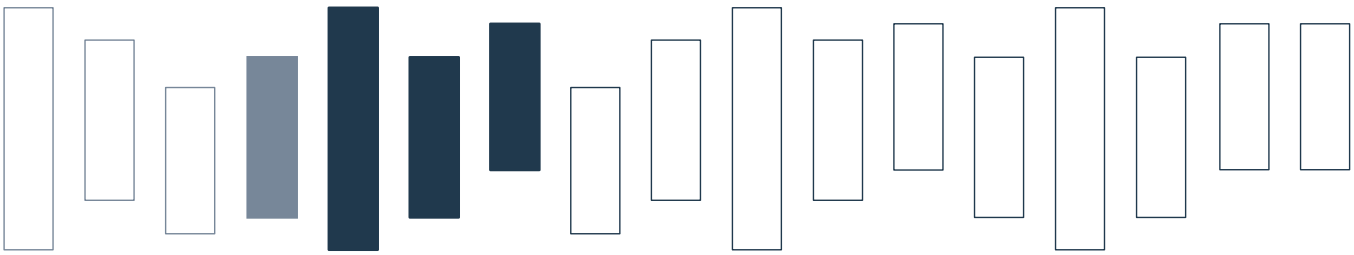
Technical Report AC-TR-15-002

September 2015

Parameterized Complexity Results for Symbolic Model Checking of Temporal Logics

Ronald de Haan and Stefan Szeider



Parameterized Complexity Results for Symbolic Model Checking of Temporal Logics

Ronald de Haan and Stefan Szeider

Algorithms and Complexity Group
TU Wien, Vienna, Austria
dehaan@ac.tuwien.ac.at, stefan@szeider.net

Abstract

The model checking problem for temporal logics is an important problem with applications in key areas of computer science. Indispensable for the state-of-the-art in solving this problem in large-scale settings is the technique of bounded model checking. We investigate the theoretical possibilities of this technique using parameterized complexity. In particular, we provide a complete parameterized complexity classification for the model checking problem for symbolically represented Kripke structures for various fragments of the temporal logics LTL, CTL and CTL*. We argue that a known result from the literature for a restricted fragment of LTL can be seen as an fpt-time encoding into SAT, and show that such encodings are not possible for any of the other fragments of the temporal logics that we consider. As a by-product of our investigation, we develop a parameterized complexity class that can be seen as a parameterized variant of the Polynomial Hierarchy.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic – Temporal Logic, D.2.4 Software/Program Verification – Model Checking

Keywords and phrases temporal logic, bounded model checking, fpt-reductions to SAT

1 Introduction

The model checking problem for temporal logics is an important problem with applications in key areas of computer science and engineering, among others in the verification of software and hardware systems (see, e.g., [2, 9, 11]). The problem consists of checking whether an abstract model of an automated system, given in the form of a labelled relational structure (a *Kripke structure*), satisfies a formal specification of desired behavior given as a temporal logic formula. Underlining the importance of temporal logic model checking in computer science, the ACM 2007 Turing Award was given for foundational research on the topic [8]. Indispensable for the state-of-the-art in solving this problem in industrial-size settings is the algorithmic technique of symbolic model checking using propositional satisfiability (SAT) solvers (called *bounded model checking*), where the SAT solvers are employed to find counterexamples [3, 4, 5, 10]. This approach works well in cases where the Kripke structure is large, but the temporal logic specification is small. Therefore, one could expect the framework of parameterized complexity theory to be well-suited for analyzing the method of bounded model checking from a theoretical point of view. Unfortunately, parameterized complexity has not been able to help identify those settings in which this technique can be applied. First of all, existing parameterized complexity analyses [12, 16, 17, 24, 25] have only considered the problem for settings where the Kripke structure is spelled-out explicitly (or consists of a small number of explicitly spelled-out components), which is highly impractical in many cases occurring in practice. In fact, the so-called state explosion problem is a major obstacle for developing practically useful techniques [7]. For this reason, the Kripke structures are often described symbolically, for instance using propositional formulas, which allows for


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

exponentially more succinct encodings of the structures. Moreover, whereas parameterized complexity analysis is traditionally focused on fixed-parameter tractability for positive results, the technique of bounded model checking revolves around encoding the problem as an instance of SAT. Therefore, the parameterized complexity analyses are bound to concentrate on very restrictive cases in order to obtain fixed-parameter tractability, unaware of some of the more liberal settings where bounded model checking can be applied.

In this paper, we investigate the possibilities and boundaries of the technique of bounded model checking from a theoretical point of view, using the framework of parameterized complexity. We do so by performing a parameterized complexity analysis of the model checking problem for fragments of various temporal logics, where we take the size of the temporal logic formula as parameter and where the Kripke structures are represented symbolically (and can thus be of size exponential in the size of their description) but enjoy a property that restricts the size of possible counterexamples. Namely, in order to give the method of bounded model checking a fighting chance, we require the size of the largest loop-free path (the *recurrence diameter*) to be polynomially bounded.

Rather than concentrating on fixed-parameter tractability for positive results, in our analysis we focus on the frontier of fpt-reducibility to SAT. Fpt-reductions to SAT have recently been identified as a way to increase the impact and usefulness of parameterized complexity for problems from various domains that are beyond NP (see, e.g., [18]). The problem of temporal logic model checking is PSPACE-complete in general, and therefore identifying settings where small parameter values can be exploited to achieve a reduction to SAT (or its co-problem UNSAT), can be considered as a positive result. Such positive results are captured by membership in para-NP or para-co-NP. On the other hand, to give evidence that no fpt-reduction to SAT exists, in certain cases we can show hardness for the known parameterized complexity class para-PSPACE. In other cases, to show hardness we need a new parameterized complexity class, PH(LEVEL), which is of independent interest and can be seen as a parameterized variant of the Polynomial Hierarchy (PH).

Contributions We consider the model checking problem for three of the most widespread temporal logics, LTL, CTL and CTL* (a precise definition of these logics can be found in Section 3). Moreover, for each of these logics, we consider also the fragments where several temporal operators (namely, U and/or X) are disallowed. We give a complete complexity classification of the problem of checking whether a given Kripke structure, specified symbolically using a propositional formula, satisfies a given temporal logic specification, parameterized by the size of the temporal logic formula. Firstly, we show that the problem is para-PSPACE-complete for all logics and all fragments if the recurrence diameter of the structure is unrestricted. Next, we identify a known result from the literature on bounded model checking as a para-co-NP-membership result for the logic LTL where both operators U and X are disallowed, and we extend this to a completeness result. Then, we show that the problem is para-PSPACE-complete for LTL (and so also for its generalization CTL*) when you allow at least one of the operators U and X. The main technical obstacle that we had to overcome to show para-PSPACE-hardness for these cases was to encode satisfiability of quantified Boolean formulas without having explicit quantification available in the logic. Finally, we show that in all remaining cases (all fragments of CTL, and the fragment of CTL* without the operators U and X) the problem is complete for PH(LEVEL). The prime difficulty for these completeness results was to identify the parameterized complexity class PH(LEVEL), and to characterize it in various ways. In short, we show that the only case (given these fragments of temporal logics) where the technique of bounded model checking

can be applied is the fragment of LTL without the operators U and X. An overview of all the completeness results that we develop in this paper can be found in Table 1.

logic $\mathcal{L}$	LTL	CTL	CTL*
$\mathcal{L}$	para-PSPACE-complete	PH(LEVEL)-complete	para-PSPACE-complete
$\mathcal{L}\backslash$X	para-PSPACE-complete	PH(LEVEL)-complete	para-PSPACE-complete
$\mathcal{L}\backslash$U	para-PSPACE-complete	PH(LEVEL)-complete	para-PSPACE-complete
$\mathcal{L}\backslash$U,X	para-co-NP-complete	PH(LEVEL)-complete	PH(LEVEL)-complete

Table 1 Parameterized complexity results for the problem SYMBOLIC-MC$^\star[\mathcal{L}]$ for the different (fragments of) logics $\mathcal{L}$. For the problem SYMBOLIC-MC$[\mathcal{L}]$, all cases are para-PSPACE-complete.

In addition, as mentioned, we introduce the parameterized complexity class PH(LEVEL), which is based on the satisfiability problem of quantified Boolean formulas parameterized by the number of quantifier alternations. We show that this class can also be characterized by means of an analogous parameterized version of first-order logic model checking, as well as by alternating Turing machines that alternate between existential and universal configurations only a small number of times (depending only on the parameter).

Related work Computational complexity analysis has been a central aspect in the study of temporal logic model checking problems, and naturally these problems have been analyzed from a parameterized complexity point of view. For instance, LTL model checking parameterized by the size of the logic formula features as an example for fixed-parameter tractability in the textbook by Flum and Grohe [16]. For the temporal logic CTL, parameterized complexity has also been used to study the problems of model checking and satisfiability [12, 17, 24, 25]. Fixed-parameter tractable reductions to SAT have also recently been considered for other problem domains, and from a theoretical point of view [14, 18, 19, 20, 21, 27]. As the SAT encoding techniques used for bounded LTL model checking result in an incomplete solving method in general, limits on the cases in which this particular encoding can be used as a complete solving method have been studied [6, 10, 22].

Outline We begin with reviewing relevant notions from (parameterized) complexity theory in Section 2. Then, in Section 3, we introduce the different temporal logics that we consider, we review known complexity results for their model checking problems, and we interpret a known result for bounded model checking for the fragment of LTL without U and X operators using notions from parameterized complexity. Next, in Section 4, we introduce the new parameterized complexity class PH(LEVEL). In Section 5 we provide the parameterized complexity results that indicate that bounded model checking cannot be applied for all other fragments of temporal logics that we consider. Finally, we conclude in Section 6. Due to space restrictions, we refer to the appendix for full proofs of statements marked with a star.

2 Preliminaries

Polynomial Space The class PSPACE consists of all decision problems that can be solved by an algorithm that uses a polynomial amount of space (memory). Alternatively, one can characterize the class PSPACE as all decision problems for which there exists a polynomial-time reduction to the problem QSAT, that is defined using quantified Boolean formulas as follows. A quantified Boolean formula (in prenex form) is a formula of the form $Q_1x_1Q_2x_2\ldots Q_nx_n.\psi$, where all x_i are propositional variables, each Q_i is either an existential or a universal quantifier, and ψ is a (quantifier-free) propositional formula over the variables $x_1,\ldots,x_n$ (called

the *matrix*). Truth for such formulas is defined in the usual way. The problem QSAT consists of deciding whether a given quantified Boolean formula is true.

