LTL Fragments are Hard for Standard Parameterisations

Martin Lück and Arne Meier
Institut für Theoretische Informatik
Leibniz Universität Hannover
Appelstr. 4, 30176 Hannover, Germany
Email: {lueck.meier}@thi.uni-hannover.de

Abstract—We classify the complexity of the LTL satisfiability and model checking problems for several standard parameterisations. The investigated parameters are temporal depth, number of propositional variables and formula treewidth, resp., pathwidth. We show that all operator fragments of LTL under the investigated parameterisations are intractable in the sense of parameterised complexity.

I. INTRODUCTION

In the last decade the research on parameterised complexity of problems increased significantly. Beyond the foundations by Downey and Fellows [5] until today several deep algorithmic techniques have been introduced and new approaches have been made; so it really is a highly prospering area of research (e.g., see for an overview of the current evolution the recent book of Downey and Fellows [6]). Essentially the main approach is to detect a parameterisation for a given problem and achieve a runtime which is independent of the parameter. For instance, given the problem of propositional satisfiability a quite naïve parameterisation is the number of variables (of the given formula φ). Then one can easily construct a deterministic algorithm solving the problem in time $2^k \cdot |φ|$ if $k$ is the number of variables in $φ$. If $k$ is assumed to be fixed then this yields a polynomial runtime wherefore one says that this problem is fixed-parameter tractable (FPT), in general, a runtime of the form $f(k) \cdot poly(n)$ where $k$ is the parameter, $n$ is the input length and $f$ is an arbitrary computable function; another name for this class is para-P. In contrast with this, runtimes of the form $n^{f(k)}$ are highly undesirable as the runtime depends on the parameter’s value—this is the runtime of algorithms in the class XP. Further some kind of parameterised intractability hierarchy is built between FPT and XP; namely the W-hierarchy. It is known that $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq XP$ but not if any of these inclusions is strict. For proving hardness results with respect to the classes of the W-hierarchy one usually uses fpt-reductions which translate, informally speaking, the instances in the usual sense from one parameterised problem to another but also take care of maintaining the parameter’s value. Hence showing $W[1]$-hardness of a problem yields the unlikeliness of it to being fixed-parameter tractable. Also classes like para-NP, similarly to para-P but using nondeterministic algorithms instead, or para-PSPACE have been introduced. The first contains the W-hierarchy and is by itself contained in para-PSPACE. Further they are incomparable to XP—under reasonable complexity class inclusion assumptions.

While the parameterised complexity theory is heavily built on top of logic, its application is relatively new in the context of logic itself. Not many significant parameterisations are known yet. Recent approaches were modal, resp., temporal operator nesting depth [15, 12] or various types of treewidth, like primal or incidence treewidth of CNF formulas [17]. Our treewidth parameter is a further generalisation and can be measured on general syntax trees of formulas and not only on CNFs.

Temporal logic as a well-known and very important area relevant in many fields of research, e.g., program verification and artificial intelligence. Its origins are traceable even to greek philosophers yet it founds its introduction by Arthur Prior in the 1960s [16]. Conceptually its main ingredients are combinations of a universal or existential path quantifier together with temporal operators referring to specific or vague moments of time, e.g., next, globally, future, until. Depending on how these quantifiers and operators may be combined the three most important logics have been defined. In Computation Tree Logic (or short CTL) one is allowed to use only pairs of a single path quantifier and a single temporal operator; in Linear Temporal Logic (LTL) one uses only temporal operators and has a single existential (or universal, depending on the definition of the logic) path quantifier in front of the formula; in the Full Branching Time Logic (CTL∗) any arbitrary combination is allowed. After a decade of seminal work from Allen Emerson, Clarke, Halpern, Schnoebelen, and
Sistla [4, 7, 19, 18]—to name only a few—the most important concepts, e.g., satisfiability and model checking have been well understood and classified with respect to their computational complexity. Recently the mentioned decision problems have been investigated in the light of a study which considers fragments of the logics in the sense of allowed operators [1, 13].

In this paper we focus on the logic LTL and its PSPACE-complete model checking. We want to investigate its parameterised complexity under the mentioned parameterisations of operator fragments.

A. Related Work.

The parameterised complexity of modal logic satisfiability has been investigated by Praveen recently [15]. He considered treewidth of some CNF-centered graph representation structure. In a work of Szeider from 2004 he discusses different parameterisation approaches with respect to propositional satisfiability [21]. In particular, he explains how to obtain primal graphs and other structural parameterisations. Recently, Lück et al. classified the parameterised complexity of satisfiability for the computation tree logic CTL [12]. Essentially we follow their used parameterisations in this paper. The results in Corollary 25 comply with the results in of Bauland et al. who investigated the existential version of LTL [1].

B. Organization of this paper.

At first we will define the relevant notions and terms around temporal logic, parameterised complexity, and our used structural parameterisations. Then in Section III we will present our classification for the different parameterisations and decision problems. We start with satisfiability and the main part is about model checking. Finally in Section IV we will conclude and give an outlook on further steps.

II. PRELIMINARIES

A. Temporal Logic

Usually temporal logic is defined through Kripke semantics. In the following we will briefly introduce the notion around it. For a deeper introduction, we confer the reader to the survey article from Meier et al. [14]. Formally a Kripke structure $A = (W, R, V)$ is a finite set $W$ of worlds (or states), $R: W \times W$ is a total transition relation (i.e., for every $w \in W$ there exists a $w' \in W$ such that $w R w'$ holds), and $V: W \rightarrow 2^{PROP}$ is a valuation function assigning sets of propositions to states, where PROP is a finite set of propositions. A path $\pi = p_0p_1 \ldots$ is an infinite sequence of states such that $p_i R p_{i+1}$ holds for $i \in \mathbb{N}$. A path starting at the root of a model is also called a run. For $w \in W$ write $\Pi(w)$ for the set of all paths starting in $w$. Write $\pi[i]$ for the world $p_i$ and $\pi^i$ for the suffix path $p_ip_{i+1} \ldots$ of $\pi$. The set of all well formed linear temporal logic formulas $LTL$ is defined inductively via the following grammar in BNF

$$\varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid X\varphi \mid F\varphi \mid \varphi U \varphi,$$

where $p \in PROP$.

Now let $A = (W, R, V)$ be a Kripke structure, $\varphi, \psi$ be two $LTL$-formulas, and $\pi$ be a path in $A$. Then it holds that

$$(A, \pi) \models p \iff p \in V(\pi[0]),$$

$$(A, \pi) \models \neg \varphi \iff (A, \pi) \not\models \varphi,$$

$$(A, \pi) \models \varphi \land \psi \iff (A, \pi) \models \varphi \land (A, \pi) \models \psi,$$

$$(A, \pi) \models X\varphi \iff (A, \pi^i) \models \varphi,$$

$$(A, \pi) \models F\varphi \iff \exists k \geq 0 : (A, \pi^k) \models \varphi,$$

$$(A, \pi) \models \varphi U \psi \iff \exists k \geq 0 \text{ such that } \forall j < k : (A, \pi^j) \models \varphi \land (A, \pi^k) \not\models \psi.$$

The usual shortcuts are obtained by combinations of these operators, e.g., $G\varphi \equiv \neg F\neg \varphi$, or $\varphi \rightarrow \psi \equiv \neg \varphi \lor \psi$.

