are our novel results.

2 Definitions

2.1 Preliminaries

We write N and Z to denote the sets of natural numbers (including zero) and integers, respectively, and [i, j] for \{k ∈ Z | i ≤ k ≤ j\}. We consider integers encoded with a binary representation. For a finite alphabet Σ, Σ* represents the set of finite words over Σ, Σ+ the set of finite non-empty words over Σ and Σω the set of infinite words over Σ. For a finite set E of elements, |E| represents its cardinality. For (finite or infinite) words and general sequences u = a0a1...ak... of length at least k + 1 > 0 we denote by u(k) = ak the (k + 1)-th element and refer to its indices 0, 1,... as positions on u. If u is finite then |u| denotes its length. For arbitrary functions f : A → B and elements a ∈ A, b ∈ B we denote by f[a → b] the function f’ that is equal to f except that f’(a) = b. We write 0 and 1 for the functions f0 : A → {0} and f1 : A → {1}, respectively, if the domain A is understood. By BA for sets A and B we denote the set of all functions from A to B.
Kripke structures. Let \( AP \) 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 \( \lambda : S \to 2^{AP} \) 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 \( (s_k, s_0) \in E \). A Kripke structure \( K \) 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 Figure 1 for an example. The classes of all Kripke structures and all flat Kripke structures are denoted \( \text{KS} \) and \( \text{FKS} \), 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 \( s_0 s_1 \ldots \in S^\omega \) of states starting in \( s_0 = s_I \) is called a run of \( S \) if there is a sequence \( \theta_0 \theta_1 \ldots \in (\mathbb{Z}^C)^\omega \) of valuation functions \( \theta_i : C \to \mathbb{Z} \) with \( \theta_0 = 0 \) 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

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

for \( p \in AP \), \( x \in V \) and \( a \in \mathbb{Z} \). Common abbreviations such as \( T \equiv p \lor \neg p \), \( \perp \equiv \neg T \), \( F \varphi \equiv T U \varphi \), \( G \varphi \equiv \neg F \neg \varphi \) 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

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

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

\[
[a](\rho, i, \theta) \overset{\text{def}}{=} a, \\
\begin{aligned}
[\tau_1 + \tau_2](\rho, i, \theta) & \overset{\text{def}}{=} [\tau_1](\rho, i, \theta) + [\tau_2](\rho, i, \theta), \\
[a \cdot \text{x}_x(\varphi)](\rho, i, \theta) & \overset{\text{def}}{=} a \cdot [\{j \in \mathbb{N} | \theta(x) \leq j \leq i, (p, j, \theta) \models \varphi\}] \end{aligned}
\]

We abbreviate $(p, i, 0) \models \varphi$ by $(p, i) \models \varphi$ and $(p, 0) \models \varphi$ by $p \models \varphi$ 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 $s_1 \models \varphi$. 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 $p \in AP$.

\[
\varphi ::= p \mid \varphi \land \varphi \mid \neg \varphi \mid \alpha \quad \beta ::= X \varphi \mid \varphi \text{ U } \varphi \quad \text{with} \quad \alpha ::= E \beta \mid A \beta \quad \text{and} \quad \text{fLTL} \text{ for } \alpha ::= \beta.
\]

With the additional rule $\alpha ::= E \beta \mid A \beta$ 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

\[
\varphi \text{ U } \varphi \overset{\text{def}}{=} \psi \lor X. F ((X \psi) \land m \cdot \text{x}_x(\varphi) \geq n \cdot \text{x}_x(\top)) \quad (1)
\]

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 Figure 1 and the $\text{CCTL}$ formula $\varphi_1 = z : A G (q \rightarrow (\#_z(p) \leq \#(\text{EX} r)))$. It basically states that on every path reaching $s_5$ there must be a position where the states $s_2$ and $s_4$ (satisfying $\text{EX} r$) together have been visited at least as often as the state $s_0$. A different, yet similar statement can be formulated using only frequency constraints: $\varphi'_1 = A (\text{EX} r \text{ U } q)$ states that $s_5$ must always be reached while visiting
Figure 1 A flat Kripke over \( AP = \{ p, q, r \} \).