Alternatively, the semantics of quantified Boolean formulas can be defined using QBF models [28]. Let $\varphi = Q_1 x_1 \dots Q_n x_n.\psi$ be a quantified Boolean formula. A *QBF model* for φ is a tree of (partial) truth assignments where (**1.**) each truth assignment assigns values to the variables $x_1, \dots, x_i$ for some $1 \leq i \leq n$, (**2.**) the root is the empty assignment, and for all assignments α in the tree, assigning truth values to the variables $x_1, \dots, x_i$ for some $1 \leq i \leq n$, the following conditions hold: (**3.**) if $i < n$, every child of α agrees with α on the variables $x_1, \dots, x_i$, and assigns a truth value to x_{i+1} (and to no other variables); (**4.**) if $i = n$, then α satisfies ψ, and α has no children; (**5.**) if $i < n$ and $Q_i = \exists$, then α has one child α' that assigns some truth value to x_{i+1}; and (**6.**) if $i < n$ and $Q_i = \forall$, then α has two children α_1 and α_2 that assign different truth values to x_{i+1}. It is straightforward to show that a quantified Boolean formula φ is true if and only if there exists a QBF model for φ. Note that this definition of QBF models is a special case of the original definition [28].

Fixed-parameter tractable reductions to SAT We assume the reader to be familiar with basic notions from parameterized complexity theory, such as fixed-parameter tractability and fpt-reductions. For more details, we refer to textbooks on the topic [13, 16, 26]. We briefly highlight some notions that are useful for investigating fpt-reductions to SAT. The propositional satisfiability problem (SAT), consists of deciding whether a given propositional formula in CNF is satisfiable. When we consider SAT as a parameterized problem, we consider the trivial (constant) parameterization. The parameterized complexity class para-NP consists of all problems that can be (many-to-one) fpt-reduced to SAT. More generally, for each classical complexity class K, the parameterized class para-K is defined as the class of all parameterized problems $L \subseteq \Sigma^* \times \mathbb{N}$, for some finite alphabet Σ, for which there exist an alphabet Π, a computable function $f : \mathbb{N} \to \Pi^*$, and a problem $P \subseteq \Sigma^* \times \Pi^*$ such that $P \in K$ and for all instances $(x, k) \in \Sigma^* \times \mathbb{N}$ of L we have that $(x, k) \in L$ if and only if $(x, f(k)) \in P$ [15]. Intuitively, the class para-K consists of all problems that are in K after a precomputation that only involves the parameter. For all classical complexity classes K, K' it holds that $K \subseteq K'$ if and only if para-$K \subseteq$ para-K'. Therefore, in particular, problems that are para-PSPACE-hard are not in para-NP, unless NP = PSPACE.

3 Model Checking for Temporal Logics

In this section, we review the definition of the temporal logics that we consider in this paper, and we introduce the problem of model checking for symbolically represented Kripke structures. In addition, we argue why the polynomial bound on the recurrence diameter of the Kripke structures is necessary to obtain an fpt-reduction to SAT. Finally, we identify a para-co-NP-membership result from the literature on bounded model checking.

3.1 Temporal Logics

We begin with defining the semantical structures for all temporal logics. In the remainder of the paper, we let P be a finite set of propositions. A *Kripke structure* is a tuple $\mathcal{M} = (S, R, V, s_0)$, where S is a finite set of *states*, $R \subseteq S \times S$ is a binary relation on the set of states called the *transition relation*, $V : S \to 2^P$ is a *valuation function* that assigns each state to a set of propositions, and where $s_0 \in S$ is the *initial state*. An example of a Kripke structure is given in Figure 1. We say that an finite sequence $s_1 \dots s_\ell$ of states $s_i \in S$ is

a *finite path* in $\mathcal{M}$ if $(s_i, s_{i+1}) \in R$ for each $1 \leq i < \ell$. Similarly, we say that an infinite sequence $s_1 s_2 s_3 \ldots$ of states $s_i \in S$ is an *infinite path* in $\mathcal{M}$ if $(s_i, s_{i+1}) \in R$ for each $i \geq 1$.

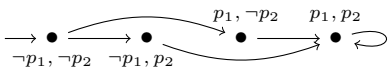


Figure 1 An example Kripke structure $\mathcal{M}_1$ for the set $P = \{p_1, p_2\}$ of propositions.

Now, we can define the syntax of the logic LTL. LTL formulas over the set P of atomic propositions are formed according to the following grammar (here p ranges over P), given by $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \mathrm{X}\varphi \mid \mathrm{F}\varphi \mid (\varphi \mathrm{U} \varphi)$. We consider the usual abbreviations, such as $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$. In addition, we let the abbreviation $\mathrm{G}\varphi$ denote $\neg\mathrm{F}\neg\varphi$. Intuitively, the formula $\mathrm{X}\varphi$ expresses that φ is true in the next (time) step, $\mathrm{F}\varphi$ expresses that φ becomes true at some point in the future, $\mathrm{G}\varphi$ expresses that φ is true at all times from now on, and $\varphi_1 \mathrm{U} \varphi_2$ expresses that φ_2 becomes true at some point in time, and until then the formula φ_1 is true at all points. Formally, the semantics of LTL formulas are defined for Kripke structures, using the notion of (infinite) paths. Let $\mathcal{M} = (S, R, V, s_0)$ be a Kripke structure, and $\overline{s}_1 = s_1 s_2 s_3 \ldots$ be a path in $\mathcal{M}$. Moreover, let $\overline{s}_i = s_i s_{i+1} s_{i+2} \ldots$ for each $i \geq 2$. Truth of LTL formulas φ on paths $\overline{s}$ (denoted $\overline{s} \models \varphi$) is defined inductively as follows (for the sake of brevity, we omit the straightforward Boolean cases):

$\overline{s}_i \models \mathrm{X}\varphi$	iff	$\overline{s}_{i+1} \models \varphi$
$\overline{s}_i \models \mathrm{F}\varphi$	iff	for some $j \geq 0$, $\overline{s}_{i+j} \models \varphi$
$\overline{s}_i \models \varphi_1 \mathrm{U} \varphi_2$	iff	there is some $j \geq 0$ such that $\overline{s}_{i+j} \models \varphi_2$ and $\overline{s}_{i+j'} \models \varphi$ for each $0 \leq j' < j$

Then, we say that an LTL formula φ is true in the Kripke structure $\mathcal{M}$ (denoted $\mathcal{M} \models \varphi$) if for all infinite paths $\overline{s}$ starting in s_0 it holds that $\overline{s} \models \varphi$. For instance, considering the example $\mathcal{M}_1$ from Figure 1, it holds that $\mathcal{M}_1 \models \mathrm{FG}p_2$.

Next, we can define the syntax of the logic CTL*, which consists of two different types of formulas: state formulas and path formulas. When we refer to CTL* formulas without specifying the type, we refer to state formulas. Given the set P of atomic propositions, the syntax of CTL* formulas is defined by the following grammar (here Φ denotes CTL* *state formulas*, φ denotes CTL* *path formulas*, and p ranges over P), given by $\Phi ::= p \mid \neg\Phi \mid (\Phi \wedge \Phi) \mid \exists\varphi$, and $\varphi ::= \Phi \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \mathrm{X}\varphi \mid \mathrm{F}\varphi \mid (\varphi \mathrm{U} \varphi)$. Again, we consider the usual abbreviations, such as $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, for state formulas as well as for path formulas. Moreover, we let the abbreviation $\mathrm{G}\varphi$ denote $\neg\mathrm{F}\neg\varphi$, and we let the abbreviation $\forall\varphi$ denote $\neg\exists\neg\varphi$. Path formulas have the same intended meaning as LTL formulas. State formulas, in addition, allow explicit quantification over paths, which is not possible in LTL.

Formally, the semantics of CTL* formulas are defined inductively as follows. Let $\mathcal{M} = (S, R, V, s_0)$ be a Kripke structure, $s \in S$ be a state in $\mathcal{M}$ and $\overline{s}_1 = s_1 s_2 s_3 \ldots$ be a path in $\mathcal{M}$. Again, let $\overline{s}_i = s_i s_{i+1} s_{i+2} \ldots$ for each $i \geq 2$. The truth of CTL* state formulas Φ on states s (denoted $s \models \Phi$) is defined as follows (again, we omit the Boolean cases): $s \models \exists\varphi$ if and only if there is some path $\overline{s}$ in $\mathcal{M}$ starting in s such that $\overline{s} \models \varphi$. The truth of CTL* path formulas φ on paths $\overline{s}$ (denoted $\overline{s} \models \varphi$) is defined as follows:

$\overline{s}_i \models \mathrm{X}\varphi$	iff	$\overline{s}_{i+1} \models \varphi$
$\overline{s}_i \models \mathrm{F}\varphi$	iff	for some $j \geq 0$, $\overline{s}_{i+j} \models \varphi$
$\overline{s}_i \models \varphi_1 \mathrm{U} \varphi_2$	iff	there is some $j \geq 0$ such that $\overline{s}_{i+j} \models \varphi_2$ and $\overline{s}_{i+j'} \models \varphi$ for each $0 \leq j' < j$

Then, we say that a CTL* formula Φ is true in the Kripke structure $\mathcal{M}$ (denoted $\mathcal{M} \models \Phi$) if $s_0 \models \Phi$. For example, again taking the structure $\mathcal{M}_1$, it holds that $\mathcal{M}_1 \models \exists(\mathrm{X}p_1 \wedge \forall\mathrm{G}p_2)$.

Next, the syntax of the logic CTL is defined similarly to the syntax of CTL*. Only the grammar for path formulas φ differs, namely $\varphi ::= \mathrm{X}\Phi \mid \mathrm{F}\Phi \mid (\Phi \mathrm{U} \Phi)$. In particular, this means that every CTL state formula, (CTL formula for short) is also a CTL* formula. The semantics for CTL formulas is defined as for their CTL* counterparts.

For each of the logics $\mathcal{L} \in \{\mathrm{LTL}, \mathrm{CTL}, \mathrm{CTL}^\star\}$, we consider the fragments $\mathcal{L}\backslash\mathrm{X}$, $\mathcal{L}\backslash\mathrm{U}$ and $\mathcal{L}\backslash\mathrm{U,X}$. In the fragment $\mathcal{L}\backslash\mathrm{X}$, the X-operator is disallowed. Similarly, in the fragment $\mathcal{L}\backslash\mathrm{U}$, the U-operator is disallowed. In the fragment $\mathcal{L}\backslash\mathrm{U,X}$, neither the X-operator nor the U-operator is allowed. Note that the logic LTL$\backslash$X is also known as UTL, and the logic LTL$\backslash$U,X is also known as UTL$\backslash$X (see, e.g., [22]).

We review some known complexity results for the model checking problem of the different temporal logics. Formally, we consider the problem MC[$\mathcal{L}$], for each of the temporal logics $\mathcal{L}$, where the input is a Kripke structure $\mathcal{M}$ and an $\mathcal{L}$ formula φ, and the question is to decide whether $\mathcal{M} \models \varphi$. Note that in this problem the Kripke structure $\mathcal{M}$ is given explicitly in the input. As parameter, we will always take the size of the logic formula. It is well-known that the problems MC[LTL] and MC[CTL*] are PSPACE-complete, and that the problem MC[CTL] is polynomial-time solvable (see, e.g., [2]). It is also well-known that the problems MC[LTL] and MC[CTL*] are fixed-parameter tractable when parameterized by the size of the logic formula (see, e.g., [2, 16]).