If $T \subseteq \{X, F, G, U\}$ is a set of temporal operators, then $LTL(T)$ is the restriction of $LTL$ to formulas containing only temporal operators from $T$. In the following we consider the relevant decision problems with respect to a fix set of temporal operators $T$ for this paper.

| Problem          | Input                                      | Question                                    |
|------------------|--------------------------------------------|---------------------------------------------|
| LTL-SAT(T)       | $LTL(T)$-formula $\varphi$                 | Does there exist a Kripke structure $A$ and a path $\pi$ in $A$ such that $(A, \pi) \models \varphi$? |
| LTL-MC(T)        | A Kripke structure $A$, a world $w \in W$ and $LTL(T)$-formula $\varphi$ | $(A, \pi) \models \varphi$ for all paths $\pi \in \Pi(w)$? |

Note that the model checking problem is also sometimes defined with existential semantics, i.e., at least one path has to satisfy the formula, which leads to complementary complexity results.

**Definition 1.** The temporal depth $td(\cdot)$ of an LTL formula is defined recursively:

$$
\begin{align*}
\text{td}(x) & := 0 \text{ if } x \in PROP \\
\text{td}(\neg \varphi) & := \text{td}(\varphi) \\
\text{td}(\varphi \land \psi) & := \max\{\text{td}(\varphi), \text{td}(\psi)\} \\
\text{td}(T\varphi) & := \text{td}(\varphi) + 1 \text{ for } T \in \{X, F, G\} \\
\text{td}(\varphi U \psi) & := \max\{\text{td}(\varphi), \text{td}(\psi)\} + 1
\end{align*}
$$

**Definition 2.** Define $LTLc(T)$ as the fragment of $LTL(T)$ which has temporal depth at most $c$. Define $LTLc-MC(T)$, resp., $LTLc-SAT(T)$ as the model checking resp. satisfiability problem restricted to formulas in $LTLc(T)$.

B. Parameterized Complexity

**Definition 3** (Parameterised problem). Let $Q \subseteq \Sigma^*$ be a decision problem and let $\kappa: \Sigma^* \rightarrow \mathbb{N}$ be a computable function. Then we call $\kappa$ a parameterisation of $Q$ and the pair $\Pi = (Q, \kappa)$ a parameterised problem.

**Definition 4** (Fixed-parameter tractable). Let $\Pi = (Q, \kappa)$ be a parameterised problem. If there is a deterministic Turing machine $M$ and a computable function $f: \mathbb{N} \rightarrow \mathbb{N}$ s.t. for every instance $x \in \Sigma^*$

- $M$ decides correctly if $x \in Q$, and
• M has a runtime bounded by $f(\kappa(x)) \cdot |x|^{O(1)}$
then we say that M is an \textit{fpt-algorithm} for II and that II is 
\textit{fixed-parameter tractable}. We define FPT as the class of all 
parameterised problems that are fixed-parameter tractable.

Similarly, we refer to a function f as \textit{fpt-computable} w.r.t. 
a parameter $\kappa$ if there is another computable function $h$ such 
that $f(x)$ can be computed in time $h(\kappa(x)) \cdot |x|^{O(1)}$.

\textbf{Definition 5} (fpt-reduction). Let $(P, \kappa)$ and $(Q, \lambda)$ be 
parameterised problems over alphabets $\Sigma$, resp., $\Delta$. Then a function 
f: $\Sigma^* \rightarrow \Delta^*$ is an \textit{fpt-reduction} if it is fpt-computable 
w.r.t. $\kappa$ and there is a computable function $h: \mathbb{N} \rightarrow \mathbb{N}$ s.t. the 
following holds f.a. $x \in \Sigma^*$:

- $x \in P \iff f(x) \in Q$ and
- $\lambda(f(x)) \leq h(\kappa(x))$, i.e., $\lambda$ is bounded by $\kappa$.

If there is an fpt-reduction from $(P, \kappa)$ to $(Q, \lambda)$ for 
parameterised problems $(P, \kappa)$ and $(Q, \lambda)$ then we call $(P, \kappa)$ 
fpt-reducible to $(Q, \lambda)$, denoted by $(P, \kappa) \leq_{\text{fpt}} (Q, \lambda)$.

\textbf{Definition 6} (W[1]). The class W[1] is the class of parameterised 
problems $(Q, \kappa)$ such that $(Q, \kappa)$ can be fpt-reduced to the 
Short Single-Tape Turing Machine Halting Problem:

| Problem: | SSTMH |
|---------|-------|
| Input:  | Non-deterministic single-tape Turing machine $M$, Integer $k$ |
| Question: | Does $M$ accept the empty string in at most $k$ steps? |
| Parameter: | $k$ |

\textbf{Definition 7} (Parameterised hardness). A problem $(P, \kappa)$ is $C$-
hard under fpt-reductions for a parameterised complexity class $C$ if $(Q, \lambda) \in C$ implies $(Q, \lambda) \leq_{\text{fpt}} (P, \kappa)$. If additionally $(P, \kappa) \in C$, we say that $(P, \kappa)$ is $C$-complete under fpt-reductions.

\textbf{Definition 8} (W[P]). The class W[P] contains the parameterised 
problems $(Q, \kappa)$ for which there is a computable function $f$ and an NTM deciding if $x \in Q$ holds in 
time $f(\kappa(x)) \cdot |x|^{O(1)}$ with at most $O(\kappa(x) \cdot \log |x|)$ non-
deterministic steps.

Flum and Grohe state how to obtain parameterised variants of 
classical complexity classes [8]. They define for “standard” 
complexity classes $C$ the corresponding parameterised versions 
para-$C$. Here, “standard” means that the class $C$ is defined 
via usual resource-restricted Turing machines. For most such 
classes $C$ we obtain para-$C$ by simply appending an additional 
factor $f(\kappa)$ to the resource bound, as done for P leading to 
FPT. This is possible for certain classes that Flum and Grohe 
call \textit{robust}, such as NP and PSPACE. This allows the following 
definitions:

\textbf{Definition 9} (para-NP). The class para-NP contains the parameterised 
problems $(Q, \kappa)$ for which there is a computable function $f$ and an non-deterministic Turing Machine deciding if $x \in Q$ holds in time $f(\kappa(x)) \cdot |x|^{O(1)}$.

\textbf{Definition 10} (para-PSPACE). The class para-PSPACE contains 
the parameterised problems $(Q, \kappa)$ for which there is 
a computable function $f$ and a deterministic Turing machine 
deciding if $x \in Q$ holds in space $f(\kappa(x)) \cdot |x|^{O(1)}$.

\textbf{Definition 11} (Slice). The $\ell$-th slice of a parameterised problem $(Q, \kappa)$ is denoted $(Q, \kappa)_\ell$ and defined as:

$$(Q, \kappa)_\ell := \{ x \mid x \in Q \text{ and } \kappa(x) = \ell \}$$

\textbf{Proposition 12} ([8]). For $C \in \{ \text{NP}, \text{coNP}, \text{PSPACE} \}$ it holds 
that a parameterised problem $(Q, \kappa)$ is hard for para-$C$ if and 
only if a finite union of slices of $(Q, \kappa)$ is hard for $C$.

