Weak Kripke Structures and LTL

Lars Kuhtz[1] and Bernd Finkbeiner[2]

[1] Microsoft Redmond
[2] Saarland University

Abstract. We revisit the complexity of the model checking problem for formulas of linear-time temporal logic (LTL). We show that the classic PSPACE-hardness result is actually limited to a subclass of the Kripke frames, which is characterized by a simple structural condition: the model checking problem is only PSPACE-hard if there exists a strongly connected component with two distinct cycles. If no such component exists, the problem is in coNP. If, additionally, the model checking problem can be decomposed into a polynomial number of finite path checking problems, for example if the frame is a tree or a directed graph with constant depth, or the frame has an SCC graph of constant depth, then the complexity reduces further to NC.

1 Introduction

Model checking, the automatic verification of a finite-state structure against a formula of a temporal logic, is one of the key advances in systems theory over the past decades. Many artifacts in modern computers, including hardware, protocols, and operating system components, can be described as finite-state structures. Model checking algorithms have also found numerous applications beyond computer-aided verification, including XML data bases, planning, and computational biology.

The complexity of the model checking problem has been the subject of intensive investigation. The fundamental result, by Sistla and Clarke in 1985, is that the model checking problem for linear-time temporal logic (LTL) is PSPACE-complete [21]. A first refinement of this result, due to Lichtenstein and Pnueli, separates the complexity in the length of the formula from the complexity in the size of the Kripke structure. It turns out that the problem really is PSPACE-complete only in the size of formula and linear in the size of the Kripke structure. Much of the subsequent work has therefore focused on detailing the complexity with respect to different classes of formulas [21,14,8,4,3].

However, the linear complexity in the size of the Kripke structure does not mean that the impact of the Kripke structure should be neglected. Consider, for example, Kripke structures that consist of a single state. The model checking problem for such Kripke structures corresponds to the problem of evaluating Boolean formulas, which is NC^1-complete [6]. The PSPACE-hardness result relies on the possibility to encode the computations of a Turing machine as paths in the Kripke structure. What happens if the frame of the Kripke structure does

not allow for such an encoding? Examples of such restricted frames occur in many different domains.

- **Paths:** The problem of checking whether a given finite path satisfies an LTL formula plays a key role in monitoring and runtime verification (cf. [9]), where individual paths are checked either online, during the execution of the system, or offline, for example based on an error report. Similarly, path checking occurs in testing [1] and in several static verification techniques, notably in Monte-Carlo-based probabilistic verification, where large numbers of randomly generated sample paths are analyzed [22].
- **Trees:** Model checking trees occurs in assertion checking, querying, debugging, and searching in all kinds of parse trees, class hierarchies, thread or process trees, abstract data types, file systems, and XML documents and XML databases (cf. [5]).
- **Restricted loops:** Control and scheduling programs often enter, after some initialization, an infinitely executed static loop [2]. The frame of the Kripke structure thus is a lasso path. If the system proceeds in stages, where each stage consists of the nondeterministically many iterations of a static loop, then the Kripke frame has several loops, but no nested loops.

In this paper, we abstract from the possible LTL formulas and the possible labelings of the Kripke structure, and instead focus entirely on the structure of the frame. A key role in our analysis is played by the *path checking* problem [19], i.e., the model checking problem where the frame is restricted to a single path. We recently showed that, for LTL formulas, path checking is in NC [13]. We generalize this result to Kripke structures for which the model checking problem can be deterministically reduced to a polynomial number of parallel path checking problems: Kripke structures that are trees or directed graphs with constant depth, or that have an SCC graph of constant depth, all have model checking problems in NC.

Our main result is that the separation between Kripke structures with a PSPACE hard model checking problem and Kripke structures with a model checking problem in coNP is a strict dichotomy. The borderline between PSPACE and coNP can in fact be characterized by a simple structural condition: We call a Kripke structure *weak* if there are no two distinct cycles within a single strongly connected component. The model checking problem for weak Kripke structures can be decomposed into the path checking problems for an exponential number of relevant paths. If a Kripke structure is weak then the model checking problem is therefore in coNP; otherwise, the model checking problem is PSPACE hard.

Related Work. The subject of this paper, the model checking problem for restricted classes of Kripke frames, is an extension of the path checking problem, which was introduced as an open problem by Demri and Schnoebelen in [8]. In addition to our recent work [13] on LTL path checking, Markey and Schnoebelen investigate the path checking problem for various extensions and restrictions of LTL [19] and also show that the complexity of the (finite) path checking problem

for the μ-calculus is P-hard [20]. Markey and Raskin [18] study the complexity of the model checking problem for restricted sets of paths for extensions of LTL to continuous time.

Another area of investigation that is closely related is the study of the state explosion problem. The state explosion problem occurs in compositional model checking when the Kripke structure is represented as some kind of product Kripke structure. Demri, Laroussinie, and Schnoebelen study the complexity of model checking parameterized by the number of factors of the product Kripke structure [7].

In classical modal logic, systems are defined via frame conditions. Starting with Ladners seminal results in [16] there is a line of research about the complexity of problems for modal logics systems under certain frame conditions (cf. [11,10] for recent results and overview on past work).

2 Computation Paths and Kripke Structures

Linear-time temporal logic reasons about linearly ordered sequences of states, which we call computation paths. In the following we define the semantic framework for the logic: propositions, states, computation paths, Kripke frames, and Kripke structures. Kripke structures symbolically represent sets of computations paths in a compact way. Kripke frames represent the topology of transition relation of a Kripke structure.

