Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Theoretische Informatik

# Parameterized Complexity of Temporal Logics

**Masterarbeit**

im Studiengang Informatik

von

**Martin Lück**

**Hannover, 9. März 2015**

## Abstract

This thesis aims at classifying the temporal logics CTL, LTL and CTL$^*$ in terms of parameterized complexity of their satisfiability and model checking problems. The studied parameters are combinations of temporal depth and syntactical treewidth of formulas. The classification is done fragment-wise and focuses on the question of tractability, i.e. for each fragment of temporal operators either an fpt algorithm or a hardness result is presented.

For the related modal logic FPT results are known for a similar parameterization by modal depth and syntactical treewidth. This thesis shows that the parameterized tractability of modal satisfiability cannot be transferred for any temporal operator but $\mathsf{X}$ and that the model checking problems of LTL and CTL$^*$ fragments stay inherently intractable under the given parameterizations.

Furthermore it is shown that the chosen parameters are both crucial for the tractability of satisfiability with $\mathsf{X}$, except for LTL where treewidth is sufficient due to the lack of branching.

## Zusammenfassung

Die vorliegende Arbeit klassifiziert die Erfüllbarkeits- und Model-Checking-Probleme der Temporallogiken CTL, LTL und CTL$^*$ hinsichtlich deren parametrisierter Komplexität. Untersuchte Parameter sind Kombinationen aus temporaler Tiefe und syntaktischer Baumweite der Eingabeformeln. Die Klassifizierung wird für alle Fragmente bezüglich der temporalen Operatoren mit Fokus auf effizienter Lösbarkeit durchgeführt, die Resultate sind daher Nachweise von FPT-Algorithmen oder Schwere-Resultate.

Für die ähnliche Modallogik sind bereits Ergebnisse für modale Tiefe und syntaktische Baumweite als Parametrisierung bekannt. Diese Arbeit zeigt, dass die parametrisierte effiziente Lösbarkeit des Erfüllbarkeitsproblems einzig für den Operator $\mathsf{X}$ übertragen werden kann und dass Model-Checking für alle Fragmente von LTL und CTL$^*$ auch derart parametrisiert inhärent schwierig bleibt.

Außerdem wird gezeigt, dass die Parameter temporale Tiefe und Baumweite beide notwendig für die effiziente Prüfung der Erfüllbarkeit mit $\mathsf{X}$ sind; lediglich bei LTL reicht die Baumweite als Parameter aus, da LTL keine Verzweigungssemantik besitzt.

# Contents

# 1 Introduction

## Motivation

Formal verification of software or hardware, i.e., proving the correctness of the behaviour of an algorithm or a dynamic system, gains importance as computational tasks become more and more complex and systems become more and more difficult to manage. This is especially true for critical systems or infrastructures where informal verification processes are likely to suffer from the vagueness and ambiguity of natural language. Typical examples for automatically verifiable systems are cryptographic protocols, circuit designs or control flows of structured programming languages.

Next to verification by proof there are various techniques like simulation, manual reviewing or testing. Yet they suffer from not being scalable, being expensive, or being inherently incomplete in the sense that they can reveal most errors but cannot prove the absence of them. On the other hand, a formal description of a fault-intolerant system can be verified thoroughly not only by hand but also by automatic theorem provers and model checkers.

Suited logical frameworks as a tool for modeling are for instance classical propositional logic or, if dealing with dynamic systems, variants of *modal logic*. Although the problem of model checking for certain logics can be feasibly done by computers, this is not the case for every logic. At the same time, the problems of *consistency* and *tautology* are already computationally hard for propositional logic and thus even harder for its modal extensions.

The theory of computational complexity provides tools to classify algorithmic problems into different degrees of (in-)tractability. Still problems that are hard in theory, i.e., require superpolynomial time to compute, are solved in practice more or less efficiently due to the use of heuristics or by being restricted to "nice" problem instances. Reflecting this development in practice and allowing a more fine-grained tractability measure, the technique of *parameterization* of a computational problem allows a much deeper insight into the properties of input instances that are the source of inherent hardness. An analysis of such a problem can provide a theoretical support for the practical efficiency, but can also itself find such a property, a *parameter*, which makes the problem easy to solve when it is itself not too complex.

Similar to classical complexity theory its offspring, parameterized complexity theory, can be used to determine computational intractability with respect to a certain parameter. In analogy to the classical case, problems can be arranged into a hierarchy of complexity classes where certain classes are considered as tractable and others as intractable. The difference between tractability (in the sense of deterministic polynomial time) and intractability is not fully understood yet even in classical complexity theory. Nevertheless, it is desirable to fully classify seemingly sensible choices of parameters of a problem based on the classical toolbox of reductions and complete problems.

## Related work

This thesis aims at classifying certain parameterizations of the model checking and satisfiability problems of temporal logics, continuing work that has been partially done for modal and temporal satisfiability. After introducing the reader to relevant foundations of parameterized complexity theory and logic, we will see a complete classification of temporal logics in the sense of *fragments*, i.e., logics with a restricted set of allowed temporal operators. A similar classification of temporal logic was thoroughly done for classical complexity by Meier [Mei11]. He showed that even the satisfiability problem, considered the "harder" one, quickly drops into realms of negligible complexity when restricted to e.g. monotone Boolean expressions, while the temporal fragments range from **NP**-complete over **PSPACE**- and **EXP**- to **2EXP**-complete. He therefore concluded that the inherent hardness of temporal logic beyond **NP** emerges from the different non-Boolean, temporal operators.

It is sure that for restricting the Boolean connectives has a high price in terms of expressivity. Hence in the world of parameterization it seems more interesting to know the consequences of restricting the temporal operators. Is there a similar sharp complexity gradient like for the Boolean connectives? Since the different temporal operators form a strict hierarchy of expressiveness [Lar95], it is safe to assume that they also differ in computational hardness in some way.

Next to Meier, the work of Praveen [Pra13] has been fundamental for this study. He successfully applied parameterization to modal logic satisfiability on various classes of Kripke frames, but shows that for these parameters the satisfiability prolem is *harder* when restricted to transitive frames. This phenomenon reappears in the classification of temporal satisfiability as an inherent source of computational hardness. Certain tractability and hardness results regarding *computation tree logic (CTL)* fragments were developed in parallel [LMS15].

# 2 Preliminaries

Logical expressivity and computational complexity are close relatives which is an insight that increasingly emerges in theoretical computer science. In this preliminary chapter we remind the reader of various definitions regarding complexity theory and logic which provide the groundwork for the presented results.

## 2.1 Complexity theory

We use the standard notation of *Turing machines (TMs)* to formalize computation. A Turing machine is an abstract automaton that receives an *input string* encoded over a finite, non-empty alphabet $\Sigma$, then executes a number of computation steps until it eventually halts, leaving an *output string* on its tape. It may be a *deterministic* Turing machine (DTM) or a *non-deterministic* one (NTM).

We call a *string* or *word* over $\Sigma$ any sequence $x = (x_1, x_2, \dots)$ where $x_i \in \Sigma$ f. a. $i \in \mathbb{N}$. If not explicitly stated otherwise, we assume a word to be finite, denoting with $|x|$ the length of the sequence. We write $\Sigma^*$ for the set of all finite words over the alphabet $\Sigma$.

**Definition 2.1** (Computable functions)**.** Let $f : \Sigma^* \to \Sigma^*$ be a function. We say that a deterministic TM $M$ *computes the function* $f$ if for every $x \in \Sigma^*$ the machine $M$ eventually halts on input $x$ with output $f(x)$. If $f$ is computed by a Turing machine we call $f$ *computable*.

**Definition 2.2** (Decision problems)**.** A set $A \subseteq \Sigma^*$ is called a *language* or a *decision problem over the alphabet* $\Sigma$. We say that a TM $M$ *decides a language* $A$ if for every $x \in A$ the machine $M$ has a computation that eventually halts in a special state $q_{\mathrm{acc}}$, and if for every $x \notin A$ the machine $M$ eventually halts in a state that is not $q_{\mathrm{acc}}$ for every computation. The machine $M$ may be non-deterministic. For a language $A$ we write $\overline{A}$ for the *complement* of $A$, i.e. the set of all strings not in $A$. A language $A$ that is decided by a TM is called *decidable*.

### Classical complexity theory

**Definition 2.3** (Landau notation)**.** Let $f : \mathbb{N} \to \mathbb{N}$. Then $\mathcal{O}(f)$ is the class of functions $g$ that are bounded by $c \cdot f$ for a constant $c$, i.e. there is a fixed $n_0$ s. t. $g(n) \leq c \cdot f(n)$

for every $n \geq n_0$.

For function classes $F, G$ write $F + G$ for $\{ f + g \mid f \in F, g \in G \}$ and similar for $F \cdot G$, $F^G$, $\mathcal{O}(F)$. Also, write $\log F$ for $\left\{ \log_b f \mid b > 1, f \in F \right\}$.

**Definition 2.4** (Time and space complexity)**.** Let $M$ be a TM. If any computation of $M$ on a word $x$ has at most $t(|x|)$ steps we say *$M$ runs in time $t$*. Similarly, *$M$ runs in space $s$* if the length of every configuration in a computation on $x$ is at most $s(|x|)$. Also, $M$ runs in time (space) $F$ for a function class $F$ if $M$ runs in time (space) $f$ for any $f \in F$.

For the rest of the thesis we fix the alphabet $\Sigma := \{0, 1\}$, i.e. a binary encoding, to encode natural numbers, graphs, formulas etc. This does neither affect the definition of the relevant complexity classes nor the validity of the results. The binary encoding will be denoted $\langle \cdot \rangle$ when explicitly mentioned and omitted otherwise.

For a function or function class $f$ define the complexity classes

$$\begin{aligned}
\mathbf{TIME}(f) &:= \{ A \subseteq \Sigma^* \mid A \text{ is decided by a DTM in time } \mathcal{O}(f) \}, \\
\mathbf{NTIME}(f) &:= \{ A \subseteq \Sigma^* \mid A \text{ is decided by an NTM in time } \mathcal{O}(f) \}, \\
\mathbf{SPACE}(f) &:= \{ A \subseteq \Sigma^* \mid A \text{ is decided by a DTM in space } \mathcal{O}(f) \}.
\end{aligned}$$

We use the following symbols to refer to well-known classes:

$$\begin{aligned}
\mathbf{P} &:= \mathbf{TIME}\left(n^{\mathcal{O}(1)}\right) & \mathbf{EXP} &:= \mathbf{TIME}\left(2^{n^{\mathcal{O}(1)}}\right) \\
\mathbf{NP} &:= \mathbf{NTIME}\left(n^{\mathcal{O}(1)}\right) & \mathbf{2EXP} &:= \mathbf{TIME}\left(2^{2^{n^{\mathcal{O}(1)}}}\right) \\
\mathbf{PSPACE} &:= \mathbf{SPACE}\left(n^{\mathcal{O}(1)}\right)
\end{aligned}$$

The term *co-$\mathcal{C}$* for a complexity class $\mathcal{C}$ refers to the complexity class which contains the complements of the problems in $\mathcal{C}$.

**Definition 2.5** (Reduction)**.** For problems $A, B$ a *$\leq_m^P$-reduction* or *polynomial time reduction* from $A$ to $B$ is a function $f$ that is computable in polynomial time such that for all $x$ it holds $x \in A \Leftrightarrow f(x) \in B$. Write $A \leq_m^P B$ if there is a *$\leq_m^P$-reduction* from $A$ to $B$. If $f$ is computable in logarithmic space, $f$ is a *log-space-reduction* from $A$ to $B$, in symbols $A \leq_m^{\log} B$.

**Definition 2.6** (Completeness)**.** A problem $A$ is *$\mathcal{C}$-hard* under *$\leq_m^P$-reductions* for a complexity class $\mathcal{C}$ if and only if every $B \in \mathcal{C}$ is $\leq_m^P$-reducible to $A$. If additionally $A \in \mathcal{C}$ then $A$ is *$\mathcal{C}$-complete*.

The term *hard* resp. *complete under $\leq_m^{\log}$-reductions* is defined analogously.

## Parameterized complexity theory

The following definitions were developed by Downey and Fellows and later by Flum and Grohe who were the first to systematically lay ground for a theory that can speak about parameterized intractability [FG06, DF99].

**Definition 2.7** (Parameterized problem)**.** Let $Q \subseteq \Sigma^*$ be a decision problem and let $\kappa \colon \Sigma^* \to \mathbb{N}$ be a computable function. Then we call $\kappa$ a *parameterization of* $Q$ and the pair $\Pi = (Q, \kappa)$ a *parameterized problem*.

**Definition 2.8** (Fixed-parameter tractable)**.** Let $\Pi = (Q, \kappa)$ be a parameterized problem. If there is a DTM $M$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ s. t. for every instance $x \in \Sigma^*$

- $M$ decides correctly if $x \in Q$ and
- $M$ has a runtime bounded by $f(\kappa(x)) \cdot |x|^{\mathcal{O}(1)}$

then we say that $M$ is an *fpt-algorithm for* $\Pi$ and that $\Pi$ is *fixed-parameter tractable*. We define **FPT** as the class of all parameterized problems that are fixed-parameter tractable.

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

*Remark:* Some authors demand that given an instance $x$ the parameter $\kappa(x)$ itself is efficiently computable, i.e. in polynomial time [FG06]. Such a restriction would prevent parameterizations like the *treewidth of graphs* which lead to interesting **FPT** results when allowed as parameter, but the existence of a polynomial time algorithm to compute their explicit value would imply $\mathbf{P} = \mathbf{NP}$. Nevertheless there are practically relevant classes of e.g. graphs with low treewidth. Flum and Grohe recommend to consider the parameter value as a part of the input to ensure its polynomial time computability.

One could also restrict the available parameters $\kappa$ to “reasonably” computable functions, an adequate upper bound could be an *fpt*-like runtime, i.e. $\kappa$ is *fpt*-computable w. r. t. itself. This still gives most of the relevant theoretical results and allows the treewidth parameter. Another possibility is to restrict the parameter size [CE12].

Besides **FPT** there is a number of important classes in theory of parameterized complexity. If we consider **FPT** as the class of “tractable” problems then the class called $\mathbf{W}[1]$ is a good candidate for the “smallest well-known intractable class” in the parameterized world.

A formal definition of intractability however first requires the concept of an *fpt-reduction* which is very similar to a reduction in the world of classical complexity.

**Definition 2.9** (fpt-reduction)**.** Let $(P,\kappa)$ and $(Q,\lambda)$ be parameterized problems over alphabets $\Sigma$ resp. $\Delta$. Then a function $f : \Sigma^* \to \Delta^*$ is an *fpt-reduction* if it is fpt-computable w. r. t. $\kappa$ and there is a computable $h\colon \mathbb{N} \to \mathbb{N}$ s. t. the following holds f. a. $x \in \Sigma^*$:

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

If there is an fpt-reduction from $(P,\kappa)$ to $(Q,\lambda)$ for pairs $(P,\kappa)$ and $(Q,\lambda)$ then we call $(P,\kappa)$ *fpt-reducible to* $(Q,\lambda)$, denoted by $(P,\kappa) \leq^{fpt} (Q,\lambda)$.

From the previous definitions it follows that **FPT** is closed under fpt-reductions, i.e. if $(P,\kappa) \leq^{fpt} (Q,\lambda)$ and $(Q,\lambda) \in \mathbf{FPT}$ then $(P,\kappa) \in \mathbf{FPT}$ follows. Note that this even holds in cases where the reduction causes a superpolynomial blow-up on the instance size.

**Definition 2.10** (**W** hierarchy)**.** Let $t \in \mathbb{N}$. We define *the $t$-th level of the* **W** *hierarchy* as

$$\mathbf{W}[t] := \left\{\, (P,\kappa) \;\middle|\; (P,\kappa) \leq^{fpt} \text{p-}\textsc{CircSat}(t,d), d \geq t \,\right\},$$

where p-$\textsc{CircSat}(t,d) = (Q,\lambda)$ is the following parameterized problem:

$$Q := \left\{ (C,k) \;\middle|\; \begin{array}{l} C \text{ is a Boolean circuit with weft } t \text{ and depth } d \\ \text{that has a satisfying assignment of weight } k \end{array} \right\}, \quad \lambda(C,k) := k$$

For the fundamental concepts of Boolean circuits and circuit complexity the reader is referred to Vollmer [Vol99]. We skip the foundations here since there are no following definitions or results directly related to circuits. Here, the *weft* of a Boolean circuit can simply be defined as the maximal number of nodes with input arity $> 2$ from any straight path from an input node to the output node. The *depth* is the maximal number of arbitrary nodes on such paths. The *weight* of an assignment is the number of input bits set to one, and an assignment is *satisfying* if the output node of the circuit produces a one for the given input, similar to propositional formulas.

**Definition 2.11** (Parameterized hardness)**.** A problem $(P,\kappa)$ is *$\mathcal{C}$-hard under fpt-reductions* for a parameterized complexity class $\mathcal{C}$ if $(Q,\lambda) \in \mathcal{C}$ implies $(Q,\lambda) \leq^{fpt} (P,\kappa)$.
If additionally $(P,\kappa) \in \mathcal{C}$, we say that $(P,\kappa)$ is *$\mathcal{C}$-complete under fpt-reductions*.

Not only the classes $\mathbf{W}[t]$ are closed under fpt-reductions by definition, it also holds that $\mathbf{W}[t]$-hardness is inherited along reductions. That is if $(P,\kappa) \leq^{fpt} (Q,\lambda)$ and $(P,\kappa)$ is $\mathbf{W}[t]$-hard then $(Q,\lambda)$ is also $\mathbf{W}[t]$-hard.

**Definition 2.12** (**XP**)**.** The class **XP** contains the parameterized problems $(Q,\kappa)$ for which there is a DTM deciding $x \overset{?}{\in} Q$ in time $\mathcal{O}\left(|x|^{f(\kappa(x))}\right) + f(\kappa(x))$ for a computable function $f$.

**Definition 2.13** (**W**[**P**])**.** The class **W**[**P**] contains the parameterized problems $(Q,\kappa)$ for which there is a computable function $f$ and an NTM deciding $x \overset{?}{\in} Q$ in time $f(\kappa(x)) \cdot |x|^{\mathcal{O}(1)}$ with at most $\mathcal{O}(\kappa(x) \cdot \log|x|)$ non-deterministic steps.

Flum and Grohe state how to obtain parameterized variants of classical complexity classes [FG02]. They define for "standard" complexity classes $\mathcal{C}$ the corresponding parameterized versions para-$\mathcal{C}$. Here, "standard" means that the class $\mathcal{C}$ is defined via usual resource-restricted Turing machines (read *resource* as time resp. space).

For most such classes $\mathcal{C}$ we obtain para-$\mathcal{C}$ 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 *robust*, such as **NP** and **PSPACE**.[1] This allows the following definitions:

**Definition 2.14** (para-**NP**)**.** The class para-**NP** contains the parameterized problems $(Q,\kappa)$ for which there is a computable function $f$ and an NTM deciding $x \overset{?}{\in} Q$ in time $f(\kappa(x)) \cdot |x|^{\mathcal{O}(1)}$.

**Definition 2.15** (para-**PSPACE**)**.** The class para-**PSPACE** contains the parameterized problems $(Q,\kappa)$ for which there is a computable function $f$ and a DTM deciding $x \overset{?}{\in} Q$ in space $f(\kappa(x)) \cdot |x|^{\mathcal{O}(1)}$.

**Definition 2.16** (Slice)**.** The $l$-th *slice* of a parameterized problem $(Q,\kappa)$ is denoted $(Q,\kappa)_l$ and defined as:

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

**Theorem 2.17** (Flum and Grohe, 2002 [FG02])**.** *Let $(Q,\kappa)$ be a parameterized problem, $Q \subsetneq \Sigma^*$, $Q \neq \emptyset$. Then $(Q,\kappa)$ is para-**NP**-hard (resp. para-**PSPACE**-hard) if and only if a union of finitely many slices of $(Q,\kappa)$ is **NP**-hard (resp. **PSPACE**-hard).*

It holds that **FPT** $=$ **W**[0] $\subseteq$ **W**[1] $\subseteq$ **W**[2] $\subseteq \cdots \subseteq$ **W**[**P**] $\subseteq$ **XP** and as well **W**[**P**] $\subseteq$ para-**NP** $\subseteq$ para-**PSPACE** (see Figure 2.1).

When judging the "tractability" of a parameterized problem, one usually assumes strictness of all inclusions above, similar to the relationship between **P** and **NP**, and therefore aims at proving **FPT**-membership or (at least) **W**[1]-hardness.

[1]But not for e.g. **LOGSPACE**.

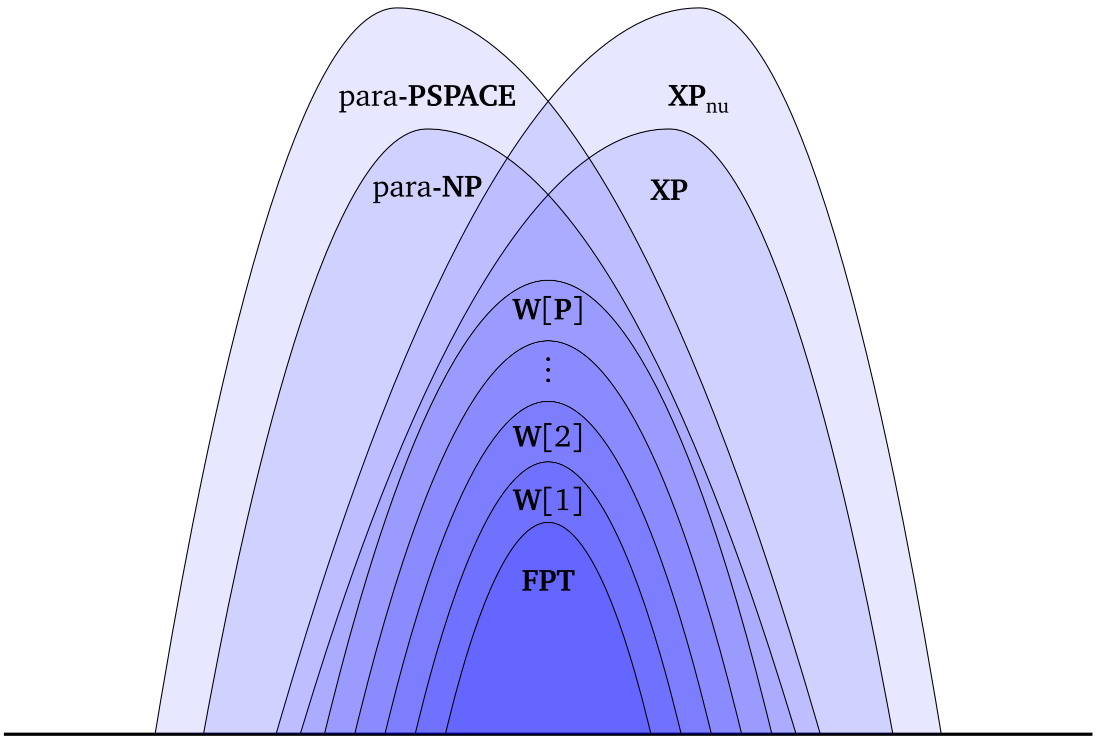


Figure 2.1: Some parameterized complexity classes

## 2.2 Logic

### Propositional logic

The classical propositional logic is attained by combining *propositional statements* with *Boolean connectives*. We follow the standard syntactic construction by giving a grammar in its Backus-Naur form,

$$\varphi ::= x \mid \top \mid \varphi \wedge \varphi \mid \neg\varphi$$

and defining the formulas $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$ and $\bot := \neg\top$ as shortcuts. Here, $x$ can generate an arbitrary propositional variable. $\mathbf{PS} = \{x_1, x_2, \ldots\}$ is the countable set of all variables or *atomic propositions*. **PL** is the set of all propositional formulas.

The semantics of truth given an assignment is inductively defined as usual.

### Modal logic

A formula in *modal logic* has a grammar slightly different from propositional logic:

$$\varphi ::= x \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box\varphi$$

The dual modal operator $\Diamond$ is also defined as a shortcut: $\Diamond\varphi := \neg\Box\neg\varphi$. We denote with **ML** the set of all modal formulas.

Evaluation of a modal formula does not take place with respect to a single assignment, but rather to a logical structure of *worlds* or *states*. This allows to formalize a multi-state system in logic while speaking about *successors* of states using formulas consisting of $\Box$ or $\Diamond$.

**Definition 2.18** (Kripke frame and structure)**.** A *Kripke frame* $\mathcal{F}$ is a finite structure $\mathcal{F} = (W, R)$ consisting of a finite set $W$ of *worlds* and a binary relation $R \subseteq W \times W$. A *Kripke structure* is a tuple $\mathcal{M} = (W, R, V)$ s. t. $(W, R)$ is a Kripke frame and $V\colon \mathbf{PS} \to \mathfrak{P}(W)$, i.e. every atomic proposition is mapped to a set of worlds. We say that the proposition is *labeled* in these worlds or *holds* in these worlds.