\textbf{Definition 13} (Complement Class). For a complexity class $C$, 
define coC as the class of problems for which their 
complement is in $C$.

\section{C. Structural Treewidth and Pathwidth}

The \textit{treewidth} of a graph is a parameter that leads to FPT 
algorithms for a wide range of otherwise intractable graph 
problems. In fact, only few known graph problems stay hard 
on trees. The treewidth of a graph is in this sense a measure 
of its “tree-likeliness”.

A \textit{path-decomposition} $P$ of a structure $A$ is similarly 
defined to tree-decompositions however $P$ has to be a path. 
Here $pw(A)$ denotes the \textit{pathwidth} of $A$. Likewise the size of 
the pathwidth describes the similarity of a structure to a path. 
Observe that pathwidth bounds treewidth from above.

\textbf{Definition 14} (Graph treewidth and pathwidth). The \textit{width} of 
a tree-decomposition $T$ is its maximal bag size minus one. 
The \textit{treewidth} of a graph $G$ is the minimal width of a 
tree-decomposition of $G$, and its \textit{pathwidth} is the minimal width 
of a path-decomposition.

\textbf{Definition 15} (Syntactical structure of formulas). We associate 
a formula $\varphi$ with a graph $S(\varphi)$, resp., $S_\varphi$ which represents the 
formula. The map $S$ is defined as follows: Let $S(\varphi) := (SF(\varphi), E)$. The set SF($\varphi$) is the set of all syntactically 
valid subformulas of $\varphi$ (counting equal subformulas multiple 
times if necessary). SF’($\varphi$) is then obtained from SF($\varphi$) 
by identifying nodes which represent the same propositional 
variable inside $\varphi$. The edge set $E$ is obtained from $\varphi$ 
by connecting each pair $(\psi, \psi') \in SF'(\varphi) \times SF'(\varphi)$ for which 
$\psi'$ is a maximal strict subformula inside $\psi$.

We assume a well-defined association with parentheses s.t. 
every node of $S(\varphi)$ represents exactly one Boolean function 
or temporal operator and its children represent its arguments. 
Then $S$ can be interpreted as a “graphical” representation of 
$\varphi$ in the sense of a syntax tree. Merging the leaves with 
equal propositional variable then leads to a cyclic graph. 
The motivation of using the “syntax graph” treewidth is that 
independent of each other. If many subformulas are connected by common 
variables, intuitively can be handled independently of each 
other. This is reflected by a high treewidth.
**Definition 16** (Formula treewidth and pathwidth). For a formula \( \varphi \), its treewidth \( \text{tw}(\varphi) \), resp., pathwidth \( \text{pw}(\varphi) \) is simply defined as \( \text{tw}(S_\varphi) \) resp. \( \text{pw}(S_\varphi) \).

The syntax graphs as defined above is a generalization of the incidence graphs of CNF formulas, and the incidence graph of a CNF is contained as a graph minor in its syntax graph: Simply merge all propositional variables with their negations, then for every clause contract all edges that belong to it. Then delete the disjunction nodes above the clauses. Therefore the structural treewidth is an upper bound for the incidence treewidth, the same holds for the pathwidth. This implies that all hardness results regarding the structural treewidth or pathwidth also hold for the incidence treewidth or pathwidth.

III. PARAMETERISED COMPLEXITY RESULTS

**A. Satisfiability**

**Theorem 17.** For \( F \in T \), \( G \in T \) or \( U \in T \), the problems (LTL-SAT(\(T\)), td + \(\text{pw}_\varphi\)) and (LTL-SAT(\(T\)), td + \(\text{tw}_\varphi\)) are \(W[1]\)-hard.

**Proof.** The result is proven by an fpt-reduction from the parameterised problem p-PW-SAT which was shown to be \(W[1]\)-hard by Praveen [15]. An instance of p-PW-SAT is a tuple \( I = (\varphi, k, (Q_i)_{i \in [k]}, (C_i)_{i \in [k]}) \) where \( \varphi \) is a propositional formula in CNF with variables \( \{q_1, \ldots, q_n\} \). The variables are partitioned into pairwise disjoint subsets \( \{Q_1, \ldots, Q_k\} \). The values \( C_i \subseteq \mathbb{N} \) are the capacities of the partitions, i.e., the exact number of variables in \( Q_i \) that must be set to one, which is the weight of the partition. An assignment is called saturated if every partition has weight equal to its capacity. For an instance \( I \) we say that \( I \) is \( p\)-PW-SAT if \( \varphi \) has an assignment that is both satisfying and saturated. The parameter of p-PW-SAT is \( \kappa(I) := k + \text{pw}(G_\varphi) \) where \( G_\varphi \) is the primal graph of the CNF \( \varphi \).

For the reduction, we consider an LTL formula \( \psi(I) \in \mathcal{LTL}(F, G) \) that has constant temporal depth and \( \kappa \)-bounded pathwidth (and therefore treewidth). The formula \( \psi(I) \) is a conjunction of the following subformulas.

\[
\begin{align*}
\psi[\text{formula}] &:= \varphi \\
\psi[\text{depth}] &:= G \bigwedge_{i=0}^{n-1} \left( (d_i \land \neg d_{i+1}) \rightarrow (m_i \mod 2 \land \neg m_{1-(i \mod 2)} \land F(d_{i+1} \land \neg d_{i+2})) \right) \\
\psi[\text{fixed-Q}] &:= \bigwedge_{i=1}^{n} \left( (q_i \rightarrow Gq_i) \land (\neg q_i \rightarrow G\neg q_i) \right) \\
\psi[\text{signal}] &:= \bigwedge_{i=1}^{n} \left[ (d_i \land \neg d_{i+1}) \rightarrow \left( \left( q_i \rightarrow T^p_{m(i)} \right) \land (\neg q_i \rightarrow T^\bot_{m(i)}) \right) \right] \\
\psi[\text{init}] &:= d_0 \land \neg d_1 \land G \bigwedge_{p=1}^{k} \left( T^0_p \land \bot_p \right)
\end{align*}
\]

The proof of correctness and \( \kappa \)-boundedness of the pathwidth is omitted. The reader is instead referred to the thesis of Lück [11]. Using the equivalences \( F\alpha \equiv \neg G\neg \alpha \), \( G\alpha \equiv \neg F\neg \alpha \) and \( F \equiv T\cup\alpha \), the reduction is possible with any temporal operator except \( X \).

**Theorem 18.** For any set \( T \), the problem (LTL-SAT(\(T\)), td) is para-NP-hard.

**Proof.** Trivial reduction from SAT to LTL-SAT(\(T\)) and application of Proposition 12.

**Theorem 19.** For \( T \subseteq \{X\} \), the problems (LTL-SAT(\(T\)), tw_\varphi) and (LTL-SAT(\(T\)), pw_\varphi) are in FPT.