Given a set of *atomic propositions* AP. A *state* $s \in 2^{\mathrm{AP}}$ is an evaluation of the atomic propositions in AP. For $p \in \mathrm{AP}$ we say that p holds in s if and only if $p \in s$. We write $s(p)$ to denote the value of p in s with $s(p) = 1$, if p holds in s, and $s(p) = 0$ otherwise. An ordered sequence $\rho = \rho_0, \rho_1, \ldots$ of states is called a *computation path* over AP. The *length of* ρ is denoted by $|\rho|$. If ρ is infinite, we set $|\rho| = \infty$; $i < \infty$ for all $i \in \mathbb{N}$. For a computation path ρ and $0 \leq i < |\rho|$ we write ρ_i for the state at position i; $\rho_{i,j}$, where $0 \leq i \leq j < |\rho|$, denotes the computation path $\rho_i, \rho_{i+1}, \ldots, \rho_j$ of length $|\rho_{i,j}| = j - i + 1$; $\rho_{i,\ldots}$ denotes the suffix of ρ at position i. The empty sequence is denoted ϵ with $|\epsilon| = 0$. We denote concatenation of computation paths as a product and write either $\sigma\rho$ or $\sigma \cdot \rho$ for the concatenation of the computation paths σ and ρ, where σ is finite. For a finite computation path σ we set $\sigma^n = \prod_0^{n-1} \sigma$, $\sigma^* = \left\{ \prod_0^{n-1} \sigma \mid n \in \mathbb{N} \right\}$, and $\sigma^\omega = \prod_0^\infty \sigma$. In the context of automata we will treat computation paths over AP as *words* over the *alphabet* $\Sigma = 2^{\mathrm{AP}}$, where a *letter* is a state. The set of all finite words over Σ is denoted as Σ^*. The set of infinite words is denoted as Σ^ω. A *language* over Σ is a subset of $\Sigma^* \cup \Sigma^\omega$. A computation path (or a word) $\rho = \rho_0, \rho_1, \ldots, \rho_n, n \in \mathbb{N}$ canonically defines a (graph theoretic) path. In the following we view ρ as a path whenever adequate.

A *Kripke structure* $\mathcal{K}$ is a four-tuple $\langle K, k_i, R, \lambda \rangle$ where K is a set of vertices, $k_i \subseteq K$ are the initial vertices, $R \subseteq K \times K$ is a transition relation, and $\lambda \colon K \to 2^{\mathrm{AP}}$ is a labeling function on the vertices of $\mathcal{K}$. The directed graph $\langle K, R \rangle$ is called the *frame* of the Kripke structure $\mathcal{K}$. By abuse of notation we sometimes

identify a state $k \in K$ with its labeling $\lambda(k)$, where we assume that $\lambda^{-1}(k)$ is determined from the context.

The *SCC graph* of a Kripke frame $\mathcal{F}$ is the graph that is obtained by collapsing every strongly connected component of $\mathcal{F}$ into a single vertex.

The *language of a Kripke structure* $\mathcal{K} = \langle K, k_i, R, \lambda \rangle$, denoted as $\text{lang}(\mathcal{K})$, is the set of (finite and infinite) computation paths

$$\{\lambda(s_0), \lambda(s_1), \cdots \mid s_0 \in k_i, \langle s_i, s_{i+1} \rangle \in R\}$$

for $i \in \mathbb{N}$ with $0 \leq i$ or $0 \leq i < n$ for some $n \in \mathbb{N}$.

A Kripke structure (respectively a Kripke frame) is called *weak*, if there are no two distinct cycles within a single strongly connected component; in other words: all cycles are pairwise disjoint. This implies that the cycles of a weak Kripke strucutre (Kripke frame, respectively) are partially ordered with respect to reachability.

3 Linear-Time Temporal Logic – LTL

We consider linear-time temporal logic (LTL) with the usual finite-path semantics, which includes a weak and a strong version of the Next operator [17]. Let AP be a set of atomic propositions. The *LTL formulas* are defined inductively as follows: every atomic proposition $p \in \text{AP}$ is a formula. If ϕ and ψ are formulas, then so are

$$\neg\phi, \quad \phi \wedge \psi, \quad \phi \vee \psi, \quad \mathrm{X}^{\exists}\, \phi, \quad \mathrm{X}^{\forall}\, \phi, \quad \phi\ \mathrm{U}\ \psi, \quad \text{and} \quad \phi\, \mathrm{R}\, \psi\ .$$

Let $p \in \text{AP}$. We use *true* to abbreviate $p \vee \neg p$ and *false* as an abbreviation for $p \wedge \neg p$. For a formula ϕ we write $\mathrm{G}\,\phi$ to abbreviate $false\, \mathrm{R}\, \phi$ and $\mathrm{F}\,\phi$ as an abbreviation for $true\, \mathrm{U}\, \phi$. The *size of a formula* ϕ is denoted by $|\phi|$.

LTL formulas are evaluated over computation paths over the set of states 2^{AP}. Given an LTL formula ϕ, a nonempty computation path ρ *satisfies* ϕ at position i $(0 \leq i < |\rho|)$, denoted by $(\rho, i) \models \phi$, if one of the following holds:

- $\phi \in \text{AP}$ and $\phi \in \rho_i$,
- $\phi = \neg\psi$ and $(\rho, i) \not\models \psi$,
- $\phi = \phi_l \wedge \phi_r$ and $(\rho, i) \models \phi_l$ and $(\rho, i) \models \phi_r$,
- $\phi = \phi_l \vee \phi_r$ and $(\rho, i) \models \phi_l$ or $(\rho, i) \models \phi_r$,
- $\phi = \mathrm{X}^{\exists}\, \psi$ and $i + 1 < |\rho|$ and $(\rho, i + 1) \models \psi$,
- $\phi = \mathrm{X}^{\forall} \psi$ and $i + 1 = |\rho|$ or $(\rho, i + 1) \models \psi$,
- $\phi = \phi_l\, \mathrm{U}\, \phi_r$ and $\exists i \leq j < |\rho|$ s.t. $(\rho, j) \models \phi_r$ and $\forall i \leq k < j$, $(\rho, k) \models \phi_l$, or
- $\phi = \phi_l\, \mathrm{R}\, \phi_r$ and $\forall i \leq j < |\rho|.(\rho, j) \models \phi_r$ or $\exists i \leq k < j$ s.t. $(\rho, k) \models \phi_l$.