3.2 Symbolically Represented Kripke Structures

A challenge occurring in practical verification settings is that the Kripke structures are too large to handle. Therefore, these Kripke structures are often not written down explicitly, but rather represented symbolically by encoding them succinctly using propositional formulas.

Let $P = \{p_1, \ldots, p_m\}$ be a finite set of propositional variables. A *symbolically represented Kripke structure* over P is a tuple $\mathcal{M} = (\varphi_R, \alpha_0)$, where $\varphi_R(x_1, \ldots, x_m, x'_1, \ldots, x'_m)$ is a propositional formula over the variables $x_1, \ldots, x_m, x'_1, \ldots, x'_m$, and where $\alpha_0 \in \{0,1\}^m$ is a truth assignment to the variables in P. The Kripke structure associated with $\mathcal{M}$ is (S, R, V, α_0), where $S = \{0,1\}^m$ consists of all truth assignments to P, where $(\alpha, \alpha') \in R$ if and only if $\varphi_R[\alpha, \alpha']$ is true, and where $V(\alpha) = \{\, p_i : \alpha(p_i) = 1 \,\}$.

▶ **Example 1.** Let $P = \{p_1, p_2\}$. The Kripke structure $\mathcal{M}_1$ from Figure 1 can be symbolically represented by (φ_R, α_0), where $\varphi_R(x_1, x_2, x'_1, x'_2) = [(\neg x_1 \wedge \neg x_2) \rightarrow (x'_1 \vee x'_2)] \wedge [(\neg x_1 \leftrightarrow x_2) \rightarrow (x'_1 \wedge x'_2)] \wedge [(x_1 \wedge x_2) \rightarrow (x'_1 \wedge x'_2)]$, and $\alpha_0 = (0,0)$. ⊣

We can now consider the symbolic variant Symbolic-MC[$\mathcal{L}$] of the model checking problem, for each of the temporal logics $\mathcal{L}$. Here the input is a symbolically represented Kripke structure $\mathcal{M}$, and an $\mathcal{L}$ formula φ, and the question is to decide whether $\mathcal{M} \models \varphi$. Similarly to the case of MC[$\mathcal{L}$], we will also consider Symbolic-MC[$\mathcal{L}$] as a parameterized problem, where the parameter is $|\varphi|$. Interestingly, for the logics LTL and CTL*, the complexity of the model checking problem does not change when Kripke structures are represented symbolically: Symbolic-MC[LTL] and Symbolic-MC[CTL*] are PSPACE-complete (see [23]). However, for the logic CTL, the complexity of the problem does show an increase. In fact, the problem is already PSPACE-hard for very simple formulas.

▶ **Proposition* 2.** Symbolic-MC[LTL] *is* PSPACE*-hard even when restricted to the case where* $\varphi = \mathrm{G}p$. Symbolic-MC[CTL] *and* Symbolic-MC[CTL*] *are* PSPACE*-hard even when restricted to the case where* $\varphi = \forall\mathrm{G}p$.

3.3 A Fixed-Parameter Tractable SAT-encoding for LTL\U,X

The result of Proposition 2 seems to indicate that the model checking problem for the temporal logics LTL, CTL and CTL* is highly intractable when Kripke structures are represented symbolically, even when the logic formulas are extremely simple. However, in the literature further restrictions have been identified that allow the problem to be solved by means of an encoding into SAT, which allows the use of practically very efficient SAT solving methods. In the hardness proof of Proposition 2, the Kripke structure has only a single path, which contains exponentially many different states. Intuitively, such exponential-length paths are the cause of PSPACE-hardness. To circumvent this source of hardness, and to go towards the mentioned setting where the problem can be solved by means of a SAT encoding, we need to restrict the recurrence diameter. The *recurrence diameter* $rd(\mathcal{M})$ of a Kripke structure $\mathcal{M}$ is the length of the longest simple (non-repeating) path in $\mathcal{M}$. We consider the following variant of Symbolic-MC[$\mathcal{L}$], where the recurrence diameter of the Kripke structures is restricted.

Symbolic-MC*[$\mathcal{L}$]
Input: a symbolically represented Kripke structure $\mathcal{M}$, $rd(\mathcal{M})$ in unary, and an $\mathcal{L}$ formula φ.
Parameter: $|\varphi|$.
Question: $\mathcal{M} \models \varphi$?

This restricted setting has been studied by Kroening et al. [22]. In particular, they showed that the model checking problem for LTL\U,X allows an encoding into SAT that is linear in $rd(\mathcal{M})$, even when the Kripke structure $\mathcal{M}$ is represented symbolically, and can thus be of exponential size. Using the result of Kroening et al., we obtain para-co-NP-completeness.

▶ **Proposition 3.** Symbolic-MC*[LTL\U,X] *is* para-co-NP-*complete.*

Proof (sketch). Kroening et al. [22] use the technique of bounded model checking [3, 5, 10], where SAT solvers are used to find a 'lasso-shaped' path in a Kripke structure that satisfies an LTL formula φ. They show that for LTL\U,X formulas, the largest possible length of such lasso-shaped paths that needs to be considered (also called the *completeness threshold*) is linear in $rd(\mathcal{M})$. However, the completeness threshold depends linearly on the size of a particular type of generalized Büchi automaton expressing φ, which in general is exponential in the size of φ. Therefore, this SAT encoding does not run in polynomial time, but it does run in fixed-parameter tractable time when the size of φ is the parameter. Their encoding of the problem of finding a counterexample to SAT can be seen as an encoding of the model checking problem to UNSAT. A para-co-NP-hardness proof can be found in the appendix. ◀

In the remainder of this paper, we will give parameterized complexity results that give evidence that this is the only case in this setting where such an fpt-reduction to SAT is possible. In order to do so, we first make a little digression to introduce a new parameterized complexity class, that can be seen as a parameterized variant of the Polynomial Hierarchy.

4 A Parameterized Variant of the Polynomial Hierarchy

In order to completely characterize the parameterized complexity of the problems Symbolic-MC*[$\mathcal{L}$], we need to introduce another parameterized complexity class, that is a parameterized variant of the Polynomial Hierarchy (PH). The PH consists of an infinite hierarchy of classes Σ_i^{p} and Π_i^{p} (see, e.g., [1, Chapter 5]). For each $i \leq 0$, the complexity class Σ_i^{p} consists of closure of the problem QSat$_i$ under polynomial-time reductions, where QSat$_i$ is the restriction of the problem QSat where the input formula starts with an existential quantifier and contains at most i quantifier alternations. The class Π_i^{p} is defined as co-Σ_i^{p}.

In other words, for each level of the PH, the number of quantifier alternations is bounded by a constant. If we allow an unbounded number of quantifier alternations, we get the complexity class PSPACE (see, e.g., [1, Theorem 5.10]). Parameterized complexity theory allows a middle way: neither letting the number of quantifier alternations be bounded by a constant, nor removing all bounds on the number of quantifier alternations, but bounding the number of quantifier alternations by a function of the parameter. We consider the parameterized problem QSat(level), where the input is a quantified Boolean formula $\varphi = \exists X_1 \forall X_2 \exists X_3 \dots Q_k X_k \psi$, where each X_i is a sequence of variables. The parameter is k, and the question is whether φ is true. We define the parameterized complexity class PH(level) to be the closure of QSat(level) under fpt-reductions. The class PH(level) lies above each level para-Σ_i^{p} of (the parameterized version of) the PH and below the class para-PSPACE. Also, for each $i > 1$, the following inclusions hold: para-NP $\cup$ para-co-NP $\subseteq$ para-$\Sigma_i^{\mathrm{p}} \cup$ para-$\Pi_i^{\mathrm{p}} \subseteq$ PH(level) $\subseteq$ para-PSPACE. Moreover, these inclusions are all strict, unless the PH collapses.

4.1 Alternative Characterizations

We can also characterize the class PH(level) using Alternating Turing machines (ATMs), which generalize regular (non-deterministic) Turing machines (see, e.g., [16, Appendix A.1]). We will use this characterization below to show membership in PH(level). The states of an ATM are partitioned into *existential* and *universal states*. Intuitively, if the ATM $\mathbb{M}$ is in an existential state, it accepts if there is some successor state that accepts, and if $\mathbb{M}$ is in a universal state, it accepts if all successor states accept. We say that $\mathbb{M}$ is ℓ-alternating for a problem Q, for $\ell \geq 0$, if for each input x of Q, for each run of $\mathbb{M}$ on x, and for each computation path in this run, there are at most ℓ transitions from an existential state to a universal state, or vice versa. The class PH(level) consists of all problems that can be solved by an ATM whose number of alternations is bounded by a function of the parameter.

▶ **Proposition⋆ 4.** *Let Q be a parameterized problem. Then $Q \in$ PH(level) if and only if there exist a computable function $f : \mathbb{N} \to \mathbb{N}$ and an ATM $\mathbb{M}$ such that: (1) $\mathbb{M}$ solves Q in fixed-parameter tractable time, and (2) for each slice Q_k of Q, $\mathbb{M}$ is $f(k)$-alternating.*

Next, to illustrate the robustnest of the class PH(level), we characterize this class using first-order logic model checking (which has also been used to characterize the classes of the well-known Weft-hierarchy, see, e.g. [16]). Consider the problem MC[FO], where the input consists of a relational structure $\mathcal{A}$, and a first-order formula $\varphi = \exists X_1 \forall X_2 \exists X_3 \dots Q_k X_k \psi$ in prenex form, where $Q_k = \forall$ if k is even and $Q_k = \exists$ if k is odd. The question is whether $\mathcal{A} \models \varphi$. The problem MC[FO] is PH(level)-complete when parameterized by k.

▶ **Proposition⋆ 5.** MC[FO] *parameterized by the number k of quantifier alternations in the first-order formula is* PH(level)*-complete.*

5 Completeness for PH(level) and para-PSPACE

In this section, we provide a complete parameterized complexity classification for the problem Symbolic-MC⋆[$\mathcal{L}$]. We already considered the case for $\mathcal{L} =$ LTL\U,X in Section 3.3, which was shown to be para-co-NP-complete. We give (negative) parameterized complexity results for the other cases. An overview of the results can be found in Table 1 on page 3. Firstly, we show that for the case of LTL, allowing at least one of the temporal operators U or X leads to para-PSPACE-completeness.

▶ **Theorem 6.** Symbolic-MC⋆[LTL\U] *is* para-PSPACE*-complete.*

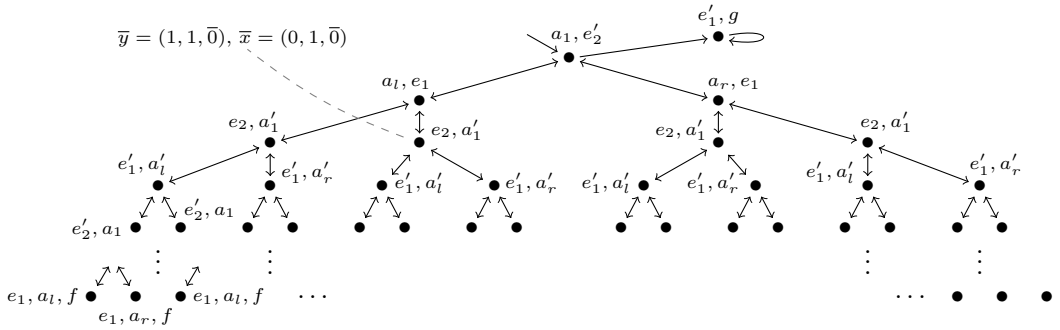


Figure 2 (The reachable part of) the structure $\mathcal{M}$ in the proof of Theorem 6.