\( s_2 \) 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 \( s_0s_1s_2s_4s_5^\omega \). The Kripke structure however satisfies \( \varphi_2 = z. \text{AG} (\neg q \rightarrow \text{EF} \#_z(p) < \#_z(r)) \) because from every state except \( s_5 \) the number of positions that satisfy \( r \) can be increased arbitrarily 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 \( \text{cLTL} \) as defined in [19]. For example, the \( \text{cLTL} \) formula \( \varphi U[\overset{a}{\bigoplus}\overset{\text{(p) + \cdots + \text{(p)}}{\#}(\varphi_1) + \cdots + \overset{\text{(p)}}{\#}(\varphi_n) \geq k}] \psi \) is equivalent to the \( \text{CLTL} \) formula \( z. \varphi U(\psi \land \overset{a}{\bigoplus}\overset{\text{(p) + \cdots + \text{(p)}}{\#}(\varphi_1) + \cdots + \overset{\text{(p)}}{\#}(\varphi_n) \geq k}) \). Admitting only natural coefficients, \( \text{cLTL} \) can be encoded even in \( \text{LTL} \) making it thus strictly less expressive than \( \text{fLTL} \). On the other hand, \( \text{cLTL}^\pm \) admits arbitrary integer coefficients, which is more general than the frequency until operator of \( \text{fLTL} \). For example, \( pU a q \) can be expressed as \( \top U[\overset{a}{\bigoplus}\overset{\text{(p) + \cdots + \text{(p)}}{\#}(\varphi_1) + \cdots + \overset{\text{(p)}}{\#}(\varphi_n) \geq 0}] q \) in \( \text{cLTL}^\pm \).

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

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

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

For temporal logics without counting variables, the model-checking problem over Kripke structure has been studied intensively and is known to be \( \text{PSPACE} \)-complete for \( \text{LTL} \) and \( \text{CTL}^* \) and in \( P \) for \( \text{CTL} \) (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 \( \text{LTL} \) is lower than in the general case [17]: it drops from \( \text{PSPACE} \) to \( \text{NP} \). As we show later, in the case of \( \text{CCTL}^* \), 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 \( \text{CCTL}^* \) 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 CTL

Satisfiability of \( \text{fLTL} \) is undecidable [5] implying the same for model-checking \( \text{fLTL}, \text{CLTL} \) and \( \text{CCTL}^* \) over Kripke structures. This applies moreover to \( \text{CCTL} \) [19]. In contrast, we show in the following that \( \text{MC}(\text{KS}, \text{fCTL}) \) is decidable using an extension of the well-known labelling algorithm for \( \text{CTL} \) (see e.g. [3]).
Let $\mathcal{K} = (S, s_I, E, \lambda)$ be a Kripke structure and $\Phi$ an fCTL formula. We compute recursively subsets $S_\varphi \subseteq S$ of the states of $\mathcal{K}$ for every subformula $\varphi \in \text{sub}(\Phi)$ 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 ($\mathbf{E} \varphi$, $\mathbf{A} \varphi$) are handled as usual, e.g. $S_p = \{ q \in S \mid p \in \lambda(q) \}$ and $S_{\mathbf{E} \varphi} = \{ q \in S \mid \exists q' \in S_\varphi : (q, q') \in \delta \}$.