**Proof.** It holds due to the path semantics of LTL that \( X(\phi \land \psi) \equiv X\phi \land X\psi \), \( X(\phi \lor \psi) \equiv X\phi \lor X\psi \) and \( X\neg \phi \equiv \neg X\phi \) for \( \phi, \psi \in \mathcal{LTL} \). Hence every LTL formula with only X-operators can efficiently be converted to an equivalent Boolean combination \( \beta \) of X-preceded variables:

\[
\varphi \equiv \beta(X^{q_1}q_1, \ldots, X^{q_n}q_n), \quad X^i := X \cdots X, \quad i \text{ times}
\]

where the \( q_i \) are propositional variables. Inconsistent literals can only occur inside the same world and therefore at the same nesting depth of \( X \). Hence the above formula \( \varphi \) is satisfiable if and only if it is satisfiable as a purely propositional formula where the expression \( X^{q_i}q_i \) is interpreted as an atomic formula (i.e., a variable).

Formally we have (LTL-SAT(\(X\)), tw_\varphi) \leq_{\text{fpt}} (SAT, tw_\varphi).

(SAT, tw_\varphi) \in FPT as a special case of CTL was shown by Lück et al. [12]. As pathwidth is an upper bound for treewidth, (LTL-SAT(\(X\)), pw_\varphi) is in FPT as well.

Unsurprisingly, all hard LTL fragments correspond to hard CTL fragments if we just supplement the operators with path quantifiers [12]. For the fragment CTL(\(A\times\mathcal{X}\)) (or equivalently modal logic on serial frames) being in FPT [15] we however need the temporal depth as an additional parameter. As satisfiable LTL formulas are already satisfied on paths and less expressive than modal logic, this extra parameter is not required for the \( \mathcal{LTL}(X) \) fragment.
In the next part we turn from satisfiability to model checking.

B. Model checking

It turns out that LTL model checking is surprisingly hard for almost all studied parameterisations. This is already the case in classical complexity theory. While model checking for CTL is P-complete, it is PSPACE-complete for LTL and CTL*, and is in fact NP-hard for every fragment with a non-empty operator set. This inherent hardness is due to the different semantics of CTL and LTL: In CTL, every subformula of a formula is what is called state formula. A polynomial time algorithm is obtained by recursively determining fulfilled subformulas in every world of the model. LTL is built from path formulas which have no truth value w.r.t. to states but only to paths from the root of the model. This forbids P algorithms in the CTL style; hardness reductions to LTL model checking usually construct Kripke structures with few worlds and (exponentially) many paths between them. In fact, LTL model checking on nonbranching structures is in P [9]. The reductions in this section follow this scheme and in general produce branching Kripke structures with certain properties.

Long before the introduction of parameterised complexity theory, statements as early as from Lichtenstein and Pnueli already distinguished between program complexity, the runtime dependent on the length of the formula $\varphi$, and structure complexity, the runtime dependent on the length of the structure $A$ to be checked [10]. They stated that the runtime factor $2^{|\varphi|}$ in their algorithm does not prohibit efficient model checking as the size of the structure is clearly dominant in practice.

In the context of parameterised complexity, this automatically yields nice fixed-parameter tractable problems:

**Corollary 20.** Let $\kappa(\varphi, A, w) := |\varphi|$. Then $(\text{LTL-MC}, \kappa) \in \text{FPT}$. 

The next logical step is to study the influence of more parameterisations on the model checking complexity: Define $\text{tw}_A$ as the treewidth of the input structure, i.e., $\text{tw}_A(\varphi, A, w) := \text{tw}(A)$. Define $\text{tw}_A$ as the structural treewidth of the input formula, i.e., $\text{tw}_A(\varphi, A, w) := \text{tw}(S^\varphi)$. Similarly define $\text{pw}_A$ and $\text{pw}_A$.

**Proposition 21** ([20, 18]). LTL-MC($X, F, G, U$) $\in$ PSPACE.

**Proposition 22** ([20, 18]). For $T \subseteq \{X\}$ or $T \subseteq \{F, G\}$ it holds LTL-MC($T$) $\in$ coNP.

**Definition 23** (Maximum branching degree). Let $A$ be a Kripke structure. Then write $\Delta(A)$ for the maximum branching degree in $A$, i.e., the smallest number $\Delta$ s.t. every world in $A$ has at most $\Delta$ successors.

**Theorem 24.** $(\text{LTL-MC}(F), \text{td} + \Delta)$ is complete for para-coNP.

**Proof.** We follow Sistla and Clarke who showed that LTL-MC($F$) (i.e., only F-operators without nesting) is coNP-hard. This is done by a reduction from the complement of the NP-complete 3SAT problem: Given a propositional formula $\varphi$ in 3CNF, is it satisfiable?

For this we construct a formula $\psi \in LTL_1(F)$ and a structure $S$ with constant branching such that $(S, w_0) \models \psi$ if and only if $\varphi$ is unsatisfiable. First assume $\varphi = \bigwedge_{i=1}^{m} (L_{i,1} \lor L_{i,2} \lor L_{i,3})$ where $L_{i,j}$ is a literal, i.e., a propositional variable or its negation. Then simply define $\psi := \bigwedge_{i=1}^{m} (F \neg L_{i,1} \land F \neg L_{i,2} \land F \neg L_{i,3})$, so $\psi$ is basically the negation of $\varphi$ supplemented with $F$ operators in front of the literals.

Assume that $\varphi$ contains variables $\{x_1, \ldots, x_n\}$. For a correct reduction the structure $S$ is now required to allow either $Fx_i$ or $F \neg x_i$ to hold for $1 \leq i \leq n$, but not both. Also, for every subset $X \subseteq \{x_1, \ldots, x_n\}$ of variables (which can be interpreted as the assignment that sets exactly the variables in $X$ to true) there should be a path through $S$ and vice versa. The structure depicted in Figure 2 has these property and therefore models propositional assignments as runs from its initial world $w_0$. This means that all runs in $S$ fulfill the path formula $\psi$ if and only if $\neg \varphi$ is satisfied by all Boolean assignments. Hence $\varphi \notin 3SAT \Leftrightarrow (\psi, S, w_0) \in \text{LTL}_1\text{-MC}(F)$. $\psi$ and $S$ are both constructible in linear time.

The para-coNP-completeness follows from Proposition 12 and Proposition 22.

**Corollary 25.** Let $T$ be a non-empty set of temporal operators. Then LTL-MC($T$) is coNP-hard.

**Proof.** For $X \subseteq T$ we modify the reduction given in the proof of Theorem 24. Simply replace the subformula $FL_i$ by $X! L_i$ in the formula and the reduction stays valid. This substitution leads only to polynomial blowup. The other cases follow from Theorem 24 as $F \varphi \equiv \neg G \neg \varphi \equiv T U \varphi$.

The last results may be surprising at first sight. As LTL is easy on nonbranching structures one could expect that bounding the branching degree leads to an easy problem as well, but in fact already a branching of degree two is sufficient to express coNP-hard model properties.