Proof. Membership follows from the PSPACE-membership of Symbolic-MC[LTL]. We show hardness by showing that the problem is already PSPACE-hard for a constant parameter value. We do so by giving a reduction from QSat. Let $\varphi_0 = \forall x_1 . \exists x_2 \ldots Q_n x_n . \psi$ be a quantified Boolean formula. We may assume without loss of generality that $(n \bmod 4) = 1$, and thus that $Q_n = \forall$. We construct a Kripke structure $\mathcal{M}$ symbolically represented by (φ_R, α_0), whose reachability diameter is polynomial in the size of φ_0, and an LTL formula φ that does not contain the U operator, in such a way that φ_0 is true if and only if $\mathcal{M} \not\models \neg\varphi$. (So technically, we are reducing to the co-problem of Symbolic-MC*[LTL\U]. Since PSPACE is closed under complement, this suffices to show PSPACE-hardness.)

The idea is to construct a full binary tree (of exponential size), with bidirectional transitions between each parent and child, and to label the nodes of this tree in such a way that a constant-size LTL formula can be used to force paths to be a transversal of this tree corresponding to a QBF model of the formula φ_0. The idea of using LTL formulas to force paths to be transversals of exponential-size binary trees was already mentioned by Kroening et al. [22]. We construct the Kripke structure $\mathcal{M}$ as depicted in Figure 2. (For a detailed treatment of how to construct φ_R and α_0 to get this structure $\mathcal{M}$, we refer to the appendix.) It is straightforward to check that the recurrence diameter $rd(\mathcal{M})$ of $\mathcal{M}$ is bounded by $2n$ as the longest simple path in $\mathcal{M}$ is from some leaf in the tree to another leaf.

More concretely, the intuition behind the construction of $\mathcal{M}$ is as follows. Every transition from the i-th level to the $(i+1)$-th level (where the root is at the 0-th level) corresponds to assigning a truth value to the variable x_{i+1}. We use variables $\overline{x} = (x_1, \ldots, x_n)$ to keep track of the truth assignment in the current position of the tree, and variables $\overline{y} = (y_1, \ldots, y_n)$ to keep track of what level in the tree the current position is (at level i, exactly the variables $y_1, \ldots, y_i$ are set to true). At the even levels i, we use the variables a_1, a_l, a_r (and a_1', a_l', a_r') to ensure that (in a single path) both possible truth assignments to the (universally quantified) variable x_{i+1} are used. At the odd levels i, we use the variables e_1, e_2 (and e_1', e_2') to ensure that one of both possible truth assignments to the (existentially quantified) variable x_{i+1} is used. We need the copies $a_1', e_1', \ldots$ to be able to enforce the intended (downward and upward) traversal of the tree. Then, the variable f is used to signal that a leaf has been reached, and the variable g is used to signal that the path is in the sink state. For a detailed specification of how to construct the LTL formula φ in such a way that it enforces such a traversal of the structure $\mathcal{M}$, we refer to the appendix.

We can then show that φ_0 is true if and only if $\mathcal{M} \not\models \neg\varphi$. By construction of $\mathcal{M}$ and φ, all paths starting in the initial state of $\mathcal{M}$ that satisfy φ naturally correspond to a QBF model of φ_0, and all QBF models of φ_0 correspond to such a path. Assume that φ_0 is true. Then there exists a QBF model of φ_0. Then there exists a path satisfying φ, and thus $\mathcal{M} \not\models \neg\varphi$. Conversely, assume that $\mathcal{M} \not\models \neg\varphi$. Then there exists a path that satisfies φ.

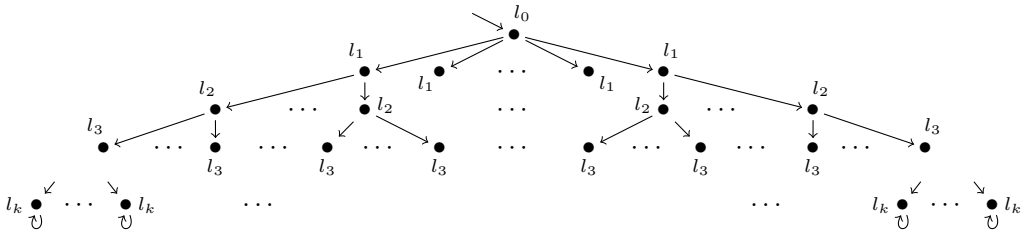


Figure 3 (The reachable part of) the structure $\mathcal{M}$ in the proof of Theorem 8.

Therefore, there exists a QBF model of φ_0, and thus φ_0 is true. ◀

▶ **Theorem 7.** Symbolic-MC*[LTL\X] *is* para-PSPACE*-complete.*

Proof. Membership follows from the PSPACE-membership of Symbolic-MC[LTL]. We show para-PSPACE-hardness by modifying the reduction in the proof of Theorem 6. The idea is to simulate the X operator using the U operator. Given an instance of QSat, we construct the Kripke structure $\mathcal{M}$ and the LTL formula φ as in the proof of Theorem 6. Then, we modify $\mathcal{M}$ and φ as follows. Firstly, we add a fresh variable x_0 to the set of propositions P, we ensure that x_0 is false in the initial state α_0, and we modify φ_R so that in each transition, the variable x_0 swaps truth values. Then, it is straightforward to see that any LTL formula of the form $\mathrm{X}\varphi'$ is equivalent to the LTL formula $(x_0 \to x_0 \mathrm{U} \varphi') \wedge (\neg x_0 \to \neg x_0 \mathrm{U} \varphi')$, on structures where x_0 shows this alternating behavior. Using this equivalence, we can recursively replace all occurrences of the X operator in the LTL formula φ. This leads to an exponential blow-up in the size of φ, but since φ is of constant size, this blow-up is permissible. ◀

Next, we show that for the case of CTL, the problem is complete for PH(level), even when both temporal operators U and X are disallowed.

▶ **Theorem 8.** Symbolic-MC*[CTL] *is* PH(level)*-complete. Moreover, hardness already holds for* Symbolic-MC*[CTL\U,X].

Proof. In order to show hardness, we give an fpt-reduction from QSat(level). Let $\varphi = \exists X_1 \forall X_2 \ldots Q_k X_k \psi$ be an instance of QSat(level). We construct a Kripke structure $\mathcal{M}$ over a set P of propositional variables represented symbolically by (φ_R, α_0), with polynomial recurrence diameter, and a CTL formula Φ such that φ is true if and only if $\mathcal{M} \models \Phi$.

The idea is to let $\mathcal{M}$ consist of a (directed) tree of exponential size, as depicted in Figure 3. The tree consists of k levels (where the root is at the 0-th level). All nodes on the i-th level are labelled with proposition l_i. Moreover, each node is associated with a truth assignment over the variables in $X = \bigcup_{1 \le i \le k} X_i$. For each node n at the i-th level (for $0 \le i < k$) with corresponding truth assignment α_n, and for each truth assignment α to the variables in X_{i+1}, there is a child node of n (at the $(i+1)$-th level) whose corresponding assignment agrees with α on the variables in X_{i+1}. Also, the truth assignment corresponding to each child of n agrees with α_n on the variables in $X_1, \ldots, X_i$. Moreover, we may assume without loss of generality that there is a propositional variable z_ψ in P that in each state is set to 1 if and only if this state sets the propositional formula ψ (over X) to true. (See Lemma 10 in the appendix for a justification of this assumption.) For a detailed treatment of how to construct φ_R and α_0 to get this structure $\mathcal{M}$, we refer to the appendix. Clearly, the longest simple path in $\mathcal{M}$ is a root-to-leaf path, which has length k.

Then, using this structure $\mathcal{M}$, we can express the quantified Boolean formula φ in CTL as follows. We define $\Phi = \exists \mathrm{F}(l_1 \wedge \forall \mathrm{F}(l_2 \wedge \exists \mathrm{F}(l_3 \wedge \cdots Q_k \mathrm{F}(l_k \wedge z_\psi) \cdots))$. By construction of Φ, we get that those subtrees of $\mathcal{M}$ that naturally correspond to witnesses for the truth of

this CTL formula Φ exactly correspond to the QBF models for φ. From this, we directly get that φ is true if and only if $\mathcal{M} \models \Phi$.

Membership in PH(LEVEL) can be proved by showing that the problem can be decided in fpt-time by an ATM that is $f(k)$-alternating. More details can be found in the appendix. ◀

Finally, we complete the parameterized complexity classification of the problem SYMBOLIC-MC* by showing membership in PH(LEVEL) for the case of CTL*\U,X.

▶ **Theorem* 9.** SYMBOLIC-MC*[CTL*\U,X] *is* PH(LEVEL)*-complete.*

Future research includes investigating the complexity for the case where the Kripke structures are specified using binary decision diagrams (BDDs) instead of (CNF) formulas, and investigating the parameterized complexity of symbolic model checking for different fragments of temporal specifications (e.g., fragments defined using automata).

6 Conclusion

An essential technique for solving the fundamental problem of temporal logic model checking is the SAT-based approach of bounded model checking. Even though its good performance in settings with large Kripke structures and small temporal logic specifications provides a good handle for a parameterized complexity analysis, the theoretical possibilities of the bounded model checking method have not been structurally investigated. We contributed to closing this gap by providing a complete parameterized complexity classification of the model checking problem for fragments of the temporal logics LTL, CTL and CTL*, where the Kripke structures are represented symbolically, and have a restricted recurrence diameter. In particular, we showed that the known case of LTL\U,X is the only case that allows an fpt-reduction to SAT, by showing completeness for the classes PH(LEVEL) and para-PSPACE for all other cases. We hope that providing a clearer theoretical picture of the settings where the powerful technique of bounded model checking can be applied helps guide engineering efforts for developing algorithms for the problem of temporal logic model checking.

References

1 Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009.

2 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

3 Armin Biere. Bounded model checking. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 457–481. IOS Press, 2009.

4 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in computers*, 58:117–148, 2003.

5 Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In Rance Cleaveland, editor, *Proceedings of TACAS 1999*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer Verlag, 1999.

6 Daniel Bundala, Joël Ouaknine, and James Worrell. On the magnitude of completeness thresholds in bounded model checking. In *Proceedings of LICS 2012*, pages 155–164. IEEE Computer Society, 2012.

7 Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Progress on the state explosion problem in model checking. In *Informatics*, volume 2000 of *Lecture Notes in Computer Science*, pages 176–194. Springer Verlag, 2001.