For $|\rho| = \infty$ and for any $i \in \mathbb{N}$ it holds that $(\rho, i) \models \mathrm{X}^{\exists}\, \psi$ if and only if $(\rho, i) \models \mathrm{X}^{\forall} \psi$. An LTL formula ϕ is *satisfied* by a nonempty path ρ (denoted by $\rho \models \phi$) if and only if $(\rho, 0) \models \phi$. A Kripke structure K satisfies the formula ϕ (denoted by $K \models \phi$) if and only if for all $\rho \in \text{lang}(K)$ it holds that $\rho \models \phi$. Two

LTL formulas ϕ and ψ are said to be equivalent ($\phi \equiv \psi$) if and only if for all paths ρ it holds that $\rho \models \phi$ if and only $\rho \models \psi$.

An LTL formula ϕ is said to be in *positive normal form* if in ϕ only atomic propositions appear in the scope of the symbol $\neg$. The following *dualities* ensure that each LTL formula ϕ can be rewritten into an equivalent formula ϕ' in positive normal form with $|\phi'| = O(|\phi|)$.

$$\begin{aligned}
\neg\neg\phi &\equiv \phi \ ; \\
\neg \mathrm{X}^{\forall}\phi &\equiv \mathrm{X}^{\exists}\neg\phi \ ; \\
\neg(\phi_l \wedge \phi_r) &\equiv (\neg\phi_l) \vee (\neg\phi_r) \ ; \\
\neg(\phi_l \,\mathrm{U}\, \phi_r) &\equiv (\neg\phi_l)\,\mathrm{R}(\neg\phi_r) \ .
\end{aligned}$$

Given a class of Kripke structures $\mathscr{K}$. The *model checking problem* of LTL over $\mathscr{K}$ (MC[LTL, $\mathscr{K}$]) is defined by

$$\mathrm{MC}[\mathrm{LTL}, \mathscr{K}] = \left\{ K \stackrel{?}{\models} \phi \mid K \in \mathscr{K}, \phi \in LTL \right\} \ .$$

In the following we often use classes of Kripke structures that are defined through a class of Kripke frames. E.g. *path* denotes the class of all Kripke structures with a frame that is a (finite) path.

Model Checking a Path. Automata-based techniques have proved very successful in establishing upper bounds on complexity of LTL model checking problems. However, there seems to be a barrier on proving sub-polynomial bounds via automata constructions. In [13], we gave a construction that uses monotone Boolean circuits in place of the usual automata-based constructions in order to prove that the LTL path checking problem is in NC.

Theorem 1 (Kuhtz and Finkbeiner (2009)).
MC[LTL, path] is in $\mathsf{AC}^1(\mathsf{logDCFL})$. □

Generalized Stutter Equivalence. It is a well known property of the star free regular languages, which is precisely captured by LTL, that those languages can only count up to a threshold but are unable to do modulo counting. In [15] Kučera and Strejček introduce the notion of *generalized stutter equivalence* which reflects the disability to count of LTL on a syntactic level.

Definition 1 (Generalized Stutter Equivalence). *Given a computation path ρ, a subsequence $\rho_{i,j}$ of ρ is* (m,n)-redundant *if $\rho_{(j+1),(j+1)+m\cdot(j-i)-m+1+n}$ is a prefix of $\rho^{\omega}_{i,j}$.*

We say that two computation paths ρ and σ are (m,n)-stutter equivalent *if ρ is obtained from σ by removing non-overlapping (m,n)-redundant subsequences, or vice versa.*

Kučera and Strejček prove the following generalized stuttering theorem for LTL. Intuitively the theorem states that an LTL formula can not distinguish between words that differ through repeated occurrence (stuttering) of a sub-word beyond a certain threshold that depends on the nesting depth of the different temporal operators.

Theorem 2 (Kučera and Strejček (2005)). *Given an LTL formula ϕ with maximal nesting depth of* U *and* R *modalities of m and with maximal nesting depth of $\mathrm{X}^{\exists}$ and $\mathrm{X}^{\forall}$ modalities of n, the set of $\{\rho \mid \rho \models \phi\}$ is closed under (m,n)-stutter equivalence.* □

Using this theorem, we will establish upper bounds for model checking of weak Kripke structures.

Model Checking LTL with a single variable. In their seminal paper [21] Sistla and Clarke prove general LTL model checking to be PSPACE-complete. The result is obtained via a reduction of the satisfiability problem to the co-model checking problem. In the same paper, LTL satisfiability is proved PSPACE-complete by encoding the computations of a PSPACE Turing machine into an LTL formula. In [8], Demri and Schnoebelen strengthen this result by showing that satisfiability is hard even for the fragment of LTL with a single atomic proposition.

Theorem 3 (Demri and Schnoebelen (2002)). *Satisfiability of LTL with a single atomic proposition is* PSPACE*-complete.* □

Restricting our attention only to formulas with a single variable, we can represent the class of all possible models in a Kripke structure with just two vertices and two different labels. When proving PSPACE-hardness of model checking of non-weak Kripke frames, we will use the fact that this simple structure can be embedded into any non-weak Kripke frame.