**Definition 2.19** (Kripke semantics)**.** Now we can evaluate the *truth* of modal formulas on Kripke structures. We say that a pair $(\mathcal{M}, w)$ of a Kripke structure $\mathcal{M} = (W, R, V)$ and a world $w \in W$ *models* a modal formula $\varphi$, in symbols $(\mathcal{M}, w) \models \varphi$, when it fulfills an inductive definition:

$$\begin{array}{lll}(\mathcal{M},w)\models x & \text{for } x\in\mathbf{PS} & \text{iff. } w\in V(x)\\ (\mathcal{M},w)\models\top & & \text{always}\\ (\mathcal{M},w)\models\neg\varphi & & \text{iff. } (\mathcal{M},w)\not\models\varphi\\ (\mathcal{M},w)\models\varphi\wedge\psi & & \text{iff. } (\mathcal{M},w)\models\varphi \text{ and } (\mathcal{M},w)\models\psi\\ (\mathcal{M},w)\models\Box\varphi & & \text{iff. } \forall w'\in W: (w,w')\in R\Rightarrow(\mathcal{M},w')\models\varphi\end{array}$$

For any Kripke structure $\mathcal{M}$ together with a world $w$ of $\mathcal{M}$ we call $(\mathcal{M},w)$ a *model* of $\varphi$ if $(\mathcal{M},w)\models\varphi$. If a modal formula $\varphi\in\mathbf{ML}$ has a model then it is *satisfiable*.
The corresponding decision problem is defined as follows:

$$\text{ML-SAT}=\left\{\varphi\ \middle|\ \begin{array}{l}\text{there is a Kripke structure } \mathcal{M}=(W,R,V)\\ \text{and } w\in W \text{ s.t. } (\mathcal{M},w)\models\varphi\end{array}\right\}$$

## Logical structures

**Definition 2.20** (Vocabulary). Define a *vocabulary* $\tau$ as a tuple $\tau=(\overline{R},\overline{f},\overline{C})$, where $\overline{R}=(R_1,R_2,\dots)$ is a sequence of relation symbols, $\overline{f}=(f_1,f_2,\dots)$ is a sequence of function symbols and $\overline{C}=(c_1,c_2,\dots)$ is a sequence of constant symbols. Every relation symbol $R$ and function symbol $f$ has a fixed arity denoted by ar$(R)$ resp. ar$(f)$. If not explicitly stated otherwise, we assume any vocabulary as finite. If a vocabulary $\tau$ consists only of relation symbols, we call $\tau$ *purely relational*.

**Definition 2.21** (Logical structure). A *logical structure* or a $\tau$*-structure* over the finite vocabulary $\tau=(R_1,\dots R_n,f_1,\dots,f_m,c_1,\dots,c_o)$ is a tuple

$$\mathcal{A}=(A,R_1^{\mathcal{A}},\dots R_n^{\mathcal{A}},f_1^{\mathcal{A}},\dots,f_m^{\mathcal{A}},c_1^{\mathcal{A}},\dots,c_o^{\mathcal{A}})$$

such that

- $A$ is a non-empty set, the *universe* or *domain* of $\mathcal{A}$,
- $R_i^{\mathcal{A}}\subseteq A^{\mathrm{ar}(R_i)}$ is a relation on $A$ for all $i\in\{1,\dots,n\}$,
- $f_i^{\mathcal{A}}$ is a function $A^{\mathrm{ar}(f_i)}\to A$ for all $i\in\{1,\dots,m\}$,
- $c_i^{\mathcal{A}}\in A$ for all $i\in\{1,\dots,o\}$.

If $\tau$ is purely relational, call $\mathcal{A}$ a *relational structure*.

In the rest of this thesis we use only finite structures with finite domains. When encoding a finite relational structure as an input for an algorithm we choose a representation that first contains the list of individuals followed by a list of tuples contained in the respective relations. The encoding size $|\mathcal{A}|$ of $\mathcal{A} = (A, \tau^{\mathcal{A}})$ is therefore

$$|\mathcal{A}| := |\tau| + |A| + \sum_{i=1}^{n} \mathrm{ar}(R_i) \cdot (|R_i^{\mathcal{A}}| + 1).$$

## Monadic second-order logic

The language of formulas one is usually working with when dealing with logical structures is *first-order logic ($\mathcal{FO}$)*. It allows to use relations, functions and constants as well as to quantify individuals from the domain of the structure. *Second-order logic ($\mathcal{SO}$)* additionally allows to quantify over relations of individuals. If the quantified relations are restricted to be unary, i.e. sets, the resulting logic is called *monadic second-order logic ($\mathcal{MSO}$)*.

**Definition 2.22** (MSO syntax)**.** Its syntax is defined

$$\varphi ::= \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \forall x \varphi \mid \forall U \varphi \mid R(\underbrace{t, \ldots, t}_{\mathrm{ar}(R)\text{ times}})$$

where $U$ stands for unary relation symbols. Here, $t$ denotes a *term* and $R$ is the binary relation "=" or a relation from the structure. Define the shortcuts $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\exists x \varphi := \neg\forall x \neg\varphi$ and $\exists U \varphi := \neg\forall U \neg\varphi$.

Terms are syntactically defined by

$$t ::= c \mid f(\underbrace{t, \ldots, t}_{\mathrm{ar}(f)\text{ times}}) \mid x$$

for constant symbols $c$, function symbols $f$ and variables $x$.

Write **MSO** for the set of all syntactically valid MSO formulas.

**Definition 2.23** (MSO semantics)**.** The evaluation of such a formula is an easy extension of the standard semantics of first-order logic formulas. To handle assignments to variables, we use an *interpretation* $\mathcal{I}$ of individual and set variables. Interpretations can be seen as a generalization of assignments to Boolean variables. They allow to inductively define the evaluation of a formula $\varphi \in \mathbf{MSO}$ over a structure $\mathcal{A}$:

$$
\begin{array}{ll}
(\mathcal{A},\mathfrak{I}) \models \top & \text{always} \\
(\mathcal{A},\mathfrak{I}) \models \neg\varphi & \text{iff. } (\mathcal{A},\theta) \not\models \varphi \\
(\mathcal{A},\mathfrak{I}) \models \varphi \wedge \psi & \text{iff. } (\mathcal{A},\mathfrak{I}) \models \varphi \text{ and } (\mathcal{A},\mathfrak{I}) \models \psi \\
(\mathcal{A},\mathfrak{I}) \models \forall x \varphi & \text{iff. } (\mathcal{A},\mathfrak{I}[x/a]) \models \varphi \text{ for all } a \in A \\
(\mathcal{A},\mathfrak{I}) \models \forall U \varphi & \text{iff. } (\mathcal{A},\mathfrak{I}[U/R]) \models \varphi \text{ for all } R \subseteq A \\
(\mathcal{A},\mathfrak{I}) \models R(t_1,\dots,t_{\mathrm{ar}(R)}) & \text{iff. } (t_1^{(\mathcal{A},\mathfrak{I})},\dots,t_{\mathrm{ar}(R)}^{(\mathcal{A},\mathfrak{I})}) \in R^{(\mathcal{A},\mathfrak{I})} \\
R^{(\mathcal{A},\mathfrak{I})} := \begin{cases} \mathfrak{I}(R) & \text{if } \mathfrak{I}(R) \text{ is defined} \\ R^{\mathcal{A}} & \text{else,} \end{cases} &
\end{array}
$$

where an *extension* of $\mathfrak{I}$ is defined as

$$
\mathfrak{I}[x/a](y) := \begin{cases} a & \text{if } x = y \\ \mathfrak{I}(y) & \text{else} \end{cases} \qquad \mathfrak{I}[U/R](V) := \begin{cases} R & \text{if } U = V \\ \mathfrak{I}(V) & \text{else} \end{cases} .
$$

Terms are recursively evaluated to a single element of $A$ as follows:

$$
\begin{array}{lll}
c^{(\mathcal{A},\mathfrak{I})} & := c^{\mathcal{A}} & \text{for a constant symbol } c \\
f(t_1,\dots,t_{\mathrm{ar}(f)})^{(\mathcal{A},\mathfrak{I})} & := f^{\mathcal{A}}(t_1^{(\mathcal{A},\mathfrak{I})},\dots,t_{\mathrm{ar}(f)}^{(\mathcal{A},\mathfrak{I})}) & \text{for a function symbol } f \\
x^{(\mathcal{A},\mathfrak{I})} & := \mathfrak{I}(x) & \text{for a variable } x
\end{array}
$$

A structure $\mathcal{A}$ models $\varphi$, i.e. $\mathcal{A} \models \varphi$, if and only if $(\mathcal{A},\emptyset) \models \varphi$.

**Example 2.24:**
Let $G = (V,E)$ be an undirected graph. It can as well be interpreted as a logical structure where the *vertices* are the individuals and the *edges* form a binary, irreflexive, symmetric relation $E$.

Then the well-known graph problem 3-Coloring can be formulated as an MSO sentence:

$$
\begin{aligned}
\exists R \exists G \exists B\ \forall x\ (Rx \vee Gx \vee Bx) & \\
\wedge \big(Rx \rightarrow (\neg Gx \wedge \neg Bx)\big) & \\
\wedge \big(Gx \rightarrow (\neg Bx \wedge \neg Rx)\big) & \\
\wedge \big(Bx \rightarrow (\neg Rx \wedge \neg Gx)\big) & \\
\wedge\ \forall u\ \forall v\ uEv \rightarrow\ \big(\neg(Ru \wedge Rv) \wedge\ \neg(Gu \wedge Gv) \wedge\ \neg(Bu \wedge Bv)\big) &
\end{aligned}
$$

# 3 Temporal logics

Propositional logic alone is not sufficient to talk about statements that change their truth over time. The first one who systematically introduced time into modern logic was Arthur N. Prior who used modal logic as a base framework [Pri57]. The resulting system was called *tense logic*.

It was Amir Pnueli who discovered the usefulness of such logics in computer science for formally describing the behaviour of dynamic systems with discrete time steps [Pnu77]. While incorporating the "Until" operator proposed by Hans W. Kamp [Kam68], further development by Pnueli [Pnu79] as well as E. Emerson, J. Halpern and E. Clarke [CE82, EH85, EH86] has produced a family of *temporal logics*: Computation Tree Logic (CTL), Linear Temporal Logic (LTL) and Full Branching Time Logic CTL*. Today, this familiy of temporal logics are widespread in many computer scientific applications as e.g. system engineering, planning or automated reasoning. Philippe Schnoebelen gave an excellent survey that in detail presents the motivation and history of temporal logics as well as important results in computational complexity of model checking and satisfiability [Sch02].

The temporal logics have in common a set of *temporal operators* which allow to not only speak about truth of a formula, but instead about truth in timesteps that are reachable from the current one. In practice, "timestep" will mostly address a step in the computation tree of a program, therefore *branching* timelines are a common concept in CTL and CTL*. In LTL however only linear timelines are considered.

We use the following syntactic constructs as temporal operators:

- $\mathsf{F}\psi$: From now on, *eventually* $\psi$ holds *(**F**uture)*.
- $\mathsf{G}\psi$: From now on, $\psi$ holds *forever (**G**lobally)*.
- $\mathsf{X}\psi$: In the next timestep $\psi$ holds *(ne**X**t)*. Synonyms: $\mathsf{N}, \mathsf{O}$
- $\varphi \mathsf{U} \psi$: Eventually $\psi$ has to hold and ***Until*** the timestep exactly before, $\varphi$ holds.

Their meaning is visualized in Figure 3.1. There are also synonymous symbols $\Diamond$, $\Box$ and $\bigcirc$ for $\mathsf{F}$, $\mathsf{G}$ and $\mathsf{X}$ respectively. Yet in this thesis we will stick to the non-symbolic versions to avoid confusion with standard modal logic. Note that sometimes there are more binary operators defined as shortcuts:

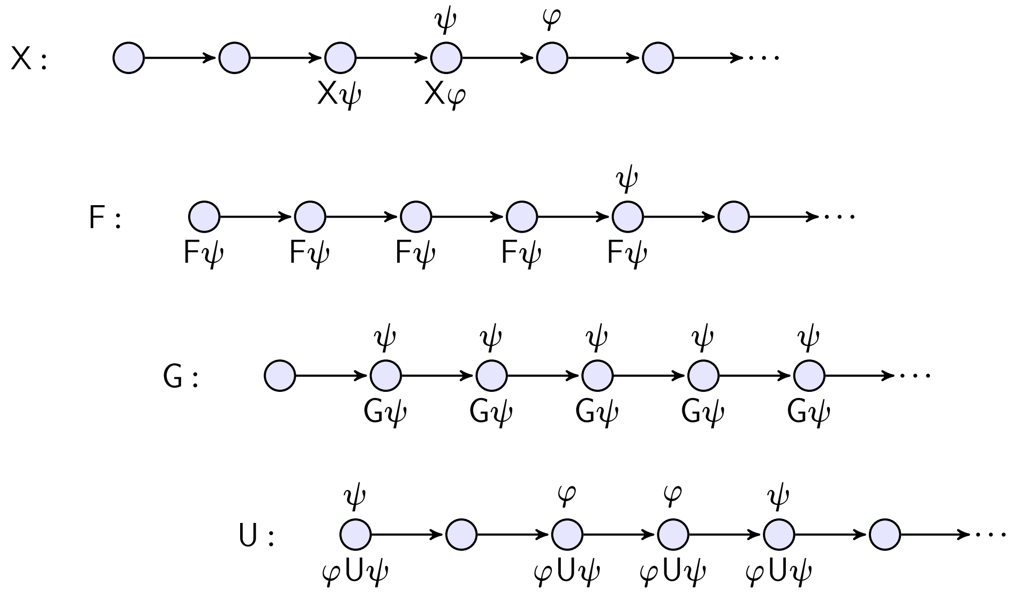


Figure 3.1: Semantics of temporal operators $X, F, G, U$

- $\varphi R\psi = G\psi \vee \psi U\varphi$: $\psi$ has to hold until $\varphi$ holds or forever ($\varphi$ ***R**eleases* $\psi$).
- $\varphi W\psi = G\varphi \vee \varphi U\psi$: $\varphi$ has to hold until $\psi$ holds or forever *(**W**eak Until)*.

While the binary operator *Until* was introduced by Kamp, the unary temporal operators are known since Prior's publications [Pri57, Kam68]. They originally defined analog operators to speak about the past, too:

- $O\psi$: In some earlier timestep $\psi$ did hold *(**O**nce)*.
- $H\psi$: In all earlier timesteps $\psi$ did hold *(**H**istorically)*.
- $P\psi$: In the previous timestep $\psi$ did hold *(**P**revious)*.
- $\varphi S\psi$: Earlier $\psi$ did hold and ***S**ince* the next timestep $\varphi$ did hold.

For e.g. LTL the version supplemented with past operators is called *PLTL*. The question whether how much these two logics differ in expressivity was open for more than 20 years. In 2003 Nicolas Markey showed that even if LTL and PLTL can express equivalent formulas, the latter one does so exponentially more succinct [Mar03]. Nevertheless we will concentrate on pure future temporal logics for reasons of simplicity.

## 3.1 Computation Tree Logic (CTL)

As the name says, CTL expresses properties of computation trees and is therefore a *branching time* logic. In general, computation trees are considered infinite as depicted in Figure 3.2a. Moreover the computation should not halt in any state. Yet we are interested in systems which can be modeled with a finite number of states, giving the advantage of being representable as a finite Kripke structure (see Definition 2.18). These two representations of an infinite computation are equivalent, the finite structure however has to be *serial* then, i.e. every state has at least one successor, as shown in Figure 3.2b. This condition essentially restricts the available Kripke frames to the ones in the class $D$ of serial frames[1] [BdV01].

To specify the semantics of temporal logics, first the concept of *state formulas* and *path formulas* has to be illustrated. The mentioned temporal operators are defined on timesteps that are ordered along a path, yet there are many paths that origin at a given state. Due to the concept of *branching time* in CTL, two *path quantifiers* called $\mathsf{E}$ and $\mathsf{A}$ are required. Any formula with temporal operators is a path formula, but to get a state formula (which can be evaluated in states of the structure) the temporal operators may only occur in the scope of a path quantifier.

The set **CTL** of syntactically valid CTL formulas is defined by following grammar:

$$\varphi ::= x \mid \top \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{AF}\varphi \mid \mathsf{AG}\varphi \mid \mathsf{AX}\varphi \mid \mathsf{A}[\varphi \mathsf{U} \varphi] \mid \mathsf{EF}\varphi \mid \mathsf{EG}\varphi \mid \mathsf{EX}\varphi \mid \mathsf{E}[\varphi \mathsf{U} \varphi]$$

Since every temporal operator is always preceded by a path quantifier, CTL is a temporal logic that has *only state formulas*.

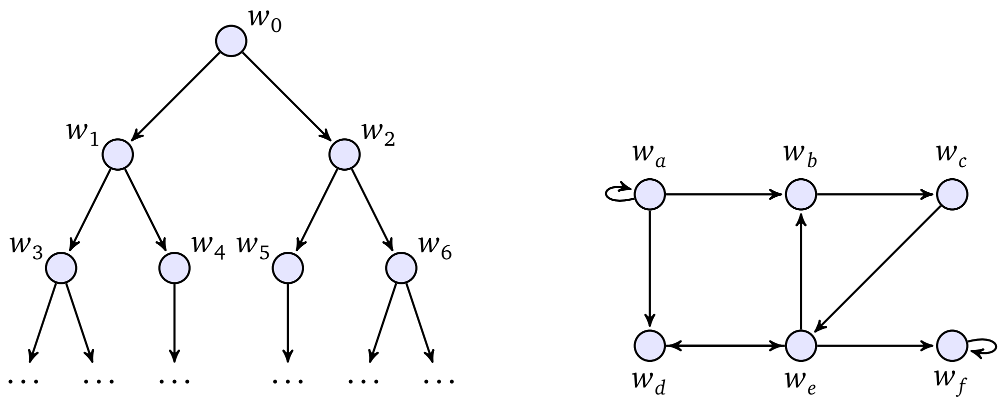
**Definition 3.1** (CTL semantics)**.** Fix a Kripke structure $\mathcal{M} = (W, R, V)$. Let

$$\Pi(w) = \{\, \pi \mid \pi = (w_1, w_2, \ldots),\ \ w_1 = w,\ \ \forall i \in \mathbb{N} : w_i R w_{i+1} \,\}$$

be the set of all paths $\pi$ in $\mathcal{M}$ of infinite length which are starting in $w$. For an infinite path $\pi = (w_1, w_2, \ldots)$, define $\pi[i] := w_i$.

Then the evaluation semantics of CTL is defined inductively and purely on state formulas.

[1]A binary relation $R$ is *serial* if every element $u$ has at least one element $v$ with $uRv$.



(a) An infinite computation tree  (b) A serial Kripke structure

Figure 3.2: Possible ways to model a non-halting computation

| | |
|---|---|
| $(\mathcal{M}, w) \models x \quad \text{for } x \in \mathbf{PS}$ | iff. $w \in V(x)$ |
| $(\mathcal{M}, w) \models \top$ | always |
| $(\mathcal{M}, w) \models \neg\varphi$ | iff. $(\mathcal{M}, w) \not\models \varphi$ |
| $(\mathcal{M}, w) \models \varphi \wedge \psi$ | iff. $(\mathcal{M}, w) \models \varphi$ and $(\mathcal{M}, w) \models \psi$ |
| $(\mathcal{M}, w) \models \mathsf{AF}\varphi$ | iff. $\forall \pi \in \Pi(w)\, \exists i \in \mathbb{N} : (\mathcal{M}, \pi[i]) \models \varphi$ |
| $(\mathcal{M}, w) \models \mathsf{AG}\varphi$ | iff. $\forall \pi \in \Pi(w)\, \forall i \in \mathbb{N} : (\mathcal{M}, \pi[i]) \models \varphi$ |
| $(\mathcal{M}, w) \models \mathsf{AX}\varphi$ | iff. $\forall \pi \in \Pi(w) : (\mathcal{M}, \pi[2]) \models \varphi$ |
| $(\mathcal{M}, w) \models \mathsf{A}[\varphi \mathsf{U} \psi]$ | iff. $\forall \pi \in \Pi(w)\, \exists i \in \mathbb{N} : (\mathcal{M}, \pi[i]) \models \psi$ |
| | and $\forall j, 1 \leq j < i : (\mathcal{M}, \pi[i]) \models \varphi$ |
| $(\mathcal{M}, w) \models \mathsf{EF}\varphi$ | iff. $\exists \pi \in \Pi(w)\, \exists i \in \mathbb{N} : (\mathcal{M}, \pi[i]) \models \varphi$ |
| $(\mathcal{M}, w) \models \mathsf{EG}\varphi$ | iff. $\exists \pi \in \Pi(w)\, \forall i \in \mathbb{N} : (\mathcal{M}, \pi[i]) \models \varphi$ |
| $(\mathcal{M}, w) \models \mathsf{EX}\varphi$ | iff. $\exists \pi \in \Pi(w) : (\mathcal{M}, \pi[2]) \models \varphi$ |
| $(\mathcal{M}, w) \models \mathsf{E}[\varphi \mathsf{U} \psi]$ | iff. $\exists \pi \in \Pi(w)\, \exists i \in \mathbb{N} : (\mathcal{M}, \pi[i]) \models \psi$ |
| | and $\forall j, 1 \leq j < i : (\mathcal{M}, \pi[i]) \models \varphi$ |

Observe that the definition via quantifiers directly implies a duality relation for the simpler temporal operators. This is however not the case for *until*-operators due to their asymmetric quantifier structure, leading to different equivalences.

$$\begin{aligned} \mathsf{EX}\psi &= \neg\mathsf{AX}\neg\psi & \mathsf{EF}\psi &= \mathsf{E}[\top\mathsf{U}\psi] \\ \mathsf{EF}\psi &= \neg\mathsf{AG}\neg\psi & \mathsf{AF}\psi &= \mathsf{A}[\top\mathsf{U}\psi] \\ \mathsf{EG}\psi &= \neg\mathsf{AF}\neg\psi & \mathsf{A}[\varphi\mathsf{U}\psi] &= \mathsf{AF}\psi \wedge \neg\mathsf{E}[\neg\psi\mathsf{U}(\neg\varphi \wedge \neg\psi)] \end{aligned}$$

This asymmetric result is surprising at first sight: The EU operator is more expressive than the AU operator despite being defined with one less quantifier alternation. When used together with AF, however, EU can define AU.

From the above equivalences one can prove that the operator set $\{\mathsf{AF}, \mathsf{AX}, \mathsf{EU}\}$ fully expresses CTL and the set $\{\mathsf{AF}, \mathsf{AX}, \mathsf{AU}\}$ does not [Lar95].

The satisfiability problem corresponding to CTL is CTL-SAT,

$$\text{CTL-SAT} := \left\{ \varphi \;\middle|\; \begin{array}{l} \varphi \in \textbf{CTL} \text{ and } (\mathcal{M}, w) \models \varphi \text{ for a serial Kripke} \\ \text{structure } \mathcal{M} \text{ and a world } w \text{ of } \mathcal{M} \end{array} \right\}.$$

The model checking problem is CTL-MC:

$$\text{CTL-MC} := \left\{ (\varphi, \mathcal{M}, w) \;\middle|\; \begin{array}{l} \varphi \in \textbf{CTL}, \mathcal{M} \text{ is a serial Kripke} \\ \text{structure and } (\mathcal{M}, w) \models \varphi \end{array} \right\}$$

## 3.2 Linear Temporal Logic (LTL)

*Linear Temporal Logic* was initially introduced by Pnueli in 1977 [Pnu77]. Unlike CTL, it is not considered a branching temporal logic but as the name suggests a *linear* one instead. LTL originated from languages of infinite strings over finite alphabets, so-called *$\omega$-words*, named after the smallest infinite ordinal $\omega$. It is possible to define finite automata that accept such *$\omega$-languages*, e.g. *Büchi-automata*, which accept an $\omega$-word if and only if they visit an accepting state infinitely often [Büc90].

Yet the definition of LTL semantics can also be carried over to serial Kripke structures. When comparing the syntax of LTL to CTL, the absence of path quantifiers is obvious. Indeed temporal operators can be nested directly in LTL and refer still to the same infinite path, which is also called a *run*.

The set **LTL** of syntactically valid LTL formulas is defined as follows:

$$\psi ::= x \mid \top \mid \psi \wedge \psi \mid \neg\psi \mid \mathsf{F}\psi \mid \mathsf{G}\psi \mid \mathsf{X}\psi \mid \psi\mathsf{U}\psi$$

In contrast to CTL, LTL is a temporal logic that has only path formulas.

**Definition 3.2** (LTL semantics)**.** Let $\pi = (w_1, w_2, \ldots)$ be an infinite path. Define $\pi_{\geq i}$ as the subpath starting at index $i$, i.e. $\pi_{\geq i} := (w_i, w_{i+1}, \ldots)$. The *truth* of an LTL formula is then again defined inductively:

$$
\begin{array}{lll}
(\mathcal{M},\pi) \models x \quad \text{for } x \in \mathbf{PS} & \text{iff. } \pi[1] \in V(x) \\
(\mathcal{M},\pi) \models \top & \text{always} \\
(\mathcal{M},\pi) \models \neg\varphi & \text{iff. } (\mathcal{M},\pi) \not\models \varphi \\
(\mathcal{M},\pi) \models \varphi \wedge \psi & \text{iff. } (\mathcal{M},\pi) \models \varphi \text{ and } (\mathcal{M},\pi) \models \psi \\
(\mathcal{M},\pi) \models \mathsf{F}\varphi & \text{iff. } \exists i \in \mathbb{N} : (\mathcal{M},\pi_{\geq i}) \models \varphi \\
(\mathcal{M},\pi) \models \mathsf{G}\varphi & \text{iff. } \forall i \in \mathbb{N} : (\mathcal{M},\pi_{\geq i}) \models \varphi \\
(\mathcal{M},\pi) \models \mathsf{X}\varphi & \text{iff. } (\mathcal{M},\pi_{\geq 2}) \models \varphi \\
(\mathcal{M},\pi) \models \varphi \mathsf{U} \psi & \text{iff. } \exists i \in \mathbb{N} : (\mathcal{M},\pi_{\geq i}) \models \psi \\
& \quad \text{and } \forall j, 1 \leq j < i : (\mathcal{M},\pi_{\geq j}) \models \varphi
\end{array}
$$

General satisfiability of an LTL-formula is equivalent to the one on path-like Kripke structures, hence to define satisfiability we simply quantify paths that start in an initial world $w$.

