Model-checking Counting Temporal Logics on Flat Structures*

Normann Decker[1], Peter Habermehl[2], Martin Leucker[1], Arnaud Sangnier[2], and Daniel Thoma[1]

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

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2017.25

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 $_{\mathsf{c}}$CTL. This formalism can state properties such as an event p will eventually occur and before that, the number of events

* Partially supported by EGIDE/DAAD-Procope (FREQS) and European Commission H2020 Project COEMS. An extended version is available [9] providing additional proof details.

28th International Conference on Concurrency Theory (CONCUR 2017).
Editors: Roland Meyer and Uwe Nestmann; Article No. 25; pp. 25:1–25:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

q is larger than two. The authors propose further an extension called (here) $_{\mathtt{c}}\mathtt{CTL}_{\pm}$ 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 $_{\mathtt{c}}\mathtt{CTL}$ is decidable in polynomial time and that the satisfiability problem for $_{\mathtt{c}}\mathtt{CTL}_{\pm}$ is undecidable. A similar extension for LTL is considered in [18] where it is proven that model checking of $_{\mathtt{c}}\mathtt{LTL}$ is ExpSpace-complete while that of $_{\mathtt{c}}\mathtt{LTL}_{\pm}$ 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 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 $\varphi \mathtt{U} \psi$ requires φ to hold at all positions before ψ. For example, the fLTL formula $p \mathtt{U}^{\frac{1}{3}} q$ 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, fLTL can be considered as a fragment of $_{\mathtt{c}}\mathtt{LTL}_{\pm}$ 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 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 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 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 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

	CTL	LTL	CTL*	fLTL	fCTL	fCTL*	CLTL	CCTL	CCTL*
KS	P	PSpace-c.	PSpace-c.	undec. [5]	**Exp**	undec.	undec.	undec. [19]	undec.
FKS	P	NP-c. [17]	PSpace	**NExp**	**Exp**	**ExpSpace**	**PH**	**PH**	**PH**

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

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 $\mathbb{N}$ 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 Σ, Σ^* represents the set of finite words over Σ, Σ^+ the set of finite non-empty words over Σ and Σ^ω 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 $u = a_0a_1 \dots a_k \dots$ of length at least $k+1 > 0$ we denote by $u(k) = a_k$ the $(k+1)$-th element and refer to its indices $0, 1, \dots$ as positions on u. If u is finite then $|u|$ denotes its length. For arbitrary functions $f : A \to B$ 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 $\mathbf{0}$ and $\mathbf{1}$ for the functions $f_0 : A \to \{0\}$ and $f_1 : A \to \{1\}$, 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 AP be a finite set of *atomic propositions*. A *Kripke structure* is a tuple $\mathcal{K} = (S, s_I, E, \lambda)$ 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 $\lambda : S \mapsto 2^{AP}$ the labelling function. A finite *path* in $\mathcal{K}$ is a sequence $u = s_0s_1 \dots s_k \in S^+$ with $(s_i, s_{i+1}) \in E$ for all $i \in [0, k-1]$. Infinite paths are defined analogously. A *run* ρ of $\mathcal{K}$ is an infinite path with $\rho(0) = s_I$. We denote by $\texttt{Runs}(\mathcal{K})$ the set of runs of $\mathcal{K}$. Due to the single initial state, we assume without loss of generality that the graph of $\mathcal{K}$ is connected, i.e. all states are reachable. A *simple loop* in $\mathcal{K}$ is a finite path $u = s_0s_1 \dots s_k$ 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 $\mathcal{K}$ is called *flat* if for each state $s \in S$ there is at most one simple loop u in $\mathcal{K}$ 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 $\mathcal{S} = (S, s_I, C, \Delta)$ 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 $\Delta \subseteq S \times \mathbb{Z}^C \times 2^{\mathfrak{G}(C)} \times S$ is the transition relation where $\mathfrak{G}(C) = \{(c < 0), (c \geq 0) \mid c \in C\}$. An infinite sequence $s_0 s_1 \ldots \in S^\omega$ of states starting in $s_0 = s_I$ is called a *run* of $\mathcal{S}$ 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 $\theta_0 = \mathbf{0}$ and a transition $(s_i, \mathbf{u}_i, G_i, s_{i+1}) \in \Delta$ for every $i \in \mathbb{N}$ such that $\theta_{i+1} = \theta_i + \mathbf{u}_i$ (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 $\mathtt{Runs}(\mathcal{S})$ the set of all such runs and assume the graph of control states underlying $\mathcal{S}$ 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 CCTL* which extends the branching-time logic 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 CCTL* formulae φ over V and AP is given by the grammar rules

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathtt{X}\,\varphi \mid \varphi\,\mathtt{U}\,\varphi \mid \mathtt{E}\,\varphi \mid x.\varphi \mid \tau \leq \tau \qquad \tau ::= a \mid a \cdot \#_x(\varphi) \mid \tau + \tau$$

for $p \in AP$, $x \in V$ and $a \in \mathbb{Z}$. Common abbreviations such as $\top \equiv p \vee \neg p$, $\bot \equiv \neg\top$, $\mathtt{F}\,\varphi \equiv \top\,\mathtt{U}\,\varphi$, $\mathtt{G}\,\varphi \equiv \neg\,\mathtt{F}\,\neg\varphi$ and $\mathtt{A}\,\varphi \equiv \neg\,\mathtt{E}\,\neg\varphi$ may also be used. The set of all *subformulae* of a formula φ (including itself) is denoted $\mathtt{sub}(\varphi)$ and $|\varphi|$ denotes the length of φ, 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 φ holds between the current position and that marked by x. The semantics of CCTL* is hence defined with respect to a Kripke structure $\mathcal{K} = (S, s_I, E, \lambda)$, a run $\rho \in \mathtt{Runs}(\mathcal{K})$, a position $i \in \mathbb{N}$ on ρ and a valuation function $\theta : V \to \mathbb{N}$ assigning a position (index) on ρ to each variable. The satisfaction relation $\models$ is defined inductively for $p \in AP$, formulae φ, ψ and terms τ_1, τ_2 by