**Theorem 26.** Let $T \subseteq \{X\}$. Then $(\text{LTL-MC}(T), \text{td}) \in \text{coW}[P]$.

**Proof.** Let a formula $\varphi \in LTL(X)$ and a structure $A$ with root $w$ be given. Now it holds that $(A, w) \not\models \varphi$ if and only if there is a path $\pi \in P(w)$ s.t. $(A, \pi) \not\models \varphi$. But this depends only on a finite prefix of $\pi$ that has length $\text{td}(\varphi)$. So to determine if the given formula is not satisfied by the structure, simply guess a finite path of length $\text{td}(\varphi)$ through $A$ and verify the formula. This requires at most $O(\text{td}(\varphi) \cdot \log |A|)$
The problem with F is that it can enforce neither order of fulfillment nor the length of a fulfilling prefix of a path. To achieve the desired effect, a rather large overhead in form of nested F operators is required. For paths π and π’ say that π’ is a j-suffix of π if there is an i ≥ j s.t. π’ = πi.

Consider the structure S with only one variable that was defined in Theorem 29. Modify it such that every world w_i^+, w_i^- and w_j has the variable p_even, labeled if i is even, and otherwise the variable p_odd. To create a matching LTL formula, first inductively define a shortcut operator Fi,j: Fi,j(α) holds on a path π ∈ Π(w_i^+) ∪ Π(w_j^-) if a j-suffix of π fulfills α. For this, we “skip” at least j worlds using the odd- and even-literals:

\[ F_{i,j}(α) := Fα \]
\[ F_{i,j+1}(α) := \begin{cases} 
F(p_{even} ∧ ¬p_{odd} ∧ F_{i+1,j}(α)) & \text{if } i \text{ is odd} \\
F(p_{odd} ∧ ¬p_{even} ∧ F_{i+1,j}(α)) & \text{if } i \text{ is even}
\end{cases} \]

At the same time we have to make sure that every F does not skip further on a path than to the immediate suffix path, i.e., the unique subpath that is a 1-suffix but not a 2-suffix. For this we use the fact that a path in S starts at w_m^+ or w_n^- exactly when it fulfills G_{p_{even}} (α is even) resp. G_{p_{odd}} (α is odd). In the rest of the proof, write p_{end} for the matching literal depending on n. Then a path π ∈ Π(w_i^+) ∪ Π(w_j^-) fulfills F_{i,j}(¬p_{end}) if after skipping j or more worlds it gets to a world which still has a proper successor in S. Thus F_{i,j}(¬p_{end}) means that there are at least j + 1 more steps between π[1] and a world with p_{end} labeled.

We aggregate the subexpressions into

\[ F_{=j}(α) := F_{0,j}(α ∧ F_{j,n−j−1}¬p_{end})) \text{ for } n > j \]
\[ F_{=n}(α) := F_{0,n}(α) \]

Due to the construction F_{=j}(α) is the desired formula; it states that for a path π ∈ Π(w_0) the expression α should hold in world π[j]: This is the only world on π which is reachable from w_0 with at least j steps but which still has distance at least n − j from the world π[n].

In the remainder we follow the proof of Theorem 29 but replace Xq by F_{=i}(q). Then again every path starting in w_0 fulfills the constructed ψ if and only if it visits the worlds in W = \{w_0^+, w_1^+, \ldots, w_m^+\} s.t. X = \{x_1, x_2, \ldots, x_m\} is a satisfying assignment for the original φ. Also the structure S and the formula ψ are again constructible in polynomial time.

The given reduction from the complement of 3SAT would likely not work with only one variable. The used “even-odd” trick is necessary since F-formulas are “compressible” in the sense that future in most temporal logics is reflexive and hence the present is a part of the future.

Also it seems that the linear temporal depth can be avoided neither for F nor for X. Therefore it is unlikely that the complement of 3SAT can offer para-coNP-hardness for model checking w.r.t. the parameterisation td + tw_φ. Therefore we
Definition 31 (Tiling). Let $C$ be a finite set of colors and $D \subseteq C^4$ a set of tiles. Every tile has four sides, namely up, down, left and right, which each have a color $c \in C$. Use the quadruple notation to explicitly write the colors of a tile: $d = (c_u, c_d, c_l, c_r)$. A $D$-tiling for a discrete area $R \subseteq \mathbb{N} \times \mathbb{N}$ is a function $\gamma : R \rightarrow D$.

Write $\gamma(x, y) = \langle (x, y)_u, (x, y)_d, (x, y)_l, (x, y)_r \rangle$. Then $\gamma$ is a valid tiling if for every $(x, y), (x', y') \in R$ holds:

- if $x' = x$ and $y' = y + 1$, then $(x, y)_d = (x', y')_u$,
- if $x' = x + 1$ and $y' = y$, then $(x, y)_r = (x', y')_l$.

Definition 32. The problem $\text{SQUARE$D$tiling}$ contains the instances $(C, D, \langle k \rangle_1)$ for which the $k \times k$-grid has a valid $D$-tiling. Here, $\langle k \rangle_1$ denotes a unary encoding.

Proposition 33 ([2]). Let $\kappa(C, D, \langle k \rangle_1) := k$. Then the parameterised problem $(\text{SQUARE$D$tiling}, \kappa)$ is W[1]-complete.

Theorem 34. (LTL-MC($X$), $td + pw_{\psi}$) and (LTL-MC($X$), $td + tw_{\psi}$) are coW[1]-hard.

Proof. The idea of the proof, a reduction from $\text{SQUARE$D$tiling}$ to the complement of the problems in the theorem, is to use the path semantics of LTL to describe a valid tiling of the $k \times k$ grid: For every $\text{SQUARE$D$tiling}$ instance $(C, D, \langle k \rangle_1)$ we construct a formula $\psi$ and structure $\mathcal{S}$. $\psi$ will have $k$-bounded temporal depth and structural width. The Kripke structure $\mathcal{S}$ however will have unbounded branching degree $\Delta$ (which is unlikely to be avoided as LTL-MC($X$) is already in FPT with parameter $td + \Delta$).

Construct $\mathcal{S}$ as follows:

- Add worlds $w_{\text{start}}$ and $w_{\text{end}}$ with the proposition $q_{\text{end}}$ labeled in $w_{\text{end}}$.
- For every tile $d \in D$ and for every $i \in [k^2]$ add a world $w_i^d$.
- Connect $w_{\text{start}}$ to $w_1^d$ for every $d \in D$.
- Connect $w_i^d$ to $w_{i+1}^d$ for every $d \in D$.
- Connect $w_i^d$ to $w_{i+1}^d$ for every $d, d' \in D$ and $1 \leq i < k^2$.
- Connect $w_{\text{end}}$ to itself.
- In every world $w_i^d$ with $d = (c_u, c_d, c_l, c_r)$ label propositional variables $c_{i,d}^u, c_{i,d}^d, c_{i,d}^l, c_{i,d}^r$.
- In every world $w_i^d$ where $i = k \cdot j$ for $j \in [k]$ label a propositional variable $q_{\text{border}}$.