The resulting problem is LTL-SAT:

$$
\text{LTL-SAT} := \left\{ \varphi \;\middle|\; \begin{array}{l} \varphi \in \mathbf{LTL} \text{ and } (\mathcal{M},\pi) \models \varphi \text{ for a serial Kripke} \\ \text{structure } \mathcal{M}, \text{ a world } w \text{ of } \mathcal{M} \text{ and all } \pi \in \Pi(w) \end{array} \right\}
$$

Considering only one path starting at $w$ instead of all paths would lead to an equivalent satisfiability problem due to the seriality of the structures.

The model checking problem is LTL-$\forall$MC:

$$
\text{LTL-}\forall\text{MC} := \left\{ (\varphi,\mathcal{M},w) \;\middle|\; \begin{array}{l} \varphi \in \mathbf{LTL},\ \mathcal{M} \text{ is a serial Kripke structure} \\ \text{and } (\mathcal{M},\pi) \models \varphi \text{ for all paths } \pi \in \Pi(w) \end{array} \right\}
$$

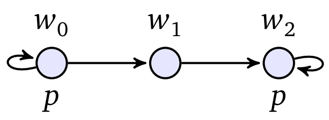
One could also define the problem LTL-$\exists$MC where at least one path from the initial world $w$ has to fulfill $\varphi$, in fact for $\varphi \in \mathbf{LTL}$ holds:

$$
(\varphi,\mathcal{M},w) \in \text{LTL-}\forall\text{MC} \Leftrightarrow (\neg\varphi,\mathcal{M},w) \notin \text{LTL-}\exists\text{MC}
$$

The universal variant is however more intuitive in practice and the results of computational complexity can easily be transferred between these two problems.

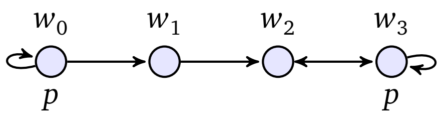
Notice that the different semantics (state formulas and path formulas) lead to incomparable expressiveness of CTL and LTL in the sense that there are CTL formulas not expressible in LTL and vice versa. Roughly spoken, LTL cannot handle branching while CTL cannot express fairness. Fairness in general has the meaning that a desired event happens infinitely often in a run, or similar, that every request gets eventually granted. An example for an LTL formula that expresses fairness is $G(\text{request} \rightarrow \mathsf{F}\,\text{grant})$.

**Example 3.3:**
Consider another well-known counter-example that has no equivalent CTL formula: $\mathsf{FG}p$. It states that there is a future timestep from whereon $p$ will forever hold. It is satisfied for example by all paths starting in $w_0$ in the following structure:



Now evaluate the CTL formula $\mathsf{AFAG}p$ where every temporal operator was just supplemented with an $\mathsf{A}$: It is false in world $w_0$. It states that every infinite path has to contain a state $w$ such that on all paths starting in $w$ globally $p$ holds. But the path $\pi = (w_0, w_0, \ldots)$ contains no such state $w$.

While $\mathsf{AFAG}p$ is stricter than $\mathsf{FG}p$, the slight variation $\mathsf{AFEG}p$ is too permissive: It states that every infinite path contains a world $w$ such that $p$ eventually holds in a path starting from $w$. This is the case for every path starting in $w_0$ in the structure below.



In contrast, the LTL formula $\mathsf{FG}p$ now is false in every world because there is always a path which can alternate between $w_2$ and $w_3$ infinitely often, preventing $\mathsf{G}p$ from ever holding on that path.

On the other hand, even the simple CTL formula $\mathsf{EX}p \wedge \mathsf{EX}\neg p$ has no equivalent in LTL since it necessitates branching.

**Theorem 3.4** (Clarke and Draghicescu, 1989 [CD89])**.** *A CTL formula $\varphi$ is either equivalent to the LTL formula obtained by deleting all path quantifiers ($\mathsf{A}, \mathsf{E}$) from it, or it has no equivalent LTL formula.*

## 3.3 Full Branching Time Logic (CTL*)

The temporal logic CTL* was first proposed by Emerson and Halpern to join CTL-like branching with LTL-like single-path nesting of temporal operators [EH86]. It allows to freely combine these operators to path formulas which then are prefixed by a path quantifier. This syntactical structure leads to a grammar using two symbols, state formulas $\varphi$ and path formulas $\psi$.

$$\begin{aligned}
\varphi &::= x \mid \top \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{A}\psi \mid \mathsf{E}\psi \\
\psi &::= \varphi \mid \psi \wedge \psi \mid \neg\psi \mid \mathsf{F}\psi \mid \mathsf{G}\psi \mid \mathsf{X}\psi \mid \psi\mathsf{U}\psi
\end{aligned}$$

A formula $\varphi$ is a valid CTL* formula, i.e. $\varphi \in \mathbf{CTL}^\star$, if and only if it is a state formula constructed by the above grammar. The interpretation of CTL* formulas follows the previously stated semantics of CTL and LTL:

**Definition 3.5** (CTL* semantics)**.**

$$\begin{array}{ll}
(\mathcal{M},w) \models x \quad \text{for } x \in \mathbf{PS} & \text{iff. } w \in V(x) \\
(\mathcal{M},w) \models \top & \text{always} \\
(\mathcal{M},w) \models \neg\varphi & \text{iff. } (\mathcal{M},w) \not\models \varphi \\
(\mathcal{M},w) \models \varphi \wedge \varphi' & \text{iff. } (\mathcal{M},w) \models \varphi \text{ and } (\mathcal{M},w) \models \varphi' \\
(\mathcal{M},w) \models \mathsf{A}\psi & \text{iff. } \forall \pi \in \Pi(w) : (\mathcal{M},\pi) \models \psi \\
(\mathcal{M},w) \models \mathsf{E}\psi & \text{iff. } \exists \pi \in \Pi(w) : (\mathcal{M},\pi) \models \psi
\end{array}$$

$$\begin{array}{ll}
(\mathcal{M},\pi) \models x \quad \text{for } x \in \mathbf{PS} & \text{iff. } \pi[1] \in V(x) \\
(\mathcal{M},\pi) \models \top & \text{always} \\
(\mathcal{M},\pi) \models \neg\psi & \text{iff. } (\mathcal{M},\pi) \not\models \psi \\
(\mathcal{M},\pi) \models \psi \wedge \psi' & \text{iff. } (\mathcal{M},\pi) \models \psi \text{ and } (\mathcal{M},\pi) \models \psi' \\
(\mathcal{M},\pi) \models \mathsf{A}\psi & \text{iff. } (\mathcal{M},\pi[1]) \models \mathsf{A}\psi \\
(\mathcal{M},\pi) \models \mathsf{E}\psi & \text{iff. } (\mathcal{M},\pi[1]) \models \mathsf{E}\psi \\
(\mathcal{M},\pi) \models \mathsf{F}\psi & \text{iff. } \exists i \in \mathbb{N} : (\mathcal{M},\pi_{\geq i}) \models \psi \\
(\mathcal{M},\pi) \models \mathsf{G}\psi & \text{iff. } \forall i \in \mathbb{N} : (\mathcal{M},\pi_{\geq i}) \models \psi \\
(\mathcal{M},\pi) \models \mathsf{X}\psi & \text{iff. } (\mathcal{M},\pi_{\geq 2}) \models \psi \\
(\mathcal{M},\pi) \models \psi \mathsf{U} \psi' & \text{iff. } \exists i \in \mathbb{N} : (\mathcal{M},\pi_{\geq i}) \models \psi' \\
 & \quad \text{and } \forall j, 1 \leq j < i : (\mathcal{M},\pi_{\geq j}) \models \psi
\end{array}$$

The satisfiability and model checking problems are defined as

$$\text{CTL}^\star\text{-SAT} := \left\{ \varphi \;\middle|\; \begin{array}{l} \varphi \in \mathbf{CTL}^\star \text{ and } (\mathcal{M},w) \models \varphi \text{ for a serial Kripke} \\ \text{structure } \mathcal{M} \text{ and a world } w \text{ of } \mathcal{M} \end{array} \right\},$$

$$\text{CTL}^\star\text{-MC} := \left\{ (\varphi,\mathcal{M},w) \;\middle|\; \begin{array}{l} \varphi \in \mathbf{CTL}^\star,\ \mathcal{M} \text{ is a serial Kripke} \\ \text{structure and } (\mathcal{M},w) \models \varphi \end{array} \right\}$$

**Definition 3.6** (Negation Normal Form)**.** A formula $\varphi$ is in *negation normal form (NNF)* if negations ($\neg$) occur only in front of propositional variables.

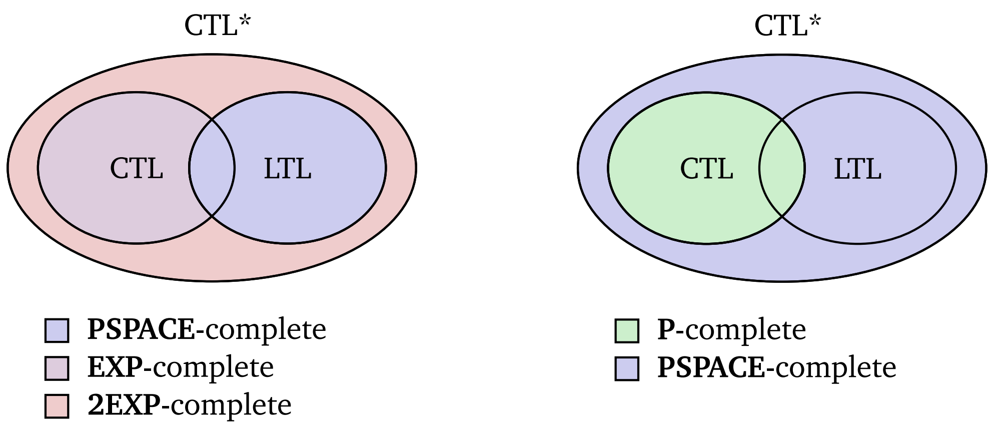
We write $\mathbf{CTL}_{\text{NNF}}$ for the subset of CTL-formulas that are in NNF, analogously we write $\mathbf{LTL}_{\text{NNF}}$ and $\mathbf{CTL}^\star_{\text{NNF}}$.

Note that every U-free temporal formula is equivalent to a formula in NNF by repeatedly applying DeMorgan's Law, replacing a temporal operator by its dual if necessary. For formulas containing U operators however this is not necessarily possible due to the lack of dual operators.

As well as the expressive powers of CTL and LTL are incomparable, there are CTL$^*$ formulas that are neither expressible in CTL nor in LTL[1]. This comes together with a higher computational complexity when deciding consistency and model validity, as Figure 3.3 shows. For both problems there are various completeness results. CTL is the only considered logic that has tractable[2] model checking while the problem is PSPACE-complete for LTL and CTL$^*$. Model checking for linear time logics is reducible to satisfiability and in fact both problems are PSPACE-complete. But for the branching logics satisfiability checking is complete for single resp. double exponential time. The upper bounds are to Clarke, Emerson, Sistla and Lei. The lower bounds are to Schnoebelen for CTL and to Sistla and Clarke for LTL (they automatically hold for CTL$^*$ as well) [SC85, CES86, EL87b, Sch02].

The reader is again referred to Schnoebelen for an excellent summary of results regarding complexity of temporal logics [Sch02].

The influence of the restriction to Boolean and temporal operator fragments on the complexity was extensively studied by Meier [Mei11].



(a) Satisfiability complexity

(b) Model checking complexity

Figure 3.3: Expressive power and complexity of temporal logics

[1]Basically any formula that necessitates branching on nested temporal operators.

[2]But inherently sequential since it is P-complete.

# 4 Treewidth and Courcelle's theorem

Graphs have been a fundamental tool for modeling many problems in computer science for a long time. Since they have immense importance for many practical tasks, they are researched thoroughly also in complexity theory where plenty of formal graph problems are defined and analyzed. Many of them are known as NP-complete and therefore considered as intractable, such as Clique, Independent Set, Dominating Set, Vertex Cover, Hamiltonicity, Coloring, Subgraph Isomorphism and Feedback Vertex Set.

Almost all important graph problems including the ones mentioned above are trivial or tractable on tree instances [Nie06]. This led to the desire to formalize the "tree-alikeness" of graphs. This property called *treewidth* was initially studied by Rudolf Halin in his work on the *S-function lattice*[1] [Hal76]. Later it was rediscovered by Neil Robertson and Paul D. Seymour [RS84]. The treewidth is a direct measure of quality of an optimal *tree decomposition*. According to the definition, trees have a treewidth of 1, cycles and serial-parallel graphs have a treewidth of 2. Any graph containing the $n$-complete subgraph has treewidth at least $n-1$. Furthermore, every treewidth $k \in \mathbb{N}$ can be characterized by a finite set of *forbidden minors* which is a consequence of the famous *graph minor theorem* by Robertson and Seymour [Ram97].

A good tree decomposition of a graph allows to solve searching problems on it much more efficiently than on the original graph — which often manifests as an fpt-algorithm parameterized by the treewidth.

## 4.1 Tree- and path-decompositions

We use the definitions 2.20 and 2.21 of vocabularies and logical structures.

**Definition 4.1** (Tree and path decomposition)**.** Let $\mathcal{A} = \left(A, \tau^{\mathcal{A}}\right)$ be a finite relational structure over $\tau = (R_1, \ldots, R_n)$.

Then a *tree-decomposition* $\mathcal{T}$ of $\mathcal{A}$ is a pair $\mathcal{T} = \left(T, (B_v)_{v\in V}\right)$ where

- $T = (V, E)$ is a finite tree with vertex set $V$ and edge set $E$,
- $(B_v)_{v\in V}$ is a family of subsets of $A$, called *bags*,

[1]This lattice of graph parameters has the treewidth as supremum and the *Hadwiger number*, the maximal size of a clique minor, as infimum.

- for every $a \in A$ the vertices of bags containing $a$ induce a non-empty connected subgraph of $T$ and
- for every relation $R$ of $\tau$ and every $(a_1, \dots, a_{\mathrm{ar}(R)}) \in R^{\mathcal{A}}$ there exists a vertex $v \in V$ with $\left\{ a_1, \dots, a_{\mathrm{ar}(R)} \right\} \subseteq B_v$.

If additionally $T$ is a path graph, i.e. acyclic with maximum degree 2, then $\mathcal{T}$ is a *path-decomposition* of $\mathcal{A}$.

**Definition 4.2** (Treewidth and pathwidth)**.** Let $\mathcal{T} = \big((V, E), (B_v)_{v \in V}\big)$ be a tree-decomposition of a relational structure $\mathcal{A} = \big(A, \tau^{\mathcal{A}}\big)$.

Then the *width* of $\mathcal{T}$ is the size of the biggest bag minus one, i.e.

$$\mathrm{w}(\mathcal{T}) := \max\{|B_v| \mid v \in V\} - 1.$$

The *treewidth* of $\mathcal{A}$ is the width of a tree-decomposition of $\mathcal{A}$ with smallest width:

$$\mathrm{tw}(\mathcal{A}) := \min\{\mathrm{w}(\mathcal{T}) \mid \mathcal{T} \text{ is a tree-decomposition of } \mathcal{A}\}$$

Such a tree-decomposition is called *optimal*.

Likewise, the *pathwidth* of $\mathcal{A}$ is the width of a path-decomposition with smallest width;

$$\mathrm{pw}(\mathcal{A}) := \min\{\mathrm{w}(\mathcal{T}) \mid \mathcal{T} \text{ is a path-decomposition of } \mathcal{A}\}$$

The offset -1 in the definition of width was historically chosen to let trees correspond exactly to the graphs with treewidth 1 and path graphs to those with pathwidth 1.

Much work has be done on treewidth and on algorithms computing optimal decompositions. In particular, a variety of results has been achieved by Bodlaender including the following theorems [BK91, Bod93a, Bod93b].

**Theorem 4.3** (Bodlaender, 1993)**.** *For every $k \in \mathbb{N}$ there is an algorithm with runtime $2^{\mathcal{O}(k^3)} \cdot \mathcal{O}(|\mathcal{A}|)$ that, given a structure $\mathcal{A}$, computes a tree-decomposition of $\mathcal{A}$ of width at most $k$ or returns "$tw(\mathcal{A}) > k$" if no such decomposition exists.*

A similar theorem holds for path-decompositions.

As given above, Bodlaender's theorem merely states the existence of a non-uniform algorithm, showing that the otherwise NP-complete problem TREEWIDTH (given a graph $G$ and a natural number $k$, decide if $\mathrm{tw}(G) \leq k$) is in **P** for each slice when parameterized by $k$ [Bod93a]. But it does not follow **XP** or even **FPT** membership from this, in fact, a non-uniform algorithm deciding treewidth is easy to find. One can e.g. determine if a graph has treewidth $> k$ by testing each element of the finite obstruction set for treewidth $\leq k$ [RS84, Ram97]. However, Bodlaender and others stated that in contrast to earlier results the new algorithm can be used “constructively” and with $k$ as part of the input. This leads to a more general version of the theorem [FG06, Thm. 11.12]:

**Corollary 4.4** (Construction of optimal decompositions)**.** *There is an algorithm that, given a structure $\mathcal{A}$, computes an optimal tree-decomposition (resp. path-decomposition) in time $f(tw(\mathcal{A})) \cdot \mathcal{O}(|\mathcal{A}|)$ $\Big($resp. $f(pw(\mathcal{A})) \cdot \mathcal{O}(|\mathcal{A}|)\Big)$ for a computable $f$.*

*Proof.* Iterate the natural numbers $k = 1, 2, \ldots$ until a decomposition is found via Theorem 4.3. □

**Lemma 4.5.** *Let $\Pi_{tw} = (P, \kappa + tw)$ and $\Pi_{pw} = (P, \kappa + pw)$ be a set $P$ of relational structures which is parameterized by $\kappa$ and additionally by treewidth resp. pathwidth. Then $\Pi_{pw} \leq^{fpt} \Pi_{tw}$.*

*Proof.* Every path-decomposition of a structure is also a tree-decomposition. Therefore pathwidth is always an upper bound for treewidth. Hence $\kappa + \mathrm{tw}(x) \leq \kappa + \mathrm{pw}(x)$, and the identity function yields a correct fpt-reduction. □

## 4.2 Recognizing MSO-definable sets

In 1990 Courcelle worked on graph grammars and graph languages and as well on monadic second order logic. He proved in a constructive way that any MSO-definable graph property can be decided in linear time on the graphs’ tree-decompositions with bounded treewidth. For this he used a finite automaton variant called *tree automaton*, which runs on tree-like, non-linear inputs. The automata-theoretic approach is also the most common proof for Courcelle’s theorem today.

Combined with Bodlaender’s results, the algorithm is usually given as in Algorithm 4.1.

**Lemma 4.6.** *Algorithm 4.1, given $\mathcal{A}$, $k$ and $\varphi$, decides whether $tw(\mathcal{A}) \leq k$ and $\mathcal{A} \models \varphi$ in fpt-time.*

*Proof.* We will sketch the common proof [Cou90]. By construction of $\varphi^*$ it holds that $\mathcal{A} \models \varphi \Leftrightarrow \mathcal{T} \models \varphi^*$. Also it holds that $\mathcal{T} \models \varphi^*$ if and only if $\mathcal{T} \in L(\mathcal{M})$, i.e. $\mathcal{M}$ accepts $\mathcal{T}$.

The runtime of step 1 and 2 is $f(k) \cdot \mathcal{O}(|G|)$ due to Theorem 4.3, step 3 runs in time $g(k, |\varphi|)$, the automaton construction in step 4 runs in time $h(k, |\varphi|)$ and finally the simulation in step 5 is possible in time $h(k, |\varphi|) \cdot |G|$. The functions $f, g, h$ are computable. The whole algorithm requires time $\big(f(k, |\varphi|) + g(k, |\varphi|) + h(k, |\varphi|)\big) \cdot \mathcal{O}(|G|)$ which is fpt and in fact fixed-parameter linear time. □

**Algorithm 4.1:** Courcelle's algorithm

```
Input  : $\mathcal{A}$ relational structure with treewidth tw($\mathcal{A}$)
         $k \in \mathbb{N}$ upper bound for treewidth
         $\varphi$ MSO formula
Output : Is tw($\mathcal{A}$) $\leq k$ and $\varphi \models \mathcal{A}$ ?
1 if $tw(\mathcal{A}) > k$ then reject
2 $\mathcal{T} \leftarrow$ tree-decomposition of $\mathcal{A}$ with width at most $k$
3 $\varphi^* \leftarrow \varphi$ transformed to its tree variant
4 $\mathcal{M} \leftarrow$ the tree automaton recognizing exactly the models of $\varphi^*$
5 Simulate $\mathcal{M}$ on $\mathcal{T}$.
6 if $\mathcal{M}$ accepted then accept else reject
```

Let MSO-MC be the model checking problem for MSO formulas on relational structures:

$$\text{MSO-MC} := \{\, (\varphi, \mathcal{A}) \mid \varphi \in \textbf{MSO}, \mathcal{A} \text{ is a relational structure and } \mathcal{A} \models \varphi \,\}$$

Then the following theorems show the fixed-parameter tractability of MSO-MC.

**Theorem 4.7** (Courcelle's meta-theorem for treewidth)**.** *The parameterized problem* (MSO-MC, $\kappa_{tw}$) *is in* **FPT**, *where* $\kappa_{tw}(\varphi, \mathcal{A}) = |\varphi| + tw(\mathcal{A})$.

*Proof.* The theorem follows from Lemma 4.6 and Corollary 4.4. If no upper bound for the treewidth is known *a priori*, one can for each natural number try to construct a tree-decomposition of that width. This leads to another factor tw($\mathcal{A}$) in the runtime. A more detailed proof is provided by Courcelle and Engelfriet [CE12, Theorem 1.24 and Chapter 6]. □

**Corollary 4.8** (Courcelle's meta-theorem for pathwidth)**.** *The parameterized problem* (MSO-MC, $\kappa_{pw}$) *is in* **FPT**, *where* $\kappa_{pw}(\varphi, \mathcal{A}) = |\varphi| + pw(\mathcal{A})$.

*Remark:* Courcelle's theorem is a result that has a purely theoretical nature due to its enormous constant factors hidden in the $\mathcal{O}$-notation and the dependence on the parameter: $g \approx 2^{32 \cdot k^3}$. The time $h(k, |\varphi|)$ required for the construction of $\mathcal{M}$ cannot even be bounded by an elementary function unless **P** = **NP** [FG04]. The number of states of $\mathcal{M}$ rather grows as an exponential "power tower" whose height is linear in the number of second-order quantifier alternations in $\varphi$. Hence applying Courcelle's theorem is usually not the end of classifying a problem but instead the beginning of searching for an efficient algorithm.

## 4.3 Meta-algorithm for uniform MSO-formula families

In Section 2.2 we saw how to define decision problems in terms of monadic second order logic. It follows from Courcelle's theorem that such a problem always is in **FPT** when the chosen parameterization is at least the treewidth of the input structure. Still there are problems which are undefinable in MSO for certain reasons.

To cover some of the problems that are not definable by a single MSO formula, a meta-algorithm is presented that dynamically selects the proper formula for a given instance. A variant of this algorithm was presented by Lück, Meier and Schindler [LMS15].

As a first step we need a way to provide infinitely many MSO formulas to an algorithm. Therefore we require a finite description of such a formula family:

**Definition 4.9** (Uniform formula families)**.** Let $\Phi := (\varphi_n)_{n\in\mathbb{N}}$ be a family of MSO formulas. We call $\Phi$ *uniform* if there is a computable function $g$ such that $g(n) = \varphi_n$, that is, $g$ computes the correct formula when a binary representation of $n$ is provided.

A computable formula family however does not automatically grant fpt-like runtime. For the theorem it is also required that the selection of a proper formula for a given instance happens in fpt-time too and that the formula cannot become too long.

**Definition 4.10** ($\kappa$-bounded functions)**.** Let $\kappa : \Sigma^* \to \mathbb{N}$ be a function. A function $f : \Sigma^* \to \Delta^*$ is *$\kappa$-bounded* if for every $x \in \Sigma^*$ the length of $f(x)$ can be computable bounded in terms of $\kappa$, i.e. there is another computable function $h : \mathbb{N} \to \mathbb{N}$ such that $\forall x \in \Sigma^* : |f(x)| \leq h(\kappa(x))$.

Altogether we get:

**Theorem 4.11.** *Let $(Q, \kappa)$ be a parameterized problem over structures $\mathcal{A}$ such that tw$(\mathcal{A})$ is $\kappa$-bounded. Let $(\varphi_n)_{n\in\mathbb{N}}$ be a uniform MSO formula family. Let $f$ be a $\kappa$-bounded, fpt-computable function w. r. t. $\kappa$. If for every $\mathcal{A}$ it holds that $\mathcal{A} \in Q \iff \mathcal{A} \models \varphi_{|f(\mathcal{A})|}$, then $(Q, \kappa) \in$* **FPT***.*

*Proof.* Let $(\varphi_n)_{n\in\mathbb{N}}$ be computed by a function $g$. Then the following algorithm correctly decides $Q$:

**Algorithm 4.2:** Meta-algorithm for MSO-family definable problems

```
  Input  : Relational structure $\mathcal{A}$
1 $n \leftarrow |f(\mathcal{A})|$
2 $\varphi \leftarrow g(n)$
3 return $\mathcal{A} \models \varphi$?
```

The runtime of $f$ in step 1 is fpt w. r. t. $\kappa$ by premise. Because $f$ is $\kappa$-bounded, the size of $n$ and therefore the runtime of $g$ in step 2 can also only depend on $\kappa$. The runtime of step 3 follows from Courcelle's theorem and is linear when parameterized by $|\varphi| + \mathrm{tw}(\mathcal{A})$. But $|\varphi|$ is bounded by the runtime of $g$ so both $|\varphi|$ and tw are $\kappa$-bounded. Formally, the runtime of step 3 is $j(|\varphi|, \mathrm{tw}(\mathcal{A})) \cdot |\mathcal{A}|$. The function $j$ can be chosen non-decreasing hence we get fpt-runtime and the theorem follows. $\square$

Note that the individual members of a formula family do not have to agree on a fixed vocabulary. A possibility to use this theorem is when dealing with sets of structures with unbounded number of relation symbols. It is impossible to define such a set with a single formula unless only finitely many relation symbols are used non-trivially. In this thesis however we will stick to finite vocabularies.