To compute if a state $s \in S$ satisfies a formula of the form $\mathbf{E} \varphi \lor \psi$ or $\mathbf{A} \varphi \lor \psi$, assume that $S_\varphi$ and $S_\psi$ are given inductively. If $s \in S_\psi$ we immediately have $s \in S_{\mathbf{E} \varphi \lor \psi}$ and $s \in S_{\mathbf{A} \varphi \lor \psi}$. For the remaining cases, the problem of deciding whether $s \in S_{\mathbf{E} \varphi \lor \psi}$ or $s \in S_{\mathbf{A} \varphi \lor \psi}$, 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 $p \in S^\omega$ in $\mathcal{K}$ as follows, in direct analogy to the semantics defined in Equation 1. Assume $r = \frac{n}{m}$ for $n, m \in \mathbb{N}$ and $n \leq m$. For passing any position on $p$ 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 $p$ if, in average, among every $m$ positions there are at least $n$ positions that satisfy $\varphi$, meaning the ratio constraint is satisfied. In $\mathcal{K}$, 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 $\hat{\mathcal{K}}_s = (S, s, \{c\}, \Delta)$ 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 $\mathbf{A} \varphi \lor \psi$ if there is no path starting in state $s$ violating the formula $\varphi \lor \psi$. 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 $\mathcal{R}_s$ of $\hat{\mathcal{K}}_s$ where every edge leading into a state $s' \in S_\psi$ is guarded by the constraint $c < 0$. Every (infinite) run of $\mathcal{R}_s$ is now a counter example for the property holding at $s$. To decide whether $s \in S_{\mathbf{A} \varphi \lor \psi}$ it suffices to check that in $\mathcal{R}_s$ no state is repeatedly reachable from $s$.

A formula $\mathbf{E} \varphi \lor \psi$ 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 $\mathcal{U}_s = (S \cup \{t\}, s, \{c\}, \Delta')$ obtained from $\hat{\mathcal{K}}_s$ featuring a new sink state $t \notin S$. The transition relation \[
\Delta' = \Delta \cup \{(s', 0, \{c \geq 0\}, t) \mid s' \in S_\varphi \} \cup \{(t, 0, \emptyset, t)\}
\]
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 $S_{\mathbf{E} \varphi \lor \psi}$ then amounts to decide whether $t$ is (repeatedly) reachable from $s$ in $\mathcal{U}_s$.

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 $\oplus^n$ 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.

$\triangleright$ Theorem 2. $\text{MC(\mathcal{K}, fCTL)}$ 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. MC(\(KS, fCTL\)) with unary number encoding is in \(P\).

4 Model-checking Frequency LTL over Flat Kripke Structures

We show in this section that model-checking \(fLTL\) is decidable over flat Kripke structures. As decision procedure we employ a guess and check approach: given a flat Kripke structure \(K\) and an \(fLTL\) formula \(\Phi\), we choose non-deterministically a set of satisfying runs \(\mathcal{P}\) to witness \(K \models \Phi\). 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 \(K\) is in fact represented by some consistent schema of size at most exponential in \(|K| + |\Phi|\). This gives rise to the following non-deterministic procedure.

1. Read as input an FKS \(K\) and an \(fLTL\) formula \(\Phi\).
2. Guess an augmented path schema \(\mathcal{P}\) in \(K\) 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 \(K = (S, sI, E, \lambda)\) and an \(fLTL\) formula \(\Phi\). For convenience we assume that \(AP \subseteq \text{sub}(\Phi)\). Omitted technical details can be found in [9].

4.1 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 \(K\) [20, 11]. 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 \(a = (s, L, G, \mathbf{u}, t) \in S \times 2^{\text{sub}(\Phi)} \times 2^{|C|} \times \mathbb{Z}^C \times \{L, R\}\) comprised of a state \(s\) of \(K\), a set of formula labels \(L\), guards \(G\) and an update \(\mathbf{u}\) over a set of counter names \(C\), and a type \(t\) indicating whether the state is part of a loop \((L)\) or a not \((R)\). We denote by \(\mathbf{st}(a) = s\), \(\mathbf{lab}(a) = L\), \(\mathbf{g}(a) = G\), \(\mathbf{u}(a) = \mathbf{u}\) 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 \((\mathbf{st}(a_i), \mathbf{st}(a_{i+1})) \in E\) 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 \((\mathbf{st}(a_n), \mathbf{st}(a_1)) \in E\) and \(\mathbf{st}(a_i) \neq \mathbf{st}(a_j)\) 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 \(P_i\) is a row or a loop. \(P_n\) is a loop and their concatenation \(P_0 \ldots P_n\) 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 indicating the possible state progressions. It features a single counter that tracks the frequency constraint of a formula \(r \mathcal{U}^4 q\) 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,
We are interested in this labelling being \( \phi \preceq \Psi = \text{loc} \circ \Psi \) same labelling with respect to following provides a sufficient criterion for correctness of the labelling of all runs of an APS.