4 LTL Model Checking Problems in NC

We start with some theorems that are corollaries from Theorem 1. In general, any class of Kripke structures for which the model checking problem can be deterministically reduced to a polynomial number of parallel path checking problems can be model checked in NC. In particular, trees can be decomposed into a linear number of paths:

Theorem 4. *MC[LTL, tree] is in* $\mathsf{AC}^1(\mathsf{logDCFL})$. □

DAGs of constantly bounded depth can be unfolded into trees with only polynomial blowup:

Theorem 5. *MC[LTL, DAG of depth $O(1)$] is in* $\mathsf{AC}^1(\mathsf{logDCFL})$. □

Markey and Schnoebelen present in [19] a reduction from the problem of checking ultimately periodic paths to the finite path checking problem. We provide a more general reduction. We start with an observation about weak Kripke structures:

Lemma 1. *Let $\mathcal{K}$ be a weak Kripke structure. Any (finite or infinite) computation path $\rho \in \mathrm{lang}(\mathcal{K})$ is of the form $\left(\prod_{i=0}^{n-1} u_i \cdot v_i^{\alpha_i}\right)$ with*

– $n \leq |\mathcal{K}|$,
– $\alpha_i \in \mathbb{N}$ *for* $i < n-1$ *and* $\alpha_{n-1} \in \mathbb{N} \cup \{\infty\}$, *and*
– u_i, v_i *are finite paths in* $\mathcal{K}$

for $0 \leq i < n$.

Proof. The statement of the lemma follows from the fact that for a weak Kripke structure all cycles are disjoint and the SCC graph is a directed acyclic graph. □

The lemma implies that we can represent a computation path ρ in a weak Kripke structure $\mathcal{K}$ as a path R in the SCC graph of $\mathcal{K}$ together with the coefficient α_i for each cycle v_i that occurs in ρ. We denote this representation of ρ by $\mathsf{R}_{\alpha_0,\ldots,\alpha_{n-1}}$.

Lemma 2. *Given an LTL formula* ϕ *and a weak Kripke structure* $\mathcal{K}$. *If there is a computation path* $\rho = \mathsf{R}_{\alpha_0,\ldots,\alpha_{n-1}} \in \mathrm{lang}(\mathcal{K})$ *with* $\rho \models \phi$ *then there is a computation path* $\rho' = \mathsf{R}_{\beta_0,\ldots,\beta_{n-1}}$ *with* $\beta_i \leq |\phi| + 1$ *such that* $\rho' \models \phi$. *In particular it holds that* $|\rho'| = O(|\phi| \cdot |\mathcal{K}|)$.

Proof. Represent the computation path ρ according to Lemma 1 and apply the generalized stuttering theorem (Theorem 2) from Kučera and Strejček. □

Lemma 2 subsumes the reduction from [19]. By reducing the problem of checking an ultimately periodic path to the finite path checking problem we get the following theorem:

Theorem 6. *MC[LTL, ultimately periodic path] is in* $\mathsf{AC}^1(\mathsf{logDCFL})$.

Proof. The computation path ρ can be represented as R_∞. Due to Lemma 2 it is sufficient to enumerate and check all computation paths R_α for $\alpha \leq |\phi| + 1$. By Theorem 1 each check can be done in $\mathsf{AC}^1(\mathsf{logDCFL})$. Since all checks are independent we can do them all in parallel.

Remark: A more careful interpretation of Theorem 2 would reveal that a single check for $\alpha = |\phi| + 1$ is actually sufficient. □

We can use Lemma 2 to generalize the result to Kripke structures with SCC graphs of constantly bounded depth.

Theorem 7. *MC[LTL, weak, SCC graph of depth* $O(1)$*] is in* $\mathsf{AC}^1(\mathsf{logDCFL})$.

Proof. Unfold the Kripke structure into a tree of polynomial size and constant depth. Each computation path in the unfolded structure can be represented as $\mathsf{R}_{\alpha_0,\ldots\alpha_n}$ where $n \in \mathbb{N}$ is a constant. Due to Theorem 2, it sufficient to enumerate (in L) and check all computation paths for $\alpha_i \leq |\phi|+1$ $(0 \leq i \leq n)$. In total there is a polynomial number of computation paths to be checked. Using Theorem 1, all checks can be done in parallel in $\mathsf{AC}^1(\mathsf{logDCFL})$. □

We can generalize the previous result a bit further: let G be the SCC graph of a Kripke structure $\mathcal{K}$. For a vertex $v \in V(G)$ let

$$\zeta(v) = \alpha(v) + \sum_{\langle w,v\rangle \in E(\mathcal{K})} \beta(v) \cdot \zeta(w)$$

where

$$\alpha(v) = \begin{cases} 1 & \text{if } v \text{ is initial in } \mathcal{K} \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

$$\beta(v) = \begin{cases} 2 & \text{if } v \text{ represents a cycle of } \mathcal{K} \text{ and} \\ 1 & \text{otherwise, and} \end{cases}$$

the empty sum equals zero. Intuitively, the function $\zeta(v)$ counts the number of paths that lead from an initial state to v, where each cycle occurs either zero or one times in a path. Let $\zeta(\mathcal{K}) = \max_{v \in V(G)} \zeta(v)$.

Theorem 8. *For any class $\mathscr{C}$ of weak Kripke structures such that ζ is polynomial in the size of the structure it holds that MC[LTL, $\mathscr{C}$] is in* $\mathsf{AC}^1(\mathsf{logDCFL})$. □

5 coNP-Complete LTL Model Checking Problems

In favor of a more concise presentation, we exclusively consider LTL over infinite paths throughout the remainder of this paper.

Theorem 9. *LTL model checking of weak Kripke structures is* coNP*-complete.*

Proof. We prove the upper bound by guessing a possibly infinite path and then using Lemma 2 to reduce the problem to the finite path checking problem. Let $\mathcal{K}$ be a weak Kripke structure. In order to decide if $\mathcal{K} \not\models \phi$ guess a path R in the SCC graph of $\mathcal{K}$ such that there is a path $\rho = \mathsf{R}_{\alpha_0,\ldots,\alpha_{n-1}} \in \mathrm{lang}(\mathcal{K})$ with $\rho \models \neg\phi$. Use Lemma 2 to reduce this to checking $\rho' \models \neg\phi$ for a finite path ρ' of polynomial length. Do this check by use of Theorem 1 in $\mathsf{AC}^1(\mathsf{logDCFL}) \subseteq \mathsf{P}$. Hence the model checking problem for weak Kripke structures is in coNP.