# 5 Parameterizing temporal satisfiability

The satisfiability problem of a logic is in general harder to solve than its model checking problem. The task of formal verification mostly deals with models. However, both problem have their importance in practice in the scope of temporal logics. Usually in the progress of verification it is stated that a certain constraint $\varphi$ should hold on a given model $\mathcal{A}$. This constraint $\varphi$ should not contain inconsistent subformulas; the existence of such a subformula obviously doesn't make sense in a verification task.

On the other hand a tautological subformula is even worse in practice; it is clearly not intended in model checking and can easily lead to wrong verification results since it is more difficult to notice. Hence a good model checking tool should also be able to do satisfiability and validity tests efficiently.

In this chapter we want to determine the parameterized complexity of satisfiability of various fragments of temporal logics for a suitable parameter. An important part of work is usually to find a meaningful parameter. Take for example the problem of propositional satisfiability, SAT: One could easily parameterize the instances (propositional formulas) by the number of occurring variables. While such a choice would immediately yield an fpt-algorithm deciding SAT in time $2^k \cdot \mathcal{O}(n)$, this result is not very helpful for a simple reason; the average number of variables is not a property that is "usually low" in common instances, and therefore this parameterization is useless in practice[1]. In general, for every decidable problem one can find artificial parameterizations permitting straightforward FPT results, as well as for every intractable problem it can be proven that certain parameterizations[2] stay fixed-parameter intractable [FG06].

Following similar research of Praveen on modal logic [Pra13], we use the temporal depth together with the structural pathwidth resp. treewidth of the formula as a parameter. This seems sensible for the following reasons: On the one hand, a temporal formula used for verification is likely designed having a rather low nesting depth of temporal operators. This is sufficient to express most important constraints like *liveness*, *safety* and *fairness*. On the other hand, the structural pathwidth and treewidth

[1] More details about choosing a good parameterization can be found in [Nie06, pp. 41–49].

[2] Take an arbitrary EXP-complete problem and the constant parameterization $\kappa = 1$.

are correlated to the degree of "interconnectedness" of disjoint subformulas in the sense of how many variables they have in common. Thus they are the lower the more the formula consists of multiple independent constraints. In this sense parameterized complexity theory allows to formally relate propositional interconnectedness to computational complexity.

The following sections are organized as follows. The fragments of the temporal logics CTL, LTL and CTL* are each analyzed in the context of parameterization by temporal depth and treewidth. Regarding the parameterization by treewidth or pathwidth we apply the result by Bruno Courcelle which was presented in Chapter 4. Last, the influence of the two parts of the parameter is studied separately.

## 5.1 Syntactical structures

To define a measure of *pathwidth* and *treewidth* on temporal formulas, we transform them into a structure that almost resembles a *syntax tree*. The main difference is that every propositional variable is represented by exactly one node. Hence in the case that a variable occurs multiple times the syntax graph is no longer a tree but a *directed acyclic graph (dag)*. From now on, we write $\mathrm{SF}(\varphi)$ for the set of all syntactically valid subformulas of $\varphi$, including $\varphi$ itself. Note that we distinguish subformulas that are lexicographically equal but occur at different positions inside $\varphi$. Therefore we define $\mathrm{SF}_\mathrm{o}(\varphi)$ similar to $\mathrm{SF}(\varphi)$ except it can contain multiple subformulas that are syntactically equivalent. Instead of considering $\mathrm{SF}_\mathrm{o}(\varphi)$ as a multiset one could also treat it as a simple set of subformula-position pairs.

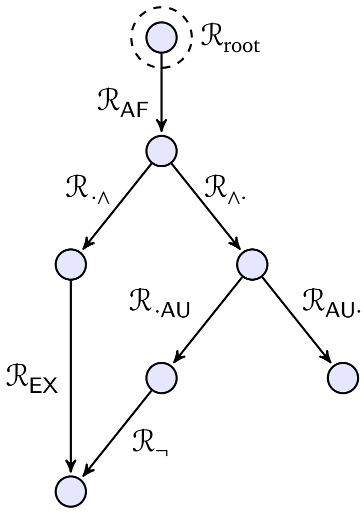
If for a temporal formula $\varphi$ we speak about the treewidth $\mathrm{tw}(\varphi)$ $\big($resp. pathwidth $\mathrm{pw}(\varphi)\big)$ we always refer to the treewidth (resp. pathwidth) of the *syntactical structure* $\mathcal{S}_\varphi$ of $\varphi$. This structure is constructed on a special relational vocabulary $\tau_\mathrm{syn}$ which contains the following relation symbols:

| | |
|---|---|
| $\mathcal{R}_\mathrm{root}(\alpha)$ | if $\alpha = \varphi$ |
| $\mathcal{R}_{\cdot T}(\alpha,\beta)$ | if $\exists\gamma$ s. t. $\alpha = \beta T \gamma$ for a binary connective $T$ |
| $\mathcal{R}_{T\cdot}(\alpha,\beta)$ | if $\exists\gamma$ s. t. $\alpha = \gamma T \beta$ for a binary connective $T$ |
| $\mathcal{R}_T(\alpha,\beta)$ | if $\alpha = T\beta$ for a unary connective $T$ |

Connectives appear as Boolean connectives $\land, \lor, \neg$ and temporal connectives $\mathsf{X}$, $\mathsf{F}$, $\mathsf{G}$, $\mathsf{U}$, $\mathsf{A}$, $\mathsf{E}$ plus their combinations to CTL operators.

Construct the syntactical structure $\mathcal{S}_\varphi := \left(\mathrm{SF}_\mathrm{o}(\varphi), (\tau_\mathrm{syn})^{\mathcal{S}_\varphi}\right)$ by using $\mathrm{SF}_\mathrm{o}(\varphi)$ as domain and then choosing the minimal interpretations of the given relations that obey the rules above.

**Example 5.1:**
Let $\varphi = \mathsf{AF}(\mathsf{EX}p \wedge \mathsf{A}[\neg p \mathsf{U} q])$ be a CTL formula. Then the corresponding structure $\mathcal{S}_\varphi$ is shown in Figure 5.1.



$\mathcal{R}_{\text{root}}$
$\mathcal{R}_{\mathsf{AF}}$
$\mathcal{R}_{\cdot\wedge}$
$\mathcal{R}_{\wedge\cdot}$
$\mathcal{R}_{\cdot\mathsf{AU}}$
$\mathcal{R}_{\mathsf{AU}\cdot}$
$\mathcal{R}_{\mathsf{EX}}$
$\mathcal{R}_{\neg}$

Figure 5.1: Example for a syntactical structure

*Remark:* Modelling a formula as a structure is not a new idea. Several graph theoretic interpretations have been considered for example in the context of the problem *Counting-SAT* (#SAT) which asks for the number of satisfying assignments of a propositional formula. In this area the fixed-parameter tractability with regard to different notions of treewidth has been studied e.g. by Marko Samer and Stefan Szeider [SS06].

A different generalization of the satisfiability problem is the *constraint satisfaction problem (CSP)*. Again Samer and Szeider did a classification with respect to parameterized complexity along a wide lattice of parameters including several notions of structural treewidth [SS10].

## 5.2 Modal Logic (ML)

Modal logic on serial frames, i.e. in the frame class D, can be seen as a kind of restricted version of CTL in the sense that the only allowed temporal operators are AX and EX, where $\mathsf{AX} \mathrel{\widehat{=}} \Box$ and $\mathsf{EX} \mathrel{\widehat{=}} \Diamond$, as illustrated in Figure 5.2. It was shown by Praveen that modal satisfiability is fixed-parameter tractable when restricted to modal CNF (*conjunctive normal form*) and parameterized by *structural treewidth* and *modal depth*. The structural treewidth of a modal CNF formula is similarly defined as in Section 5.1. The modal depth of a formula $\varphi$ is the maximal nesting depth of modal operators in $\varphi$.

Formally:

$$\begin{array}{llll}
\mathrm{md}(x) & := 0 \text{ if } x \in \mathbf{PS} & \mathrm{md}(\top) & := 0 \\
\mathrm{md}(\varphi \wedge \psi) & := \max\{\mathrm{md}(\varphi), \mathrm{md}(\psi)\} & \mathrm{md}(\neg\varphi) & := \mathrm{md}(\varphi) \\
\mathrm{md}(\Box\varphi) & := \mathrm{md}(\varphi) + 1 & &
\end{array}$$

An example of a property that is not expressible in modal logic is the *reachability property*: From the world $w$, is there a sequence of successors such that one finally arrives in a world $w'$ where $\psi$ holds? In general, every formula $\varphi$ is satisfiable if and only if it has a model with depth bounded by $\mathrm{md}(\varphi)$. Here, the depth of a Kripke structure is the maximum depth of a world, where the depth of a world $w$ is the length of the shortest path to $w$ from the root of the structure.

Hence reachability over a path with arbitrary length is not expressible with $\Box$ and $\Diamond$ [BdV01]. This however not longer holds if the Kripke structures are restricted to e.g. frames with a transitive successor relation. Such restrictions can have significant impact on the hardness of the satisfiability problem. This holds in the classical complexity theory as well as when parameterized by modal depth.

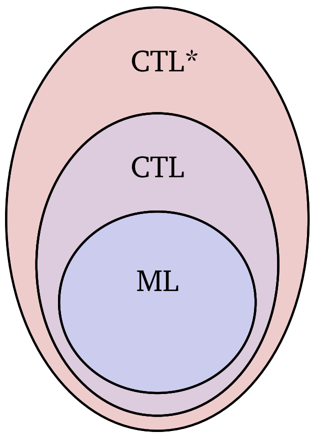
**Theorem 5.2** (Ladner, 1977). ML-SAT *is* **PSPACE**-*complete under* $\leq^{\log}_{\mathrm{m}}$-*reductions. If the frames are restricted to the class S5 of* Euclidean frames[1]*, the resulting satisfiability problem is* **NP**-*complete under* $\leq^{\log}_{\mathrm{m}}$-*reductions and therefore as easy as propositional satisfiability [Lad77].*

**Theorem 5.3** (Praveen, 2013). ML-SAT *is* **FPT** *on CNF formulas when parameterized by structural treewidth and modal depth,* **W**[1]-*hard when restricted to transitive frames, and again* **FPT** *for any frame class that is additionally Euclidean [Pra13].*

Note that the classical complexity drops to **NP** for the class S5 because of the *polynomial model property*: A modal formula $\varphi$ is satisfied in S5 if and only if it is satisfied by an Euclidean model with size $|\varphi|^{\mathcal{O}(1)}$. This is not the case for the other frame classes since there are formulas that enforce non-Euclidean models of exponential size.

In the parameterized version, the **FPT** results stem from a similar property: Every satisfiable modal formula $\varphi$ has a model of depth at most $\mathrm{md}(\varphi)$ which can also be constructed Euclidean, but not necessarily transitive. The bounded model depth allows to express satisfiability by an MSO formula over the formula structure which essentially quantifies sets of subformulas corresponding to worlds in a model, using $\mathcal{O}(\mathrm{md}(\varphi))$ quantifiers overall. Then **FPT** membership follows from Courcelle's theorem as a special case of Theorem 4.11.

[1]A binary relation $R$ is *Euclidean* if $wRu \wedge wRv \rightarrow uRv$.



▢ **PSPACE**-complete
▢ **EXP**-complete
▢ **2EXP**-complete

Figure 5.2: Modal logic embedded into temporal logics

## 5.3 Computation Tree Logic (CTL)

In CTL we distinguish between the operator fragments depicted in Figure 5.3. Every black line between two fragments $T, T'$ represents an increase of expressive power in the sense that every formula expressible in $T$ can also be expressed using the operators in $T'$ but not the other way around. The operators $\mathsf{EF}, \mathsf{EG}, \mathsf{EX}$ are not considered explicitly because they can be expressed in their dual operators $\mathsf{AG}, \mathsf{AF}, \mathsf{AX}$ together with negation:

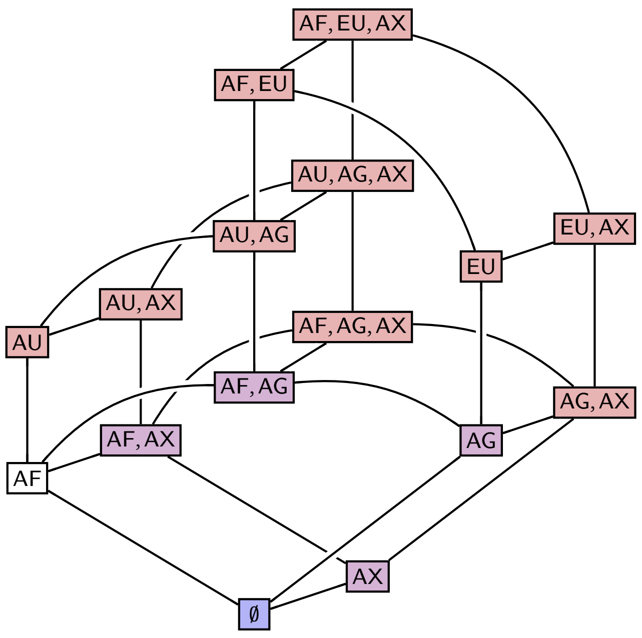
**Definition 5.4.** For sets $T$ and $T'$ of temporal operators, $T$ is *simulated* by $T'$, written $T \sqsubseteq T'$, if every temporal operator in $T$ is also in $T'$ or can be written with operators from $T'$ together with Boolean connectives. Write $T \equiv T'$ if $T \sqsubseteq T'$ and $T' \sqsubseteq T$.

**Example 5.5:**
$\{\mathsf{AF}, \mathsf{AX}\} \equiv \{\mathsf{EX}, \mathsf{EG}\}$. $\{\mathsf{F}, \mathsf{X}\} \sqsubseteq \{\mathsf{U}, \mathsf{X}\}$. $\{\mathsf{A}, \mathsf{F}\} \equiv \{\mathsf{E}, \mathsf{A}, \mathsf{G}, \mathsf{F}\}$. $\{\mathsf{AU}\} \sqsubseteq \{\mathsf{AF}, \mathsf{EU}\}$.

Let $T$ be a set of temporal operators. For the set of CTL formulas using only the operators that can be simulated using $T$, write $\mathbf{CTL}(T)$ (we will omit the set brackets when appropriate). Use the shortcuts CTL-SAT$(T)$ := $\mathbf{CTL}(T) \cap$ CTL-SAT and $\mathbf{CTL}_{\text{NNF}}(T) := \mathbf{CTL}(T) \cap \mathbf{CTL}_{\text{NNF}}$.

Following classical complexity results are known from Meier [Mei11]:

- The fragments $\emptyset$ and $\{\mathsf{AF}\}$ are **NP**-hard, as this property already arises for propositional logic, and in **NP** for the $\emptyset$ case.



🟦 **NP**-complete
🟪 **PSPACE**-complete
🟥 **EXP**-complete
☐ Open

Figure 5.3: Classical complexity of CTL satisfiability

- The fragments { AG }, { AG, AF }, { AX }, { AX, AF } are PSPACE-complete. Even if this comes with a slightly modified proof when AX is not available, the hardness directly carries over from the possibility to embed model logic in these fragments. **PSPACE** membership on the other hand is obtained using an algorithm with polynomial recursion depth.

- Any fragment containing { AU }, { EU } or { AG, AX } can express computations of Alternating Turing Machines with polynomial space, which itself is equivalent to exponential time. Therefore the satisfiability problem of these fragments is **EXP**-complete.

In this section the satisfiability problem of CTL is studied with regard to parameterization by treewidth of syntactic structures and nesting depth of temporal operators. Certain operator fragments turn out to be fixed-parameter tractable for reasons that are similar to the classical case with some exceptions, namely having models of bounded depth. Those fragments which can enforce "long models" are provided with a matching lower bound, i.e. **W**[1]-hardness.

The parameter *temporal depth*, td, is defined similar to modal logic:

$$\begin{aligned}
&\mathrm{td}(x) := 0 \text{ if } x \in \mathbf{PS} && \mathrm{td}(\top) := 0\\
&\mathrm{td}(\varphi \wedge \psi) := \max\{\mathrm{td}(\varphi), \mathrm{td}(\psi)\} && \mathrm{td}(\neg\varphi) := \mathrm{td}(\varphi)\\
&\mathrm{td}(T\varphi) := \mathrm{td}(\varphi) + 1 \quad \text{if } T \in \{\mathsf{AX}, \mathsf{AF}, \mathsf{AG}, \mathsf{AU}, \mathsf{EX}, \mathsf{EF}, \mathsf{EG}, \mathsf{EU}\}
\end{aligned}$$

## 5.3.1 The tractable fragment AX

The following result regarding a tractable problem makes intensive use of Courcelle's theorem and therefore relies on MSO-definability of the corresponding decision problem. The main idea is to express the existence of a model in an MSO formula by consecutively quantifying each world and as many successors as necessary. The syntactical structures of formulas are used as the logical structures for the MSO formula. Then a particular world is quantified as a unary relation, and therefore a subset of individuals which essentially is the set subformulas labeled in this world.

This representation of a model has its pitfalls: To check consistency of a quantified model, in particular it has to be ensured that every "eventuality" imposed by temporal operators is fulfilled in a future world. Strictly spoken, the linear order of quantifiers is not suitable for expressing cyclic models, but every acyclic "tree" model of a temporal formula has to be infinite (see Chapter 3). Therefore we rely on the fragments being finitely verifiable, i.e. that it is sufficient to inspect only a small number of worlds of an infinite tree model to prove its correctness.

First we introduce the notions of *quasi-models* which are required for working with such infinite tree models.

**Definition 5.6** (Quasi-labels)**.** Let $\Gamma$ be a set of CTL formulas in negation normal form. A *quasi-label* $H$ over $\Gamma$ is a subset of $\Gamma$ such that every element is *propositionally implied*, i.e. for every formula $\alpha \wedge \beta \in H$ it follows that $\alpha \in H$ and $\beta \in H$ and for every $\alpha \vee \beta \in H$ it follows that $\alpha \in H$ or $\beta \in H$.

These quasi-label conditions are also called *local conditions*.

Quasi-labels which contain both $\psi$ and $\neg\psi$ for a formula $\psi$ are *inconsistent*, otherwise they are *consistent*.

**Definition 5.7** (Quasi-structures)**.** A *quasi-structure* is almost a Kripke structure. It is a tuple $(W, R, L)$ with the following properties:

- $W$ is a set of worlds,
- $R \subseteq W \times W$ is a serial successor relation,

- $L\colon W \mapsto \mathfrak{P}(\mathbf{CTL}_{\mathrm{NNF}})$ meaning that for every $w \in W$ the set $L(w)$ is a quasi-label over $\mathbf{CTL}_{\mathrm{NNF}}$.

Call a quasi-structure *finite* if $W$ is finite. A quasi-structure is *consistent* if $L(w)$ is consistent for every $w \in W$.

The main difference to a usual model is that the latter can only label simple propositional variables to worlds.

**Definition 5.8** (Quasi-models)**.** A *quasi-model* of a formula $\varphi \in \mathbf{CTL}_{\mathrm{NNF}}$ is a tuple $(\mathcal{M}, w_0)$ with $\mathcal{M} = (W, R, L)$ being a quasi-structure and the following quasi-model conditions satisfied:

- $\mathcal{M}$ is consistent,
- $\varphi \in L(w_0)$,
- $L(w)$ is a quasi-label over $\mathrm{SF}(\varphi)$ for every $w \in W$.

Additionally the temporal subformulas have to be satisfied for every $w \in W$:

- if $\mathsf{AX}\psi \in L(w)$ then $\forall \pi \in \Pi(w) :\ \psi \in L(\pi[2])$
- if $\mathsf{EX}\psi \in L(w)$ then $\exists \pi \in \Pi(w) :\ \psi \in L(\pi[2])$
- if $\mathsf{AF}\psi \in L(w)$ then $\forall \pi \in \Pi(w)\ \exists i \in \mathbb{N} :\ \psi \in L(\pi[i])$
- if $\mathsf{EF}\psi \in L(w)$ then $\exists \pi \in \Pi(w)\ \exists i \in \mathbb{N} :\ \psi \in L(\pi[i])$
- if $\mathsf{AG}\psi \in L(w)$ then $\forall \pi \in \Pi(w)\ \forall i \in \mathbb{N} :\ \psi \in L(\pi[i])$
- if $\mathsf{EG}\psi \in L(w)$ then $\exists \pi \in \Pi(w)\ \forall i \in \mathbb{N} :\ \psi \in L(\pi[i])$
- if $\mathsf{A}[\psi \mathsf{U} \chi] \in L(w)$ then $\forall \pi \in \Pi(w)\ \exists i \in \mathbb{N} :\ \chi \in L(\pi[i])$ and $\forall j, 1 \leq j < i :\ \psi \in L(\pi[i])$
- if $\mathsf{E}[\psi \mathsf{U} \chi] \in L(w)$ then $\exists \pi \in \Pi(w)\ \exists i \in \mathbb{N} :\ \chi \in L(\pi[i])$ and $\forall j, 1 \leq j < i :\ \psi \in L(\pi[i])$

A quasi-model is *state-minimal* for a formula $\varphi$ if deletion of a state leads to the structure not longer satisfying $\varphi$ in the quasi-model sense. It is *label-minimal* if every quasi-label is minimal, i.e. deletion of a subformula from a quasi-label leads to violation of a quasi-label condition w. r. t. $\varphi$. If a quasi-model is state-minimal and label-minimal just call it *minimal*.

In their definition, quasi-models expose similarities to the so-called *Hintikka structures* which themselves are also structures with extended labelings. Different notions of quasi-models, pseudo-models or Hintikka structures are defined in this context: Either quasi-labels require to have every formula labeled that is true at the given state; or they require to have at least the formulas labeled that are required by another formula labeled somewhere in the model. The definitions given above make use of the second meaning.

The fragmentary problem CTL-SAT($\mathsf{AX}$) is already **PSPACE**-complete in classical complexity theory, whereas the hardness disappears in the parameterization by temporal depth. The reason for this is similar as in modal logic and is called *finite tree model property*.

We will indeed start proving the tractability of CTL-SAT($\mathsf{AX}$) by restricting ourselves to easily verifiable models that are characteristic for temporal $\mathsf{X}$-formulas and as well for modal formulas.

**Definition 5.9.** A Kripke model $(\mathcal{M}, r)$ is *tree-like* if the following properties hold for every world $w$ of $\mathcal{M}$:

- either $w = r$ and $w$ has no predecessors,
- or $w$ has exactly one predecessor,
- or $w$ has exactly two predecessors of which one must be $w$ itself (inducing a loop in $\mathcal{M}$) and $w$ has only itself as a successor ($w$ is a *pseudo-leaf*)

**Definition 5.10.** Let $\mathcal{T}$ be a tree quasi-model for a formula $\varphi \in \mathbf{CTL}_{\mathrm{NNF}}$. We say that $\mathcal{T}$ is *branching-normalized* if for every world $w$ in $\mathcal{T}$ it holds: For every distinct $\mathsf{E}$-prefixed subformula $\psi$ labeled in $w$ there is exactly one successor of $w$ where $\psi$ is labeled.

**Theorem 5.11.** *Let $\varphi \in \mathbf{CTL}_{\mathrm{NNF}}(\mathsf{AX}, \mathsf{EX})$. Then the following statements are equivalent:*

*(1) $\varphi$ is satisfiable.*

*(2) $\varphi$ has a tree quasi-model.*

*(3) $\varphi$ has a branching-normalized minimal tree quasi-model.*

*(4) $\varphi$ has a tree-like Kripke model of depth $td(\varphi)$.*

*Proof.* $(4) \Rightarrow (1)$ is clear.

$(1) \Rightarrow (2)$:

A finite model $(\mathcal{M}, r)$ of $\varphi$ can easily be transformed into such a tree $\mathcal{T}$ by "unrolling" $\mathcal{M}$ beginning at the root $r$, repeatedly appending copies of worlds $v_1, \ldots, v_n$ to a leaf $w$, if $v_1, \ldots, v_n$ are the corresponding successors of $w$ in the finite model. Since $\mathcal{M}$ is a model of $\varphi$ we can label $\varphi$ at the root of $\mathcal{T}$ and extend every quasi-label $L(w)$ of $\mathcal{T}$ to fulfill the quasi-label conditions.

(2) $\Rightarrow$ (3):

We normalize the tree quasi-model $\mathcal{T}$ as follows. Assume that $\mathcal{T}$ is minimal since we can always chop worlds and delete formulas from quasi-labels if it is not.

Let then $w$ be a state of $\mathcal{T}$ that has the E-prefixed formulas $\mathsf{EX}\gamma_1, \dots, \mathsf{EX}\gamma_n$ labeled. Then there are (not necessarily distinct) successors $w_1, \dots, w_n$ of $w$ which each have the corresponding $\gamma_1, \dots, \gamma_n$ labeled. Every formula $\gamma_i$ is labeled in exactly one successor of $w$ (otherwise $\mathcal{T}$ is not minimal). If there is a successor where $\gamma_i, \gamma_j$ for $i \neq j$ are labeled, duplicate the subtree below $w_i$ into two subtrees s. t. $w_i$ has labeled $\gamma_i$ but not $\gamma_j$ and the copy of $w_i$ has labeled $\gamma_j$ but not $\gamma_i$. The remaining formulas are labeled in both states. Therefore a branching-normalized minimal tree pseudo-model of $\varphi$ can be constructed.

(3) $\Rightarrow$ (4):

For modal logic the result is proven as in [GO07, Lemma 35] (*finite tree model property*). Since the temporal operators AX and EX correspond (besides seriality) to the modal operators $\Box$ and $\Diamond$, the proof basically is the same as for modal logic.