Since properties can be verified over flat counter systems in non-deterministic polynomial time. An augmented path \( \text{su}(\ell) \) of an APS is non-empty iff \( \text{su}(\ell) \neq \emptyset \). Since every run \( \sigma \in \text{Runs}(\mathcal{P}) \) corresponds, by construction of \( \mathcal{P} \), to a path \( \text{st}(\mathcal{P}, \ell) \in Q^* \in K \) we define the satisfaction of an fLTL formula \( \varphi \) at position \( i \) by \( (\sigma, i) \models \varphi \) iff \( (\text{st}(\mathcal{P}, \sigma), i) \models \varphi \).

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.

> **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 \( \Psi = \varphi \Box^d \Diamond^e \psi \) if the number \( d = |\{0 \leq i < |u| \mid \varphi \in \text{lab}(u(i))\}| \) of positions labelled with \( \varphi \) is larger than \( (d > \frac{e}{d} \cdot |u|) \) equal to \( (d = \frac{e}{d} \cdot |u|) \) or smaller than \( (d < \frac{e}{d} \cdot |u|) \), respectively, the fraction \( \frac{e}{d} \) of all positions of \( u \). A tuple \( (P_0, \ldots, P_n) \) of rows 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 \( 0 \leq i < |P_k| \).

> **Definition 6** (Consistency). Let \( \mathcal{P} = (P_0, \ldots, P_n) \) be an APS in \( K \), \( k \in [0, n] \) 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 applies.
Model-Checking Counting Temporal Logics on Flat Structures

1. \( \Psi \in AP \) and \( \Psi \in \text{lab}_P(\ell) \) \( \Rightarrow \) \( \Psi = \lambda(st_P(\ell)) \), or \( \Psi = \varphi \lor \psi \) and \( \Psi \in \text{lab}_P(\ell) \) \( \Leftrightarrow \varphi, \psi \in \text{lab}_P(\ell) \), or \( \Psi = \neg \varphi \) and \( \Psi \in \text{lab}_P(\ell) \) \( \Leftrightarrow \varphi \notin \text{lab}_P(\ell) \).

2. \( \Psi = \chi \varphi \) and \( \forall \ell' \in \text{succ}_P(\ell) : \Psi \in \text{lab}_P(\ell) \) \( \Leftrightarrow \varphi \in \text{lab}_P(\ell') \).

3. \( \Psi = \varphi \cup^\ast \psi \) and one of the following holds:
   a. \( \Psi, \varphi \in \text{lab}_P(\ell) \)
   b. \( \Psi \in \text{lab}_P(\ell) \) and \( P_n \) is good for \( \Psi \) and \( \exists \ell' \in \text{loc}_P(n) : \psi \in \text{lab}_P(\ell') \)
   c. \( t_P(\ell) = \mathbb{R} \) and there is a counter \( c \in C \) such that \( \forall \ell' < \ell : u_P(\ell')(c) = 0 \) and \( \forall \ell' > \ell : \varphi \notin \text{lab}_P(\ell') \Rightarrow u_P(\ell')(c) = y - x \) and \( \forall \ell' \in \text{lab}_P(\ell') \) then \( \psi \notin \text{lab}_P(\ell') \) and \( \forall \ell' > \ell : \psi \in \text{lab}_P(\ell') \Rightarrow (c < 0) \in g_P(\ell') \) and
   d. There is \( k' \in [0, n] \) such that all locations \( \ell' \in \text{loc}_P(k') \) are consistent wrt. \( \Psi \) and
      - if \( k = n \) then \( k' < k \) and \( (P_k', P_{k+1}, \ldots, P_k) \) \( \{ \varphi, \psi, \Psi \} \)-periodic,
      - if \( k < n \) and \( P_k \) is good or neutral for \( \Psi \) and \( \Psi \notin \text{lab}_P(\ell') \), or \( P_k \) is bad for \( \Psi \) and \( \Psi \in \text{lab}_P(\ell') \) then \( k' < k < n \) and \( (P_k, P_{k+1}, \ldots, P_k+1) \) \( \{ \varphi, \psi, \Psi \} \)-periodic, and
      - if \( k < n \) and \( P_k \) is good or neutral for \( \Psi \) and \( \Psi \in \text{lab}_P(\ell') \), or \( P_k \) is bad for \( \Psi \) and \( \Psi \notin \text{lab}_P(\ell') \) then \( k < k' < n \) and \( (P_k, P_{k+1}, \ldots, P_k+1) \) \( \{ \varphi, \psi, \Psi \} \)-periodic.