8 Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: algorithmic verification and debugging. *Communications of the ACM*, 52(11):74–84, 2009.

9 Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

10 Edmund M. Clarke, Daniel Kroening, Joël Ouaknine, and Ofer Strichman. Completeness and complexity of bounded model checking. In *Proceedings of VMCAI 2004*, volume 2937 of *Lecture Notes in Computer Science*, pages 85–96. Springer Verlag, 2004.

11 Edmund M. Clarke and Jeannette M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.

12 Stéphane Demri, François Laroussinie, and Philippe Schnoebelen. A parametric analysis of the state-explosion problem in model checking. *J. of Computer and System Sciences*, 72(4):547–575, 2006.

13 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer Verlag, 2013.

14 Johannes Klaus Fichte and Stefan Szeider. Backdoors to normality for disjunctive logic programs. In *Proceedings AAAI 2013*, pages 320–327. AAAI Press, 2013.

15 Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

16 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin, 2006.

17 Stefan Göller. The fixed-parameter tractability of model checking concurrent systems. In *Proceedings of CSL 2013*, volume 23 of *LIPIcs*, pages 332–347, 2013.

18 Ronald de Haan and Stefan Szeider. Compendium of parameterized problems at higher levels of the Polynomial Hierarchy. Technical Report TR14–143, *Electronic Colloquium on Computational Complexity* (ECCC), 2014.

19 Ronald de Haan and Stefan Szeider. Fixed-parameter tractable reductions to SAT. In Uwe Egly and Carsten Sinz, editors, *Proceedings of SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 85–102. Springer Verlag, 2014.

20 Ronald de Haan and Stefan Szeider. The parameterized complexity of reasoning problems beyond NP. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Proceedings of KR 2014*. AAAI Press, 2014.

21 Ronald de Haan and Stefan Szeider. Machine characterizations for parameterized complexity classes beyond para-NP. In *Proceedings of SOFSEM 2015*, volume 8939 of *Lecture Notes in Computer Science*, pages 217–229. Springer Verlag, 2015.

22 Daniel Kroening, Joël Ouaknine, Ofer Strichman, Thomas Wahl, and James Worrell. Linear completeness thresholds for bounded model checking. In *Proceedings of CAV 2011*, volume 6806 of *LNCS*, pages 557–572. Springer Verlag, 2011.

23 Orna Kupferman, Moshe Y Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *J. of the ACM*, 47(2):312–360, 2000.

24 Martin Lück, Arne Meier, and Irina Schindler. Parameterized complexity of CTL. In *Proceedings of LATA 2015*, volume 8977 of *Lecture Notes in Computer Science*, pages 549–560. Springer Verlag, 2015.

25 Arne Meier. *On the Complexity of Modal Logic Variants and their Fragments*. PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2011.

26 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford, 2006.

27 Andreas Pfandler, Stefan Rümmele, and Stefan Szeider. Backdoors to abduction. In Francesca Rossi, editor, *Proceedings of IJCAI 2013*. AAAI Press/IJCAI, 2013.

28 Horst Samulowitz, Jessica Davies, and Fahiem Bacchus. Preprocessing QBF. In *Proceedings of CP 2006*, volume 4204 of *Lecture Notes in Computer Science*, pages 514–29. Springer Verlag, 2006.

A Additional Proofs

Proof of Proposition 2. We give a polynomial-time reduction from QSAT. Let $\varphi = \exists x_1.\forall x_2.\ldots\exists x_{m-1}.\forall x_m.\psi$ be a quantified Boolean formula. We construct a symbolically represented Kripke structure $\mathcal{M}$ as follows. We consider the following set of variables:

$$Z = \{\, x_i, y_i : 1 \leq i \leq m \,\} \cup \{d, t\}.$$

The initial state α_0 is the all-zeroes assignment to Z.

We construct the formula φ_R representing the transition relation of $\mathcal{M}$. We let:

$$\varphi_R(Z, Z') = \varphi_{R,0}(Z, Z') \vee \bigwedge_{1 \leq j \leq 4} \varphi_{R,j}(Z, Z'),$$

where we define the formulas $\varphi_{R,j}(Z, Z')$ below. The intuition behind the construction of $\mathcal{M}$ is that any path will correspond to a strategy for choosing the valuation of the existentially quantified variables. We use the variables y_i to indicate which variables x_i have already been assigned a value. In fact, we ensure that in every reachable state, the variables y_i that are set to true are a consecutive sequence $y_1, \ldots, y_i$ for some $1 \leq i \leq m$. We use the following formula $\varphi_{R,1}(Z, Z')$ to do this:

$$\varphi_{R,1}(Z, Z') = \bigwedge_{1 \leq i < m} \neg y'_i \to \neg y'_{i+1}.$$

Moreover, we ensure for any transition from state α to state α', that α and α' differ on at most one variable y_i, using the following formula $\varphi_{R,2}(Z, Z')$:

$$\varphi_{R,2}(Z, Z') = \bigwedge_{1 \leq i \leq m} [\neg(y_i \leftrightarrow y'_i) \to \bigwedge_{i < i' \leq m} (y_{i'} \leftrightarrow y'_{i'})].$$

Furthermore, below we will use the following auxiliary formulas, that ensure that for any transition, the number of variables y_i that are true strictly increases (if not all y_i are set to true) or decreases (if not all y_i are set to false), respectively:

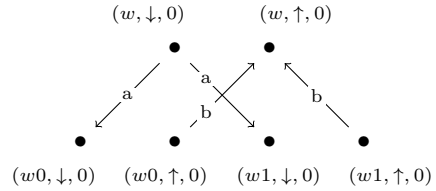
$$\begin{aligned} \varphi_{y\text{-incr}}(Z, Z') = &\quad (\neg y_1 \to y'_1) \wedge \bigwedge_{1 \leq i < m} (y_i \wedge \neg y_{i+1}) \to y'_{i+1}, \\ \varphi_{y\text{-decr}}(Z, Z') = &\quad (y_m \to \neg y'_m) \wedge \bigwedge_{1 \leq i < m} (y_i \wedge \neg y_{i+1}) \to \neg y'_i. \end{aligned}$$

Next, we ensure that in all reachable states, whenever y_i is false, x_i also has to be false. We do so using the following formula $\varphi_{R,3}(Z, Z')$:

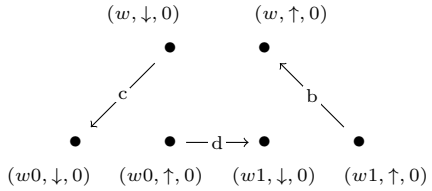
$$\varphi_{R,3}(Z, Z') = \bigwedge_{1 \leq i < m} \neg y'_i \to \neg x'_i.$$

Because of the above restrictions, we can restrict our attention to states α for which holds (1) that $y_1, \ldots, y_i$ are true, for some $1 \leq i \leq m$, and all remaining variables y_j are false, and (2) that all variables x_j for $i < j \leq m$ are false. We will denote these states by tuples (w, e, t), where $w \in \{0,1\}^i$, $e \in \{\uparrow, \downarrow\}$ and $t \in \{0, 1\}$. A tuple (w, d, t) with $|w| = i$ denotes the state α that sets $y_1, \ldots, y_i$ to true, sets $x_1, \ldots, x_i$ according to w, sets d to true if and only if $e = \uparrow$, and sets the variable t according to the value in the tuple.

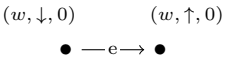
The idea behind how we continue constructing φ_R is that we piece together all possible instantiations of the gadgets in Figure 4. This results in a large directed acyclic graph containing states (w, e, t), with the property that any path that visits a state $(w, \downarrow, 0)$



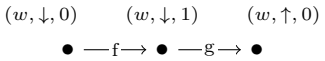
(a) Gadget for words w of odd length $< m$.



(b) Gadget for words w of even length $< m$.



(c) Gadget for words w of length m that satisfy ψ.



(d) Gadget for words w of length m that do not satisfy ψ.

Figure 4 Gadgets for the proof of Proposition 2. The labels on the relations indicate what part of $\varphi_{R,4}$ is used to encode the relations.

ultimately also visits the state $(w,\uparrow,0)$. This property allows us to use the gadgets in the following way. The gadget for a word w of odd length $i < m$ enforces that whenever a path visits the state $(w,\downarrow,0)$, it must also visit the state $(wb,\downarrow,0)$ for some $b \in \{0,1\}$. Intuitively, this simulates existential quantifiers. This property allows us to use the gadgets in the following way. The gadget for a word w of even length $i < m$ enforces that whenever a path visits the state $(w,\downarrow,0)$, it must also visit both states $(wb,\downarrow,0)$ for $b \in \{0,1\}$. Intuitively, this simulates universal quantifiers. Moreover, the gadgets for words w of length m enforce that on the way from $(w,\downarrow,0)$ to $(w,\uparrow,0)$ the state $(w,\downarrow,1)$ is visited if and only if w corresponds to a truth assignment to the variables in X that does not satisfy ψ.

We make sure that φ_R encodes exactly the transitions from $\alpha_1 = (w_1,e_1,t_1)$ to $\alpha_2 = (w_2,e_2,t_2)$ from the gadgets described above by means of the following (sub)formulas of φ_R. We distinguish seven cases. The labels on the arrows in Figure 4 indicate which case applies to which relation in the gadgets.

(a) The string w_1 is of odd length less than m and $e_1 = \downarrow$ and $t_1 = 0$. We ensure that $w_2 = w_1 b$ for some $b \in \{0,1\}$, that $e_2 = \downarrow$ and that $t_2 = 0$.

(b) It holds that $e_1 = \uparrow$, $t_1 = 0$ and the string w_1 either (i) is of even length less than m or (ii) is of odd length less than m and ends with 1. We ensure that w_2 is the string w_1 without the last symbol, that $e_2 = \uparrow$, and that $t_2 = 0$.

(c) The string w_1 is of even length less than m and $e_1 = \downarrow$ and $t_1 = 0$. We ensure that $w_2 = w_1 0$, that $e_2 = \downarrow$ and that $t_2 = 0$.

(d) It holds that $e_1 = \uparrow$, $t_1 = 0$ and the string w_1 is of odd length less than m and ends with 0. We ensure that w_2 is the string w_1 where the last symbol is replaced by a 1, that $e_2 = \downarrow$, and that $t_2 = 0$.

(e) The string w_1 is of length m, $e_1 = \downarrow$ and $t_1 = 0$. Moreover, w_1 satisfies ψ. We ensure that $w_2 = w_1$, that $e_2 = \uparrow$ and that $t_2 = 0$.

(f) The string w_1 is of length m, $e_1 = \downarrow$ and $t_1 = 0$. Moreover, w_1 does not satisfy ψ. We ensure that $w_2 = w_1$, that $e_2 = \downarrow$ and that $t_2 = 1$.

(g) The string w_1 is of length m, $e_1 = \downarrow$ and $t_1 = 1$. We ensure that $w_2 = w_1$, that $e_2 = \uparrow$ and that $t_2 = 0$.