Alternatively consider a minimal tree quasi-model $\mathcal{T}$. Write $\mathrm{td}(w)$ as the maximal temporal depth of a formula labeled in $w$, i.e. $\mathrm{td}(w) := \max\{\, \mathrm{td}(\psi) \mid \psi \in L(w) \,\}$. Then for every world $w$ and successor $w'$ it holds that $\mathrm{td}(w') < \mathrm{td}(w)$: Assume $\mathrm{td}(w') \geq \mathrm{td}(w)$ for contradiction. Let $\psi$ be a formula of maximal temporal depth labeled in $w'$ such that $\psi$ is not implied by a local quasi-label condition. Such a formula $\psi$ must exist in $L(w)$. However $\psi$ can then only be necessary because in $w$ there is a formula $\mathsf{AX}\psi$ or $\mathsf{EX}\psi$ labeled. But $\mathrm{td}(w) \leq \mathrm{td}(w')$ and $\psi$ had maximal temporal depth in $w'$.

This shows that every quasi-label at depth $\mathrm{td}(\varphi)$ can merely contain propositional formulas. In the minimal infinite tree model replace the successors of such worlds by a self-loop, then delete every labeled formula except propositional variables. The resulting structure is a finite Kripke model (because of the finite branching of $\mathcal{T}$), tree-like, of depth $\mathrm{td}(\varphi)$ and a model of $\varphi$. $\Box$

*Remark:* For many other logics often the so-called *filtration technique* can be applied to obtain finite models, which is used as follows. Take an infinite tree pseudo-model $\mathcal{T}$. Since $\mathcal{T}$ can be assumed minimal we have $\mathcal{T}$ to have only subformulas labeled that indeed occur in $\varphi$. Hence we have only finitely many different quasi-labels appearing in $\mathcal{T}$. The worlds then form an equivalence relation $\equiv$ in the sense that $w \equiv w'$ iff. $L(w) = L(w')$. $\equiv$ has finite index (and indeed index $\leq 2^{|\varphi|}$). Doing a quotient construction, i.e. collapsing equivalence classes of states into a single state results in a finite structure. Unfortunately this construction introduces additional loops in the model which causes problems with A operators. Therefore an additional step is required. A correct construction is done by Emerson, which is basically a quotient construction followed by an extraction of acyclic substructures that satisfy A-formulas. This yields the finite model [Eme90, Theorem 6.14].

**Theorem 5.12.** *The problem* CTL-SAT(AX) *is fixed-parameter tractable when parameterized by temporal depth and structural treewidth, i.e.* $(\text{CTL-SAT}(\mathsf{AX}), \kappa) \in \mathbf{FPT}$ *for* $\kappa(\varphi) := td(\varphi) + tw(\mathcal{S}_\varphi)$.

*Proof.* Let $n \in \mathbb{N}$ be a natural number. Then the following MSO formulas existentially quantify a consistent quasi-structures of a formula $\varphi \in \mathbf{CTL}_{\text{NNF}}(\mathsf{AX}, \mathsf{EX})$.

$$\begin{aligned}\theta_{\text{Boolean}}(H) := \forall x \forall y \forall z \\ H(x) \rightarrow \Big(&\mathcal{R}_{\cdot\wedge}(x,y) \wedge \mathcal{R}_{\wedge\cdot}(x,z) \rightarrow (H(y) \wedge H(z)) \;\wedge \\ &\mathcal{R}_{\cdot\vee}(x,y) \wedge \mathcal{R}_{\vee\cdot}(x,z) \rightarrow (H(y) \vee H(z)) \;\wedge \\ &\mathcal{R}_{\neg}(x,y) \rightarrow \neg H(y)\Big)\end{aligned}$$

$$\theta^0_{\text{world}}(H) := \;\theta_{\text{Boolean}}(H)$$

$$\begin{aligned}\theta^n_{\text{world}}(H) := \;&\theta_{\text{Boolean}}(H) \wedge \exists H_{\mathsf{AX}}\Big( \\ &\forall x \big(H_{\mathsf{AX}}(x) \leftrightarrow (H(x) \wedge \exists y\, \mathcal{R}_{\mathsf{AX}}(x,y))\big) \wedge \\ &\exists H' \left(\theta^{n-1}_{\text{world}}(H') \;\wedge\; \forall z H_{\mathsf{AX}}(z) \rightarrow H'(z)\right) \wedge \\ &\forall x \forall y \;(H(x) \wedge \mathcal{R}_{\mathsf{EX}}(x,y)) \\ &\qquad\qquad \rightarrow \left(\exists H'\, \theta^{n-1}_{\text{world}}(H') \wedge H'(y) \wedge \forall z H_{\mathsf{AX}}(z) \rightarrow H'(z)\right)\Big)\end{aligned}$$

Now it holds for a CTL formula $\varphi$ in NNF that $\varphi \in \text{CTL-SAT}(\mathsf{AX}, \mathsf{EX})$ if and only if $\mathcal{S}_\varphi \models \exists H\, \exists x\, \mathcal{R}_{\text{root}}(x) \wedge H(x) \;\wedge\; \theta^{\text{td}(\varphi)}_{\text{world}}(H)$.

## Correctness

From Theorem 5.11 we know that it is sufficient to verify a tree quasi-model of $\varphi$ only up to depth $\leq \text{td}(\varphi)$. The formula $\theta_{\text{Boolean}}$ makes sure that every quantified quasi-label $H$ is consistent and $\wedge, \vee$-expressions are satisfied. This check is also sufficient at maximal depth of the model where only propositional formulas are labeled. The formula $\theta^n_{\text{world}}(H)$ for $n > 0$ additionally requires at least one successor world to exist at the next level to ensure seriality, and also that for every EX-formula in the quasi-label $H$ there is a branching to a successor of the current world, satisfying exactly that

EX-formula. Also, every successor of the current world is required to fulfill any AX-formula labeled in the current quasi-label $H$. This procedure is sufficient due to the normalized-branching property.

### FPT Runtime

Clearly there is a computable function $g(n) = \exists H \exists x \mathcal{R}_{\mathrm{root}}(x) \land H(x) \land \theta^{n}_{\mathrm{world}}(H)$. Also there is a function $f$ with $|f(\varphi)| = \mathrm{td}(\varphi)$ that is computable in linear time. Since $\kappa(\varphi) = \mathrm{td}(\varphi) + \mathrm{tw}(\mathcal{S}_{\varphi})$, both tw and $f$ are $\kappa$-bounded. We may assume that the formula $\varphi$ is in NNF since such a conversion can again be done in linear time.

From $\varphi \in \text{CTL-SAT}(\mathsf{AX}, \mathsf{EX}) \iff \mathcal{S}_{\varphi} \models g(|f(\varphi)|)$ and Theorem 4.11 follows the fixed-parameter tractability. □

## 5.3.2 Intractable fragments

AX- and EX-formulas can only speak about structures in a restricted way, i.e. they cannot speak about worlds that have greater depth than the temporal depth of the formula. Other CTL operators however have a greater expressive power by being able to enforce "deep" models, i.e. the model depth cannot be bounded by the parameter alone, lifting the algorithmic complexity of the satisfiability problem outside the range of fixed-parameter tractability. This will be proven in two steps: First we will introduce another **W**[1]-hard parameterized problem and then show that it can be formulated in terms of different CTL operators.

### Partitioned Weighted Satisfiability

Previously used by Praveen to show intractability of transitive modal logic, the problem of *partitioned weighted satisfiability* further generalizes the problem of weighted satisfiability [Pra13].

An instance is a tuple $I = \big(\varphi, k, (X_i)_{i\in[k]}, (C_i)_{i\in[k]}\big)$ where $\varphi$ is a propositional formula in CNF over variables $x_1, \ldots, x_n$, and the variables are partitioned into disjoint sets $X_1, \ldots, X_k$. $[k]$ is a shorthand for the set $\{1, \ldots, k\}$. Each partition $X_i$ has an assigned *capacity* $C_i \in \mathbb{N}$.

An assignment $\theta$ is called *saturated* for an instance $I$ if in every partition $X_i$ there are exactly $C_i$ variables set to one by $\theta$.

We define the problem of finding a saturated assignment:

$$\textsc{PartWeight-SAT} := \left\{ I = \big(\varphi, k, (X_i)_{i\in[k]}, (C_i)_{i\in[k]}\big) \;\middle|\; \begin{array}{l} \varphi \text{ is CNF and has a saturated} \\ \text{satisfying assignment } \theta \end{array} \right\}$$

The parameter is $\kappa(I) := \mathrm{pw}(G_{\varphi}) + k$, where $G_{\varphi}$ is the *primal graph* of $\varphi$.

The primal graph of a propositional formula $\varphi$ in CNF is the graph that contains a vertex $v_x$ for every variable $x$ in $\varphi$ and edges $(v_x, v_y)$ if $x$ and $y$ are distinct variables that occur together in at least one clause.

As a shortcut for the parameterized problem (PARTWEIGHT-SAT, $\kappa$) we will write p-PW-SAT.

*Remark:* At this point it is important to understand the different notions of graphs representing a formula. Another known notion besides the primal graph is the *incidence graph* of $\varphi$ that in contrast contains a vertex $v_x$ for every variable $x$ in $\varphi$, one vertex $v_C$ for every clause $C$ of $\varphi$ and edges $(v_x, v_C)$ if $x \in C$.

The primal graph in general has a much simpler structure and lower treewidth and pathwidth than the incidence graph. The next results require the use of primal graphs to maintain low pathwidths, whereas the syntactical structures defined earlier contain an embedded incidence graph of their formula (but are not restricted to formulas in CNF).

**Theorem 5.13** (Praveen, 2013 [Pra13]). *The problem p-PW-SAT is* $\mathbf{W}[1]$*-hard.*

*Proof.* To prove the hardness we use another $\mathbf{W}[1]$-hard problem, NUMBERLISTCOLORING (NLC), which was introduced by Fellows et al. [FFL+07]

An instance of NLC is again a tuple $I = (V, E, k, (L_v)_{v\in V}, (C_i)_{i\in[k]})$. Here, $(V, E)$ is an undirected simple graph for which a vertex coloring has to be found; $k$ is the total number of colors. The parameter is $k$ plus the pathwidth of $(V, E)$. For a vertex $v \in V$, $L_v \subseteq [k]$ is the list of colors that are available for coloring the vertex $v$. Similar to p-PW-SAT, $C_i$ is the capacity of color $i$. A coloring of a graph is called *proper* if no two adjacent vertices share the same color, and every color of a vertex is picked from its corresponding list of allowed colors, and additionally we call a coloring *saturated* if the number of occurences of any color is equal to its capacity.

We use a slight variation of the problem: The set pw2-NLC is the set of all instances $I$ such that the given graph has a proper saturated coloring but additionally has pathwidth at most 2. Fellows et al. showed that this restricted problem is already $\mathbf{W}[1]$-hard when using only $k$ as parameter, the total number of colors.

Now the reductions to the problem p-PW-SAT works as follows: First check if the input graph has pathwidth at most 2 in linear time using Theorem 4.3 (Bodlaender's algorithm); return an arbitrary negative instance if pw $> 2$. Otherwise continue as follows. For every combination of a vertex $v$ and one of its colors $i \in L_v$, introduce a propositional variable $q_{v,i}$. The variables are partitioned in classes depending on which color they represent:

$$X_i := \{\, q_{v,i} \mid v \in V, i \in L_v \,\}$$

The propositional formula $\varphi$ is defined to check if the coloring is proper:

$$\varphi := \bigwedge_{v\in V} \underbrace{\Big(\bigvee_{i\in L_v} q_{v,i}\Big)}_{T_1\text{: } \geq 1 \text{ color for } v} \wedge \bigwedge_{v\in V} \underbrace{\bigwedge_{\substack{i,j\in L_v\\ i\neq j}} (\neg q_{v,i} \vee \neg q_{v,j})}_{T_2\text{: } \leq 1 \text{ color for } v} \wedge \bigwedge_{(u,v)\in E} \underbrace{\bigwedge_{i\in L_u\cap L_v} (\neg q_{u,i} \vee \neg q_{v,i})}_{T_3\text{: different colors for } u \text{ and } v}$$

We define $h$ as the function that computes the mapping

$$I = \big(V, E, k, (L_v)_{v\in V}, (C_i)_{i\in[k]}\big) \mapsto \big(\varphi, k, (X_i)_{i\in[k]}, (C_i)_{i\in[k]}\big) = I'.$$

$h$ is an fpt-reduction from pw2-NLC to p-PW-SAT: The described transformations are possible in polynomial time. Also a proper vertex coloring of $(V, E)$ results in a satisfying assignment of $\varphi$, while any satisfying assignment of $\varphi$ has to be of the form that it induces a proper vertex coloring of $(V, E)$.

Since the weights and capacities of the partitions in the output instance correspond to the color lists and capacities of the input instance and since the selection of a color for a vertex sets exactly one variable of $\varphi$ to one, there is a saturated proper coloring of $(V, E)$ if and only if there is a saturated satisfying assignment of $\varphi$.

For $h$ being an fpt-reduction it remains to show that the parameter of $I'$ is bounded by the parameter of $I$ which is $\kappa(I') = \mathrm{pw}(G_\varphi) + k$. The parameter of the left hand side is $k$, hence we finish the proof by bounding the resulting pathwidth of $G_\varphi$ in $k$.

Consider an optimal path decomposition $\mathcal{P}$ of $(V, E)$. $\mathcal{P}$ has width two. Now construct a decomposition $\mathcal{P}'$: For each bag $B$ of $\mathcal{P}$, replace every vertex $v \in B$ by the set $\{\, q_{v,i} \mid i \in L_v \,\}$. Every bag of $\mathcal{P}'$ has now at most size $3k$.

The resulting decomposition $\mathcal{P}'$ is a correct path decomposition of $G_\varphi$, the primal graph of $\varphi$: For every variable $q_{v,i}$ the bags containing it induce a non-empty connected subpath in $\mathcal{P}'$ since the vertex $v$ did the same in $\mathcal{P}$ before. For every edge $(q_{v,i}, q_{v',i'})$ of $G_\varphi$ to be covered in $\mathcal{P}'$ we show that there is a bag $B'$ in $\mathcal{P}'$ that contains both variables.

Assume that $v = v'$. Then the edge must stem from the two variables being together in a clause of the $T_1$ or $T_2$ type (ensuring that $v$ receives exactly one color of its list). But then every bag previously containing $v$ also contains $q_{v,i}$ and $q_{v,i'}$ in $\mathcal{P}'$.

The other case is $v \neq v'$. Such an edge always stems from a clause of type $T_3$, i.e. ensuring that the coloring is proper. But a clause of type $T_3$ is only added for vertices $v, v'$ that are adjacent in $(V, E)$. Therefore the path composition $\mathcal{P}$ already had a bag containing both $v$ and $v'$. Similar to the argumentation above the corresponding bag of $\mathcal{P}'$ contains $q_{v,i}$ and $q_{v',i'}$ and therefore covers the edge. $\square$

### Expressing partitioned weighted satisfiability in CTL

Praveen proved the **W**[1]-hardness of modal satisfiability in transitive frames by formulating the problem of partitioned weighted satisfiability in modal logic. The rough idea is as follows: Consider an instance $I = \big(\varphi, k, (X_i)_{i\in[k]}, (C_i)_{i\in[k]}\big)$ of PARTWEIGHT-SAT with $\varphi$ containing variables $q_1, \dots, q_n$. Then a modal formula is constructed which enforces the existence of worlds $w_0 \mapsto w_1 \mapsto w_2 \mapsto \dots \mapsto w_n$ forming a chain. For $i \geq 1$ the world $w_i$ then has the purpose to assign to $q_i$ either $\top$ or $\bot$. In the last world $w_n$ another subformula eventually checks the number of variables set to $\top$ in each partition; the number has to equal the respective capacity.

Crafting such a type of formula is not a new technique, but doing it in the context of a parameterized reduction requires careful assembling of subformulas in a way that keeps the pathwidth low. Significant parts of technical work in [Pra13] are devoted to prove that the structural pathwidth of the produced formula is bounded by the parameter.

We will cover the different CTL fragments in three steps: First the reduction from saturated satisfiability is presented in detail for general CTL. The transition from modal logic to temporal logic that is required in this step is not hard. The second part of this section describes how to obtain a path decomposition of low structural pathwidth from the resulting formula. In the final step the result is transferred to the remaining fragments of CTL.

**Lemma 5.14.** *For every instance $I$ of* PARTWEIGHT-SAT *there is a temporal formula $\psi(I) \in$ **CTL** that is satisfiable if and only if $I$ has a saturated satisfying assignment.*

*Proof.* The formula $\psi(I)$ is a conjunction of several subformulas which will be presented next. In the construction we assume reasonable (and efficiently checkable) properties of the instance, e.g. that the capacity of a partition is at most its size, and that $\varphi$ is a syntactically valid CNF.

The original formula $\varphi$ should be true in the initial world $w_0$ to ensure its satisfiability.

$$\psi[\text{formula}] \quad := \varphi \tag{5.1}$$

Enforce a model that contains the chain of worlds mentioned above. For this we use "depth" variables $d_0, d_1, d_2, \dots$ that have to be labeled in the desired order.

$$\psi[\text{depth}] \quad := \mathsf{AG} \bigwedge_{i=0}^{n} [(d_i \land \neg d_{i+1}) \to \mathsf{AX}(d_{i+1} \land \neg d_{i+2})] \tag{5.2}$$

Let $Q$ be the subset of the variables $\{q_1, \ldots, q_n\}$ that is labeled in $w_0$. $\psi[\text{formula}]$ ensures that $Q$ represent a satisfying assignment of $\varphi$. To check the saturation of $Q$ w. r. t. the given capacities the set $Q$ should be repeatedly labeled in each consecutive world.

$$\psi[\text{fixed-}Q] \quad := \mathsf{AG} \bigwedge_{i=1}^{n} [q_i \leftrightarrow \mathsf{AX} q_i] \tag{5.3}$$

Next let $p(i) \in [k]$ denote the partition number of $q_i$. We introduce new propositional variables $\top^{\uparrow}_{p(i)}$ which signal that the number of labeled variables from partition $p(i)$ has increased.

$$\psi[\text{signal}] \quad := \mathsf{AG} \bigwedge_{i=1}^{n} \left[ (d_i \wedge \neg d_{i+1}) \rightarrow \left( q_i \leftrightarrow \top^{\uparrow}_{p(i)} \right) \right] \tag{5.4}$$

To count the total number of labeled variables per partition we need several variables named $\top^{j}_{p}$ here. Let $N(p)$ denote the size of the partition $p$. Then whenever an increment signal for partition $p$ is encountered, increment the counter from $j$ to the next integer $j+1$.

$$\begin{aligned}\psi[\text{count}] \quad := \mathsf{AG} \bigwedge_{p=1}^{k} \bigwedge_{j=0}^{N(p)} & \Big[ \top^{\uparrow}_{p} \rightarrow \left( \top^{j}_{p} \leftrightarrow \mathsf{AX}\, \top^{j+1}_{p} \right) \\ & \wedge \neg \top^{\uparrow}_{p} \rightarrow \left( \top^{j}_{p} \leftrightarrow \mathsf{AX}\, \top^{j}_{p} \right) \Big]\end{aligned} \tag{5.5}$$

Make sure that all the used variables do a consistent counting.

$$\psi[\text{monotone}] := \mathsf{AG} \left[ \bigwedge_{i=1}^{n} (d_i \rightarrow d_{i-1}) \wedge \bigwedge_{p=1}^{k} \bigwedge_{j=1}^{N(p)+1} \left( \top^{j}_{p} \rightarrow \top^{j-1}_{p} \right) \right] \tag{5.6}$$

Begin the counting correctly at the initial world.

$$\psi[\text{init}] \quad := d_0 \wedge \neg d_1 \wedge \bigwedge_{p=1}^{k} \left[ \neg \top^{\uparrow}_{p} \wedge \neg \top^{1}_{p} \right] \wedge \mathsf{AG} \bigwedge_{p=1}^{k} \top^{0}_{p} \tag{5.7}$$

Require that the counted number of positive variables per partition equals the capacity.

$$\psi[\text{target}] \quad := \mathsf{AG} \bigwedge_{p=1}^{k} \left[ d_{n+1} \rightarrow \left( \top^{C_p}_{p} \wedge \neg \top^{C_p+1}_{p} \right) \right] \tag{5.8}$$

It follows the proof of the lemma.

"⇒": Assume $\big(\varphi, k, (X_i)_{i\in[k]}, (C_i)_{i\in[k]}\big) \in$ PARTWEIGHT-SAT. Construct a model for $\psi(I)$ as follows. Start with the world $w_0$. $\varphi$ is satisfied by setting a saturated subset $Q$ of variables to one, i.e. the number of ones in a partition equals its capacity. Label $Q$ in the world $w_0$ so $\varphi$ is satisfied there as well (remember that $\varphi$ is purely propositional). Construct successor worlds $w_1, \ldots, w_{n+1}$, add a self-loop to $w_{n+1}$ and label the minimal amount of propositional variables in $w_0, w_1, \ldots$ to satisfy $\psi$[init], $\psi$[depth] and $\psi$[fixed-$Q$]. Note that this step is always possible. Now label the variables $\top^{\uparrow}_{p(i)}$ where necessary to fulfill $\psi$[signal]. This leads to exactly $C_{p(i)}$ occurences of $\top^{\uparrow}_{p(i)}$ since $Q$ is chosen saturated. For this reason, the formula $\psi$[count] allows for every partition $p$ that its counter is incremented exactly $C_p$ times. This construction does not violate the $\psi$[monotone] condition and allows to satisfy $\psi$[target] in the world $w_{n+1}$.

"⇐": Let $\mathcal{M}$ be a model of $\psi(I)$. It has to contain a tree of worlds which fulfill the conditions $\psi$[init], $\psi$[depth] and $\psi$[fixed-$Q$]. This tree of worlds can be shrunk to a path $w_0, \ldots, w_{n+1}$ by removing unnecessary labels and then identifying worlds at the same depth (which have to share exactly the same labels since no $\mathsf{E}$ operator is present). Because $\psi$[target] and $\psi$[monotone] have to hold in all worlds including $w_{n+1}$ (which has $d_{n+1}$ labeled), on the path $w_0, \ldots, w_{n+1}$ the $\top^{\uparrow}_p$ signal is labeled exactly $C_p$ times for every partition $p$. But $\psi$[signal] allows this if and only if the corresponding variable $q_i$ is set to one in the world $w_i$. This proves the existence of a saturating assignment $\theta$ for $\varphi$. $\theta$ also is satisfying for $\varphi$ since $w_0$ was labeled consistently. □

**Lemma 5.15.** *For an instance $I$ the formula $\psi(I)$ from Lemma 5.14 can be constructed to have a $\kappa$-bounded structural pathwidth, where $\kappa$ is the parameter of p-PW-SAT.*

*Proof.* Write $\mathcal{S}$ for $\mathcal{S}_{\psi(I)}$. Let $\mathcal{P}$ denote an optimal path decomposition of the primal graph of $\varphi$. It is to show how $\mathcal{P}$ can be extended to a path-decomposition $\mathcal{P}'$ of $\mathcal{S}$.

For this we first assume a special structure of $\mathcal{P}$, the *one-step addition property* that was already used in [Pra13]. It says that the bag $B_i$ in $\mathcal{P}$ introduces exactly one variable $q$, i.e. $q$ and only $q$ is present in $B_i$ but was not present in $B_{i-1}$. A bag that introduces no new variable can be deleted, and a bag introducing multiple variables can be split into multiple bags. Therefore assume the one-step addition property. Also use a renaming of bags $B_i$ and variables $q_i$ s. t. the bags are ordered along their number and bag $B_i$ introduces variable $q_i$.

The process of *augmenting a bag $B$ with $x$* means inserting a copy $B'$ of $B$ between $B$ and its successor bag and placing the additional element $x$ there. It holds that $|B'| = |B| + 1$. Augmenting a bag does preserve the one-step addition property in the sense that there always is a "leftmost" bag introducing a variable $q_i$.

Now use the following procedure to construct $\mathcal{P}'$:

1. For $1 \le p \le k$ add the variable $\top^{\uparrow}_p$ to every bag. This increases every bag size by the number $k$ of partitions.

2. $\psi$[formula]: $\varphi$ is in CNF. For every clause the primal graph of $\varphi$ has to contain a clique of its variables, therefore $\mathcal{P}$ already must contain a bag $B$ covering all these variables. Assume that this clause has size $m$ and is represented as a subformula $((((L_1 \vee L_2) \vee L_3) \vee \ldots) \vee L_m)$ where every literal $L_i$ is a variable $q$ or its negation $\neg q$. Augment $B$ with the $\vee$-nodes in the following way: Create $m$ copies of $B$. Add the $j$-th $\vee$-node to the $j$-th and the $(j+1)$-th copy of $B$. This results in the bags containing an $\vee$-node inducing a connected component. Refer to the outmost $\vee$-operators as the *primary* $\vee$*-nodes*. Proceed similar for the $\wedge$-nodes: Select two primary $\vee$-clauses that are "neighbors" in the path decomposition and add a $\wedge$-node to all bags that connect them.

   Remember that an optimal path decomposition can be computed in fpt by Corollary 4.4. Hence the structure $\mathcal{S}$ can be constructed in a way that allows the arguments above *a priori*, e.g. the placement of parentheses in $\psi(I)$ can always be chosen to associate literals in ascending order of variables in $\mathcal{P}$, and then associate clauses in ascending order of primary $\vee$-nodes. Then in the structure the edge linking these primary $\vee$-nodes and their common conjunction is covered and every bag receives at most two additional $\vee$-nodes and at most two additional $\wedge$-nodes. Figure 5.4 illustrates the procedure.