The APS \( P \) 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 Figure 2 can be labelled consistently with \( X \mu P \) since all its successor (0 and 1) are labelled with \( p \).

Case 3, concerning the (frequency) until operator, is more involved.

Assume that \( \Phi = \varphi \cup^\ast \psi \) 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 \( \Psi = \mu r \cup q \) and the APS shown in Figure 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 1/2 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 [9] for further details). Hence we obtain the following lemma that guarantees correctness of our decision procedure.
Lemma 7 (Correctness). If there is an APS $\mathcal{P}$ in $\mathcal{K}$ such that $\mathcal{P}$ is consistent wrt. $\Phi$ and $\Phi \in \text{lab}_P(0)$ and $\text{Runs}(\mathcal{P}) \neq \emptyset$ then $\mathcal{K} \models \Phi$.

4.3 Constructing Consistent APS

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

Concerning the base case where $\Phi \in AP$, all paths in a flat structure can be represented by a path schema of linear size $[20, 11]$. Intuitively, since $\mathcal{K}$ is flat, every subpath $s_is_{i+1} \ldots s_{i'} \ldots s_{i''}$ of $\rho$ where a state $s_i = s_{i'} = s_{i''}$ occurs more than twice is equal to $(s_is_{i+1} \ldots s_{i-1})^ks_{i'}$ for some $k \in \mathbb{N}$. 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 $|u_0u_1 \ldots u_m| \leq 2|S|$. 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 $\mathcal{st}_\mathcal{P}(\sigma) = \rho$. 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 $\chi \varphi$. If $\Phi = \chi \varphi$ 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 $\varphi \forall^* \psi$. The most involved case is to label a location $\ell$ by $\Phi = \varphi \forall^* \psi$. 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 3a or 3b of Definition 6 applies. The conditions of case 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 at $\ell'$ 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 Figure 2 location 5 and a formula $\varphi = r \forall^2 q$. 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 $P_9$ 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 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 \( P = P[\ell_0] \ldots P[\ell_{|P|-1}] \) be a non-terminal loop in \( P \) with corresponding location sequence \( v = \ell_0 \ldots \ell_{|P|-1} \) and \( \hat{n} = |P| \cdot y \) for some \( y > 0 \). For every run \( \sigma = uv^n w \in \text{Runs}(P) \) where \( n \geq \hat{n} + 2 \) there are \( n_1 \) and \( n_2 \) such that \( \sigma = uv^{n_1} v^0 v^{n_2} w \) and for all positions \( i \) on \( \sigma \) with \( |i| \leq n_1 < |uv^{n_1} v^0| \) or \( |uv^{n_1} v^0| \leq i < |uv^{n_1} v^0 v^{n_2}| \) we have \( (\sigma, i) \models \Phi \) iff \( (\sigma, i + |P|) \models \Phi \).

**Example 9.** Consider again the APS \( P \) in Figure 2, a run \( \sigma \in \text{Runs}(P) \) and the location 3. Whether or not \( \varphi = r \mathcal{U} \bar{q} \) 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 \( i \) (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 Figure 3 and \( \sigma \) can easily be mapped onto this modified structure.

The presented procedure for constructing an APS from the run \( \rho \) in \( K \) performs only linearly many steps in \( |\Phi| \), namely one step for each subformula. It starts with a structure of size at most \( 2|K| \) 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 \( K \). 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.

**Lemma 10** (Completeness). If \( K \models \Phi \) then there is a consistent APS \( P \) in \( K \) of at most exponential size in \( K \) and \( \Phi \) where \( \Phi \in \text{lab}(P(0)) \) and \( P \) 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 \( |K| \). 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.