Formally, we construct the formula $\varphi_{R,4}(Z, Z')$ as follows:

$$
\begin{aligned}
&\varphi_{R,4}(Z,Z') = \\
&\quad \bigwedge_{1\le i\le m, i \text{ odd}} [[\neg y_m \wedge d \wedge \neg t \wedge y_i \wedge \neg y_{i+1}] \to [d' \wedge \neg t' \wedge y'_{i+1} \wedge \bigwedge_{1\le i'\le i} (x_i \leftrightarrow x'_i)]] && \text{(a);}\\
\wedge\quad &\bigwedge_{1\le i\le m, i \text{ even}} [[\neg y_m \wedge \neg d \wedge \neg t \wedge y_i \wedge \neg y_{i+1}] \to [\neg d' \wedge \neg t' \wedge \neg y'_i \wedge \bigwedge_{1\le i'< i} (x_i \leftrightarrow x'_i)]] && \text{(b.i);}\\
\wedge\quad &\bigwedge_{1\le i\le m, i \text{ odd}} [[\neg y_m \wedge \neg d \wedge \neg t \wedge y_i \wedge \neg y_{i+1} \wedge x_i] \to [\neg d' \wedge \neg t' \wedge \neg y'_i \wedge \bigwedge_{1\le i'< i} (x_i \leftrightarrow x'_i)]] && \text{(b.ii);}\\
\wedge\quad &\bigwedge_{1\le i\le m, i \text{ even}} [[\neg y_m \wedge d \wedge \neg t \wedge y_i \wedge \neg y_{i+1}] \to [d' \wedge \neg t' \wedge y'_{i+1} \wedge \neg x'_{i+1} \wedge \bigwedge_{1\le i'\le i} (x_i \leftrightarrow x'_i)]] && \text{(c);}\\
\wedge\quad &\bigwedge_{1\le i\le m, i \text{ odd}} [[\neg y_m \wedge \neg d \wedge \neg t \wedge y_i \wedge \neg y_{i+1} \wedge \neg x_i] \to \\
&\qquad\qquad [d' \wedge \neg t' \wedge y'_i \wedge \neg y'_{i+1} \wedge x'_i \wedge \bigwedge_{1\le i'< i} (x_i \leftrightarrow x'_i)]] && \text{(d);}\\
\wedge\quad &[d \wedge \neg t \wedge \psi \wedge \bigwedge_{1\le i\le m} y_i] \to [\neg d' \wedge \neg t' \wedge \bigwedge_{1\le i\le m} (y'_i \wedge (x_i \leftrightarrow x'_i))] && \text{(e);}\\
\wedge\quad &[d \wedge \neg t \wedge \neg\psi \wedge \bigwedge_{1\le i\le m} y_i] \to [d' \wedge t' \wedge \bigwedge_{1\le i\le m} (y'_i \wedge (x_i \leftrightarrow x'_i))] && \text{(f);}\\
\wedge\quad &[d \wedge t \wedge \neg\psi \wedge \bigwedge_{1\le i\le m} y_i] \to [\neg d' \wedge \neg t' \wedge \bigwedge_{1\le i\le m} (y'_i \wedge (x_i \leftrightarrow x'_i))] && \text{(g);}
\end{aligned}
$$

Finally, we make sure that the state $(\epsilon, \uparrow, 0)$ has a self-loop, by means of the following formula $\varphi_{R,0}$:

$$\varphi_{R,0}(Z,Z') = \bigwedge_{z\in Z\setminus\{d\}} \neg z \wedge \bigwedge_{z'\in Z'\setminus\{d'\}} \neg z'.$$

We let the temporal logic formula whose truth is to be checked be Gt in the case of LTL, and $\forall$Gt in the case of CTL or CTL* (these formulas are equivalent).

We claim that φ has a QBF model if and only if $\mathcal{M} = (\varphi_R, \alpha_0) \models$ Gt. This holds because there is a correspondence between QBF models for φ and paths in $\mathcal{M}$ that satisfy the proposition t in each state. Each such path in $\mathcal{M}$ can be transformed into a QBF model for φ by removing the direction of the arrows, removing self-loops, merging states $(w, \downarrow, 0)$ and $(w, \uparrow, 0)$ into a single truth assignment corresponding to the word w. Because such a path does not visit any state where t is true, the leaves of the resulting tree satisfy ψ, and therefore the resulting tree is a QBF model for φ. Vice versa, each QBF model can be used similarly to obtain a path in $\mathcal{M}$ that satisfies Gt. ◀

Proof of Proposition 3 (continued). We show para-co-NP-hardness by showing that the problem Symbolic-MC*[LTL\U,X] is co-NP-hard already for formulas of constant size. We do so by a reduction from UNSAT. Let ψ be a propositional formula over the variables $x_1, \ldots, x_n$. We construct an instance of Symbolic-MC*[LTL\U,X] as follows. We consider the set $P = \{y_0, y_1, x_1, \ldots, x_n\}$ of propositional variables. We represent assignments $\alpha : P \to \{0,1\}$ as binary vectors $(c_0, c_1, b_1, \ldots, b_n)$ of length $n+2$, where c_0 indicates the value $\alpha(y_0)$, where c_1 indicates the value $\alpha(y_1)$, and where b_i indicates the value $\alpha(y_i)$ for each $1 \le i \le n$.

We then construct the symbolically represented Kripke structure $\mathcal{M}$ given by (φ_R, α_0) as follows. This structure $\mathcal{M}$ is depicted in Figure 5. We let $\alpha_0 = \overline{0}$, i.e., the all-zeroes

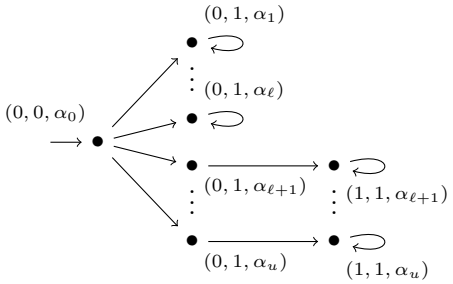


Figure 5 (The reachable part of) the structure $\mathcal{M}$ in the proof of Proposition 3. The assignments $\alpha_1, \ldots, \alpha_\ell$ are the ones not satisfying ψ, and the assignments $\alpha_{\ell+1}, \ldots, \alpha_u$ are the ones satisfying ψ.

assignment. Then we define φ_R as follows:

$$
\begin{aligned}
& \varphi_{R,4}(y_0, y_1, \overline{x}, y_0', y_1', \overline{x}') = \\
& (\neg y_1 \rightarrow (\neg y_0' \wedge y_1')) && \textit{(a)};\\
\wedge\ & (y_1 \rightarrow \bigwedge_{1 \leq i \leq n} (x_i \leftrightarrow x_i')) && \textit{(b)};\\
\wedge\ & ((y_1 \wedge \psi(x_1, \ldots, x_n)) \rightarrow (y_0' \wedge y_1')) && \textit{(c)};\\
\wedge\ & ((y_1 \wedge \neg\psi(x_1, \ldots, x_n)) \rightarrow (\neg y_0' \wedge y_1')) && \textit{(d)}.
\end{aligned}
$$

The transition relation given by φ_R allows a transition from α_0 to the state $(0, 1, \alpha)$ for any truth assignment α to the variables $x_1, \ldots, x_n$. Then, if this assignment α satisfies ψ, a transition is allowed to the looping state $(1, 1, \alpha)$. Otherwise, if α does not satisfy ψ, the only transition from state $(0, 1, \alpha)$ is to itself. Finally, we define the LTL formula to be $\varphi = \mathrm{G}\neg y_0$.

The reduction clearly runs in polynomial time. Moreover, $rd(\mathcal{M}) = 2$, and the LTL formula φ is of constant size, and contains only the temporal operator G.

We claim that $\mathcal{M} \models \varphi$ if and only if ψ is unsatisfiable. Firstly, assume that $\mathcal{M} \models \varphi$. We proceed indirectly, and assume that ψ is satisfiable. Let α be the truth assignment to the variables $x_1, \ldots, x_n$ that satisfies ψ. Then the path $\alpha_0, (0, 1, \alpha), (1, 1, \alpha), (1, 1, \alpha), \ldots$ is an infinite path in $\mathcal{M}$ that does not satisfy $\mathrm{G}\neg y_0$, which is a contradiction with our assumption that $\mathcal{M} \models \varphi$. Therefore, we can conclude that ψ is unsatisfiable.

Conversely, assume that ψ is unsatisfiable. We show that $\mathcal{M} \models \varphi$. By construction of φ_R, the only way in which a path in $\mathcal{M}$ can reach a state not satisfying $\neg y_0$ is with a transition from $(0, 1, \alpha)$ to $(1, 1, \alpha)$, for some truth assignment α to the variables $x_1, \ldots, x_n$. However, such a transition only occurs in $\mathcal{M}$ if α satisfies ψ. This is a contradiction with our assumption that ψ is unsatisfiable, and therefore $\mathcal{M} \models \varphi$. ◀

Proof of Proposition 4. First of all, we observe that the class of parameterized problems that can be solved in fpt-time by an $f(k)$-alternating ATM is closed under fpt-reductions. Next, we describe how an instance $\varphi = \exists X_1 \forall X_2 \ldots \forall X_k \psi$ of the problem QSAT(LEVEL) can be solved by a (fixed) ATM $\mathbb{M}$ that is k-alternating. Using nondeterminism in its first existential phase, $\mathbb{M}$ guesses truth values for the variables in X_1. Then, using nondeterminism in the subsequent universal phase, $\mathbb{M}$ guesses truth values for the variables in X_2. Similarly, using alternating existential and universal phases, $\mathbb{M}$ guesses truth values for all variables in all other X_i. Then, using deterministic computation, $\mathbb{M}$ verifies whether the guessed truth values satisfy ψ, and accepts the input if and only if ψ is satisfied. Clearly, φ is true if and only if $\mathbb{M}$ accepts φ.