3. For $1 \le i \le n$ add the variables $d_{i-1}, d_i, d_{i+1}, d_{n+2}$ and the nodes representing their negations to the bag $B$ that introduces $q_i$ as well the nodes representing $(d_i \wedge \neg d_{i+1})$, $(d_{i+1} \wedge \neg d_{i+2})$ and $(d_i \rightarrow d_{i-1})$. Because of the one-step addition property regarding the $q_i$'s every inserted node induces a connected component. This adds a constant number of items to every bag. The same holds when adding the necessary AX-nodes and $\rightarrow$-nodes to every bag to cover each conjunct of $\psi$[depth]. Process its $\bigwedge$-node like before, adding at most two items to every bag, and also add its AG-node to every bag to completely cover this formula.

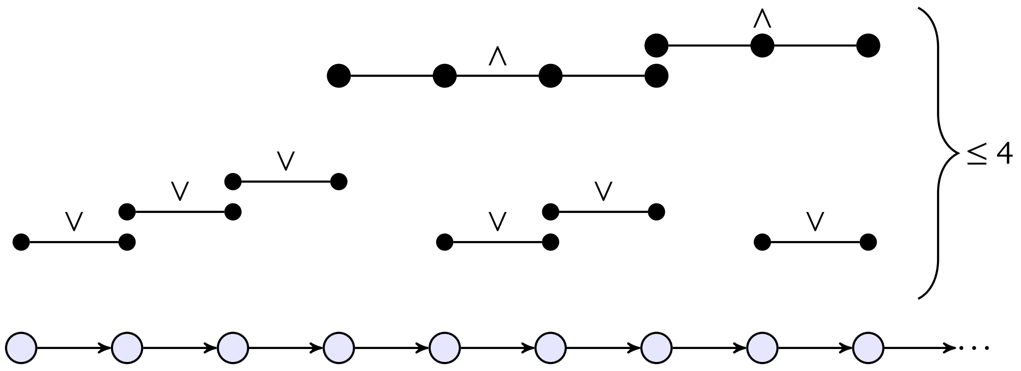


Figure 5.4: Bag augmentation for many small ∧,∨-nodes

4. $\psi$[fixed-$Q$]: Add the AX- and $\leftrightarrow$-nodes for the $i$-th conjunct exactly to the respective bags that introduce $q_i$, increasing the size of each bag by at most two. Process $\bigwedge$ and AG as before.

5. $\psi$[signal]: Again augment the bag introducing $q_i$ by the nodes $\leftrightarrow$, $(d_i \wedge \neg d_{i+1})$ and $\rightarrow$. The signal variable $\top^{\uparrow}_{p(i)}$ is already added in every bag, then add $\bigwedge$ and AG as before.

6. For the remaining formulas, jump to the last bag $B$. W.l.o.g. assume it contains the variable $d_{n+1}$. Let $C$ again be the maximum of the capacities. For every $1 \leq j \leq C$ now do the follwing: Append one bag that is a copy of $B$, but the $j$-th appended bag additionally contains $\top^j_p$, $\top^{j+1}_p$ and $\top^{j-1}_p$ for every partition $p$. This increases every bag size by $3k$. Since $d_{n+1}$ also is in these bags, the nodes representing subformulas containing $\top^j_p$'s of $\psi$[count], $\psi$[monotone], $\psi$[init] and $\psi$[target] can be added (each a constant number of items). Note that $\psi$[target] actually requires these nodes to be added after the last bag so the bags containing $d_{n+1}$ are connected.

7. The remaining subformulas of $\psi(I)$ are either conjunctions of size $C$ over $\top^j_p$'s (which are be covered by augmenting the bags introduced in the previous step by constantly many items), conjunctions of size $k$ and connectives of constant depth to link the subformulas together to get $\psi(I)$ (which all can be added to every bag).

The above construction results in a path decomposition of the structure $\mathcal{S}$ whose width is $\kappa$-bounded. The use of signal variables $\top^{\uparrow}_p$ is crucial for the construction; Them being the only “link” between variables $q_i$ and partition weight counters $\top^j_p$ is necessary for keeping the pathwidth low.

Observe that the pathwidth resulting from this construction will in general be higher than the similar approach for modal CNF in [Pra13] (which is only $4\text{pw}(G_\varphi) + 2k + 9$). The reason lies in the chosen structural representation of modal formulas in the previous work, which relies on CNF: For every clause, only one node is added to the structure, whereas the syntactical structure defined in Section 5.1 only allow binary connectives, i.e. of arity at most two. Therefore the number of $\wedge, \vee$-connectives in $\psi(I)$ itself is not $\kappa$-bounded, but by carefully choosing the association order of subformulas the items can be added with bag size increasing only by a constant number. □

**Lemma 5.16.** CTL-SAT(AG, AX) *is* $\mathbf{W}[1]$*-hard when parameterized by structural pathwidth (resp. treewidth) and temporal depth.*

*Proof.* Consider an $\leq^{fpt}$-reduction from the $\mathbf{W}[1]$-hard problem p-PW-SAT as described in Lemma 5.14. This is a valid $\leq^{fpt}$-reduction: The CTL formula $\psi(I)$ can be computed in fpt-time, and the parameter is bounded since the temporal depth of $\psi(I)$ is constant and the structural pathwidth is bounded by the parameter of p-PW-SAT according to Lemma 5.15. The same hardness result for treewidth instead of pathwidth follows from pathwidth being an upper bound for treewidth. $\square$

**Lemma 5.17.** CTL-SAT(AF, AX) *is* $\mathbf{W}[1]$*-hard when parameterized by structural pathwidth (resp. treewidth) and temporal depth.*

*Proof.* We re-prove Lemma 5.14 for the fragment $\{\mathsf{AF}, \mathsf{AX}\}$. Observe that the formulas were deliberately chosen to have AG operators occuring at temporal depth zero only. Instead of constructing $\psi(I)$ as a conjunction of those AG-formulas, construct a semantically equivalent AG-formula of conjunctions since $\bigwedge \mathsf{AG}\alpha_i \equiv \mathsf{AG} \bigwedge \alpha_i$ holds. Therefore assume $\psi(I)$ to be in the form $\psi(I) = \alpha \land \mathsf{AG}\gamma$ for a purely propositional formula $\alpha$. Then $\psi(I)' = \alpha \land \mathsf{EG}\gamma$ and equivalently $\psi(I)'' = \alpha \land \neg\mathsf{AF}\neg\gamma$ suffice for the proof of Lemma 5.14, because already from one single path of worlds $w_0, \ldots, w_n$ with correctly labeled depth propositions one can construct a saturated assignment of $\varphi$. The structural pathwidth of $\psi(I)''$ can be bounded almost exactly like in Lemma 5.15 since "$\alpha\, \land$" is constant and only "$\bigwedge \mathsf{AG}$" has to be changed to "$\neg\mathsf{AF}\neg \bigwedge$" in the proof. $\square$

**Lemma 5.18.** CTL-SAT(AG) *is* $\mathbf{W}[1]$*-hard when parameterized by structural pathwidth (resp. treewidth) and temporal depth.*

*Proof.* To prove the theorem some changes in the reduction given in Lemma 5.14 are required. The formulas which contain AX have to be changed for two reasons.

**Stutter-invariance.** One problem is a property that is known as the *stutter-invariance* of the operators F, G and U. Basically it says that they cannot distinguish between a path $\pi$ and a path $\pi'$ where arbitrary worlds of $\pi$ have been duplicated. In this sense it leads to an incorrect reduction when depth proposition $d_i$ are enforced with F operators only, since then they can occur twice, which would result in counting a variable $q_i$ for its partition twice.

**No irreflexive operators.** All operators but X are *reflexive* in the sense that the present is part of the future. However, the incrementation of the $\mathsf{T}_p^j$ counters must be implemented irreflexive, otherwise the counter would immediately jump to its maximal value at the first labeled $\mathsf{T}_p^{\uparrow}$.

Replace the formulas in the following way.

In $\psi[\text{depth}]$ we replace $\mathsf{AX}$ with the branching operator $\neg\mathsf{AG}\neg \equiv \mathsf{EF}$. Note that it is then not longer the case that all paths reachable from $w_0$ form the desired chain of worlds. However the branch of the model that satisfies one of the $\mathsf{EF}$-formulas has again to branch correctly at least once for the next depth level because of the nesting inside an $\mathsf{AG}$ operator, therefore at least one path starting in $w_0$ contains the correct labels. The next difference is that an arbitrary number of states now can share the same depth. But eventually the depth indicator has to increase in a satisfying model due to the semantics of $\mathsf{EF}$. Hence at least one path reachable from $w_0$ has the desired form. To deal with the problem of irreflexivity, we enforce some kind of alternation in terms of variables. Label a new variable $m_0$ resp. $m_1$ in worlds of parity 0 resp. 1.

$$\psi[\text{depth}]' := \mathsf{AG} \bigwedge_{i=0}^{n} [(d_i \wedge \neg d_{i+1}) \rightarrow \mathsf{EF}(d_{i+1} \wedge \neg d_{i+2})] \tag{5.9}$$

$$\psi[\text{alternation}] := \mathsf{AG} \bigwedge_{i=0}^{n-1} \left[(d_i \wedge \neg d_{i+1}) \rightarrow (m_{i \bmod 2} \wedge \neg m_{1-(i \bmod 2)})\right] \tag{5.10}$$

Fixing the chosen subset of variables $q_i$ is easily done using only $\mathsf{AG}$.

$$\psi[\text{fixed-}Q]' := \bigwedge_{i=1}^{n} [(q_i \rightarrow \mathsf{AG} q_i) \wedge (\neg q_i \rightarrow \mathsf{AG}\neg q_i)] \tag{5.11}$$

The signal counting formula has to be adapted to the fact that there can now exist multiple consecutive worlds having the same depth proposition labeled. Also, the counting procedure has to be implemented differently for the case that no increment signal is set. Without $\mathsf{AX}$ we cannot express that the labeled counter propositions may *not* change in the next world. To maintain correctness of the reduction, we have to introduce a second type of counters for variables set to zero, $\bot_p^{\uparrow}$. The following formulas are inserted to ensure the correctness of the new counter.

$$\psi[\text{signal}]_2 := \mathsf{AG} \bigwedge_{i=1}^{n} \left[(d_i \wedge \neg d_{i+1}) \rightarrow \left(\neg q_i \leftrightarrow \bot_{p(i)}^{\uparrow}\right)\right] \tag{5.12}$$

$$\psi[\text{init}]_2 := \bigwedge_{p=1}^{k} \left[\neg\bot_p^{\uparrow} \wedge \neg\bot_p^{1}\right] \wedge \mathsf{AG} \bigwedge_{p=1}^{k} \bot_p^{0} \tag{5.13}$$

$$\psi[\text{monotone}]_2 := \mathsf{AG} \bigwedge_{p=1}^{k} \bigwedge_{j=1}^{N(p)+1} \left(\bot_p^{j} \rightarrow \bot_p^{j-1}\right) \tag{5.14}$$

Check if for partition $p$ at most $C_p$ variables have been set to one and at most $N(p)-C_p$ variables have been set to zero.

$$\psi[\text{target}] \quad := \mathsf{AG} \bigwedge_{p=1}^{k} \left[ \neg \top_p^{C_p+1} \wedge \neg \bot_p^{N(p)-C_p+1} \right] \tag{5.15}$$

At last, the existing counting procedure has to be replaced and split up into counting of ones and zeros.

$$\begin{aligned} \psi[\text{count}]_1 \quad &:= \mathsf{AG} \bigwedge_{p=1}^{k} \bigwedge_{j=0}^{N(p)} \bigwedge_{i=0}^{1} \Big[ \left( \top_p^{\uparrow} \wedge \top_p^{j} \wedge m_i \right) \\ &\qquad \rightarrow \mathsf{AG} \left( m_{1-i} \rightarrow \mathsf{AG}\, \top_p^{j+1} \right) \Big] \end{aligned} \tag{5.16}$$

$$\begin{aligned} \psi[\text{count}]_2 \quad &:= \mathsf{AG} \bigwedge_{p=1}^{k} \bigwedge_{j=0}^{N(p)} \bigwedge_{i=0}^{1} \Big[ \left( \bot_p^{\uparrow} \wedge \bot_p^{j} \wedge m_i \right) \\ &\qquad \rightarrow \mathsf{AG} \left( m_{1-i} \rightarrow \mathsf{AG}\, \bot_p^{j+1} \right) \Big] \end{aligned} \tag{5.17}$$

As explained above there is at least one path of worlds where every depth proposition is reached at least once. If a depth proposition $d_i$ is reached with a signal variable $\top_p^{\uparrow}$ or $\bot_p^{\uparrow}$ labeled, then the corresponding counter value increases during the next parity change of $i$. That means that if a partition $p$ has weight $k$, then on this path there are at least $k$ parity changes with the proposition $\top_p^{\uparrow}$ labeled, and at least $N(p)-k$ parity changes with the proposition $\bot_p^{\uparrow}$ labeled. This leads to the counter $\top_p^{j}$ having a value $j \geq k$ and the counter $\bot_p^{j}$ having a value $j \geq N(p)-k$ in world $w_{n+1}$. This contradicts $\psi[\text{target}]$ unless $j$ is exactly $k$ resp. $N(p)-k$ and the partition is saturated.

The structural pathwidth increases only by a constant when considering the changes of the two formulas $\psi[\text{depth}]'$ and $\psi[\text{fixed-}Q]'$. The formula $\psi[\text{alternation}]$ can be handled by augmenting the bags which introduce $d_i$. To add the new counting and target formulas the same procedure as in Lemma 5.15 can be used: Treat every variable of the type $\bot_p^{\uparrow}$, $\bot_p^{j}$ like its $\top_p^{\uparrow}$ or $\top_p^{j}$ counterpart to preserve the $\kappa$-boundedness. □

**Lemma 5.19.** CTL-SAT(AU) *is* **W**[1]*-hard when parameterized by structural pathwidth (resp. treewidth) and temporal depth.*

*Proof.* Only minor changes from Lemma 5.18 are required to show the **W**[1]-hardness of the fragment {AU}. Change the formulas as follows: Introduce an additional depth proposition $d_{n+2}$ that has to hold after $d_{n+1}$. Then every subformula $\mathsf{AG}\gamma$ can be replaced by $\mathsf{A}[\gamma \mathsf{U} d_{n+2}]$. This is possible since AG never occurs negated except in $\psi[\text{depth}]$, also the depth formula has to be modified to make sure that no "rogue" occurence of $d_{n+2}$ appears before the propositions $d_1, \ldots, d_{n+1}$ occured in that order.

$$\psi[\text{depth}]' := \bigwedge_{i=0}^{n} \mathsf{A}\Big[(d_i \wedge \neg d_{i+1}) \rightarrow \big(m_{i \bmod 2} \wedge \neg m_{1-(i \bmod 2)} \\ \wedge \mathsf{A}[\neg d_{n+2} \mathsf{U} (d_{i+1} \wedge \neg d_{i+2})]\big) \mathsf{U} d_{n+2}\Big]$$

□

### 5.3.3 The open fragment AF

The tractability result for the AX fragment cannot be simply transferred to AF. There are two reasons.

First, AF together with negation obviously has not the finite tree property, it is not the case that quasi-labels of a certain depth can be assumed as empty in an infinite tree model. A single formula with AF's dual EG is enough to have infinite paths with non-empty label. The question for such paths is: Is the path "good" in the sense that it fulfills all eventualities imposed by labeled AF formulas, or is it "bad" in the sense that the EG-prefixed formula is labeled forever but prevents the AF-prefixed formulas from being labeled? In this context the terms "good" path and "bad" paths are coined by Mark Reynolds for a tableau algorithm that decides CTL$^\star$-SAT [Rey11]. Tableau algorithms in general try to traverse a formula top-down or bottom-up while creating a canonical model, or finding out that none exists. The MSO formula that defines CTL-SAT(AX) can be interpreted as such a tableau algorithm. But a procedure deciding satisfiability for **CTL**(AF) additionally has to deal with the question of good and bad paths, and then has to terminate after only verifying a finite number of states on a path. Reynolds's algorithm terminates, but he states that bad paths are followed for a prefix of length that is at least doubly exponential in $\varphi$. For an fpt-algorithm, a similar bound has to be found that depends only on the parameter, i.e. on the temporal depth and structural treewidth of $\varphi$.

This leads to the second reason: AF and EG operators can enforce long models (i.e. not bounded by the parameter) s. t. the used approach is not possible with a MSO formula that quantifies each world of the model.

**Theorem 5.20.** *The minimal depth of a model of a formula* $\varphi \in$ **CTL**(AF) *cannot be bounded by* $td(\varphi) + pw(\mathcal{S}_\varphi)$.

*Proof.* The proof is given using a formula family $(\varphi_n)_{n\in\mathbb{N}} \in$ **CTL**(AF) with constant structural pathwidth and temporal depth s. t. $\varphi_n$ has no model with depth less than $n$.

Define $\beta_i := q_i \wedge \mathsf{EG}(\neg q_{i-1})$.

Define $\varphi_n := q_0 \wedge \mathsf{AF}\beta_1 \wedge \mathsf{AF}\beta_2 \wedge .... \wedge \mathsf{AF}\beta_n$.

$\varphi_n$ has constant temporal depth of two. $\varphi_n$ has constant pathwidth:

Obtain a path-decomposition $\mathcal{P}$ by introducing for $i \in [n]$ a bag $B_i$ and inserting all subformulas of $\mathsf{AF}\beta_i$ into $B_i$, which is a constant number of nodes. Insert $q_0$ in another bag $B_0$. For $i = 0, \dots, n-1$ then connect bag $B_i$ and $B_{i+1}$. To cover the $\wedge$-nodes, insert the $i$-th $\wedge$ into the bags $B_i$ and $B_{i+1}$. This increases the width of $\mathcal{P}$ by at most two.

Now it holds: $\varphi_n$ has a model of size $n+1$ and depth $n$, but no model with size $< n+1$ or depth $< n$. As a model of $\varphi_n$ we can choose a chain of $n+1$ worlds s. t. that the variables $q_0, \dots, q_n$ are consecutively labeled in ascending order. After the last world a loop is added. This chain model has size $n+1$ and depth $n$.

Now arbitrarily choose another model $(\mathcal{M}, r)$ of $\varphi_n$. We show that $(\mathcal{M}, r)$ has depth at least $n$ and therefore size at least $n+1$. First extend $\mathcal{M}$ to a quasi-model. Next we use some notions introduced by Emerson [Eme90]: For a formula $\mathsf{AF}\beta$ labeled in a world $w$, write $\mathrm{DAG}[w, \beta]$ for the finite dag that starts at $w$ and contains all worlds reachable from $w$ up to the first occurence of the formula $\beta$ in a quasi-label. Such a finite dag always has to exist. Furthermore the dag is not only contained in $M$, but *embedded* in $M$, which means that every path through $M$ that leads out of the dag has to go through its leaves. The leaves of $\mathrm{DAG}[w, \beta]$ are also called *frontier worlds* and the non-leaves are called *interior worlds*.

In this proof shortly write $\mathrm{DAG}[i]$ for $\mathrm{DAG}[r, \beta_i]$.

**Claim.** *DAG*$[i]$ *is contained in the interior worlds of DAG*$[i+1]$.

*Proof of claim.* It suffices to show that $\mathrm{DAG}[i]$ is contained in $\mathrm{DAG}[i+1]$. The two dags cannot have common frontier worlds as those worlds would have both $q_i$ and $\neg q_i$ labeled.

Let $w$ be a frontier world of $\mathrm{DAG}[i+1]$. Then $\beta_{i+1} \in L(w)$. But then $\mathsf{AF}\beta_i \notin L(w)$ for the following reason: As $\beta_{i+1} \in L(w)$ implies $\mathsf{EG}\neg q_i \in L(w)$ and $\mathsf{AF}\beta_i$ implies $\mathsf{AF}q_i$, the formula $\mathsf{AF}\beta_i$ is false in $w$. Let $\pi \in \Pi(w)$ be the path that satisfies $\mathsf{G}\neg q_i$.

Every path $\pi' \in \Pi(r)$ which runs through $w$ has to visit a "shallower" world $w'$ with $\beta_i \in L(w')$ before $w$: Otherwise the path

$$(r = \pi'[1], \pi'[2], \dots, w = \pi[1], \pi[2], \dots)$$

would be a path starting in $r$ but not fulfilling $\mathsf{F}\beta_i$. Then $(\mathcal{M}, r)$ would not be a model.

This implies that on every path to a frontier node of $\mathrm{DAG}[i+1]$ there occurs a frontier node of $\mathrm{DAG}[i]$.

From the above claim it follows that no frontier node of $\mathrm{DAG}[n]$ is reachable from $r$ with less than $n$ steps. In fact, every model $(\mathcal{M}, r)$ of $\varphi_n$ contains the chain of length $n$. □

If we consider $\mathsf{EG}$-free formulas only, then long models can only be enforced by $\mathsf{AF}$-formulas with many common variables and therefore presumably a high structural treewidth: The required number of worlds is at most the chromatic number of the primal graph of the formula. Graph theoretic approaches were made to find a correlation between the chromatic number of the primal graph and the treewidth of the syntax graph, but so far without success.

**Example 5.21:**

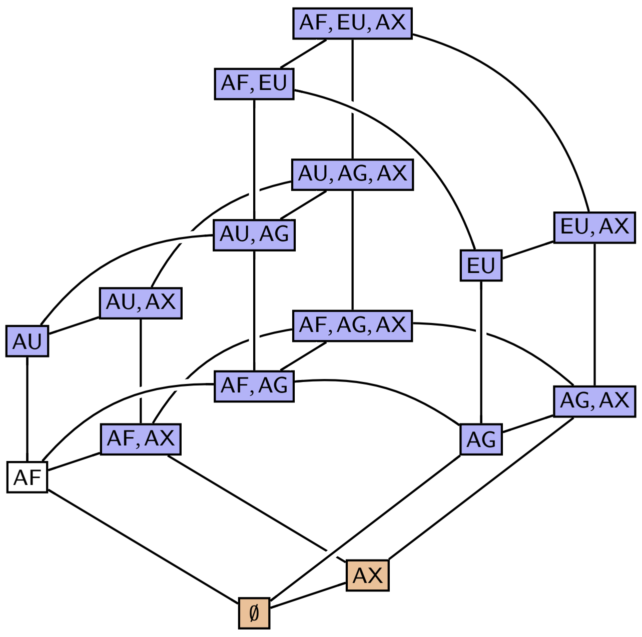
$$\varphi_n := \bigwedge_{i=1}^{n} \mathsf{AF}\left(\bigwedge_{j=1}^{i-1} \neg q_j \wedge q_i\right)$$

The smallest model of the formula $\varphi_n$ is a chain $w_1 \mapsto \ldots \mapsto w_n$ s. t. in the world $w_i$ exactly the variable $q_i$ is labeled but $q_j$ for $j < i$ is not. It is not possible to satisfy two of the $n$ $\mathsf{AF}$-eventualities in the same world, and $\varphi_n$ has always temporal depth $\mathrm{td}(\varphi_n) = 1$, but to enforce $n$ distinct worlds to exist seems not possible with a structural treewidth less than $\log n$.

Unfortunately the reduction from p-PW-SAT to CTL-SAT also seems to be impossible when using only $\mathsf{AF}$ and $\mathsf{EG}$ operators with nesting depth bounded by $k$. For the remaining fragments however the results are summed up in the following theorem.

**Theorem 5.22** (Parameterized complexity of CTL-SAT)**.** *Consider* CTL-SAT$(T)$ *parameterized by structural pathwidth (resp. treewidth) and temporal depth. Then the problem is* **FPT** *if* $T \sqsubseteq \{\mathsf{AX}\}$ *and* $\mathbf{W}[1]$*-hard if* $\{\mathsf{AU}\} \sqsubseteq T, \{\mathsf{AG}\} \sqsubseteq T$ *or* $\{\mathsf{AX}, \mathsf{AF}\} \sqsubseteq T$.