$$\begin{array}{lcl}
(\rho, i, \theta) \models p & \stackrel{\mathsf{def}}{\Leftrightarrow} & p \in \lambda(\rho(i)), \\
(\rho, i, \theta) \models \mathtt{X}\,\varphi & \stackrel{\mathsf{def}}{\Leftrightarrow} & (\rho, i+1, \theta) \models \varphi, \\
(\rho, i, \theta) \models \varphi\,\mathtt{U}\,\psi & \stackrel{\mathsf{def}}{\Leftrightarrow} & \exists k \geq i : (\rho, k, \theta) \models \psi \text{ and } \forall j \in [i, k-1] : (\rho, j, \theta) \models \varphi, \\
(\rho, i, \theta) \models \mathtt{E}\,\varphi & \stackrel{\mathsf{def}}{\Leftrightarrow} & \exists \rho' \in \mathtt{Runs}(\mathcal{K}) : \forall j \in [0, i] : \rho'(j) = \rho(j) \text{ and } (\rho', i, \theta) \models \varphi, \\
(\rho, i, \theta) \models x.\varphi & \stackrel{\mathsf{def}}{\Leftrightarrow} & (\rho, i, \theta[x \mapsto i]) \models \varphi, \\
(\rho, i, \theta) \models \tau_1 \leq \tau_2 & \stackrel{\mathsf{def}}{\Leftrightarrow} & [\![\tau_1]\!](\rho, i, \theta) \leq [\![\tau_2]\!](\rho, i, \theta),
\end{array}$$

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

$$\begin{array}{rcl}
[\![a]\!](\rho, i, \theta) & \stackrel{\mathsf{def}}{=} & a, \\
[\![\tau_1 + \tau_2]\!](\rho, i, \theta) & \stackrel{\mathsf{def}}{=} & [\![\tau_1]\!](\rho, i, \theta) + [\![\tau_2]\!](\rho, i, \theta), \\
[\![a \cdot \#_x(\varphi)]\!](\rho, i, \theta) & \stackrel{\mathsf{def}}{=} & a \cdot |\{j \in \mathbb{N} \mid \theta(x) \leq j \leq i, (\rho, j, \theta) \models \varphi\}|.
\end{array}$$

We abbreviate $(\rho, i, \mathbf{0}) \models \varphi$ by $(\rho, i) \models \varphi$ and $(\rho, 0) \models \varphi$ by $\rho \models \varphi$ and say that ρ satisfies φ (at position i) in these cases. Moreover, we say a state $s \in S$ satisfies φ, denoted $s \models \varphi$ if there are $\rho_s \in \mathtt{Runs}(\mathcal{K})$ and $i \in \mathbb{N}$ such that $\rho_s(i) = s$ and $(\rho_s, i) \models \varphi$. The Kripke structure

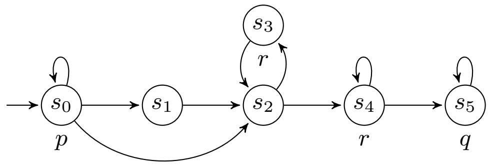

Figure 1 A flat Kripke over $AP = \{p, q, r\}$.

$\mathcal{K}$ satisfies φ, denoted by $\mathcal{K} \models \varphi$, 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 $\mathtt{CCTL}^*$ in analogy to the classical logics $\mathtt{LTL}$ and $\mathtt{CTL}$. The *linear* time fragment $\mathtt{CLTL}$ consists of those $\mathtt{CCTL}^*$ formulae that do not use the path quantifiers $\mathtt{E}$ and $\mathtt{A}$. The *branching* time logic $\mathtt{CCTL}$ restricts the use of temporal operators $\mathtt{X}$ and $\mathtt{U}$ such that each occurrence must be preceded immediately by either $\mathtt{E}$ or $\mathtt{A}$. 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 $\mathtt{fCTL}^*$, $\mathtt{fLTL}$ and $\mathtt{fCTL}$ as fragments of $\mathtt{CCTL}^*$. Consider the following grammar defining the syntax of formulae φ for natural numbers $n, m \in \mathbb{N}$ with $0 < m$, $n \leq m$ and $p \in AP$.

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \alpha \qquad\qquad \beta ::= \mathtt{X}\,\varphi \mid \varphi\,\mathtt{U}^{\frac{n}{m}}\,\varphi$$

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

$$\varphi\,\mathtt{U}^{\frac{n}{m}}\,\psi \stackrel{\mathsf{def}}{\equiv} \psi \vee x\,.\,\mathtt{F}\,\left((\mathtt{X}\,\psi)\;\wedge\; m\cdot\#_x(\varphi) \geq n\cdot\#_x(\top)\right) \tag{1}$$

for $\mathtt{fCTL}^*$ formulae φ and ψ and a variable $x \in V$ not being used in either φ or ψ.