The proof of the lower bound reduces the satisfiability problem of propositional logic to the co-model checking problem of weak Kripke structures. The reduction is very similar to the reduction used by [21] to show that the co-model checking problem for the fragment of LTL that has F as the only modality is NP-hard.

Given a propositional logic formula f over the set of variables $\{v_0, \ldots, v_n\}$, $n \in \mathbb{N}$. We obtain the LTL formula ϕ from f by substituting for all $0 \leq i \leq n$ each occurrence of the variable v_i by the LTL formula $\mathrm{X}^{\exists 2i+1}\, p$, where $p \in \mathrm{AP}$. The formula f is satisfiable if and only if $\neg\phi$ does not hold on the Kripke structure $\mathcal{K}$ shown in Figure 1. □

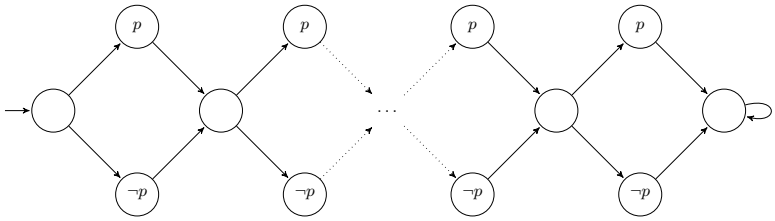

Fig. 1. Kripke structure used to reduce propositional SAT to LTL model checking

The above proof actually provides a slightly stronger result than stated in Theorem 9:

Corollary 1. *The problem of model checking LTL on planar acyclic graphs is* coNP*-hard.* □

The hardness result from Theorem 9 can be sharpened to more restricted classes with a coNP lower bound on the model checking problem. In order to prepare the proof of the next theorem we show here the construction for Kripke structures with an SCC graph that is a path. In fact self loops, i.e. state-stuttering, is sufficient. The idea is similar to the lower bound from the previous theorem, but the diamond shaped substructures are replaced by self loops. For a propositional formula f with variables $v_0, \ldots, v_{n-1}$ we build a Kripke structure $\mathcal{K}$ that is a sequence of n self loops as shown in Figure 2 where each vertex is labeled with a unique state (represented as a propositional formula) p_i. The LTL formula ϕ is obtained by substituting in f each variable v_i with the LTL formula $\mathrm{F}(p_i \wedge \mathrm{X}^{\exists} p_i)$. It is easy to check that f is satisfiable if and only if $\mathcal{K} \not\models \neg\phi$.

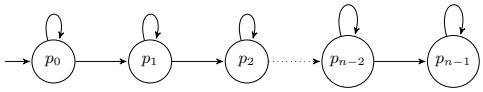

Fig. 2. Kripke structure used to reduce SAT to LTL model checking of Kripke structures for which the SCC graph is a path.

The next theorem is a refined (though more technical) version of Theorem 9. Recall that the function ζ from Section 4 counts the number of paths in a weak Kripke structure where each cycle occurs at most once.

Theorem 10. *For any class $\mathscr{C}$ of weak Kripke structures for which ζ is exponential in the size of the structure it holds that MC[LTL, $\mathscr{C}$] is complete for* coNP*.*

The upper bound remains the same as in Theorem 9; the lower bound is refined through a stronger constraint on the classes of structures.

Proof. The proof for the lower bound combines the proof for the lower bound for planar DAGs and the lower bound for paths Kripke structures with an SCC graph that is a path. Again, we reduce SAT to the co-model checking problem on $\mathscr{C}$. Let f a propositional formula with variables $v_0, \ldots, v_{n-1}$. There is a Kripke structure $\mathcal{K}$ in $\mathscr{C}$ such that the SCC graph of $\mathcal{K}$ contains sequence of vertices $v_0, \ldots, v_{n-1}$, where v_i is reachable from v_{i-1}. Due to the exponential growth of ζ there are $O(n)$ many i with $\zeta(v_i) = O(2 \cdot v_{i-1})$. Moreover, we know that either there are two distinct paths from v_{i-1} to v_i with two distinguishing vertices v_i^0 and v_i^1, or v_i represents a cycle. In the former case we label v_i^0 with a unique label p_i (represented by a propositional formula) and substitute v_i in f with the LTL formula $\mathrm{F}\, p_i$. In the latter case we label v_i with a unique label p_i (represented as a propositional formula) and substitute any occurrence of v_i with the LTL formula $\mathrm{F}(p_i \wedge \mathrm{X}^{\exists}(\mathit{true}\, \mathrm{U}\, p_i))$. We call the resulting LTL formula ϕ. Again, it is easy to check that $\mathcal{K} \not\models \neg\phi$ if and only if f is satisfiable. □

The classification of Kripke structures between NC and coNP-hardness is not a complete dichotomy. There is a gap concerning the structures with $n^{O(1)} \ll \zeta(k) \ll O(2^n)$. This is illustrated via the following theorem.

Theorem 11. *For any class $\mathscr{C}$ of weak Kripke structures with $\zeta = O(n^{\log^{O(1)} n})$, where n is the size of a Kripke structure, it holds that MC[LTL, $\mathscr{C}$] is in* polyL.

Proof. We can enumerate all computation paths of polynomial length that are relevant according to Lemma 2 in polyL. Each computation path can be checked in $\mathsf{AC}^1(\mathsf{logDCFL}) \subseteq \mathsf{polyL}$. □

6 PSPACE-Complete LTL Model Checking Problems

We now investigate the borderline between the frames whose model checking problems are in coNP and those whose model checking problems are PSPACE-hard. It turns out that LTL model checking is PSPACE-complete for *any* non-weak Kripke frame. In contrast to the previous PSPACE hardness results, we get a lower bound that does not depend on the asymptotic behavior of a class of Kripke structures, but holds for each non-weak Kripke frame. Moreover, together with Theorem 9 we obtain a dichotomic classification.