For the other direction, let Q be an arbitrary parameterized problem that is decided in fpt-time by an ATM $\mathbb{M}$ that is $f(k)$-alternating. We assume without loss of generality that $f(k)$ is even. We give an fpt-reduction from Q to QSat(level), using standard ideas from the proof of the Cook-Levin Theorem. We assume without loss of generality that for each nondeterministic step that $\mathbb{M}$ takes, there are only two possible transitions. If this were not the case, we could straightforwardly transform $\mathbb{M}$ so that it satisfies this property. Let y be an arbitrary instance for Q, with $|y| = n$. We know that $\mathbb{M}$ runs in fpt-time, i.e., in time $g(k)n^c$, for some computable function g and some constant c. Therefore, we know in particular that in each phase (existential or universal), $\mathbb{M}$ makes at most $g(k)n^c$ many nondeterministic (binary) choices. We introduce $f(k)$ many sets $X_1, \ldots, X_{f(k)}$, where each set X_i contains $g(k)n^c$ many propositional variables $x_{i,1}, \ldots, x_{i,g(k)n^c}$. The interpretation of these variables is that variable $x_{i,j}$ specifies which transition to take in the j-th nondeterministic step in the i-th phase of the computation. Using (the truth values of) these variables in $X_1, \ldots, X_{f(k)}$, we can then in fpt-time decide whether $\mathbb{M}$ ends up in an accepting state using these transitions (when given input y). Therefore, we can in fpt-time construct a Boolean circuit C over the variables in $X_1, \ldots, X_{f(k)}$ that captures this simulation procedure. Then, the quantified Boolean circuit $\exists X_1 \forall X_2 \ldots \forall X_{f(k)} C$ is true if and only if $y \in Q$. Finally, we can easily transform this to a quantified Boolean formula of the right form, for instance by using a standard Tseitin transformation to transform C into an equivalent universally quantified DNF formula $\exists Z \psi$. The result of the reduction is then the quantified Boolean formula $\exists X_1 \forall X_2 \ldots \forall X_{f(k)} \forall Y \psi$, which is an instance of QSat(level), and which is true if and only if $y \in Q$. ◀

Proof of Proposition 5. We fpt-reduce QSat(level) and MC[FO] to each other. The reduction from QSat(level) to MC[FO] is very straightforward. We construct a relational structure $\mathcal{A}$ with two elements $0, 1$ in its domain, and two unary predicates T and F, where $T^{\mathcal{A}} = \{1\}$ and $F^{\mathcal{A}} = \{0\}$. Then we transform the quantified Boolean formula φ to a first-order formula φ' by transforming each positive literal x to the first-order atom $T(x)$ and transforming each negative literal $\neg x$ to the atom $F(x)$. It is easy to verify the correctness of this reduction.

Next, for the other direction, we describe the fpt-reduction from MC[FO] to QSat(level). Let $\varphi = \exists X_1 \forall X_2 \ldots \forall X_k \psi$, together with the relational structure $\mathcal{A}$ with domain A, be an instance of MC[FO] (we assume without loss of generality that k is even). We replace each first-order variable $x \in X_i$ with $|A|$ many propositional variables x^a, for $a \in A$. Let X_i' denote the set of propositional replacement variables x^a for $x \in X_i$. Then, for each i, we construct a formula χ_i that ensures that for each $x \in X_i$ there is exactly one x_a that is true. Next, we transform ψ into a propositional formula ψ' as follows. Each occurrence of an atom $R(x_1, \ldots, x_r)$ in ψ, where $R^{\mathcal{A}} = \{(a_{1,1}, \ldots, a_{1,r}), \ldots, (a_{\ell,1}, \ldots, a_{\ell,r})\}$, we replace by the disjunction $\bigvee_{1 \leq i \leq \ell} (x_1^{a_{i,1}} \wedge \cdots \wedge x_r^{a_{i,r}})$. Finally, we construct the propositional formula $\psi'' = \chi_1 \wedge (\chi_2 \rightarrow (\chi_3 \wedge (\chi_4 \rightarrow \cdots (\chi_k \rightarrow \psi')) \cdots)$. The final result of the reduction is the quantified Boolean formula $\exists X_1' \forall X_2' \ldots \forall X_k' \psi''$. The correctness of this reduction can be verified straightforwardly. ◀

▶ **Lemma 10.** *Given a symbolically represented Kripke structure $\mathcal{M}$ given by (φ_R, α_0) over the set P of propositional variables, and a propositional formula ψ over P, we can construct in polynomial time a Kripke structure $\mathcal{M}'$ given by (φ_R', α_0') over the set $P \cup \{z\}$ of variables (where $z \notin P$) such that:*

- *there exists an isomorphism ρ between the states in the reachable part of $\mathcal{M}$ and the states in the reachable part of $\mathcal{M}'$ that respects the initial states and the transition relations,*
- *each state s in the reachable part of $\mathcal{M}$ agrees with $\rho(s)$ on the variables in P, and*
- *for each state s in the reachable part of $\mathcal{M}$ it holds that $\rho(s) \models z$ if and only if $s \models \psi$.*

Proof. Intuitively, the required Kripke structure $\mathcal{M}'$ can be constructed by adding the variable z to the set P of propositions, and modifying the formula φ_R specifying the transition relation and the initial state α_0 appropriately. In the new initial state α_0', the variable z gets the truth value 1 if and only if $\alpha_0 \models \psi$. Moreover, the transition relation specified by φ_R' ensures that in any reached state α, the variable z gets the truth value 1 if and only if $\alpha \models \psi$.

Concretely, we define $\mathcal{M}' = (\varphi_R', \alpha_0')$ over the set $P \cup \{z\}$ as follows. We let $\alpha_0'(p) = \alpha_0(p)$ for all $p \in P$, and we let $\alpha_0'(z) = 1$ if and only if $\alpha_0 \models \psi$. Then, we define φ_R' by letting:

$$\varphi_R'(\overline{x}, z, \overline{x}', z') = \varphi_R(\overline{x}, \overline{x}') \wedge (z' \leftrightarrow \psi(\overline{x})).$$

The isomorphism ρ can then be constructed as follows. For each state α in $\mathcal{M}$, $\rho(\alpha) = \alpha \cup \{z \mapsto 1\}$ if $\alpha \models \psi$, and $\rho(\alpha) = \alpha \cup \{z \mapsto 0\}$ if $\alpha \not\models \psi$. ◀

Proof of Theorem 6 (continued). We first show how to construct $\mathcal{M} = (\varphi_R, \alpha_0)$. Remember that $P = \{x_1, \ldots, x_n, y_1, \ldots, y_n, a_1, a_l, a_r, a_1', a_l', a_r', e_1, e_2, e_1', e_2', f, g\}$. We let α_0 be the assignment that sets only the propositional variables a_1, e_2' to true, and all other propositional variables to false. Then, we define $\varphi_R(\overline{p}^\star, \overline{p})$ to be the conjunction of the following subformulas. The first conjunct

$$(g \leftrightarrow e_1' \wedge \bigwedge_{p \in P \setminus \{e_1', g\}} \neg p)$$

ensures that g is only true in the sink state, and the second conjunct

$$(g \rightarrow (a_1 \wedge e_2' \wedge \bigwedge_{p \in P \setminus \{a_1, e_2'\}} \neg p))$$

ensures that the sink state is only reachable from the initial state. Next, we make sure that the states in the tree have the correct truth values for the propositional variables $y_1, \ldots, y_n$, i.e., that for each node in the i-th level of the tree, exactly the variables $y_1, \ldots, y_i$ are true. This is ensured by the following conjuncts:

$$\bigwedge_{1<i\leq n} (y_i \rightarrow y_{i-1}),$$
$$\bigwedge_{1\leq i<n} (\neg y_i \rightarrow \neg y_{i+1}), \text{ and}$$
$$\bigwedge_{1\leq i<n} (\neg(y_i \wedge y_{i+1} \wedge \neg y_i^\star \wedge \neg y_{i+1}^\star) \wedge \neg(y_i^\star \wedge y_{i+1}^\star \wedge \neg y_i \wedge \neg y_{i+1})).$$

We ensure that the propositional variable f is true exactly in the leafs of the tree, using the conjunct:

$$(f \leftrightarrow \bigwedge_{1\leq i\leq n} y_i).$$

Then, we ensure that each node at the i-th level of the tree corresponds to a partial truth assignment to the variables $x_1, \ldots, x_i$ that agrees with its parent node on the variables $x_1, \ldots, x_{i-1}$. We do so by means of the following conjuncts:

$$\bigwedge_{1\leq i\leq n} (\neg y_i \rightarrow \neg x_i), \text{ and}$$
$$\bigwedge_{1\leq i\leq n} ((y_i \leftrightarrow y_i^\star) \rightarrow (x_i \leftrightarrow x_i^\star)).$$

Finally, we enforce the intended truth values for the propositional variables in $A = \{a_1, a_l, a_r, a'_1, a'_l, a'_r\}$ and in $E = \{e_1, e_2, e'_1, e'_2\}$. In order to do so, we introduce two auxiliary formulas $\varphi_{\text{l-ch}}$ and $\varphi_{\text{r-ch}}$, that encode whether a node in the tree is a left or a right child:

$$\varphi_{\text{l-ch}}(\overline{p}) = \bigvee_{1 \leq i \leq n} (y_i \wedge \neg y_{i+1} \wedge \neg x_i) \vee (y_n \wedge \neg x_n), \text{ and}$$
$$\varphi_{\text{r-ch}}(\overline{p}) = \bigvee_{1 \leq i \leq n} (y_i \wedge \neg y_{i+1} \wedge x_i) \vee (y_n \wedge x_n).$$

Moreover, we introduce another auxiliary formula $\varphi_{\text{down}}(\overline{p}^\star, \overline{p})$, that encodes whether the transition goes down the tree:

$$\varphi_{\text{down}}(\overline{p}^\star, \overline{p}) = \bigvee_{1 \leq i \leq n} (\neg y_i \wedge y_i^\star).$$

Using these auxiliary formulas, we construct the following conjuncts of φ_R, that enforce the intended interpretation of the propositional variables in A and E:

$$(\neg g \wedge \varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge a_1^\star \wedge \varphi_{\text{l-ch}}(\overline{p})) \to (a_l \wedge e_1),$$
$$(\neg g \wedge \varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge a_1^\star \wedge \varphi_{\text{r-ch}}(\overline{p})) \to (a_r \wedge e_1),$$
$$(\neg g \wedge \varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge (a'_1)^\star \wedge \varphi_{\text{l-ch}}(\overline{p})) \to (a'_l \wedge e'_1),$$
$$(\neg g \wedge \varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge (a'_1)^\star \wedge \varphi_{\text{r-ch}}(\overline{p})) \to (a'_r \wedge e'_1),$$
$$(\neg g \wedge \varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge e_1^\star) \to (e_2 \wedge a'_1),$$
$$(\neg g \wedge \varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge (e'_1)^\star) \to (e'_2 \wedge a_1),$$
$$(\neg g \wedge \neg\varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge a_1^\star \wedge \varphi_{\text{l-ch}}(\overline{p})) \to (a'_l \wedge e'_1),$$
$$(\neg g \wedge \neg\varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge a_1^\star \wedge \varphi_{\text{r-ch}}(\overline{p})) \to (a'_r \wedge e'_1),$$
$$(\neg g \wedge \neg\varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge (a'_1)^\star \wedge \varphi_{\text{l-ch}}(\overline{p})) \to (a_l \wedge e_1),$$
$$(\neg g \wedge \neg\varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge (a'_1)^\star \wedge \varphi_{\text{r-ch}}(\overline{p})) \to (a_r \wedge e_1),$$
$$(\neg g \wedge \neg\varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge e_1^\star \to (a_1 \wedge e'_2),$$
$$(\neg g \wedge \neg\varphi_{\text{down}}(\overline{p}^\star, \overline{p}) \wedge (e'_1)^\star \to (a'_1 \wedge e_2),$$
$$\bigwedge_{c \in A} (c \to \bigwedge_{c' \in A \setminus \{c\}} \neg c'), \text{ and}$$
$$\bigwedge_{c \in E} (c \to \bigwedge_{c' \in E \setminus \{c\}} \neg c').$$

This concludes our construction of the Kripke structure $\mathcal{M}$ as depicted in Figure 2.