▶ **Example 1.** *Consider the Kripke structure given by Fig. 1 and the $\mathtt{CCTL}$ formula $\varphi_1 = z.\,\mathtt{A}\,\mathtt{G}\,(q \to (\#_z(p) \leq \#_z(\mathtt{E}\,\mathtt{X}\,r)))$. It basically states that on every path reaching s_5 there must be a position where the states s_2 and s_4 (satisfying $\mathtt{E}\,\mathtt{X}\,r$) 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: $\varphi_1' = \mathtt{A}((\mathtt{E}\,\mathtt{X}\,r)\,\mathtt{U}^{\frac{1}{2}}\,q)$ 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 φ_1 and φ_1' are violated, e.g. by the path $s_0^3 s_1 s_2 s_4 s_5^\omega$. The Kripke structure however satisfies $\varphi_2 = z.\,\mathtt{A}\,\mathtt{G}\,(\neg q \to \mathtt{E}\,\mathtt{F}\,\#_z(p) < \#_z(r))$ 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 $\mathtt{CCTL}^*$ are a very flexible way of defining the scope of a constraint, frequency constraints in $\mathtt{fCTL}^*$ are always bound to the scope of an until operator. The same applies to the counting constraints of $_{\mathtt{c}}\mathtt{LTL}$ as defined in [19]. For example, the $_{\mathtt{c}}\mathtt{LTL}$ formula $\varphi\,\mathtt{U}_{[a_1\#(\varphi_1)+\cdots+a_n\#(\varphi_n)\geq k]}\,\psi$ is equivalent to the $\mathtt{CLTL}$ formula $z.\varphi\,\mathtt{U}(\psi \wedge a_1\#_z(\varphi_1) + \cdots + a_n\#_z(\varphi_n) \geq k)$. Admitting only natural coefficients, $_{\mathtt{c}}\mathtt{LTL}$ can be encoded even in $\mathtt{LTL}$ making it thus strictly less expressive than $\mathtt{fLTL}$. On the other hand,

$_{\mathtt{c}}\mathtt{LTL}_{\pm}$ admits arbitrary integer coefficients, which is more general than the frequency until operator of $\mathtt{fLTL}$. For example, $p\,\mathtt{U}^{\frac{a}{b}}\,q$ can be expressed as $\top\,\mathtt{U}_{[b\#(p)-a\#(\top)\geq 0]}\,q$ in $_{\mathtt{c}}\mathtt{LTL}_{\pm}$. The relation between $_{\mathtt{c}}\mathtt{CTL}_{\pm}$ and $\mathtt{fCTL}$, as well as $_{\mathtt{c}}\mathtt{CTL}^{*}_{\pm}$ and $\mathtt{fCTL}^{*}$ is analogous.

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

Input: A Kripke structure $\mathcal{K} \in \mathfrak{K}$ and a formula $\varphi \in \mathcal{L}$. **Decide:** Does $\mathcal{K} \models \varphi$ hold?

For temporal logics without counting variables, the model-checking problem over Kripke structure has been studied intensively and is known to be PSPACE-complete for $\mathtt{LTL}$ and $\mathtt{CTL}^{*}$ and in P for $\mathtt{CTL}$ (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 $\mathtt{LTL}$ is lower than in the general case [17]: it drops from PSPACE to NP. As we show later, in the case of $\mathtt{CCTL}^{*}$, 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 $\mathtt{CCTL}^{*}$ 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 $\mathtt{fLTL}$ is undecidable [5] implying the same for model-checking $\mathtt{fLTL}$, $\mathtt{CLTL}$ and $\mathtt{CCTL}^{*}$ over Kripke structures. This applies moreover to $\mathtt{CCTL}$ [19]. In contrast, we show in the following that $\mathtt{MC}(\mathtt{KS}, \mathtt{fCTL})$ is decidable using an extension of the well-known labelling algorithm for $\mathtt{CTL}$ (see e.g. [3]).

Let $\mathcal{K} = (S, s_I, E, \lambda)$ be a Kripke structure and Φ an $\mathtt{fCTL}$ formula. We compute recursively subsets $S_\varphi \subseteq S$ of the states of $\mathcal{K}$ for every subformula $\varphi \in \mathtt{sub}(\Phi)$ of Φ 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_Φ then solves the problem. Propositions ($p \in AP$), negation ($\neg\varphi$), conjunction ($\varphi \wedge \psi$) and *temporal next* ($\mathtt{E}\,\mathtt{X}\,\varphi$, $\mathtt{A}\,\mathtt{X}\,\varphi$) are handled as usual, e.g. $S_p = \{q \in S \mid p \in \lambda(q)\}$ and $S_{\mathtt{E}\,\mathtt{X}\,\varphi} = \{q \in S \mid \exists q' \in S_\varphi : (q, q') \in \delta\}$.

To compute if a state $s \in S$ satisfies a formula of the form $\mathtt{E}\,\varphi\,\mathtt{U}^{r}\,\psi$ or $\mathtt{A}\,\varphi\,\mathtt{U}^{r}\,\psi$, assume that S_φ and S_ψ are given inductively. If $s \in S_\psi$ we immediately have $s \in S_{\mathtt{E}\,\varphi\,\mathtt{U}^{r}\,\psi}$ and $s \in S_{\mathtt{A}\,\varphi\,\mathtt{U}^{r}\,\psi}$. For the remaining cases, the problem of deciding whether $s \in S_{\mathtt{E}\,\varphi\,\mathtt{U}^{r}\,\psi}$ or $s \in S_{\mathtt{A}\,\varphi\,\mathtt{U}^{r}\,\psi}$, 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 $\mathcal{K}$ 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 passing any position on ρ we pay a fee of n and for those positions that satisfy φ we gain a reward of m. Thus, we obtain a non-negative balance of rewards and gains at some position on ρ if, in average, among every m positions there are at least n positions that satisfy φ, 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

$$\Delta = \{(t, \mathbf{u}, \emptyset, t') \mid (t, t') \in E,\ t \notin S_\varphi \Rightarrow \mathbf{u}(c) = -n,\ t \in S_\varphi \Rightarrow \mathbf{u}(c) = m - n\}.$$

The state s satisfies the formula $\mathtt{A}\,\varphi\,\mathtt{U}^{r}\,\psi$ if there is no path starting in state s violating the formula $\varphi\,\mathtt{U}^{r}\,\psi$. The latter is the case if at every position where ψ 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 $s \in S_{\mathtt{A}\,\varphi\,\mathtt{U}^r\,\psi}$ it suffices to check that in $\mathcal{R}_s$ no state is repeatedly reachable from s.