**Theorem 11.** \( MC(f_{K\Phi}, f_{\text{LT}}) \) is in \( \text{NEXP} \).

This result immediately extends to \( f_{\text{CTL^*}} \). For a state \( q \) of a flat Kripke structure \( K \) and an arbitrary \( f_{\text{LT}} \) formula \( \varphi \), the procedure allows us to decide in \( \text{NEXP} \) whether \( q \models f_{\text{LT}} \varphi \).
holds. It allows us further to decide if \( q \models A \varphi \) holds in \( \text{ExpSpace} \) by the dual formulation \( q \not\models E \neg\varphi \) 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 \( |K| + |\Phi| \).

\[ \text{Theorem 12. } \mathcal{MC}(\mathsf{FKS}, \mathsf{fCTL}^*) \text{ is in } \text{ExpSpace}. \]

5 On model-checking \( \mathsf{CCTL}^* \) over flat Kripke structures

In this section, we prove decidability of \( \mathcal{MC}(\mathsf{FKS}, \mathsf{CCTL}^*) \). 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 \( \text{CLTL}, \mathsf{CCTL} \) and \( \mathsf{CCTL}^* \).

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 \( \mathsf{PH} \). The syntax of \( \mathsf{PH} \) formulae \( \varphi \) and \( \mathsf{PH} \) terms \( \tau \) over a set of variables \( V \) is defined by the grammar

\[
\varphi ::= \tau \leq \tau \mid \neg \varphi \mid \varphi \lor \varphi \mid \exists x. \varphi \mid \exists^=x_1 \varphi \\
\tau ::= a \mid a \cdot x \mid \tau + \tau
\]

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 \( \tau \) as expected, e.g., \( \eta(3 \cdot x + 1) = 3 \cdot \eta(x) + 1 \). We define the satisfaction relation \( \models_{\mathsf{PH}} \) as usual for first-order logics and by \( \models_{\mathsf{PH}} \exists^=x_1 \varphi \), \( \mathbb{N} \ni \{b \in \mathbb{N} \mid \eta[y \mapsto b] \models_{\mathsf{PH}} \varphi \} = \eta(x) \) for the Härtig quantifier. Notice that the solution set has to be finite.

The satisfiability problem of \( \mathsf{PH} \) consists in determining whether for a \( \mathsf{PH} \) formula \( \varphi \) there exists a valuation \( \eta \) such that \( \eta \models_{\mathsf{PH}} \varphi \). It is decidable [2, 24, 25] via eliminating the Härtig quantifier, but its complexity is not known. For what concerns classic Presburger arithmetic, the complexity of its satisfiability problem lies between \( 2\text{Exp} \) and \( 2\text{ExpSpace} \) [4].

Lower bound for \( \mathcal{MC}(\mathsf{FKS}, \mathsf{CCTL}^*) \). Let \( K \) be the flat Kripke structure over \( \mathsf{AP} = \emptyset \) that consists of a single loop of length one. We can encode satisfiability of a \( \mathsf{PH} \) formula \( \Phi \) into the question whether the (unique) run \( \rho \) of \( K \) satisfies a \( \mathsf{CLTL} \) formula \( \hat{\Phi} \). Assume without loss of generality that \( \Phi \) has no free variables. Let \( V_\Phi \) be the variables used in \( \Phi \) and \( z_1, z_2, \ldots \not\in V_\Phi \) additional variables. Recall that \( \rho \models \hat{\Phi} \) if \( (\rho, \emptyset, 0) \models \hat{\Phi} \) for some valuation \( \emptyset \) of the positional variables in \( \hat{\Phi} \).