Theorem 12. *The LTL model checking problem is* PSPACE*-complete for any non-weak Kripke frame.*

Proof. Given a non-weak Kripke frame $\mathcal{F}$. We reduce the validity problem for LTL to the co-model checking problem on a structure $\mathcal{K}$ with frame $\mathcal{F}$.

We start by choosing an adequate labeling for $\mathcal{K}$. Let $s, t, u \in 2^{\mathrm{AP}}$ be pairwise disjoint states represented as Boolean formulas. Because $\mathcal{F}$ is non-weak we know that there are two distinct cycles that share a common vertex x. Label x with

state s. There is a vertex y that is present in only one of the two cycles. Label y with state t. Label all remaining vertices of $\mathcal{K}$ with u. Figure 3 provides a schematic view of $\mathcal{K}$.

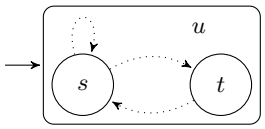

Fig. 3. Non-weak Kripke structure with the labeling used in the proof of Theorem 12

Given some formula ζ with only a single variable. By Theorem 3, deciding validity for an LTL formula with only a single atomic proposition is PSPACE-hard. ζ is valid if and only if $\neg\zeta$ is not satisfiable. To decide if $\phi = \neg\zeta$ is satisfiable is the co-model checking problem on a universal Kripke structure. We will reduce the latter for ϕ to the co-model checking problem on the Kripke structure $\mathcal{K}$ with the labeling given above for a formula ϕ^* that can be constructed from ϕ in L. Since PSPACE is closed under complement, the model checking problem for $\mathcal{K}$ is thus PSPACE hard.

We assume that the unique atomic proposition that occurs in ϕ is p. A Kripke structure with two states, namely p and $\neg p$, that represents the universal language $\{p, \neg p\}^\omega$ is shown in Figure 4.

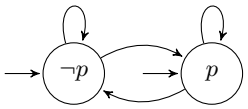

Fig. 4. The Kripke structure that represents the universal language $\{p, \neg p\}^\omega$.

In the following we call a suffix of a computation path an a-suffix if the first state of the suffix is a. We call a cycle in a Kripke structure an a-cycle if it starts in an a-state. We identify the cycle with the corresponding state sequence.

The construction of ϕ^* is as follows: First, transform ϕ into positive normal form in L. Next, we define inductively a formula ϕ'. For the cases that ϕ is either an atomic proposition or a negated atomic proposition let

- $p' = s \wedge \mathrm{X}^\exists\,(u\,\mathrm{U}\,s)$ and
- $(\neg p') = s \wedge \mathrm{X}^\exists\,\big(u\,\mathrm{U}\,\big(t \wedge \mathrm{X}^\exists\,(u\,\mathrm{U}\,s)\big)\big)$.

The idea is that the formula p' holds exclusively on an s-cycle that does not include the t state, whereas $(\neg p)'$ holds on any s-cycle that visits the t state.

This way, each s-cycle encodes a state of the original Kripke structure. The formula will translate each single step in the original Kripke structure into an s-cycle in $\mathcal{K}$. The remaining cases for ϕ' are defined inductively as follows:

- $\psi' \wedge \chi'$ for $\phi = \psi \wedge \chi$,
- $\psi' \vee \chi'$ for $\phi = \psi \vee \chi$,
- $s \wedge \mathrm{X}^{\exists}(\neg s \,\mathrm{U}\, \psi')$ for $\phi = \mathrm{X}^{\exists}\psi$,
- $s \wedge ((s \rightarrow \psi') \,\mathrm{U}\, \chi')$ for $\phi = \psi \,\mathrm{U}\, \chi$, and
- $s \wedge (\psi' \,\mathrm{R}\, (s \rightarrow \chi'))$ for $\phi = \psi \,\mathrm{R}\, \chi$.

For any formula ϕ we want that a computation path that models ϕ' is "meaningful" in the sense that it can be mapped to a computation path having only p and $\neg p$ states. Therefore we defined $'$ such that a formula ϕ' holds on a computation path ρ only if ρ starts in the s state.

Finally, we set $\phi^* = (\mathrm{G}\,\mathrm{F}\, s) \wedge (\neg s \,\mathrm{U}\, \phi')$. By requiring that any computation path ρ with $\rho \models \phi^*$ returns to s infinitely often, we prevent that ρ "gets stalled" in some "meaningless" loop. The formula allows the computation to reach s initially, and then it forces the remaining computation path to satisfy ϕ' which encodes on $\mathcal{K}$ the original meaning of ϕ.

We claim that for each computation path ρ it holds that $\rho \models \phi$ if and only if there is a computation path $\rho^* \in \mathrm{lang}(\mathcal{K})$ such that $\rho^* \models \phi^*$. Let c be a s-cycle in $\mathcal{K}$ that does not visit t and let d an s-cycle in $\mathcal{K}$ that visits t. For proving the "only if" part of the claim assume that $\rho \models \phi$. Let ρ' be defined as follows:

$$\rho' = \begin{cases} c \cdot \rho'_{1,\ldots} & \text{if } \rho_0(p), \text{ and} \\ d \cdot \rho'_{1,\ldots} & \text{otherwise.} \end{cases}$$

It holds that $\rho' \in \mathrm{lang}(\mathcal{K})$, every suffix of ρ' contains an s-state, and ρ' starts with s. Further $'$ induces a surjective mapping from the suffixes of ρ to s-suffixes of ρ'. This mapping is monotonic with respect to the order of start position of the suffixes.