A formula $\mathtt{E}\,\varphi\,\mathtt{U}^r\,\psi$ 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 $\mathcal{U}_s = (S \uplus \{\mathtt{t}\}, s, \{c\}, \Delta')$ obtained from $\hat{\mathcal{K}}_s$ featuring a new sink state $\mathtt{t} \notin S$. The transition relation

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

extends Δ 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 $\mathtt{t}$. Checking if s is supposed to be contained in $S_{\mathtt{E}\,\varphi\,\mathtt{U}^r\,\psi}$ then amounts to decide whether $\mathtt{t}$ 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 $\ominus^n$ 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 Φ, 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.** *MC(KS, 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 $\mathcal{K}$ and an fLTL formula Φ, 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 fLTL formula Φ.
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 Φ.

We fix for this section a flat Kripke structure $\mathcal{K} = (S, s_I, E, \lambda)$ and an fLTL formula Φ. For convenience we assume that $AP \subseteq \mathtt{sub}(\Phi)$. Omitted technical details can be found in [9].

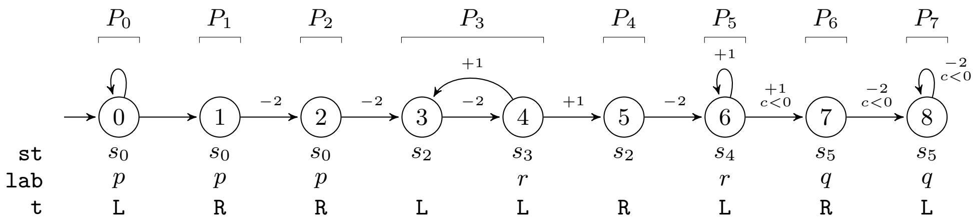

Figure 2 An APS $\mathcal{P} = (P_0, \ldots, P_7)$ of the flat Kripke structure in Fig. 1

4.1 Augmented Path Schemas

The set of runs of $\mathcal{K}$ can be represented as a finite number of so-called path schemas that consist of a sequence of paths and simple loops consecutive in $\mathcal{K}$ [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 $\mathcal{K}$ is a tuple $a = (s, L, G, \mathbf{u}, t) \in S \times 2^{\mathtt{sub}(\Phi)} \times 2^{\mathfrak{G}(C)} \times \mathbb{Z}^C \times \{\mathtt{L}, \mathtt{R}\}$ comprised of a state s of $\mathcal{K}$, a set of formula labels L, guards G and an update $\mathbf{u}$ over a set of counter names C, and a *type* indicating whether the state is part of a loop ($\mathtt{L}$) or a not ($\mathtt{R}$). We denote by $\mathtt{st}(a) = s$, $\mathtt{lab}(a) = L$, $\mathtt{g}(a) = G$, $\mathtt{u}(a) = \mathbf{u}$ and $\mathtt{t}(a) = t$ the respective components of a. An *augmented path* in $\mathcal{K}$ is a sequence $u = a_0 \ldots a_n$ of augmented states a_i such that $(\mathtt{st}(a_i), \mathtt{st}(a_{i+1})) \in E$ for $i \in [0, n-1]$. If $\mathtt{t}(a_i) = \mathtt{R}$ 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 $(\mathtt{st}(a_n), \mathtt{st}(a_1)) \in E$ and $\mathtt{st}(a_i) \neq \mathtt{st}(a_j)$ for $i \neq j$ and $\mathtt{t}(a_i) = \mathtt{L}$ for all $i \in [0, n-1]$.

An *augmented path schema (APS)* in $\mathcal{K}$ 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 indicating the possible state progressions. It features a single counter that tracks the frequency constraint of a formula $r \, \mathtt{U}^{\frac{2}{3}} \, q$ 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, $\mathtt{loc}_{\mathcal{P}}(k) = \{\ell \mid |P_0 P_1 \ldots P_{k-1}| \leq \ell < |P_0 P_1 \ldots P_k|\}$ denotes for $0 \leq k \leq n$ the set of locations belonging to component P_k and for all locations $\ell \in \mathtt{loc}_{\mathcal{P}}(k)$ we denote the corresponding component index in $\mathcal{P}$ by $\mathtt{comp}_{\mathcal{P}}(\ell) = k$. For example, in Fig. 2 we have $\mathtt{loc}_{\mathcal{P}}(3) = \{3, 4\}$ and $\mathtt{comp}_{\mathcal{P}}(6) = 5$ 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., $\mathtt{st}_{\mathcal{P}}(\ell_1 \ell_2)$ for $\mathtt{st}(\mathcal{P}[\ell_1])\mathtt{st}(\mathcal{P}[\ell_2])$ and $\mathtt{u}_{\mathcal{P}}(\ell)$ for $\mathtt{u}(\mathcal{P}[\ell])$.