The idea is essentially to encode the value given to a variable \( x \in V_\Phi \) of \( \Phi \) into the distance between the positions assigned to two variables of \( \hat{\Phi} \). Technically, a mapping \( Z \in \mathbb{N}_k \) associates with each variable \( x \in V_\Phi \) an index \( j = Z(x) \) and the constraints that \( \Phi \) imposes on \( x \) are translated to constraints on positional variables \( z_j \) and \( z_{j-1} \) (more precisely, the distance \( \emptyset(z_j) - \emptyset(z_{j-1}) \) between the assigned positions). The following transformation \( t : \mathsf{PH} \times \mathbb{N}_k \times \mathbb{N} \rightarrow \mathsf{CLTL} \) constructs the \( \mathsf{CLTL} \) formula from \( \Phi \). When a variable is encountered, the mapping \( Z \) is updated by assigning to it the next free index (third parameter). Let

\[
\begin{align*}
\tau(\varphi_1 \odot \varphi_2, Z, i) &= \tau(\varphi_1, Z, i) \odot \tau(\varphi_2, Z, i) \\
\tau(a \cdot x, Z, i) &= a \cdot \#z_{i-1}(\top) - a \cdot \#z_i(\top) \\
\tau(\exists x. \varphi, Z, i) &= F z_i . \tau(\varphi, Z[x \mapsto i], i + 1) \\
\tau(\exists^=x_1 \varphi, Z, i) &= G \{ t(x, Z, i) = \#z_{i-1}(z_i . \tau(\varphi, Z[y \mapsto i], i + 1)) \}
\end{align*}
\]

for \( x, y \in V_\Phi \), \( a \in \mathbb{N} \) and \( \odot \in \{\land, \leq, +\} \). Then, we obtain \( \hat{\Phi} = z_0 . t(\Phi, 1, 1) \), initialising \( Z \) and the first free index with 1. Notice that the translation of the Härtig quantifier instantiates.
the scope effectively twice when substituting the equality and thus the size of $\hat{\Phi}$ may at worst double with each nesting. Finally, we can equivalently add path quantifiers to all temporal operators in $\hat{\Phi}$ and obtain, syntactically, a CCTL formula.

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

**Deciding $MC(FKS, CCTL^*)$.** We provide a polynomial reduction to the satisfiability problem of $PH$. Given a flat Kripke structure $\mathcal{K}$ we can represent each run $\rho$ by a fixed number of naturals. 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 $\text{Runs}(\mathcal{K})$. Such predicates were shown to be definable by Presburger arithmetic formulae of polynomial size and used to encode $MC(FKS, CCTL^*)$ [12, 10]. We adopt this idea for $MC(FKS, CCTL^*)$ and $PH$. Let $\mathcal{K} = (S, s_1, E, \lambda)$ and assume $S \subseteq \mathbb{N}$ without loss of generality. For $N \in \mathbb{N}$ 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.

**Lemma 14 ([10]).** There is a number $N \in \mathbb{N}$, a mapping $\text{enc} : N^N \rightarrow S^*$ and predicates $Conf(r_1, \ldots, r_N, i, s)$ and $Run(r_1, \ldots, r_N)$ such that for all valuations $\eta : V_N \rightarrow \mathbb{N}$ we have

1. $\eta \models_{PH} Run(r_1, \ldots, r_N) \iff \text{enc}(\eta(r_1), \ldots, \eta(r_N)) \in \text{Runs}(\mathcal{K})$ and
2. if $\eta \models_{PH} Run(r_1, \ldots, r_N)$ then $\eta \models_{PH} Conf(f_1, \ldots, f_m, r_N, i, s) \iff \text{enc}(\eta(r_1), \ldots, \eta(r_N))(\eta(i)) = \eta(s)$.

Both predicates are definable by PH formulae over variables $V \supseteq V_N$ of polynomial size in $|\mathcal{K}|$.