*Proof.* The tractability result follows from Theorem 5.12. The case $T = \emptyset$ can trivially be reduced to CTL-SAT$(\mathsf{AX})$. The hardness results follow from Lemma 5.18, Lemma 5.17 and Lemma 5.19. Note that the length of a formula can increase exponentially when simulating $\mathsf{AU}$ operators, but since the reductions have to keep the temporal depth bounded by $k$ and the blow-up is at most $2^{\mathrm{td}(\varphi)}$, the reduction is still possible in fpt-time. $\square$



🟧 in **FPT**
🟦 **W**[1]-hard
☐ Open

Figure 5.5: Complexity of CTL-SAT parameterized by $\kappa = (\mathrm{td} + \mathrm{tw})$

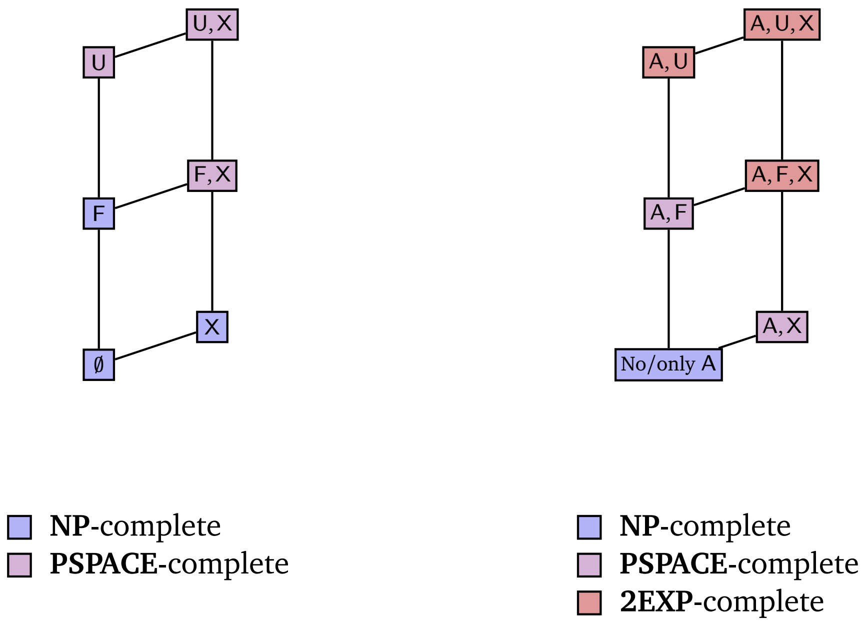
## 5.4 Linear Temporal Logic (LTL) and Full Branching Time Logic (CTL$^*$)

For $\mathcal{L} \in \{\mathbf{LTL}, \mathbf{CTL}^\star\}$, define the sets $\mathcal{L}(T), \mathcal{L}\text{-SAT}(T), \mathcal{L}_{\mathrm{NNF}}(T)$ as for CTL. The following classical results are known for LTL and CTL$^*$:

**Theorem 5.23** (Bauland, Schneider, Schnoor, Schnoor and Vollmer, 2007 [BSS+07])**.** *The problem* LTL-SAT$(T)$ *is* **NP***-complete for* $T \sqsubseteq \{\mathsf{X}\}$ *and* $T \sqsubseteq \{\mathsf{F}\}$ *and* **PSPACE***-complete for other fragments.*

**Theorem 5.24** (Meier, Thomas, Vollmer and Mundhenk, 2009 [MTVM09])**.** *The problem* CTL$^\star$-SAT$(T)$ *is* **NP***-complete when* $T \sqsubseteq \{\mathsf{A}\}$ *or* $\{\mathsf{A}\} \not\sqsubseteq T$*, it is* **PSPACE***-complete for* $T \equiv \{\mathsf{A}, \mathsf{F}\}$ *or* $T \equiv \{\mathsf{A}, \mathsf{X}\}$*, and it is* **2EXP***-complete for other fragments.*

In the world of parameterized complexity we choose the similar definitions of structural treewidth resp. pathwidth as before. Also define the temporal depth like for



(a) Complexity of LTL satisfiability (b) Complexity of CTL* satisfiability

Figure 5.6: Classical complexity of temporal logics

CTL. Observe that the path quantifiers A and E do not count as temporal operators for themselves.

$$\begin{array}{llll}
\mathrm{td}(x) & := 0 \text{ if } x \in \mathbf{PS} & \mathrm{td}(\top) & := 0 \\
\mathrm{td}(\varphi \wedge \psi) & := \max\{\mathrm{td}(\varphi), \mathrm{td}(\psi)\} & \mathrm{td}(\neg\varphi) & := \mathrm{td}(\varphi) \\
\mathrm{td}(\mathsf{E}\varphi) & := \mathrm{td}(\varphi) & \mathrm{td}(\mathsf{A}\varphi) & := \mathrm{td}(\varphi) \\
\mathrm{td}(T\varphi) & := \mathrm{td}(\varphi) + 1 \quad \text{if } T \in \{\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}\} & &
\end{array}$$

For the next theorems we use the notion of quasi-models as we did in Definition 5.8 for CTL. The only important difference is that a quasi-label of a state can only contain state formulas to make sense. We therefore introduce *path labels* to deal with CTL* formulas.

**Definition 5.25** (CTL* quasi-models)**.** A CTL* *quasi-model* of a formula $\varphi \in \mathbf{CTL}^{\star}_{\mathrm{NNF}}$ is a tuple $(\mathcal{M}, w_0)$ with $\mathcal{M} = (W, R, L_s, L_p)$ and $(W, R, L_s)$ being a quasi-structure. $L_p$ maps infinite paths starting at states in $W$ to a quasi-label. The following quasi-model conditions have to be satisfied:

- $\varphi \in L_s(w_0)$,
- $L_s(w)$ is a consistent quasi-label over $\mathrm{SF}(\varphi)$ for every $w \in W$.

- $L_s(w)$ contains only state formulas,
- $L_p(\pi)$ is a consistent quasi-label over $\mathrm{SF}(\varphi)$ for every infinite path $\pi$ through $M$.

Note that the function $L_\pi$ is in general not finite since most Kripke structures contain infinitely many paths.

For state formulas there are two new consistency rules:

- if $\mathsf{A}\psi \in L_s(w)$ then $\forall \pi \in \Pi(w) : \ \psi \in L_p(\pi)$
- if $\mathsf{E}\psi \in L_s(w)$ then $\exists \pi \in \Pi(w) : \ \psi \in L_p(\pi)$

The remaining consistency rules are to path labels:

- if $\psi \in L_p(\pi)$ and $\psi$ is also a state formula then $\psi \in L_s(\pi[1])$
- if $\mathsf{X}\psi \in L_p(\pi)$ then $\psi \in L_p(\pi_{\geq 2})$
- if $\mathsf{F}\psi \in L_p(\pi)$ then $\exists i \in \mathbb{N} : \ \psi \in L_p(\pi_{\geq i})$
- if $\mathsf{G}\psi \in L_p(\pi)$ then $\forall i \in \mathbb{N} : \ \psi \in L_p(\pi_{\geq i})$
- if $\psi \mathsf{U} \chi \in L_p(\pi)$ then $\exists i \in \mathbb{N} : \ \chi \in L_p(\pi_{\geq i})$ and $\forall j, 1 \leq j < i : \ \psi \in L_p(\pi_{\geq j})$

**Lemma 5.26.** *Every satisfiable formula $\varphi \in \mathbf{CTL}^\star_{\mathrm{NNF}}$ has a minimal tree quasi-model.*

*Proof.* Constructed by "unrolling" as for CTL (cf. Theorem 5.11). □

**Lemma 5.27.** *Let $\varphi \in \mathbf{CTL}^\star_{\mathrm{NNF}}(\mathsf{A}, \mathsf{E}, \mathsf{X})$. Every minimal tree quasi-model $\mathcal{T}$ of $\varphi$ has the property that every world at depth $\geq td(\varphi)$ and every path starting at such a world can only have propositional formulas labeled.*

*Proof.* Like for CTL in Theorem 5.11 write $\mathrm{td}(w) := \max\{\, \mathrm{td}(\psi) \mid \psi \in L_s(w) \,\}$ as the maximal temporal depth of a formula labeled in $w$.

Similar write $\mathrm{td}(\pi) := \max\left\{ \mathrm{td}(\psi) \;\middle|\; \psi \in L_p(\pi) \right\}$ for quasi-labels of paths.

First we show that $\forall w \forall \pi \in \Pi(w) : \mathrm{td}(\pi) \leq \mathrm{td}(w)$. Consider a world $w$ and a path $\pi$ starting in $w$ with $\mathrm{td}(\pi) > \mathrm{td}(w)$. Let $\psi$ be a formula of maximal temporal depth in $L_p(\pi)$ such that $\psi$ is not implied by a local quasi-label condition. Then $\psi$ can only be labeled in $L_p(\pi)$ because $\mathsf{A}\psi$ or $\mathsf{E}\psi$ is labeled in $\pi[1] = w$. But this contradicts $\mathrm{td}(\pi) > \mathrm{td}(w)$. Therefore $\mathrm{td}(\pi) \leq \mathrm{td}(w)$.

Second we show that $\forall w \forall \pi \in \Pi(w) : \mathrm{td}(\pi) > \mathrm{td}(\pi_{\geq 2})$. Assume there is a path $\pi$ with $\mathrm{td}(\pi) \leq \mathrm{td}(\pi_{\geq 2})$. Similar as above let $\psi$ be a subformula labeled in $L_p(\pi_{\geq 2})$ s. t. $\psi$ has maximal temporal depth and is not locally required. But this implies that $\mathsf{X}\psi \in L_p(\pi)$ since otherwise $\psi$ could be deleted from $L_p(\pi_{\geq 2})$. Again we get a contradiction to $\mathrm{td}(\pi) \leq \mathrm{td}(\pi_{\geq 2})$.

The last statement can be generalized to $\forall w \, \forall \pi \in \Pi(w) : \mathrm{td}(\pi_{\geq i}) - \mathrm{td}(\pi_{\geq i+j}) \geq j$. We prove this by induction on $j$. For $j = 0$ we have $\mathrm{td}(\pi_{\geq i}) - \mathrm{td}(\pi_{\geq i}) = 0$. For $j > 0$:

$$\begin{array}{rrcl} & \pi_{\geq i+(j+1)} & = & \left(\pi_{\geq i+j}\right)_{\geq 2} \\ \Rightarrow & \mathrm{td}\left(\pi_{\geq i+(j+1)}\right) & < & \mathrm{td}\left(\pi_{\geq i+j}\right) \\ \Rightarrow & \mathrm{td}\left(\pi_{\geq i}\right) - \mathrm{td}\left(\pi_{\geq i+(j+1)}\right) & > & \mathrm{td}\left(\pi_{\geq i}\right) - \mathrm{td}\left(\pi_{\geq i+j}\right) \\ \Rightarrow & \mathrm{td}\left(\pi_{\geq i}\right) - \mathrm{td}\left(\pi_{\geq i+(j+1)}\right) & \geq & \mathrm{td}\left(\pi_{\geq i}\right) - \mathrm{td}\left(\pi_{\geq i+j}\right) + 1 \\ \Rightarrow & \mathrm{td}\left(\pi_{\geq i}\right) - \mathrm{td}\left(\pi_{\geq i+(j+1)}\right) & \geq & j + 1 \end{array}$$

For the proof we do the same chopping of states as in CTL. To prove that no quasi-label is violated we show that in the minimal model for every world $w$ in depth $\mathrm{td}(\varphi)$ it holds $\mathrm{td}(w) = 0$ and $\mathrm{td}(\pi) = 0 \, \forall \pi \in \Pi(w)$. This is obvious for $\mathrm{td}(\varphi) = 0$.

Assume now that $\mathrm{td}(w) > 0$ for a world $w$ in depth $\mathrm{td}(\varphi)$. Then let $\psi$ be a subformula labeled in $L_s(w)$ s. t. $\mathrm{td}(\psi) > 0$ and $\psi$ is not locally required. That means that there is a path $\pi$ and $i > 1$ s. t. $w = \pi[i]$ and $\psi \in L_p\left(\pi_{\geq i}\right)$, since $w$ is not the root of the model, and $\mathrm{td}\left(\pi_{\geq i}\right) > 0$. But $\left(\pi_{\geq i}\right)$ has to be of the form $\left(\pi'_{\geq \mathrm{td}(\varphi)+1}\right)$ for some $\pi' \in \Pi(r)$, i.e. a path that starts at the root of the model, so

$$\mathrm{td}\left(\pi'\right) - \mathrm{td}\left(\pi_{\geq i}\right) = \mathrm{td}\left(\pi'_{\geq 1}\right) - \mathrm{td}\left(\pi'_{\geq \mathrm{td}(\varphi)+1}\right) \geq \mathrm{td}(\varphi)$$

and $\mathrm{td}\left(\pi'\right) \geq \mathrm{td}\left(\pi_{\geq i}\right) + \mathrm{td}(\varphi) > \mathrm{td}(\varphi)$. But a formula $\psi$ with $\mathrm{td}(\psi) > \mathrm{td}(\varphi)$ cannot be labeled in a minimal model, hence the assumption that $\mathrm{td}(w) > 0$ for a world $w$ in depth $\mathrm{td}(\varphi)$ is false. For such a world it automatically also holds $\forall \pi \in \Pi(w) : \mathrm{td}(\pi) \leq \mathrm{td}(w)$ which we showed earlier. $\square$

*Remark:* Note that unlike for CTL the filtration technique alone (i.e. construction of equivalence classes of states) does not work for CTL* for the direction from right to left. One cannot simply identify states that have the same set of labeled formulas in the quasi-model as adding this type of "back edges" would lead to unfulfilled path formulas.

Considerations like this also take place in certain tableau construction methods for satisfiability. Such methods were well-known for modal logic or CTL whereas there was no such algorithm known for CTL* due to the difficulties imposed by nested path operators, until in 2011 Reynolds presented his sophisticated tableau algorithm that correctly handles the "back edge" construction in the tree [Rey11].

The direction from right to left has been left out in Lemma 5.26 and it is unknown to the author if a similar "infinite tree compaction" as for CTL in [Eme90] has been done. But for CTL* restricted to X operators we again (almost) fall down to plain modal logic and immediately get back the finite tree model property:

**Theorem 5.28.** *Let $\varphi \in \mathbf{CTL}^{\star}_{\mathrm{NNF}}(\mathsf{A}, \mathsf{E}, \mathsf{X})$. Then the following statements are equivalent:*

*(1) $\varphi$ is satisfiable.*

*(2) $\varphi$ has a tree quasi-model.*

*(3) $\varphi$ has a minimal tree quasi-model.*

*(4) $\varphi$ has a tree-like Kripke model of depth $td(\varphi)$.*

*Proof.* (4) $\Rightarrow$ (1) is clear. (1) $\Rightarrow$ (2) and (2) $\Rightarrow$ (3) are due to Lemma 5.26.
(3) $\Rightarrow$ (4) follows from Lemma 5.27. □

**Corollary 5.29.** *A formula $\varphi \in \mathbf{CTL}^{\star}_{\mathrm{NNF}}(\mathsf{A}, \mathsf{E}, \mathsf{X})$ is satisfiable iff there is a tree quasi-structure $\mathcal{T}$ such that the following holds:*

- *$\varphi$ is labeled at the root state of $\mathcal{T}$,*
- *$\mathcal{T}$ is consistent up to depth $td(\varphi)$,*
- *quasi-label conditions are satisfied in $\mathcal{T}$ for states and paths up to depth $td(\varphi)$.*

The separation of state formulas and path formulas that happens in CTL$^*$ bears its own subtleties. Even when $\mathsf{X}$ is the only present temporal operator, the placement of path quantifiers $\mathsf{A}, \mathsf{E}$ still influences the semantics.

**Example 5.30:**
Compare the following two formulas from which one is a valid CTL formula and the other one is only a CTL$^*$ formula.

$$\varphi_1 := \mathsf{EX}p \wedge \mathsf{EX}\neg p \wedge (\mathsf{AX}p \vee \mathsf{AX}\neg p)$$
$$\varphi_2 := \mathsf{EX}p \wedge \mathsf{EX}\neg p \wedge \mathsf{A}(\mathsf{X}p \vee \mathsf{X}\neg p)$$

$\varphi_1$ is unsatisfiable since it requires all successors to either satisfy $p$ or all successors to satisfy $\neg p$, but there also must be successors which fulfill $p$ resp. $\neg p$. In $\varphi_2$ every path outgoing from the initial world has to fulfill the path formula $(\mathsf{X}p \vee \mathsf{X}\neg p)$. This does not require that all paths consequently decide for $\mathsf{X}p$ or $\mathsf{X}\neg p$.
In general it holds $\mathsf{A}(\phi \wedge \psi) \equiv \mathsf{A}\phi \wedge \mathsf{A}\psi$, but $\mathsf{A}(\phi \vee \psi) \not\equiv \mathsf{A}\phi \vee \mathsf{A}\psi$. In the converse we have $\mathsf{E}(\phi \vee \psi) \equiv \mathsf{E}\phi \vee \mathsf{E}\psi$ but $\mathsf{E}(\phi \wedge \psi) \not\equiv \mathsf{E}\phi \wedge \mathsf{E}\psi$.

Note that the sheer number of paths that exists in a model already prohibits to existentially quantify all related quasi-labels of paths. Therefore Theorem 5.12 cannot be directly transferred to work with path formulas. The approach for the next theorem relies on a property of "path equivalence" that holds in formulas containing only $\mathsf{A}, \mathsf{E}, \mathsf{X}$ as temporal operators. Call a formula $\psi$ for which $\mathsf{A}\psi$ is labeled on a path *A-scoped* and for which $\mathsf{E}\psi$ is labeled *E-scoped*. Call pure path subformulas of $\psi$ (i.e. no intermediate $\mathsf{A}, \mathsf{E}$-operator) also $\mathsf{A}$- resp. $\mathsf{E}$-scoped. The proof is based on a PSPACE algorithm for $\mathrm{CTL}^\star\text{-SAT}(\mathsf{A}, \mathsf{X})$ given by Meier, Thomas, Vollmer and Mundhenk [MTVM09]. This algorithm recursively evaluates a formula $\varphi$ in depth-first search and checks for a contradiction up to depth $\mathrm{td}(\varphi)$.

The main argument is that only two sets $P_A$ and $P_E$ have to be remembered by the algorithm which are exactly the sets of $\mathsf{A}$-scoped and $\mathsf{E}$-scoped path formulas. Consider for example a path formula $\psi$ that is labeled on a path $\pi$. If $\psi$ is $\mathsf{A}$-scoped, then $\psi$ has to be labeled in all paths starting at $\pi[1]$, and if $\psi$ is $\mathsf{E}$-scoped, then $\psi$ has to be labeled in at least one path starting at $\pi[1]$, independent of the total number of paths that start in $\pi[1]$ (which can be higher than the number of immediate successors of $\pi[1]$). Due to the path equivalence we then only need to remember these two sets $P_A$ and $P_E$ of $\mathsf{A}$- resp. $\mathsf{E}$-scoped formulas: If a path formula $\psi$ in $\pi$ has the form $\mathsf{X}\xi$ then we can just label $\xi$ at all successors resp. one successor of $\pi[1]$, and for nested path formulas the membership in $P_A$ resp. $P_E$ then recursively is inherited to subpaths starting at the successor.

Note that when other temporal operators besides $\mathsf{X}$ occur then the path equivalence property does not hold anymore: The point of fulfillment of imposed eventualities is not longer determined over all paths but instead can be different.

**Theorem 5.31.** $\mathrm{CTL}^\star\text{-SAT}(\mathsf{A}, \mathsf{X})$ *is in* **FPT** *when parameterized by structural treewidth and temporal depth.*

*Proof.* We encode the PSPACE algorithm as the following MSO formula which verifies a tree model for $\varphi$ up to depth $\mathrm{td}(\varphi)$.

$$
\begin{aligned}
\theta_{\text{Boolean}}(L) := \forall x \forall y \forall z\\
L(x) \to \Big( & \mathcal{R}_{\cdot\wedge}(x,y) \wedge \mathcal{R}_{\wedge\cdot}(x,z) \to (L(y) \wedge L(z)) \ \wedge \\
& \mathcal{R}_{\cdot\vee}(x,y) \wedge \mathcal{R}_{\vee\cdot}(x,z) \to (L(y) \vee L(z)) \ \wedge \\
& \mathcal{R}_{\neg}(x,y) \to \neg L(y) \Big)
\end{aligned}
$$

$$
\theta_{\text{inherit}\,R}(P, P') := \forall x \forall y \Big( \big(P(x) \wedge \mathcal{R}_R(x,y)\big) \to P'(y) \Big)
$$

$$
\begin{aligned}
\theta^{n}_{\text{path}}(P_A,P_E) :=\ & \theta_{\text{Boolean}}(P_A) \wedge \theta_{\text{Boolean}}(P_E) \wedge \\
& \exists H\ \Big(\theta_{\text{Boolean}}(H) \wedge \forall z\ \big((P_A(z) \vee P_E(z)) \rightarrow H(z)\big) \wedge \\
& \quad \theta_{\text{inherit}\ \mathsf{A}}(H,P_A)\ \wedge\ \theta_{\text{inherit}\ \mathsf{E}}(H,P_E)\Big) \wedge \\
& \exists P'_A\, \exists P'_E\ \Big(\theta^{n-1}_{\text{path}}(P'_A,P'_E) \wedge \theta_{\text{inherit}\ \mathsf{X}}(P_A,P'_A)\Big) \wedge \\
& \forall x\, \forall y\ \Big((P_E(x) \wedge \mathcal{R}_{\mathsf{X}}(x,y)) \rightarrow \\
& \quad \exists P'_A\, \exists P'_E \Big(\theta^{n-1}_{\text{path}}(P'_A,P'_E)\ \wedge\ P'_E(y)\ \wedge\ \theta_{\text{inherit}\ \mathsf{X}}(P_A,P'_A)\Big)\Big)
\end{aligned}
$$

$$
\begin{aligned}
\theta^{0}_{\text{path}}(P_A,P_E) :=\ & \theta_{\text{Boolean}}(P_A) \wedge \theta_{\text{Boolean}}(P_E) \wedge \\
& \exists H\ \Big(\theta_{\text{Boolean}}(H) \wedge \forall z\ \big((P_A(z) \vee P_E(z)) \rightarrow H(z)\big) \wedge \\
& \quad \theta_{\text{inherit}\ \mathsf{A}}(H,P_A)\ \wedge\ \theta_{\text{inherit}\ \mathsf{E}}(H,P_E)\Big)
\end{aligned}
$$

Set $m := \text{td}(\varphi)$. Define

$$\theta := \exists P_A\, \exists P_E\ \exists x\, \mathcal{R}_{\text{root}}(x)\ \wedge P_A(x)\ \wedge\ \theta^{m}_{\text{path}}(P_A,P_E).$$

Now it holds for a formula $\varphi \in \mathbf{CTL}^{\star}_{\text{NNF}}(\mathsf{A},\mathsf{X})$ that $\varphi \in \text{CTL}^{\star}\text{-SAT} \Longleftrightarrow \mathcal{S}_{\varphi} \models \theta$.

**Correctness**

"$\Leftarrow$":

We construct a tree quasi-model $\mathcal{T}$ for $\varphi$ from a valid interpretation of $\theta$.

In the formula $\theta^{m}_{\text{path}}(P_A,P_E)$ for the initial $P_A, P_E$ a path $\pi$ is represented which starts in a state with label $H$. Insert a root $r$ into $\mathcal{T}$ with empty quasi-label. Then add an infinite path starting at $r$ as well with empty quasi-label.

Repeat the following to ensure the quasi-label conditions up to depth $i \in \{0,\dots,m\}$:

Choose a world $w$ at depth $i$ which is not yet processed, starting with $r$.

There is exactly one path $\pi \in \Pi(w)$ as well as instances of $P_A, P_E$ which satisfy $\theta^{n}_{\text{path}}(P_A,P_E)$ and correspond to $\pi$. $P_A$ is the set of path formulas that must hold at all paths starting in $w$ and $P_E$ is the set of path formulas that must hold in at least one path starting in $w$. Set the quasi-label $L_S(w)$ to the state formulas in $H$. (If $r = w$

then $\varphi \in H$.) We know that for every formula $\mathsf{A}\psi \in H$ there is also $\psi \in P_A$ and for every formula $\mathsf{E}\psi \in H$ there is also $\psi \in P_E$. Therefore we "split up" $\pi$ as follows: For every formula $\psi \in P_E$ attach a new infinite path to $w$ which has $\psi$ labeled. Now for every path added to $w$ this way label every $\psi$ with $\psi \in P_A$ to it. This cannot create an inconsistent quasi-label of the common origin $w$ of these paths since the quasi-label $H$ is consistent. Also the new paths are not affected by residual path formulas labeled in $L_P(\pi)$. Now $L_S(w)$ fulfills all quasi-label conditions for state formulas.

Then for $\pi$ and every added path jump one step forward to the world $w'$ in depth $i+1$. This is done in the MSO formula by quantifying a unique successor path $\pi' \in \Pi(w')$ for every added path as well as for $\pi$. In $\pi'$ every $\psi$ can be labeled for which $\mathsf{X}\psi$ was labeled in the unique path $(w, w', \ldots)$. Hence for every path $\pi \in \Pi(w)$ the quasi-label conditions are also satisfied.

This step can be repeated up to depth $m$. $\varphi$ is in $\mathbf{CTL}^\star_{\text{NNF}}(\mathsf{A}, \mathsf{E}, \mathsf{X})$. Due to Corollary 5.29 we know that if a tree quasi-model $\mathcal{T}$ is consistent until depth $m$ and has $\varphi$ labeled at its root, then $\mathcal{T}$ can be truncated to a minimal model which has only propositional formulas in depth $m$, which are sufficiently checked by $\theta^0_{\text{path}}$.

"$\Rightarrow$":

A quasi-model $\mathcal{M}$ of $\varphi$ can be traversed to choose correct instances for the set variables occuring in $\theta$: Use quasi-labels of paths for sets $P_A, P_E, P'_A, P'_E$ and use quasi-labels of states for sets $H$.

Choose an arbitrary path starting at the root $r$ of $\mathcal{M}$ for the sets $P_A$ and $P_E$, choose a suitable superset of $L_S(r)$ for the first instance of $H$.

For the remaining instances of $\theta^n_{\text{path}}$ that correspond to a path $\pi$ in $\mathcal{M}$ use the quasi-label of $\pi_{\geq 2}$ to instanciate the first $P'_A$ and $P'_E$, and for formulas $\mathsf{E}\psi$ corresponding to elements of $P_E$ choose an arbitrary path $\pi'$ starting in a successor of $\pi[1]$ where $\psi$ is labeled. The quasi-label of $\pi'$ can again be used to instanciate the second $P'_A$ and $P'_E$ for every $\mathsf{E}$-scoped formula in $P_E$. Note that a path in $\mathcal{M}$ can be used to assign the same set to multiple occurences of set variables $P'_A, P'_E$.

Due to the local quasi-label conditions the formula $\theta_{\text{Boolean}}$ is always satisfied, and due to the temporal quasi-label conditions the formulas $\theta_{\text{inherit}\,R}$ are always satisfied in the above construction.

## FPT Runtime

Proven similar as Theorem 5.12 since the length of $\theta$ depends only on $\text{td}(\varphi)$. $\square$

**Lemma 5.32.** LTL-SAT($\mathsf{F}$) *is* $\mathbf{W}[1]$*-hard when parameterized by structural pathwidth and temporal depth.*

*Proof.* The result is proven by a parameterized reduction from the problem p-PW-SAT similar to the CTL-SAT(AG) case (cf. Lemma 5.18): Let $I = \big(\varphi,\, k,\, (X_i)_{i\in[k]},\, (C_i)_{i\in[k]}\big)$

be an instance of p-PW-SAT. We consider an equivalent LTL formula $\psi(I)$ that has a low structural pathwidth.