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

Finally, notice that $\mathtt{CS}(\mathcal{P})$ is in fact a *flat* counter system. It is shown in [11] that $\mathtt{LTL}$ properties can be verified over flat counter systems in non-deterministic polynomial time. Since $\mathtt{LTL}$ can express that each location of $\mathtt{CS}(\mathcal{P})$ 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 σ the labelling $L_i = \mathtt{lab}_{\mathcal{P}}(\sigma(i))$. We are interested in this labelling being *correct* with respect to some $\mathtt{fLTL}$ formula Φ 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 $\mathtt{fLTL}$ formula $\Psi = \varphi\, \mathtt{U}^{\frac{x}{y}}\, \psi$ if the number $d = |\{0 \leq i < |u| \mid \varphi \in \mathtt{lab}(u(i))\}|$ of positions labelled with φ is larger than $(d > \frac{x}{y} \cdot |u|)$, equal to $(d = \frac{x}{y} \cdot |u|)$ or smaller than $(d < \frac{x}{y} \cdot |u|)$, respectively, the fraction $\frac{x}{y}$ 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 $L \subseteq \mathtt{sub}(\Phi)$ 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 $\mathtt{lab}(P_k(i)) \cap L = \mathtt{lab}(P_{k+1}(i)) \cap L$ for all $0 \leq i < |P_k|$.

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

1. *$\Psi \in AP$ and $\Psi \in \mathtt{lab}_{\mathcal{P}}(\ell) \Leftrightarrow \Psi \in \lambda(\mathtt{st}_{\mathcal{P}}(\ell))$, or $\Psi = \varphi \wedge \psi$ and $\Psi \in \mathtt{lab}_{\mathcal{P}}(\ell) \Leftrightarrow \varphi, \psi \in \mathtt{lab}_{\mathcal{P}}(\ell)$, or $\Psi = \neg\varphi$ and $\Psi \in \mathtt{lab}_{\mathcal{P}}(\ell) \Leftrightarrow \varphi \notin \mathtt{lab}_{\mathcal{P}}(\ell)$.*
2. *$\Psi = \mathtt{X}\,\varphi$ and $\forall \ell' \in \mathtt{succ}_{\mathcal{P}}(\ell) : \Psi \in \mathtt{lab}_{\mathcal{P}}(\ell) \Leftrightarrow \varphi \in \mathtt{lab}_{\mathcal{P}}(\ell')$.*
3. *$\Psi = \varphi\, \mathtt{U}^{\frac{x}{y}}\, \psi$ and one of the following holds:*
   - **a.** *$\Psi, \psi \in \mathtt{lab}_{\mathcal{P}}(\ell)$*
   - **b.** *$\Psi \in \mathtt{lab}_{\mathcal{P}}(\ell)$ and P_n is good for Ψ and $\exists \ell' \in \mathtt{loc}_{\mathcal{P}}(n) : \psi \in \mathtt{lab}_{\mathcal{P}}(\ell')$*
   - **c.** *$\mathtt{t}_{\mathcal{P}}(\ell) = \mathtt{R}$ and there is a counter $c \in C$ such that $\forall \ell' < \ell : \mathtt{u}_{\mathcal{P}}(\ell')(c) = 0$ and $\forall \ell' \geq \ell : \varphi \in \mathtt{lab}_{\mathcal{P}}(\ell') \Rightarrow \mathtt{u}_{\mathcal{P}}(\ell')(c) = y - x$ and $\forall \ell' \geq \ell : \varphi \notin \mathtt{lab}_{\mathcal{P}}(\ell') \Rightarrow \mathtt{u}_{\mathcal{P}}(\ell')(c) = -x$ and*
     - *if $\Psi \notin \mathtt{lab}_{\mathcal{P}}(\ell)$ then $\psi \notin \mathtt{lab}_{\mathcal{P}}(\ell)$ and $\forall \ell' > \ell : \psi \in \mathtt{lab}_{\mathcal{P}}(\ell') \Rightarrow (c < 0) \in \mathtt{g}_{\mathcal{P}}(\ell')$ and*
     - *if $\Psi \in \mathtt{lab}_{\mathcal{P}}(\ell)$ then $\exists \ell' > \ell : \psi \in \mathtt{lab}_{\mathcal{P}}(\ell') \wedge (c \geq 0) \in \mathtt{g}_{\mathcal{P}}(\ell')$.*
   - **d.** *There is $k' \in [0, n]$ such that all locations $\ell' \in \mathtt{loc}_{\mathcal{P}}(k')$ are consistent wrt. Ψ 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 Ψ and $\Psi \notin \mathtt{lab}_{\mathcal{P}}(\ell)$, or P_k is bad for Ψ and $\Psi \in \mathtt{lab}_{\mathcal{P}}(\ell)$ then $k' < k < n$ and $(P_{k'}, P_{k'+1}, \ldots, P_{k+1})$ is $\{\varphi, \psi, \Psi\}$-periodic, and*
     - *if $k < n$ and P_k is good or neutral for Ψ and $\Psi \in \mathtt{lab}_{\mathcal{P}}(\ell)$, or P_k is bad for Ψ and $\Psi \notin \mathtt{lab}_{\mathcal{P}}(\ell)$ 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 Ψ 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 $\mathtt{X}\,p$ since all its sucessor (0 and 1) are labelled with p. Case **3**, concerning the (frequency) until operator, is more involved.