Now, let $\Phi$ be a $CCTL^*$ formula to be verified on $\mathcal{K}$. Without loss of generality we assume that all comparisons $\varphi \leq \psi$ in $\text{sub}(\Phi)$ of the form $\varphi_1 \leq \varphi_2$ have the shape $\varphi_1 = \sum_{i=1}^{k} a_i \cdot \#x_i(\varphi_i) + b \leq \sum_{i=k+1}^{m} a_i \cdot \#x_i(\varphi_i) + c$ for some $k, m, b, c \in \mathbb{N}$, coefficients $a_i \in \mathbb{N}$ and subformulae $\varphi_i$. As it is done in [10] for CTL, using the predicates $Conf$ and $Run$, we construct a PH formula that is satisfiable if and only if $\mathcal{K} \models \Phi$. Given the encoding of relevant runs into natural numbers we can express path quantifiers with quantification over the variables $r_1, \ldots, r_N$. Temporal operators can be expressed by using $Conf$ to access specific positions. Storing of positions is done explicitly by assigning them as value to specific variables $x$. Variables $z$ are introduced to hold the number of positions satisfying a formula and can then be used in constraints. For example, to translate a term $\#x(\varphi)$ we specify a variable, e.g., $z_1$ holding this value by $\exists z_1. \exists z_2. i \leq i' \wedge \varphi$ where $i$ holds the current position and $\varphi$ expresses that $\varphi$ holds at position $i'$ of the current run. Constraints like $\#x(\varphi) + 1 \leq \#x(\psi)$ can now directly be translated to, e.g., $z_1 + 1 \leq z_2$. We use a syntactic translation function $chk$ that takes the formula $\varphi$ to be translated, the names of $N$ variables encoding the current run and the name of the variable holding the current position. Let

\[
\begin{align*}
\text{chk}(p, r_1, \ldots, r_N, i) &= \exists s. Conf(r_1, \ldots, r_N, i, s) \land V_{\eta[p \in \lambda(a)]} \land s = a \\
\text{chk}(\varphi \land \psi, r_1, \ldots, r_N, i) &= \text{chk}(\varphi, r_1, \ldots, r_N, i) \land \text{chk}(\psi, r_1, \ldots, r_N, i) \\
\text{chk}(\neg \varphi, r_1, \ldots, r_N, i) &= \neg \text{chk}(\varphi, r_1, \ldots, r_N, i) \\
\text{chk}(X \varphi, r_1, \ldots, r_N, i) &= \exists i'. i' = i + 1 \land \text{chk}(\varphi, r_1, \ldots, r_N, i') \\
\text{chk}(\varphi \lor \psi, r_1, \ldots, r_N, i) &= \exists i'. i \leq i' \land \text{chk}(\psi, r_1, \ldots, r_N, i') \\
\text{chk}(E \varphi, r_1, \ldots, r_N, i) &= \exists r_1, \ldots, r_N. \text{Run}(r_1, \ldots, r_N) \land \text{chk}(\varphi, r_1, \ldots, r_N, i) \\
\text{chk}(x. \varphi, r_1, \ldots, r_N, i) &= \exists x. x = i \land \text{chk}(\varphi, r_1, \ldots, r_N, i) \\
\text{chk}(\varphi \leq, r_1, \ldots, r_N, i) &= \exists z_1, \ldots, z_m. (\bigwedge_{i=1}^{m} \exists z_i. i \leq i' \leq \text{chk}(\varphi, r_1, \ldots, r_N, i')) \land a_1 \cdot z_1 + \ldots + a_k \cdot z_k + b \leq a_{k+1} \cdot z_{k+1} + \ldots + a_m \cdot z_m + c
\end{align*}
\]
for $\varphi \leq \sum_{i=1}^{k} a_i \cdot \#x_i(\varphi_i) + b \leq \sum_{i=k+1}^{m} a_i \cdot \#x_i(\varphi_i) + c$. Primed variables denote fresh copies of the corresponding input variables, e.g. $i'$ becomes $(i')' = i''$ and $i''$ becomes $i'''$.

Now, $\Phi \models K$ if and only if $\exists r_1 \ldots \exists r_N. \exists i. Run(r_1, \ldots, r_N) \land i = 0 \land chk(\Phi, r_1, \ldots, r_N, i)$ is satisfiable.

▶ Theorem 15. $MC(FKS, CCTL^*)$ is reducible to 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 $fCTL$, $fLTL$ and $fCTL^*$ 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 CLTL and counter systems and as future work we plan to study automata-based formalisms inspired by $fLTL$ where we will equip our automata with some counters whose role will be to evaluate the relative frequency of particular events.

References


Model-Checking Counting Temporal Logics on Flat Structures