Let e be the (possibly empty) sequence of states on a shortest path in $\mathcal{K}$ that leads from an initial state to s. Let $\rho^* = e \cdot \rho'$. Every suffix of ρ^* contains an s state and therefore it holds that $\rho^* \models \mathrm{G}\,\mathrm{F}\, s$. Since all states in e contradict s from the definition of ρ^* it follows directly that $\rho^* \models (\neg s) \,\mathrm{U}\, \phi'$ if $\rho' \models \phi'$. We prove this by induction over ϕ:

- For $\phi = p$ it holds that if ρ starts with p then ρ' starts with c followed by an s state. Hence $\rho' \models s \wedge \mathrm{X}^{\exists}(u \,\mathrm{U}\, s)$.
- For $\phi = \neg p$ it holds that if ρ starts with $\neg p$ then ρ' starts with d followed by an s state. Hence $\rho' \models s \wedge \mathrm{X}^{\exists}\big(u \,\mathrm{U}\, \big(t \wedge \mathrm{X}^{\exists}(u \,\mathrm{U}\, s)\big)\big)$.
- For $\phi \in \{\psi \wedge \chi, \psi \vee \chi\}$ the claim follows directly from the induction hypothesis and the semantics of LTL.
- For $\phi = \mathrm{X}^{\exists}\psi$ it holds that $\rho_{1,\ldots} \models \psi$. By induction hypothesis $(\rho_{1,\ldots})' \models \psi'$ and by definition of ρ' it holds that $\rho' \models s \wedge \mathrm{X}^{\exists}(\neg s \,\mathrm{U}\, \psi')$.
- For $\phi = \psi \,\mathrm{U}\, \chi$ there is a position i such that $\rho_{i,\ldots} \models \chi$ and $\rho_{j,\ldots} \models \phi$ for all $j < i$. By induction hypothesis there is an l such that $\rho'_{l,\ldots} \models \chi'$. Since $'$ is

surjective and monotonic on the s-suffixes of ρ' from the induction hypothesis it follows that $\rho'_{j,\dots} \models \psi'$ for all s-suffixes with $j < l$. Further recall that ψ' holds only on computation paths that start with s. Hence, for all non-s-suffix $\rho'_{j,\dots}$ with $j < l$ the formula $(s \to \psi')$ holds trivially. Thus, for all $j < l$ it holds that $\rho'_{j,\dots} \models (s \to \psi')$ and we get $\rho' \models s \wedge (s \to \psi') \,\mathrm{U}\, \chi'$.
- The case for $\phi = \psi \,\mathrm{R}\, \chi$ is analogous to the previous case.

We now prove the "if" part of the claim. Given a computation path in $\sigma \in \mathrm{lang}(\mathcal{K})$ such that $\sigma \models \phi^*$. There is a position i_0 such that for all $j < i_0$ it holds that $\sigma_{j,\dots} \models \neg s$ and $\sigma_{i_0,\dots} \models \phi'$. Let $\sigma^0 = \sigma_{i_0,\dots}$. We show that $\sigma^0 \models \phi'$ implies that there is a computation path σ' with $\sigma' \models \phi$. We know that σ^0 contains only s, t, and u states. Moreover, we know from the definition of ϕ^* that any suffix of σ^0 contains an s state. The computation path σ' is defined as follows:

$$\sigma' = \begin{cases} p \cdot \sigma'_{s_0,\dots} & \text{if } \sigma^0_{0,s_0} \text{ contains no } t \text{ state, and} \\ \neg p \cdot \sigma'_{s_0,\dots} & \text{otherwise,} \end{cases}$$

where s_0 is the position of the first s-state in $\sigma_{1,\dots}$. Note that $'$ induces a monotonic and surjective mapping from the s-suffixes of σ^0 to the suffixes of σ. We show by induction over ϕ that $\sigma' \models \phi$:

- For $\phi = p$ we have $\sigma^0 \models s \wedge \mathrm{X}^{\exists}(u \,\mathrm{U}\, s)$. This implies that σ^0_{0,s_0} does not contain any t state and therefore $\sigma'_0(p)$.
- For $\phi = \neg p$ we have $\sigma^0 \models s \wedge \mathrm{X}^{\exists}\big(u \,\mathrm{U}\, \big(t \wedge \mathrm{X}^{\exists}(u \,\mathrm{U}\, s)\big)\big)$. Therefore σ^0_{0,s_0} contains a t state and hence $\sigma'_0(\neg p)$.
- For $\phi \in \{\psi \wedge \chi, \psi \vee \chi\}$ the claim follows directly from the induction hypothesis and the semantics of LTL.
- For $\phi = \mathrm{X}^{\exists}\psi$ we have $\sigma^0 \models s \wedge \mathrm{X}^{\exists}(\neg s \,\mathrm{U}\, \psi')$. This implies that $\sigma^0_{s_0,\dots} \models \psi'$. From the induction hypothesis it follows that $\sigma'_{1,\dots} \models \psi$ and thus $\sigma' \models \mathrm{X}^{\exists}\psi$.
- For $\phi = \psi \,\mathrm{U}\, \chi$ we have that $\sigma^0 \models s \wedge ((s \to \psi') \,\mathrm{U}\, \chi')$. Therefore there is an $i \in \mathbb{N}$ such that $\sigma^0_{i,\dots} \models \chi'$ and for all $j < i$ it holds that $\sigma^0_{j,\dots} \models (s \to \psi')$. By induction hypothesis there is an $l \in \mathbb{N}$ such that $\sigma'_{l,\dots} \models \chi$. For all s-suffixes $\sigma^0_{j,\dots}$ with $j < i$ it holds that $\sigma^0 j, \dots \models \psi'$. Recall that $'$ induces a monotonic and surjective function from the s-suffixes of σ^0 to the suffixes of σ'.
  Together with the induction hypothesis we deduce that for all $j < l$ it holds that $\sigma'_{j,\dots} \models \psi$. We conclude that $\sigma' \models \psi \,\mathrm{U}\, \chi$.
- The case for $\phi = \psi \,\mathrm{R}\, \chi$ is analogous to the previous case. Note, however, that there are infinitely many s-states in σ^0. □

7 Conclusions

We have developed a classification of Kripke structures with respect to the complexity of the model checking problem for LTL. We showed that the model checking problem for a Kripke frame is PSPACE-complete if and only if the frame is not weak. The problem is coNP-complete for the class of all weak Kripke structures. The problem is in NC for any class of Kripke structures for which the

model checking problem can be reduced to a polynomial number of path checking problems. Examples of such classes include finite paths, ultimately periodic path, finite trees, directed graphs of constant depths, and classes of Kripke structures with an SCC graph of constant depth.