Assume that $\Phi = \varphi \, \mathtt{U}^{\frac{x}{y}} \, \psi$ is an until formula and that the labelling of $\mathcal{K}$ by φ and ψ is consistent. In some cases, it is obvious that Φ holds, namely at positions labelled by ψ (case **3a**) or if the final loop already guarantees that Φ 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 Φ holds (case **3c**). Recall that to validate (or invalidate) the labelling of a location by the formula Φ 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 Φ there should exist a location in the future where ψ holds and the balance counter is non-negative. For those not labelled with Φ all locations in the future where ψ 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 Φ 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 Φ 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 $\Psi = r \, \mathtt{U}^{\frac{2}{3}} \, q$ and the APS shown in Fig. 2. It is consistent to *not* label location 1 by Ψ 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 Ψ 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 Φ then for every run of the APS, each time the formula Φ 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. Φ and $\Phi \in \mathtt{lab}_{\mathcal{P}}(0)$ and $\mathtt{Runs}(\mathcal{P}) \neq \emptyset$ then $\mathcal{K} \models \Phi$.*

4.3 Constructing Consistent APS

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

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 $s_i s_{i+1} \ldots s_{i'} \ldots s_{i''}$ of ρ where a state $s_i = s_{i'} = s_{i''}$ occurs more than twice is equal to $(s_i s_{i+1} \ldots s_{i'-1})^k s_{i''}$ for some $k \in \mathbb{N}$. Hence, there are simple subpaths $u_0, \ldots, u_m \in S^+$ of ρ and positive numbers of iterations $n_0, \ldots, n_{m-1} \in \mathbb{N}$ such that $\rho = u_0^{n_0} u_1^{n_1} \ldots u_{m-1}^{n_{m-1}} u_m^{\omega}$ and $|u_0 u_1 \ldots u_m| \leq 2|S|$. 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 Φ and a run $\sigma \in \mathtt{Runs}(\mathcal{P})$ with $\mathtt{st}_{\mathcal{P}}(\sigma) = \rho$. If $\Phi = \varphi \wedge \psi$ or $\Phi = \neg\varphi$, Definition 6 determines for each augmented state of $\mathcal{P}$ whether it is supposed to be labelled by Φ or not. It remains hence to deal with the next and frequency until operators.

Labelling $\mathcal{P}$ by $\mathtt{X}\,\varphi$. If $\Phi = \mathtt{X}\,\varphi$ the labelling at some location ℓ is extended according to the labelling of its successors. These may disagree upon φ (only) if ℓ has more than one

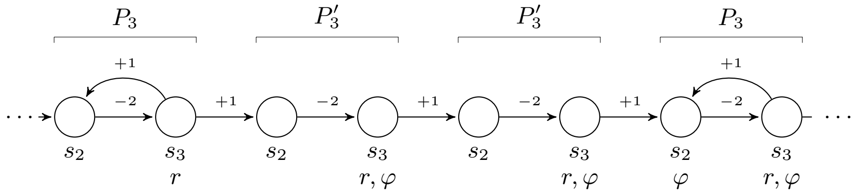

Figure 3 A decomposition of loop P_3 from Fig. 2 allowing for a correct labelling wrt. $\varphi = r\,\mathtt{U}^{\frac{2}{3}}\,q$.

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 σ: 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 $\mathtt{R}$ instead of $\mathtt{L}$. If otherwise σ 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, σ 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 $\varphi\,\mathtt{U}^r\,\psi$. The most involved case is to label a location ℓ by $\Phi = \varphi\,\mathtt{U}^r\,\psi$. First, assume that ℓ is part of a row. Whether it must be labelled by Φ is uniquely determined by σ. 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 Φ holds at ℓ but every location ℓ' witnessing this (by being reachable with sufficient frequency and labelled by ψ) is part of some loop P'. Adding the required guard directly to ℓ' may be too strict if σ traverses P' more than once. However, the first iteration (if P' is bad for Φ) or the last iteration (if P' is good) on σ contains a position (labelled with ψ) witnessing that Φ 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 $\varphi = r\,\mathtt{U}^{\frac{2}{5}}\,q$. Location 8 could witness that φ holds but a corresponding guard would be violated eventually since P_7 is bad for φ. The first iteration is thus the optimal choice. The unfolding P_6 separates it such that location 7 can be guarded instead without imposing unnecessary constraints.

Now assume that location ℓ, to be labelled or not with Φ, is part of a loop P which is *stable* in the sense that Φ 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, σ may go through ℓ several, say $n > 1$, times where Φ 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 $\sigma = uv^n w \in \mathtt{Runs}(\mathcal{P})$ where $n \geq \hat{n} + 2$ there are n_1 and n_2 such that $\sigma = uv^{n_1}v^{\hat{n}}v^{n_2}w$ and for all positions i on σ with $|u| \leq i < |uv^{n_1-1}|$ or $|uv^{n_1}v^{\hat{n}}| \leq i < |uv^{n_1}v^{\hat{n}}v^{n_2-2}|$ we have $(\sigma, i) \models_{\mathcal{P}} \Phi$ iff $(\sigma, i + |P|) \models_{\mathcal{P}} \Phi$.*

▶ **Example 9.** *Consider again the APS $\mathcal{P}$ in Fig. 2, a run $\sigma \in \mathtt{Runs}(\mathcal{P})$ and the location 3. Whether or not $\varphi = r\,\mathtt{U}^{\frac{2}{3}}\,q$ holds at some position i with $\sigma(i) = 3$ depends on how often σ traverses the good loop P_5 (the more the better) and how often it repeats P_3 after position i*

(the more the worse). Assume σ traverses P_5 exactly five times and P_3 sufficiently often, say 10 times. Then, during the last three iterations of P_3, φ 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 σ can easily be mapped onto this modified structure.