Next, we give a detailed specification of the LTL formula φ, that enforces a traversal of the tree $\mathcal{M}$ corresponding to a QBF model of the formula φ_0. The LTL formula φ consists of a conjunction of several subformulas. The first conjunct of φ is $\mathrm{X}a_l$, which ensures that the path starts by going down to the left child of the root of the tree. The next conjuncts are $\mathrm{G}[(a_1 \wedge \mathrm{X}e_1 \wedge \neg\mathrm{X}f) \to \mathrm{XX}e_2]$ and $\mathrm{G}[(a'_1 \wedge \mathrm{X}e'_1 \wedge \neg\mathrm{X}f) \to \mathrm{XX}e'_2]$, which ensure that after a transition corresponding to setting a universal variable to some value, the path goes further down the tree (if possible). The conjuncts $\mathrm{G}[(e_1 \wedge \mathrm{X}e_2) \to \mathrm{XX}a'_l]$ and $\mathrm{G}[(e'_1 \wedge \mathrm{X}e'_2) \to \mathrm{XX}a_l]$ ensure that after setting an existential variable to some value, the path goes further down the tree by setting the next universal variable to 0. The conjunct $\mathrm{G}[(a_1 \wedge \mathrm{X}e_1 \wedge \mathrm{X}f) \to \mathrm{XX}a_1]$ ensures that after reaching a leaf of the tree, the path goes back up. The conjuncts $\mathrm{G}[(a_l \wedge \mathrm{X}a_1) \to \mathrm{XX}a_r]$ and $\mathrm{G}[(a'_l \wedge \mathrm{X}a'_1) \to \mathrm{XX}a'_r]$ ensure that after the path goes back up a transition corresponding to setting a universal variable x_i to 0, the path continues (downwards) with the transition corresponding to setting the variable x_i to 1. The conjuncts $\mathrm{G}[(a_r \wedge \mathrm{X}a_1) \to \mathrm{XX}e'_1]$ and $\mathrm{G}[(a'_r \wedge \mathrm{X}a'_1) \to \mathrm{XX}e_1]$ ensure that after the path goes back up a transition corresponding to setting a universal variable x_i to 0, the path continues (upwards) by going back up on the transition corresponding to setting variable x_{i-1}. The conjuncts $\mathrm{G}[(e_2 \wedge \mathrm{X}e_1) \to \mathrm{XX}a_1]$ and $\mathrm{G}[(e'_2 \wedge \mathrm{X}e'_1 \wedge \mathrm{X}\neg g) \to \mathrm{XX}a'_1]$ ensure that after

the path goes back up a transition corresponding to setting an existential variable x_i to some value, the path continues (upwards) by going back up on the transition corresponding to setting variable x_{i-1} (if possible).

Finally, we need to ensure that this tree traversal corresponds to a QBF model for the formula φ_0, i.e., that all total assignments that appear along the path satisfy the matrix ψ of φ_0. We do so by adding a last conjunct to φ. However, to keep the LTL formula φ of constant size, we need to introduce a propositional variable that abbreviates the truth of ψ. By Lemma 10, we may assume without loss of generality that there is a propositional variable z_ψ in P that in each state is set to 1 if and only if this state sets the formula ψ to true. We then let the final conjunct of φ be $\mathrm{G}[f \to z_\psi]$. Clearly, the size of φ is constant. ◀

Proof of Theorem 8 (continued). First, we show how to construct the Kripke structure $\mathcal{M} = (\varphi_R, \alpha_0)$. Remember that $P = X_1 \cup \cdots \cup X_k \cup \{l_0, l_1, \ldots, l_k\}$ (for the sake of simplicity, we leave treatment of the propositional variable z_ψ to the technique discussed in the proof of Lemma 10). We let α_0 be the truth assignment that sets only the propositional variable l_0 to true, and all other propositional variables to false. Then, we define $\varphi_R(\overline{p}, \overline{p}')$ as the conjunction of several subformulas. The first conjuncts ensure that in each level of the tree, the propositional variables l_i get the right truth value:

$$\bigwedge_{0 \le i < k} l_i \to l'_{i+1}, \quad (l_k \to l'_k) \quad \text{and} \quad \bigwedge_{0 \le i < i' \le k} \neg(l'_i \wedge l'_{i'}).$$

The following conjunct ensures that the partial truth assignment of a node at the i-th level of the tree agrees with its parent on all variables in $X_1, \ldots, X_{i-1}$.

$$\bigwedge_{1 \le i \le n} (l'_i \to \bigwedge_{x \in X_1 \cup \cdots \cup X_{i-1}} (x \leftrightarrow x')).$$

This concludes our construction of the Kripke structure $\mathcal{M}$ as depicted in Figure 3.

In order to prove membership in PH(LEVEL), we show that SYMBOLIC-MC*[CTL] can be decided in fpt-time by an ATM $\mathbb{M}$ that is $f(k)$-alternating. The algorithm implemented by $\mathbb{M}$ takes a different approach than the well-known dynamic programming algorithm for CTL model checking for explicitly encoded Kripke structures (see, e.g., [2, Section 6.4.1]). Since symbolically represented Kripke structures can be of size exponential in the input, this bottom-up algorithm would require exponential time. Instead, we employ a top-down approach, using (existential and universal) non-determinism to quantify over the possibly exponential number of states.

We consider the function `CTL-MC`, given in pseudo-code in Algorithm 1, which takes as input the Kripke structure $\mathcal{M}$ in form of its representation (φ_R, α_0), a state α in $\mathcal{M}$, a CTL formula Φ and the recurrence diameter $rd(\mathcal{M})$ of $\mathcal{M}$ (in unary), and outputs 1 if and only if α makes Φ true. Note that in this algorithm, we omit the case for the operator F, as any CTL formula $\exists \mathrm{F}\Phi$ is equivalent to $\exists \top \mathrm{U} \Phi$. It is readily verified that this algorithm correctly computes whether $\mathcal{M}, \alpha \models \Phi$. Therefore, $\mathcal{M} \models \Phi$ if and only if `CTL-MC`($\mathcal{M}$, α_0, Φ, m) returns 1, where m is the unary encoding of $rd(\mathcal{M})$.

It remains to verify that the algorithm `CTL-MC` can be implemented in fpt-time by an $f(k)$-alternating ATM $\mathbb{M}$. We can construct $\mathbb{M}$ in such a way that the existential guesses are done using the existential non-deterministic states of $\mathbb{M}$, and the universal guesses by the universal non-deterministic states. Note that the recursive call in the case for $\neg\Phi_1$ is preceded by a negation, so the existential and universal non-determinism swaps within this recursive call. The recursion depth of the algorithm is bounded by $|\Phi| = k$, since each recursive call strictly decreases the size of the CTL formula used. Moreover, in each call of the function

Algorithm 1: Recursive CTL model checking using bounded alternation.

```
function CTL-MC(M, α, Φ, m):
    switch Φ do
        case p ∈ P: return α(p) ;                                  /* Boolean cases */
        case ¬Φ1: return not CTL-MC(M, α, Φ1, m) ;
        case Φ1 ∧ Φ2: return CTL-MC(M, α, Φ1, m) and CTL-MC(M, α, Φ2, m) ;
        case ∃XΦ1:
            (existentially) pick a state α' in M ;          /* guess successor state */
            if φR(α, α') is false then return 0 ;               /* check transition */
            return CTL-MC(M, α', Φ1, m) ;                                /* recurse */
        case ∃Φ1UΦ2:
            (existentially) pick some m' ≤ m ;               /* guess length of path */
            (existentially) pick states α1, ..., αm' in M ;          /* guess path */
            (universally) pick some 1 ≤ j < m' ;     /* check all states in path */
            if φR(αj, αj+1) is false then return 0 ;          /* check transition */
            if CTL-MC(M, αj, Φ1, m) = 0 then return 0 ;               /* recurse */
            return CTL-MC(M, αm', Φ2, m) ;                            /* recurse */
```

CTL-MC, at most two recursive calls are made (not counting recursive calls at deeper levels of recursion). Therefore, the running time of $\mathbb{M}$ is bounded by $2^k\text{poly}(n)$, where n is the input size. Also, since in each call of the function at most two alternations between existential and universal non-determinism are used (again, not counting at deeper levels of recursion), we know that $\mathbb{M}$ is 2^k-alternating. (This bound on the number of alternations needed can be improved with a more careful analysis and some optimizations to the algorithm.) ◀

Proof of Theorem 9. Hardness for PH(LEVEL) follows from Theorem 8. We show membership in PH(LEVEL), by describing an algorithm A to solve the problem that can be implemented by an ATM $\mathbb{M}$ that runs in fpt-time and that is $f(k)$-alternating. The algorithm works similarly to Algorithm 1, described in the proof of Theorem 8, and recursively decides the truth of a CTL* formula in a state. The difference with Algorithm 1 is that it does not look only at the outermost temporal operators of the CTL* formula in a recursive step, but considers possibly larger subformulas in each recursive step. Let $\exists\varphi$ be a CTL* formula, and let s be a state in $\mathcal{M}$. The algorithm A then considers all maximal subformulas $\psi_1, \ldots, \psi_\ell$ of φ that are CTL* state formulas as atomic propositions $p_1, \ldots, p_\ell$, turning the formula φ into an LTL formula. Since φ does not contain the operators U and X, we know that in order to check the existence of an infinite path satisfying φ, it suffices to look for lasso-shaped paths of bounded length (linear in $rd(\mathcal{M})$ and exponential in the size of φ), i.e., a finite path followed by a finite cycle [22]. The algorithm A then uses (existential) nondeterminism to guess such a lasso-shaped path π, and to guess for each state which of the propositions $p_1, \ldots, p_\ell$ are true, and verifies that π witnesses truth of $\exists\varphi$. Then, in order to ensure that it correctly determines whether $\exists\varphi$ is true, the algorithm needs to verify that it guessed the right truth values for $p_1, \ldots, p_\ell$ in π. It does so by recursively determining, for each state s' in the lasso-shaped path π, and each p_i, whether ψ_i is true in s' if and only it guessed p_i to be true in s'. (In order to ensure that in each level of recursion there are only a constant number of recursive calls, like Algorithm 1, the algorithm A uses universal nondeterminism iterate over each p_i and each s'.) The algorithm then reports that $\exists\varphi$ is true in s if and only if (1) the guesses for π and the truth values of $p_1, \ldots, p_\ell$ together form a correct witness for truth

of $\exists\varphi$, and (2) for each p_i and each s' it holds that p_i was guessed to be true in s' if and only if ψ_i is in fact true in s'. The recursive cases for CTL* formulas where the outermost operator is not temporal are analogous to Algorithm 1. Like Algorithm 1, the algorithm runs in fpt-time and is 2^k-alternating. ◀