The structure $\mathcal{S}$ is illustrated in Figure 3. It models (not necessarily valid) tilings $\gamma$ as runs from $w_{\text{start}}$ by “serializing” the square into a path: It contains $k$ worlds for the first row, another $k$ worlds for the second row appended to the first $w$ worlds, and so on to the $k$-th row, resulting in a total of $k^2$ worlds on each path (besides $w_{\text{start}}$ and $w_{\text{end}}$). At the same time, there are $|D|$ successors that a path can use to select the next tile in the current (or next) row: For every place $(i, j) \in [k] \times [k]$ a tile $d$ is selected by visiting the corresponding world $w_{d(i-1)+j}$.

Now we give the path formulas that verify that the tiling $\gamma$ described by a run $\pi \in \Pi(w_{\text{start}})$ is valid.

$$
\psi_{c,r} := [q_{\text{border}} \lor (c_r^i \rightarrow X(q_{\text{end}} \lor c_{i+1}^r))]
$$

$$
\psi_{c,d} := [c_d^i \rightarrow X^k (q_{\text{end}} \lor c_{i+k}^d)]
$$

The formula $\psi_{c,r}$ is true in a path $\pi$ starting in a world $w_1$ (which has color $c$ to the right) if $\pi$ chooses a matching successor: Either $w_1$ is a border and the color to the right is not relevant, or $w_1$ has $w_{\text{end}}$ as immediate successor and the tiling is complete, or the color matches the left color of the next tile.

Similar, the formula $\psi_{c,d}$ ensures that the tile directly below the current one (which lies in distance exactly $k$ in the structure) has the matching up color or is already beyond the last row.

We form the conjunction over every color $c$ and every $i$:

$$
\psi := \bigwedge_{i=1}^{k^2} \left[ X^i \bigwedge_{c \in C} (\psi_{c,r} \land \psi_{c,d}^i) \right]
$$

Claim. $\mathcal{S}$ and $\psi$ can be constructed in fpt-time.

Proof of claim. The structure $\mathcal{S}$ can be constructed in time $O(|D|^2 \cdot k^2)$ and the formula $\psi$ can be constructed in time $O(|C| \cdot k^3)$, which is both polynomial since $k$ is encoded unary in $\text{SQUARE$D$tiling}$.

Claim. Let $\pi = (w_{\text{start}}, w_1^d, w_2^d, \ldots, w_{k^2}^d, w_{\text{end}}, \ldots)$ be a run through $\mathcal{S}$. Then $\pi \models \psi$ if and only if $d_1, d_2, \ldots, d_k$ form a valid tiling of $[k] \times [k]$.

Proof of claim. “$\Rightarrow$”: Assume there are $(x, y)$ and $(x', y')$ such that the tiling conditions are violated.

- Case 1: $x' = x + 1$ and $y' = y$. Then $x \neq k$, $(x, y)$ has right color $c$ and $(x + 1, y)$ has left color $c' \neq c$. Let $i := (x-1) \cdot k + y$. By definition of $\pi$ it holds that $\pi[1] = w_{\text{start}}$ and $\pi[i+1] = w_{d_i}$. But as $w_1^d$ is not a border and $\pi_{2i+1} \models \psi_{c,r}$, the successor has $c$ as left color. But this means that $c_{i+1}^r$ is labeled in $w_{d_i}$ and $c_{i+1}^r$ is labeled in $w_{d_{i+1}}$, contradiction to $c' \neq c$.
- Case 2: $x' = x$ and $y' = y + 1$ which is similar proven as Case 1.

Figure 3. Structure that models square tilings as runs.
“⇐”: Let $d_1, d_2, \ldots, d_k$ be a valid tiling of $[k] \times [k]$. Assume that $\neg\psi$ holds, i.e., there is a color $c$ and an $i$ s.t.\ $\pi_{i \rightarrow i+1}$ does not satisfy $\psi_{c,r} \land \psi_{c,d}$. If $\psi_{c,r}$ is false, then $w^i_d$ is not a border but also has a right different color than its successor on $\pi$ has as left color. But then $\gamma$ would not be a valid tiling. The case that $\psi_{c,d}$ is false can be handled analogously.

It is easy to see that every run $\pi$ through $S$ from $w_{\text{start}}$ has the form as in the above claim, i.e., 

$\pi = \langle w_{\text{start}}, w^1_d, w^2_d, \ldots, w^n_d, w_{\text{end}}, \ldots \rangle$. Hence we get $(C, D, k) \in \text{SQUARETILING}$ if and only if $\exists \pi \in \Pi(w_{\text{start}}) : (S, \pi) \models \psi$ if and only if $(\neg\psi, S, w_{\text{start}}) \not\in \text{ILTL-MC}(X)$.

**Claim.** The formula $\psi$ has temporal depth $k^2 + k$ and structural pathwidth at most $2k^2 + k + 15$.

**Proof of claim.** The temporal depth of $k^2 + k$ is the nesting depth of $X$ operators in $\psi$.

For the pathwidth we construct a path-decomposition $P$ of $\psi$ as follows: For every $i \in [k^2]$ and every color $c \in C$ we create an isolated bag $B^i_c$. The bag $B^i_c$ contains the nodes representing

- the Boolean connectives $\lor, \land$ in $\psi_{c,r}$,
- the Boolean connectives $\lor, \land$ in $\psi_{c,d}$,
- the variables $q_{\text{border}}, q_{\text{end}}, c^1, c^{i+1}, c^k$ and $c^{i+k}$,
- the single $X$-operator in $\psi_{c,r}$,
- the $k$ $X$-operators in $\psi_{c,d}$.

The isolated bag covers every edge between nodes representing subformulas of $\psi_{c,r}$ and $\psi_{c,d}$ with a width of $|B^i_c| = 3 + 2 + 6 + 1 + k = k + 12$. Also every subformula of $\psi_{c,r}$ and $\psi_{c,d}$ except $q_{\text{border}}$ and $q_{\text{end}}$ occurs exactly once in $\psi$, hence every such subformula of $\psi$ trivially induces a connected path in $P$. But as $q_{\text{border}}$ and $q_{\text{end}}$ are added into every bag $B^i_c$ they also induce a connected path as soon as the bags are connected.

To handle the remaining connectives including the “big conjunctions” of size $|C|$, proceed as follows: For every formula $\xi^i_c := (\psi_{c,r} \land \psi_{c,d})$, add $\xi^i_c$ to $B^i_c$.

Assume that the colors are ordered as $c_1, c_2, \ldots, c|C|$, and that the big conjunctions have the structure $((\xi^1_c \land \xi^2_c) \land \xi^3_c) \ldots \land \xi^{|C|}_c$. For every $j, 1 \leq j < |C|$ then connect the bags $B^i_{c_j}$ and $B^i_{c_{j+1}}$ by inserting an edge in $P$, and add the $j$-th $\land$-node into both bags. Then after inserting the last conjunction, add the $i$ nodes for $X^i$ to $B^i_{c|C|}$, and finally add the nodes for the conjunction of size $k^2$ to every bag. These steps increase the size of every bag by at most $2k^2 + 3$.

As $P$ now consists of $k^2$ disconnected sequences of $|C|$ bags each, concatenate them into a path in arbitrary order. This leads to $P$ being a connected path; and the variables $q_{\text{border}}$ and $q_{\text{end}}$ as well as the nodes of the $k^2$-conjunction now induce connected subpaths.