The presented procedure for constructing an APS from the run ρ 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 $\mathcal{P}_\Phi$ of size at most exponential in the length of Φ and polynomial in the number of states of $\mathcal{K}$. This consistent APS still contains a run corresponding to ρ and hence its first location must be labelled by Φ because $(\rho, 0) \models \Phi$ and we have seen that consistency implies correctness.

▶ **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 Φ where $\Phi \in \mathtt{lab}(\mathcal{P}(0))$ and $\mathcal{P}$ 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 $|\mathcal{K}|$. 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.

▶ **Theorem 11.** *$\mathtt{MC}(\mathtt{FKS}, \mathtt{fLTL})$ is in* NExp*.*

This result immediately extends to $\mathtt{fCTL}^*$. For a state q of a flat Kripke structure $\mathcal{K}$ and an arbitrary $\mathtt{fLTL}$ formula φ, the procedure allows us to decide in NExp whether $q \models \mathtt{E}\,\varphi$ holds. It allows us further to decide if $q \models \mathtt{A}\,\varphi$ holds in ExpSpace by the dual formulation $q \not\models \mathtt{E}\,\neg\varphi$ and Savitch's theorem. Following otherwise the standard labeling procedure for $\mathtt{CTL}$ (cf. Section 3) requires to invoke the procedure a polynomial number of times in $|\mathcal{K}| + |\Phi|$.

▶ **Theorem 12.** *$\mathtt{MC}(\mathtt{FKS}, \mathtt{fCTL}^*)$ is in* ExpSpace*.*

5 On model-checking $\mathtt{CCTL}^*$ over flat Kripke structures

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

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 $\mathtt{PH}$. The syntax of $\mathtt{PH}$ formulae φ and $\mathtt{PH}$ terms τ over a set of variables V is defined by the grammar

$$\varphi ::= \tau \leq \tau \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\varphi \mid \exists^{=x} y.\varphi \qquad\qquad \tau ::= a \mid a \cdot x \mid \tau + \tau$$

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 $\eta : V \to \mathbb{N}$ 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 $\models_{\mathtt{PH}}$ as usual for first-order logics and by $\eta \models_{\mathtt{PH}} \exists^{=x} y.\varphi \overset{\text{def}}{\Leftrightarrow} \mathbb{N} \ni |\{b \in \mathbb{N} \mid \eta[y \mapsto b] \models_{\mathtt{PH}} \varphi\}| = \eta(x)$ for the Härtig quantifier. Notice that the solution set has to be finite.

The *satisfiability problem* of PH consists in determining whether for a PH formula φ there exists a valuation η such that $\eta \models_{\mathtt{PH}} \varphi$. 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 2Exp and 2ExpSpace [4].

Lower bound for MC(FKS, CCTL*). Let $\mathcal{K}$ be the flat Kripke structure over $AP = \emptyset$ that consists of a single loop of length one. We can encode satisfiability of a PH formula Φ into the question whether the (unique) run ρ of $\mathcal{K}$ satisfies a CLTL formula $\hat{\Phi}$. Assume without loss of generality that Φ has no free variables. Let V_Φ be the variables used in Φ and $z_1, z_2, \ldots \notin V_\Phi$ additional variables. Recall that $\rho \models \hat{\Phi}$ if $(\rho, \theta, 0) \models \hat{\Phi}$ for some valuation θ of the positional variables in $\hat{\Phi}$.

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

$$\begin{array}{lcllcl}
\mathtt{t}(\varphi_1 \odot \varphi_2, Z, i) & = & \mathtt{t}(\varphi_1, Z, i) \odot \mathtt{t}(\varphi_2, Z, i) & \mathtt{t}(\neg\varphi, Z, i) & = & \neg\mathtt{t}(\varphi, Z, i) \\
\mathtt{t}(a \cdot x, Z, i) & = & a \cdot \#_{z_{Z(x)-1}}(\top) - a \cdot \#_{z_{Z(x)}}(\top) & \mathtt{t}(a, Z, i) & = & a \\
\mathtt{t}(\exists x.\varphi, Z, i) & = & \mathtt{F}\, z_i.\mathtt{t}(\varphi, Z[x \mapsto i], i+1) & & & \\
\mathtt{t}(\exists^{=x} y.\varphi, Z, i) & = & \mathtt{F}\,\mathtt{G}\left(\mathtt{t}(x, Z, i) = \#_{z_{i-1}}(z_i.\mathtt{t}(\varphi, Z[y \mapsto i], i+1))\right) & & &
\end{array}$$

for $x, y \in V_\Phi$, $a, i \in \mathbb{N}$ and $\odot \in \{\wedge, \leq, +\}$. Then, we obtain $\hat{\Phi} = z_0.\mathtt{t}(\Phi, \mathbf{1}, 1)$, 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, CLTL) 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 ρ by a fixed number of naturals. We use a predicate $Conf$ that allows for accessing the i-th state on ρ given its encoding and a predicate Run characterising all (encodings of) runs in $\mathtt{Runs}(\mathcal{K})$. Such predicates were shown to be definable by Presburger arithmetic formulae of polynomial size and used to encode MC(FKS,CTL*) [12, 10]. We adopt this idea for MC(FKS, CCTL*) and PH. Let $\mathcal{K} = (S, s_I, E, \lambda)$ 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 $enc : \mathbb{N}^N \to S^\omega$ and predicates $Conf(r_1, \ldots, r_N, i, s)$ and $Run(r_1, \ldots, r_N)$ such that for all valuations $\eta : V_N \to \mathbb{N}$ we have **1.** $\eta \models_{PH} Run(r_1, \ldots, r_N) \Leftrightarrow enc(\eta(r_1), \ldots, \eta(r_N)) \in \mathtt{Runs}(\mathcal{K})$ and **2.** if $\eta \models_{PH} Run(r_1, \ldots, r_N)$ then $\eta \models_{PH} Conf(r_1, \ldots, r_N, i, s) \Leftrightarrow enc(\eta(r_1), \ldots, \eta(r_N))(\eta(i)) = \eta(s)$. Both predicates are definable by PH formulae over variables $V \supseteq V_N$ of polynomial size in $|\mathcal{K}|$.*