Open questions and future work. There are several open questions that deserve further study. Albeit small, there is a gap between $\mathsf{AC}^1(\mathsf{logDCFL})$ and the best known lower bound, NC^1 for LTL path checking. There is some hope to further reduce the upper bound towards NC^1, the currently known lower bound, because the construction in [13] relies on the algorithms for evaluating monotone planar Boolean circuits with all constant gates on the outer face. The circuits that appear in the construction actually exhibit much more structure. Another way to improve the upper bounds of the path and tree checking algorithms would be to prove a better upper bound for the problem of evaluating one-input-face monotone planar Boolean circuits.

Another area that requires more attention is the model checking problem on restricted frames for branching-time temporal logic. In the first author's thesis [12], it is shown that the model checking problem for CTL on finite trees is in NC (more precisely, in $\mathsf{AC}^2(\mathsf{logDCFL})$). The gap between NC as an upper bound and L as a lower bound for tree structures and P for general structures is comparably small. Still, beyond finite trees we do know very little about other classes for which the model checking problem for CTL is in NC. What are the properties of Kripke structures that allow for efficiently parallel model checking for CTL? What is the complexity of tree checking for CTL+past and CTL with a sibling axis? What is the complexity of CTL* tree checking?

Finally, an important question concerns the translation of the improved complexity bounds for restricted sets of frames into efficient practical model checking algorithms. The proofs in this paper and the underlying proofs for the path checking problem [13] are based on a variety of different computational models: Boolean circuits, space-restricted Turing machines, time-restricted Turing machines, Turing machines with push-down store, and parallel random access memory machines (PRAM). Are there practical parallel implementations for parallel path and tree checking?

Complexity classes that are characterized by efficient parallel algorithms and complexity classes that are characterized by space-efficient algorithms are tightly coupled through simulation theorems. Considering that in modern hardware architectures cache-efficiency and I/O-efficiency matter more than the simple number of computation steps, the following question seems even more important than the previous one: Can we derive practical space-efficient implementations from parallel path and tree checking algorithms? With the fast growing number of available cores in modern computing devices, on the one hand, and the tight resource restrictions on mobile devices, on the other hand, good trade-offs between cache-efficiency, I/O-efficiency, and CPU-usage become increasingly important.

References

1. C. Artho, H. Barringer, A. Goldberg, K. Havelund, S. Khurshid, M. Lowry, C. Pasareanu, G. Rosu, K. Sen, W. Visser, and R. Washington. Combining test case generation and runtime verification. *Theoretical Computer Science*, 336(2-3):209 – 234, 2005.
2. T. P. Baker. The cyclic executive model and ada. *The Journal of Real-Time Systems*, 1:120–129, 1989.
3. M. Bauland, M. Mundhenk, Th. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The tractability of model-checking for ltl: The good, the bad, and the ugly fragments. *Electr. Notes Theor. Comput. Sci.*, 231:277–292, 2009.
4. M. Bauland, Th. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The complexity of generalized satisfiability for linear temporal logic. *Logical Methods in Computer Science*, 5(1), 2009.
5. M. Benedikt, L. Libkin, and F. Neven. Logical definability and query languages over ranked and unranked trees. *ACM Trans. Comput. Log.*, 8(2), 2007.
6. S.R. Buss. The boolean formula value problem is in ALOGTIME. In *STOC*, pages 123–131, New York, NY, USA, 1987. ACM.
7. S. Demri, F. Laroussinie, and Ph. Schnoebelen. A parametric analysis of the state-explosion problem in model checking. *J. Comput. Syst. Sci.*, 72(4):547–575, 2006.
8. S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Inf. Comput.*, 174(1):84–103, 2002.
9. K. Havelund and G. Roşu. Monitoring programs using rewriting. In *ASE*, pages 135 – 143. IEEE Computer Society, 2001.
10. E. Hemaspaandra. The complexity of poor man's logic. *J. Log. Comput.*, 11(4):609–622, 2001.
11. E. Hemaspaandra and H. Schnoor. On the complexity of elementary modal logics. In Susanne Albers and Pascal Weil, editors, *STACS*, volume 1 of *LIPIcs*, pages 349–360. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008.
12. L. Kuhtz. *Model Checking Finite Paths and Trees.* PhD thesis, Universität des Saarlandes, 2010.
13. L. Kuhtz and B. Finkbeiner. LTL path checking is efficiently parallelizable. In *ICALP'09*, volume 5556 of *LNCS*, pages 235 – 246. Springer, 2009.
14. O. Kupferman and M.Y. Vardi. Relating linear and branching model checking. In *PROCOMET*, pages 304–326, New York, June 1998. Chapman & Hall.
15. A. Kučera and J. Strejček. The stuttering principle revisited. *Acta Inf.*, 41(7-8):415–434, 2005.
16. R.E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.
17. O. Lichtenstein, A. Pnueli, and L.D. Zuck. The glory of the past. In *Proceedings of the Conference on Logic of Programs*, pages 196–218, London, UK, 1985. Springer.
18. N. Markey and J.-F. Raskin. Model checking restricted sets of timed paths. *Theoretical Computer Science*, 358(2-3):273 – 292, 2006. CONCUR 2004.
19. N. Markey and Ph. Schnoebelen. Model checking a path (preliminary report). In *CONCUR*, volume 2761 of *LNCS*, pages 251–265. Springer, 2003.
20. N. Markey and Ph. Schnoebelen. Mu-calculus path checking. *Inf. Process. Lett.*, 97(6):225–230, 2006.
21. A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.
22. H.L.S Younes and R.G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, volume 2404 of *LNCS*. Springer, 2002.