From the claims a valid fpt-reduction follows and thereby we conclude the theorem.

**Theorem 35.** Let $F \in T$, $G \in T$ or $U \in T$. Then $(\text{LTL-MC}(T), \text{td} + \text{pw}_\varphi)$ and $(\text{LTL-MC}(T), \text{td} + \text{tw}_\varphi)$ are coW[1]-hard.

**Proof.** We adapt the reduction given in the proof of Theorem 34. First label a new depth proposition $d_i$ in every world $w^i_d$ for $d_i \in D$, $1 \leq i \leq k^2$. Then change the formulas as follows:

$\psi_{c,r} := \left[q_{\text{border}} \lor \left(c^i \lor F(c_{i+1}^i)\right)\right]$

$\psi_{c,d} := \begin{cases} \left[c^i \lor F(c_{i+k}^i)\right] & \text{if } i + k \leq k^2 \\ F & \text{otherwise} \end{cases}$

$\psi := \bigwedge_{i=1}^{k^2-1} \left(F \left(d_i \land \bigwedge_{c \in C} \left(\psi_{c,r} \land \psi_{c,d}\right)\right)\right)$

This does increase the pathwidth at most by $k^2$ for $d_1, \ldots, d_k$. Also $F \alpha$ can be replaced by $\neg G \neg \alpha \land T \neg U \alpha$ for the remaining cases.

**Definition 36.** The problem RECTANGLETILING contains the tuples $(C, c_0, c_1, D)$ for which there is an $m \in \mathbb{N}$ such that the $|D| \times m$-grid has a valid $D$-tiling $\gamma$ with the color $c_0$ at the top edge and $c_1$ at the bottom edge.

**Proposition 37 ([3]).** The problem RECTANGLETILING is PSPACE-complete.

**Theorem 38.** The problems $(\text{LTL-MC}(X, F), \text{pw}_\varphi)$ and $(\text{LTL-MC}(X, F), \text{tw}_\varphi)$ are para-PSPACE-complete.

**Proof.** We consider a $\leq_p$-reduction from RECTANGLETILING to LTL-MC($X, F$) such that only LTL formulas with constant pathwidth (and therefore constant treewidth) are produced. This proves the theorem according to Proposition 12 and Proposition 21. This reduction originally from Sistla and Clarke is to show the PSPACE-hardness of general LTL model checking. We modify it to obtain a constant pathwidth.

Write the shortcut $n := |D|$. Then similar to the proof of Theorem 34 we construct a Kripke structure $S$ that models $n \times m$-tilings as runs:

- Add worlds $w_{\text{left}}, w_{\text{right}}$ and $w_{\text{end}}$ which each have only one proposition labeled, namely $p_{\text{left}}, p_{\text{right}}$, and $p_{\text{end}}$.
- For every tile $d \in D$ and every $i \in [n]$ add a world $w_i^d$.
- Connect $w_{\text{left}}$ to $w^1_d$ for every $d \in D$.
- Connect $w^n_d$ to $w_{\text{right}}$ for every $d \in D$.
- Connect $w^d_i$ to $w^{d+1}_i$ for every $d, d' \in D$ and $1 \leq i < n$.
- Connect $w_{\text{right}}$ to $w_{\text{end}}$, $w_{\text{right}}$ to $w_{\text{left}}$, and $w_{\text{end}}$ to itself.
- In every world $w^d_{\text{end}}$ with $d = \langle c_a, c_d, c_l, c_r \rangle$ label propositional variables $c_a, c_d, c_l, c_r$.

The structure $S$ is shown in Figure 4 and models tilings as follows: A run $\pi$ starts in $w_{\text{left}}$ and visits a row of $n$ worlds $w^i_d$. These worlds are the first row of the tiling. In every of the $n$ steps, $\pi$ may decide for any of the $|D|$ possible successors (which correspond to tiles). The back edge from $w_{\text{right}}$ to $w_{\text{left}}$ may be used then an arbitrary number of times, constructing a tiling consisting of many rows. The path may then enter the state $w_{\text{end}}$ and stay there forever.

We use the following formulas to check if the tiling is valid. First ensure that the complete first row has up color $c_0$:

$\psi_{\text{first}} := \bigwedge_{i=1}^n X^i(c_0)_u$
Check the neighbor to the right and below (if it is not the border):

\[
\psi_{c,r} := [c_r \rightarrow X (q_{\text{right}} \lor c_l)] \\
\psi_{c,d} := [c_d \rightarrow X^{n+2} (q_{\text{end}} \land c_u)]
\]

The last row must exist and have down color \( c_1 \):

\[
\psi_{\text{last}} := F \left( q_{\text{left}} \land (X^{n+2} q_{\text{end}}) \land \bigwedge_{i=1}^{n} X^i (c_1)_d \right)
\]

The whole tiling is expressed by \( \psi \):

\[
\psi := \psi_{\text{first}} \land \psi_{\text{last}} \land \neg F \neg \bigcap_{c \in C} (\psi_{c,r} \land \psi_{c,d})
\]

Similar to Theorem 34 is the case that there is a valid \( n \times m \)-tiling if and only if a path starts in \( w_{\text{left}} \) and satisfies \( \psi \).

**Claim.** The formula \( \psi \) has constant structural pathwidth.

**Proof of claim.** We construct a path-decomposition \( \mathcal{P} \) of \( \psi \) as follows. Ignore the variables \((c_0)_u, (c_1)_d, q_{\text{left}}, q_{\text{right}}\) and \( q_{\text{end}} \) as they can be added to every bag at the end, increasing every bag size only by five.

Now first process the formula \( \psi_{\text{left}} \). It contains \( n \) “chains” of X-operators. For each such chain create a row of bags. For \( j = 1, \ldots, n-1 \) then add the \( j \)-th and the \( (j+1) \)-th X-operator node to the \( j \)-th bag, so the edge in the syntactical structure between them is covered. Then the big conjunction is handled as in the proof of Theorem 34, connecting the \( n \) rows of bags to a single path, increasing the width by at most two.

The formulas \( \psi_{c,r} \) have each constant length, so for every \( c \in C \) put all of the nodes of subformulas of \( \psi_{c,r} \) into a single bag and append it to \( \mathcal{P} \). This does not violate the path-decomposition rules as every variable \( c_r, c_l \) appears only once in the whole formula \( \psi \).

The length of \( \psi_{c,d} \) depends on \( n \), but the chain of \( n + 2 \) X-operators can be decomposed like in \( \psi_{\text{first}} \), leading to \( n + 2 \) new bags with each constant size.

The length of \( \psi_{\text{last}} \) is again not constant; it contains a big conjunction of chains of X-operators as well as another single chain with \( q_{\text{end}} \) inside. Decompose the chains and the big conjunction as in \( \psi_{\text{first}} \). The edges connecting them to the remaining number of constantly many \( \land \)-nodes and the \( F \) node can then be covered by adding the nodes to every bag, increasing the size only by a constant.