Now, let Φ be a CCTL* formula to be verified on $\mathcal{K}$. Without loss of generality we assume that all comparisons $\varphi_\leq \in \mathtt{sub}(\Phi)$ of the form $\tau_1 \leq \tau_2$ have the shape $\varphi_\leq =$

$\sum_{\ell=1}^{k} a_\ell \cdot \#_{x_\ell}(\varphi_\ell) + b \leq \sum_{\ell=k+1}^{m} a_\ell \cdot \#_{x_\ell}(\varphi_\ell) + c$ for some $k, m, b, c \in \mathbb{N}$, coefficients $a_\ell \in \mathbb{N}$ and subformulae φ_ℓ. 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 $\exists z_1.\exists^{=z_1} i'.x \leq i' \leq i \wedge \hat{\varphi}$ where i holds the current position and $\hat{\varphi}$ expresses that φ 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 φ to be translated, the names of N variables encoding the current run and the name of the variable holding the current position. Let

$$
\begin{aligned}
\mathtt{chk}(p, r_1, \ldots, r_N, i) &= \exists s.\mathit{Conf}(r_1, \ldots, r_N, i, s) \wedge \textstyle\bigvee_{a \mid p \in \lambda(a)} s = a \\
\mathtt{chk}(\varphi \wedge \psi, r_1, \ldots, r_N, i) &= \mathtt{chk}(\varphi, r_1, \ldots, r_N, i) \wedge \mathtt{chk}(\psi, r_1, \ldots, r_N, i) \\
\mathtt{chk}(\neg\varphi, r_1, \ldots, r_N, i) &= \neg\mathtt{chk}(\varphi, r_1, \ldots, r_N, i) \\
\mathtt{chk}(\mathtt{X}\,\varphi, r_1, \ldots, r_N, i) &= \exists i'.i' = i + 1 \wedge \mathtt{chk}(\varphi, r_1, \ldots, r_N, i') \\
\mathtt{chk}(\varphi\,\mathtt{U}\,\psi, r_1, \ldots, r_N, i) &= \exists i''.i \leq i'' \wedge \mathtt{chk}(\psi, r_1, \ldots, r_N, i'') \wedge \\
&\quad\ \forall i'.(i \leq i' \wedge i' < i'') \rightarrow \mathtt{chk}(\psi, r_1, \ldots, r_N, i') \\
\mathtt{chk}(\mathtt{E}\,\varphi, r_1, \ldots, r_N, i) &= \exists r'_1 \ldots \exists r'_N.\mathit{Run}(r'_1, \ldots, r'_N) \wedge \mathtt{chk}(\varphi, r'_1, \ldots, r'_N, i) \wedge \forall i'. \\
&\quad\ (i' \leq i) \rightarrow \exists s.\mathit{Conf}(r_1, \ldots, r_N, i', s) \wedge \mathit{Conf}(r'_1, \ldots, r'_N, i', s) \\
\mathtt{chk}(x.\varphi, r_1, \ldots, r_N, i) &= \exists x.x = i \wedge \mathtt{chk}(\varphi, r_1, \ldots, r_N, i) \\
\mathtt{chk}(\varphi_{\leq}, r_1, \ldots, r_N, i) &= \exists z_1 \ldots \exists z_m.\,(\textstyle\bigwedge_{\ell=1}^{m} \exists^{=z_\ell} i'.x_\ell \leq i' \leq i \wedge \mathtt{chk}(\varphi_\ell, r_1, \ldots, r_N, i')) \\
&\quad\ \wedge a_1 \cdot z_1 + \ldots + a_k \cdot z_k + b \leq a_{k+1} \cdot z_{k+1} + \ldots + a_m \cdot z_m + c
\end{aligned}
$$

for $\varphi_{\leq} = \sum_{\ell=1}^{k} a_\ell \cdot \#_{x_\ell}(\varphi_\ell) + b \leq \sum_{\ell=k+1}^{m} a_\ell \cdot \#_{x_\ell}(\varphi_\ell) + c$. Primed variables denote fresh copies of the corresponding input variables, e.g. i' becomes $(i')' = i''$ and i'' becomes i'''. Now, $\Phi \models \mathcal{K}$ if and only if $\exists r_1 \ldots \exists r_N.\exists i.\mathit{Run}(r_1, \ldots, r_N) \wedge i = 0 \wedge \mathtt{chk}(\Phi, r_1, \ldots, r_N, i)$ 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

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. doi: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 complexitiy of logical theories. *Theor. Comput. Sci.*, 11:71–77, 1980. doi: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. doi: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. doi: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. doi: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. doi:10.1145/1592761.1592781.

9 Normann Decker, Peter Habermehl, Martin Leucker, Arnaud Sangnier, and Daniel Thoma. Model-checking counting temporal logics on flat structures. *CoRR*, abs/1706.08608, 2017. URL: `http://arxiv.org/abs/1706.08608`.

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. doi: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. doi:10.1016/j.ic.2015.03.007.

12 Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Model-checking CTL* over flat Presburger counter systems. *Journal of Applied Non-Classical Logics*, 20(4):313–344, 2010. doi:10.3166/jancl.20.313-344.

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. doi: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. doi:10.1145/567067.567081.

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 462–477. 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.