Finally for covering the whole formula \( \psi \) in \( \mathcal{P} \), we need to insert the remaining small \( \land \)-operators, negations, and the \( F \) into every bag; and to decompose the big conjunction for every color \( c \) append the bags of the \( \psi_{c,d} \) formulas in the right order, again adding the small conjunction parts of the big conjunction.

From the above claims and Proposition 21 the para-PSPACE-completeness follows.

**Theorem 39.** \((\text{LTL-MC}(U), td + pw_{tw})\) and \((\text{LTL-MC}(U), td + tw_{tw})\) are para-PSPACE-complete.

**Proof.** For the Until operator the reduction from the proof of Theorem 38 is possible in constant temporal depth. Similar to the proof of Theorem 35 adapt the structure \( \mathcal{S} \) and supplement the labeled color variables \( c_{u}, c_{d}, c_{l}, c_{r} \) by their depth-aware versions, i.e., \( c_{u}^{0}, c_{d}^{0}, c_{l}^{0}, c_{r}^{0} \) for \( 1 \leq i \leq n \).

Modify the formulas as follows:

\[
\begin{align*}
\psi_{\text{first}} & := [q_{\text{left}} \lor (c_u)_u] U q_{\text{right}} \\
\psi_{\text{last}} & := \neg F [q_{\text{left}} \land \neg (q_{\text{right}} \land c_l)] U q_{\text{end}} \\
\psi_{c,r} & := [c_r] U [c_r U [c_r U [c_r U [c_r U (q_{\text{end}} \land c_l)]]]] \\
\psi_{c,d} & := [c_d] U [\neg c_d U (q_{\text{end}} \land c_l)] \\
\psi & := \psi_{\text{first}} \land \psi_{\text{last}} \land \neg [F \neg \bigcap_{i=1}^{n} (\psi_{c,r}^{i} \land \psi_{c,d}^{i})]
\end{align*}
\]

The variables \((c_0)_u, (c_1)_d, q_{\text{left}}, q_{\text{right}}\) can again be added to every bag of a path-decomposition \( \mathcal{P} \) of \( \psi \). The only part of \( \psi \) that is not constant is the conjunction over the \( n \cdot |C| \) subformulas \( \psi_{c,r}^{i} \) and \( \psi_{c,d}^{i} \). But each such subformula \( \psi_{c,r}^{i} \), resp., \( \psi_{c,d}^{i} \) can be covered by a single isolated bag: It has only a constant number of nodes and every occurring variable is either subformula-local or is already added to every bag.

Then it remains to decompose the big conjunctions which can be done in two steps. First connect the isolated bags for the inner conjunction and add the small conjunction nodes as needed. Then connect the resulting chains of length \( |C| \) to finalize the path-decomposition \( \mathcal{P} \) that has a constant width.

**IV. Conclusion**

We showed that the intractable satisfiability and model checking problems of LTL cannot be tamed by applying the toolbox of parameterised complexity theory, at least not for the chosen well-known parameters. The model checking hardness is solely caused by the path semantics of LTL in branching Kripke structures. This conclusion holds both for the full set of LTL operators and for every operator fragment; the only exception is the case with only the \( X \) operator and a special parameterisation that allows a depth-bounded search tree algorithm. As LTL is a special case of CTL*, the hardness results immediately hold for CTL* as well.
A future research possibility is to continue the search for tractable parameters of LTL. The ultimate goal is to find a non-trivial parameter, i.e., lower than the one of Lichtenstein and Pnueli, that allows fixed-parameter tractability for all LTL operators. A first step could be LTL instances of simultaneously bounded formula treewidth and input structure treewidth.

ACKNOWLEDGMENT

The authors acknowledge the partial support by DFG grant ME 4279/1-1.

REFERENCES

[1] M. Bauland, M. Mundhenk, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The tractability of model checking for LTL: the good, the bad, and the ugly fragments. ACM Transactions on Computational Logic (TOCL), 12(2):13:1–13:28, January 2011.
[2] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. On the parameterized complexity of short computation and factorization. Archive for Mathematical Logic, 36:321–337, 1997.
[3] B. S. Chlebus. Domino-tiling games. Journal of Computer and System Sciences, 32(3):374–392, 1986.
[4] E. M. Clarke and E. A. Emerson. Desing and synthesis of synchronisation skeletons using branching time temporal logic. In Logic of Programs, volume 131 of Lecture Notes in Computer Science, pages 52–71. Springer Verlag, 1981.
[5] R. G. Downey and M. R. Fellows. Parameterized Complexity. Springer-Verlag, 1999.
[6] R. G. Downey and M. R. Fellows. Fundamentals of Parameterized Complexity. Springer, 2013.
[7] E. Allen Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences, 30(1):1–24, 1985.
[8] J. Flum and M. Grohe. Describing Parameterized Complexity Classes. In Helmut Alt and Afonso Ferreira, editors, STACS 2002, volume 2285 of Lecture Notes in Computer Science, pages 359–371. Springer Berlin Heidelberg, 2002.
[9] V. Goranko. Tutorial notes on introduction to temporal logics for specification and verification. First PhD Autumn School on Modal Logic, IT University of Copenhagen, 2009.
[10] O. Lichtenstein and A. Pnueli. Checking That Finite State Concurrent Programs Satisfy Their Linear Specification. In Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, pages 97–107, New York, NY, USA, 1985. ACM.
[11] M. Lück. Parameterized Complexity of Temporal Logics. Master’s thesis, Leibniz Universität Hannover, 2015.
[12] M. Lück, A. Meier, and I. Schindler. Parameterized Complexity of CTL: A Generalization of Courcelle’s Theorem. In Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France. Proceedings, volume 8977 of Lecture Notes in Computer Science, pages 549–560. Springer, 2015.
[13] A. Meier, M. Mundhenk, M. Thomas, and H. Vollmer. The Complexity of Satisfiability for Fragments of CTL and CTL*. International Journal of Foundations of Computer Science, 20(5):901–918, 2009.
[14] A. Meier, J.-S. Müller, M. Mundhenk, and H. Vollmer. Complexity of Model Checking for Logics over Kripke models. Bulletin of the EATCS, 108, 2012.
[15] M. Praveen. Does Treewidth Help in Modal Satisfiability? ACM Transactions on Computational Logic (TOCL), 14(3):18:1–18:32, 2013.
[16] A. N. Prior. Time and Modality. Clarendon Press, Oxford, 1957.
[17] M. Samer and S. Szeider. A Fixed-Parameter Algorithm for #SAT with Parameter Incidence Treewidth. CoRR, abs/cs/0610174, 2006. URL http://arxiv.org/abs/cs/0610174.
[18] P. Schnoebelen. The Complexity of Temporal Logic Model Checking. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, Advances in Modal Logic, pages 393–436. King’s College Publications, 2002.
[19] A. Sistla and E. Clarke. The complexity of propositional linear temporal logics. Journal of the ACM, 32(3):733–749, 1985.
[20] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. Journal of the ACM, 32(3):733–749, 1985.
[21] S. Szeider. On fixed-parameter tractable parameterizations of SAT. In Proceedings of the 16th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003), volume 2919 of Lecture Notes in Computer Science, pages 188–202, 2004.