The formula $\psi(I) \in \mathbf{LTL}(\mathsf{F})$ is a conjunction of the following subformulas.

$$\psi[\text{formula}] := \varphi \tag{5.18}$$

$$\psi[\text{depth}] := \mathsf{G}\bigwedge_{i=0}^{n-1}\big[(d_i \wedge \neg d_{i+1}) \to (m_{i \bmod 2} \wedge \neg m_{1-(i \bmod 2)} \wedge \mathsf{F}(d_{i+1} \wedge \neg d_{i+2}))\big] \tag{5.19}$$

$$\psi[\text{fixed-}Q] := \bigwedge_{i=1}^{n}[(q_i \to \mathsf{G}q_i) \wedge (\neg q_i \to \mathsf{G}\neg q_i)] \tag{5.20}$$

$$\psi[\text{signal}] := \mathsf{G}\bigwedge_{i=1}^{n}\left[(d_i \wedge \neg d_{i+1}) \to \left(\left(q_i \leftrightarrow \top^{\uparrow}_{p(i)}\right) \wedge \left(\neg q_i \leftrightarrow \bot^{\uparrow}_{p(i)}\right)\right)\right] \tag{5.21}$$

$$\psi[\text{count}] := \mathsf{G}\bigwedge_{p=1}^{k}\bigwedge_{j=0}^{n(p)}\bigwedge_{i=0}^{1}\left[\left(\top^{\uparrow}_p \wedge \top^j_p \wedge m_i\right) \to \mathsf{G}\left(m_{1-i} \to \mathsf{G}\top^{j+1}_p\right)\right] \tag{5.22}$$

$$\wedge\left[\left(\bot^{\uparrow}_p \wedge \bot^j_p \wedge m_i\right) \to \mathsf{G}\left(m_{1-i} \to \mathsf{G}\bot^{j+1}_p\right)\right] \tag{5.23}$$

$$\psi[\text{monotone}] := \mathsf{G}\left[\bigwedge_{i=1}^{n}(d_i \to d_{i-1}) \wedge \bigwedge_{p=1}^{k}\bigwedge_{j=1}^{n(p)+1}\left[\left(\top^j_p \to \top^{j-1}_p\right) \wedge \left(\bot^j_p \to \bot^{j-1}_p\right)\right]\right] \tag{5.24}$$

$$\psi[\text{target}] := \mathsf{G}\bigwedge_{p=1}^{k}\left[d_{n+1} \to \left(\top^{C_p} \wedge \neg\top^{C_p+1} \wedge \bot^{n(p)-C_p}_p \wedge \neg\bot^{n(p)-C_p+1}_p\right)\right] \tag{5.25}$$

$$\psi[\text{init}] := d_0 \wedge \neg d_1 \wedge \mathsf{G}\bigwedge_{p=1}^{k}\left[\top^0_p \wedge \bot^0_p\right] \tag{5.26}$$

The $\kappa$-boundedness of the structural pathwidth of $\psi(I)$ is proven as in Lemma 5.15.

The pathwidth increases only marginally when replacing $\mathsf{G}$ by $\neg\mathsf{F}\neg$ in the given formulas. The correctness follows from the argumentation already done in Lemma 5.18: In the CTL case the formula enforces at least one path which does the correct counting of variables in their respective partitions. The given LTL formula, which is a path formula, ensures the same behaviour on a single path. $\square$

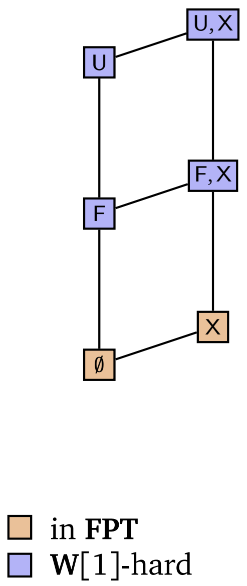
Note that the path semantics of LTL allows this reduction using the $\mathsf{F}$ operator only. For CTL on the other hand the different semantics of $\mathsf{AF}$ and $\mathsf{EF}$ resp. $\mathsf{AG}$ and $\mathsf{EF}$ are crucial for this method: Replacing $\mathsf{AG}$ by $\mathsf{EG}$ would result in an incorrect reduction. The problem already occurs in $\psi[\text{fixed-}Q]$. Using only $\mathsf{EG}$ it is not possible to force even two variables $q_1, q_2$ to maintain their value on a single path; they can immediately "branch off" onto different paths.

**Theorem 5.33.** LTL-SAT$(T)$ *parameterized by structural pathwidth (resp. treewidth) and temporal depth is* **FPT** *if* $T \subseteq \{\mathsf{X}\}$ *and* $\mathbf{W}[1]$*-hard if* $\mathsf{F} \in T$ *or* $\mathsf{U} \in T$.

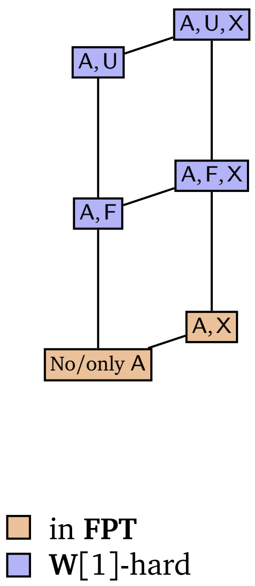
*Proof.* For $T \subseteq \{\mathsf{X}\}$ the problem is $\leq^{fpt}$-reducible to CTL$^\star$-SAT$(\mathsf{A},\mathsf{X})$ by the function $f(\varphi) = \mathsf{A}\varphi$. For $T = \{\mathsf{F}\}$ refer to Lemma 5.32. For $\{\mathsf{F}\} \subseteq T$ the hardness is shown by the identity reduction from LTL-SAT$(\mathsf{F})$ to LTL-SAT$(T)$ which is an $\leq^{fpt}$-reduction. For $\{\mathsf{U}\} \subseteq T$ we modify the reduction and replace every formula $\mathsf{F}\psi$ by $[\top\mathsf{U}\psi]$, increasing the pathwidth only marginally. $\square$

**Theorem 5.34.** CTL$^\star$-SAT$(T)$ *parameterized by structural pathwidth (resp. treewidth) and temporal depth is* **FPT** *if* $T \sqsubseteq \{\mathsf{A},\mathsf{X}\}$ *or* $\{\mathsf{A}\} \not\sqsubseteq T$*, and* $\mathbf{W}[1]$*-hard if* $\{\mathsf{A},\mathsf{F}\} \sqsubseteq T$ *or* $\{\mathsf{A},\mathsf{U}\} \sqsubseteq T$.

*Proof.* The case $T \sqsubseteq \{\mathsf{A},\mathsf{X}\}$ is shown by Theorem 5.31. If $\{\mathsf{A}\} \not\sqsubseteq T$, it is only possible to express propositional formulas which is equivalent to $T = \emptyset \sqsubseteq \{\mathsf{A},\mathsf{X}\}$. For $\{\mathsf{A},\mathsf{F}\} \sqsubseteq T$ and $\{\mathsf{A},\mathsf{U}\} \sqsubseteq T$ one can $\leq^{fpt}$-reduce LTL-SAT$(\mathsf{F})$ resp. LTL-SAT$(\mathsf{U})$ to CTL$^\star$-SAT$(T)$ via $f(\varphi) = \mathsf{A}\varphi$. $\square$



(a) Complexity of LTL satisfiability



(b) Complexity of CTL* satisfiability

Figure 5.7: Complexity of temporal logics parameterized by $\kappa = (\mathrm{td} + \mathrm{tw})$

## 5.5 Variations of the parameterization

The tractability or intractability of a temporal logic fragment can depend on the chosen parameterization. In this section the influence of different parameters is studied with respect to the temporal fragments.

**Lemma 5.35.** LTL-SAT(X) *is in* **FPT** *when parameterized by structural treewidth.*

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

$$\varphi \equiv \beta(\mathsf{X}^{n_1}q_1, \ldots, \mathsf{X}^{n_m}q_m), \quad \text{where } \mathsf{X}^i := \underbrace{\mathsf{X}\ldots\mathsf{X}}_{i \text{ times}} \text{ and } q_i \text{ is a variable.}$$

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 $\mathsf{X}^{n_i}q_i$ is interpreted as an atomic formula (i.e. a variable).

Formally we have

$$(\text{LTL-SAT}(\mathsf{X}), \mathrm{tw}_\varphi) \leq^{\mathit{fpt}} (\text{SAT}, \mathrm{tw}_\varphi) \leq^{\mathit{fpt}} (\text{CTL-SAT}(\emptyset), \mathrm{tw}_\varphi + \mathrm{td}) \in \textbf{FPT}. \qquad \square$$

**Lemma 5.36.** CTL-SAT(AF) *is* **W**[1]*-hard when parameterized by structural treewidth or pathwidth.*

*Proof.* Similar as in Lemma 5.14 we give a reduction from the problem p-PW-SAT. Due to the restricted power of the AF and EG operators we choose a different approach for expressing the existence of a saturated assignment in CTL.

Let $I = \big(\varphi, k, (X_i)_{i\in[k]}, (C_i)_{i\in[k]}\big)$ be a given instance of PARTWEIGHT-SAT. $\varphi$ is a propositional formula in CNF, $k$ is the number of partitions, $X_i$ is the subset of variables in partition $i$ and $C_i$ is the capacity of partition $i$, i.e. the desired weight inside the partition of a satisfying assignment of $\varphi$.

We construct a CTL formula $\psi(I)$ which is a conjunction of the formulas described next. Use the following notation: Let the propositional variables that occur in $\varphi$ be named $q_1, \ldots, q_n$. Then $p(i)$ is the number of the unique partition that contains $q_i$. For an arbitrary partition number $p$, write $C(p)$ for its capacity and $N(p)$ for its size as before.

Assuming some assignment to $q_1, \ldots, q_n$, the idea of the reduction is to measure the weight of a partition in terms of worlds which have labeled a special propositional variable $p^+$, where $p$ is a partition number. $p^+$ is used to bound the weight of $p$ from above. Similar use a propositional variable $p^-$ to bound the weight of $p$ from below.

First enforce the model of $\psi(I)$ to have a special "ordered" form like in the proof of Theorem 5.20:

$$\begin{aligned}\beta_i^d :=& \left[q_i \to \mathsf{AF}\left(p(i)^+ \land d_i \land \mathsf{EG}(\neg e_{i-1})\right)\right] \land \\ & \left[\neg q_i \to \left(\mathsf{AF}(p(i)^- \land d_i \land \mathsf{EG}(\neg e_{i-1})\right)\right] \\ \beta_i^e :=& \mathsf{AF}\left(e_i \land \mathsf{EG}(\neg d_i) \land \bigwedge_{p=1}^{k} \neg p^+ \land \neg p^-\right)\end{aligned}$$

Let $(\mathcal{M}, r)$ be a model which has $\beta_i^d$ and $\beta_i^e$ labeled in $r$ for $i \in [n]$. It holds $\mathrm{DAG}[\beta_1^d] \prec \mathrm{DAG}[\beta_1^e] \prec \mathrm{DAG}[\beta_2^d] \prec \mathrm{DAG}[\beta_2^e] \prec \ldots \prec \mathrm{DAG}[\beta_n^d] \prec \mathit{DAG}[\beta_n^e]$, where $\prec$ means "is contained in the interior worlds of". This can be proven similar to Theorem 5.20: Now, additionally to the $n$ "depth" propositions we also have $n$ "emptiness" frontiers in the sense that they have neither the $p^+$ and $p^-$ propositions labeled, while the frontier worlds of $\beta_i^d$ formulas have at least $p^+$ or $p^-$ or both labeled.

Call the set of frontier worlds of a dag simply its *frontier*.

**Claim.** *For any assignment of $q_1, \ldots, q_n$ in $r$ holds: If a partition $p$ has weight $h$, then every path $\pi \in \Pi(r)$ visits a set of frontiers $H = \{A_1, \ldots, A_h, B_1, \ldots, B_{N(p)-h}\}$ s. t. no two frontiers of $H$ have common worlds, all worlds of each $A_i$ have $p^+$ labeled and all worlds of each $B_i$ have $p^-$ labeled.*

*Proof of claim.* Consider the assignment of variables $q_1, \ldots, q_n$ in $r$. The frontier of

DAG[$\beta_i^d$], i.e. the worlds that are witness for the AF-subformula in $\beta_i^d$, have $p(i)^+$ labeled if $q_i$ holds in $r$, and they have $p(i)^-$ labeled if $\neg q_i$ holds in $r$. These frontiers are distinct since DAG[$\beta_i^d$] $\prec$ DAG[$\beta_{i+1}^d$] for each $i$ and $\prec$ is transitive.

So for every partition $p$ and every path $\pi$ from $r$ through $\mathcal{M}$, $\pi$ visits at least $h$ times a frontier with $p^+$ and $N(p)-h$ times a frontier with $p^-$ labeled if the partition $p$ was assigned weight $h$. We use the following recursively defined formulas to limit the weight to the capacity:

$$\gamma^+(p,0) := \mathsf{EG}\left(\neg p^+ \to \mathsf{EG}\neg p^+\right)$$
$$\gamma^+(p,j) := \mathsf{EG}\left(\neg p^+ \to \mathsf{EG}(p^+ \to \gamma^+(p,j-1))\right)$$

These formulas, when labeled in $r$, can be interpreted as follows: $\gamma^+(p,0)$ is true when there is no frontier with $p^+$ labeled, i.e. none of the variables $q_i$ with $p(i)=p$ was assigned one (and no rogue $p^+$ is labeled somewhere in the model). The formula $\gamma^+(p,j)$ means: As soon as on a path first $\neg p^+$ and then $p^+$ is encountered, at most $j-1$ more worlds with $p^+$ may be encountered.

Similar define formulas $\gamma^-(p,j)$ which limit the occurences of worlds with $p^-$. Also label $\neg p^+ \wedge \neg p^-$ in $r$ for each partition number $p$.

**Claim.** *Let $p$ be a partition and $0 \le h \le N(p)$. If the formulas $\gamma^+(p,h)$ and $\gamma^-(p,N(p)-h)$ are both true in $r$, then the variables $q_1,\dots,q_n$ are assigned in $r$ s. t. $p$ has weight $h$.*

*Proof of claim.* For a contradiction assume that $p$ has weight $h' > h$. Then there are at least $h'$ distinct frontiers which have $p^+$ labeled. Between two frontiers of DAG[$\beta_i^d$] and DAG[$\beta_{i+1}^d$] there is always another frontier of DAG[$\beta_i^e$] where neither $p^+$ nor $p^-$ is labeled.

Write $F_1,\dots,F_{h'}$ for the $h'$ distinct frontiers with $p^+$ labeled and assume they are ordered from smallest to largest distance relative to $r$. Now assume that $\gamma^+(p,h)$ is true in $r$. Since $p^+$ does not hold in $r$, a path $\pi$ starting from $r$ has to satisfy $\mathsf{G}(p^+ \to \gamma^+(p,h-1))$. As $h' > 0$, $\pi$ will eventually encounter a world $w$ with $p^+$ labeled (in $F_1$ or earlier). But due to a frontier with $\neg p^+$ the formula $\mathsf{G}(p^+ \to \gamma^+(p,h-2))$ has to hold on another path $\pi'$ that yet has to reach $F_2$.

When repeating this $h$ times we get that in a world $w^*$ the formula $\gamma^+(p,0)$ has to hold, but every path starting at $w^*$ has yet to reach the frontier $F_{h'}$, which contradicts the assumption.

The case $h' < h$ is proven similar: If $h' < h$, then there are more than $N(p)-h$ frontiers with $p^-$ labeled and $\gamma^-(p,N(p)-h)$ cannot be true in $r$. Note that both cases also work when there are rogue propositions labeled somewhere in the model; they can only make a valid formula $\gamma^+$ or $\gamma^-$ false, but not vice versa.

Figure 5.8 illustrates this argumentation.

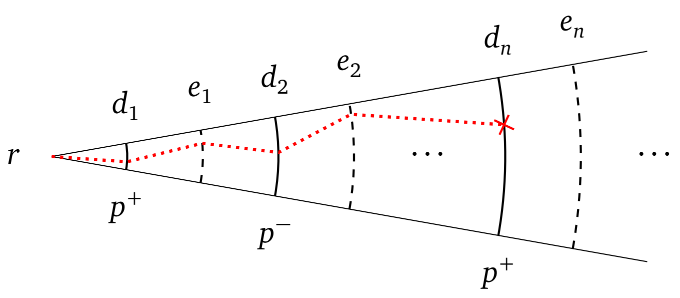


Figure 5.8: $p$ has weight at least 2: All paths $\pi \in \Pi(r)$ fail to fulfill $\gamma^+(p,1)$

From the above claims it follows that in every model $(\mathcal{M}, r)$ of the formula

$$\psi(I) := \varphi \wedge \bigwedge_{p=1}^{k} \left[\neg p^+ \wedge \neg p^- \wedge \gamma^+(p, C(p)) \wedge \gamma^-(p, N(p) - C(p))\right] \wedge \bigwedge_{i=1}^{n} [\beta_i^d \wedge \beta_i^e]$$

the root label $L(r)$ contains a subset of $\{q_1, \dots, q_n\}$ that forms a saturated and satisfying assignment of $I$.

On the other hand, a saturated, satisfying assignment of $I$ can be used to construct a model of $\psi(I)$. A chain of $2n+1$ worlds suffices: Label the satisfying assignment in the root and fill the remaining worlds with depth propositions and emptiness propositions in a way that satisfies $\psi(I)$. From this it follows

$$I \in \text{PARTWEIGHT-SAT} \Leftrightarrow \psi(I) \in \text{CTL-SAT}(\mathsf{AF}).$$

The formula is computable in fpt-time w. r. t. to the parameter $\kappa$ of the left hand side, i.e. the number $k$ of partitions and the pathwidth of the CNF formula $\varphi$. To complete the reduction it remains to show that the structural pathwidth of $\psi$ (and therefore its structural treewidth) is bounded by $\kappa$. For this, obtain a path-decomposition of $\psi(I)$ as follows.

Assume that $\mathcal{P}$ is a path-decomposition of the structure of $\varphi$ s. t. there are bags $B_1, \dots, B_n$ which introduce the variables $q_1, \dots, q_n$ in this order. $\mathcal{P}$ can be obtained using the same argumentation as in Lemma 5.15. Obtain a path-decomposition $\mathcal{P}'$ of $\mathcal{S}_\psi$ as follows: For every variable $q_i$, insert all nodes of non-atomic subformulas of $[\beta_i^d \wedge \beta_i^e]$ into the bag $B_i$. Add the variables $d_{i-1}$, $d_i$, $e_{i-1}$, $e_i$ to the bags $B_i$ and $B_{i+1}$. Every bag has now at most the width of $\mathcal{P}$ plus a constant. The $n-1$ conjunctions between the formulas $[\beta_i^d \wedge \beta_i^e]$ also can be added with increasing the bag size by at most two.

To cover a formula $\gamma^+(p, C(p))$, append $C(p)+1$ new bags to $\mathcal{P}'$ that successively cover $(\mathrm{SF}(\gamma^+(p,j)) \setminus \mathrm{SF}(\gamma^+(p,j-1))) \cup \{\gamma^+(p,j-1)\}$ for each $j = 0, \ldots, C(p)$. Every such constructed bag has only constant width. Handle the $\gamma^-$ instances identically. The only variables contained in $\gamma$-subformulas are $p^+$ and $p^-$ for $p = 1, \ldots, k$. These variables as well as the remaining connectives of $\psi(I)$ can be added to every bag, increasing the width of $\mathcal{P}'$ only by a computable function of $k$. □

For the next theorem we require a result about complexity of modal logic. Define ML-SAT($\{p\}, D$) as the set of formulas $\varphi$ that have at most one propositional variable and are satisfied by a serial Kripke structure.

**Lemma 5.37.** *ML-SAT($\{p\}, D$) is PSPACE-complete.*

*Proof.* It was shown by Joseph Y. Halpern that the problem of modal satisfiability with only one variable is already PSPACE-complete [Hal95]. The upper bound is clear due to the problem being a restriction of ML-SAT. Halpern states a reduction from the problem *Quantified Boolean Formula Truth (QBF)* for the lower bound as follows.

A *quantified Boolean formula (qbf)* is of the form $Q_1 p_1 Q_2 p_2 \ldots Q_m p_m \phi$ where every $p_i$ is a propositional variable, $Q_i \in \{\exists, \forall\}$ and $\phi$ is a propositional formula on the variables $p_1, \ldots, p_m$. A qbf $\exists p \phi$ is defined as true if and only if $\phi[p/\top]$ or $\phi[p/\bot]$ is true, i.e. $p$ is replaced by $\top$ resp. $\bot$ in $\phi$, and a qbf $\forall p \phi$ is defined as true if both $\phi[p/\top]$ and $\phi[p/\bot]$ are true.

In a first step, we encode truth of qbf's in a modal logic formula. The formula uses modal operators to span a full binary tree, modeling assignments to the quantified variables as leaves. In the second step we will reduce the number of required propositional variables to one. For these parts of the proof we follow Halpern [Hal95, Sec. 3]. We will then restrict the problem to serial frames.

Assume a qbf of the form $Q_1 p_1 Q_2 p_2 \ldots Q_m p_m \phi$. A binary tree of depth $m+1$ is built using the following modal formulas:

Enforce monotonicity of the depth predicates $d_1, \ldots, d_{m+1}$:

$$\psi[\text{monotone}] := \bigwedge_{i=1}^{m+1} (d_i \rightarrow d_{i-1}) \tag{5.27}$$

At depth $i$ choose a value for $p_i$ and keep it fixed for deeper worlds:

$$\psi[\text{determined}] := \bigwedge_{i=1}^{m} \left[ d_i \rightarrow \Big( \big(p_i \rightarrow \Box(d_i \rightarrow p_i)\big) \wedge \big(\neg p_i \rightarrow \Box(d_i \rightarrow \neg p_i)\big) \Big) \right] \tag{5.28}$$

Introduce successors for every world that correctly handle the branching depending on the quantifiers in the qbf. For a universal ($\forall$) quantifier have two successors:

$$\begin{aligned}\psi[\text{branch}]_{i,\forall} \quad &:= \big[(d_{i-1} \land \neg d_i) \to \big(\Diamond(d_i \land \neg d_{i+1} \land p_i) \\ &\qquad\qquad\qquad\qquad \land \Diamond(d_i \land \neg d_{i+1} \land \neg p_i)\big)\big]\end{aligned} \tag{5.29}$$

For an existential ($\exists$) quantifier have at least one successor with arbitrary assignment of $p_i$:

$$\psi[\text{branch}]_{i,\exists} \quad := \big[(d_{i-1} \land \neg d_i) \to \Diamond(d_i \land \neg d_{i+1})\big] \tag{5.30}$$

The constructed modal formula $\psi$ is then:

$$\psi := d_0 \land \neg d_1 \land \bigwedge_{i=1}^{m} \Box^i \big[\psi[\text{monotone}] \land \psi[\text{determined}] \land \psi[\text{branch}]_{i,Q_i} \land (d_m \to \phi)\big]$$

It holds that $\psi$ is satisfiable if and only if $Q_1 p_1 Q_2 p_2 \ldots Q_m p_m \phi$ is true.

In the next step we reduce the number of propositional variables by replacing them by so-called *primitive-proposition-like (pp-like)* formulas. Intuitively, pp-like formulas are “independent enough” from each other so they can be used as an immediate replacement for propositional variables occuring in a modal formula.

Let $p$ be a single propositional variable. We construct a family $(q_j)_{j\in\mathbb{N}}$ of pp-like formulas:

$$q_j = \Diamond(\neg p \land \Diamond^j p)$$

Intuitively, $q_j$ is true at a state $w_0$ iff. there is a path $(w_0, \ldots, w_j)$ starting in $w_0$ such that $p$ is false in $w_1$ but true in $w_j$. Halpern shows that this formula family is indeed pp-like, thus in the formula $\psi$ given above the $2m+1$ variables can be replaced by $q_1, \ldots, q_{2m+1}$ and the reduction stays valid.

In the last step we show that the formula $\psi$ is satisfiable if and only if it is satisfied in a serial Kripke structure by making a satisfying structure serial (the direction from right to left comes for free). First observe that the family $(q_j)_{j\in\mathbb{N}}$ is still pp-like in serial frames. Every state which is affected by a $\Box$-subformula also has to fulfill a $\Diamond$-subformula (since it has a depth predicate labeled), hence it already has a successor. Vice versa, states without successors cannot have a depth predicate labeled, hence we can arbitrarily add successors or loops without affecting the validity of the reduction from QBF. $\Box$

**Lemma 5.38.** CTL-SAT(AX) *and* CTL$^\star$-SAT(A, X) *are complete for* para-**PSPACE** *when parameterized by structural treewidth.*

*Proof.* As CTL$^\star$-SAT(A, X) $\in$ **PSPACE**, the membership follows automatically for any parameter [Mei11]. For the hardness consider the problem ML-SAT($\{p\}, D$) which is PSPACE-complete (see Lemma 5.37). Now define a function $g : \mathbf{ML} \to \mathbf{CTL}(\mathsf{AX})$ via

$$g(\varphi) := \begin{cases} \varphi[\Box/\mathsf{AX}, \Diamond/\neg\mathsf{AX}\neg] & \text{if } \varphi \text{ has at most one variable,} \\ \bot & \text{otherwise} \end{cases}$$

i.e. $g$ simply replaces the modal operators $\Box$, $\Diamond$ with the corresponding temporal operators ($\mathsf{EX} \equiv \neg\mathsf{AX}\neg$). Indeed $g$ is computable in linear time. For any $\psi = g(\varphi)$ we have $\mathrm{tw}(\mathcal{S}_\psi) \leq 2$. Therefore $g$ is a $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from ML-SAT($\{p\}, D$) to $(\text{CTL-SAT}(\mathsf{AX}), \mathrm{tw})_2$. As now a single slice of it is already **PSPACE**-hard, the parameterized problem $(\text{CTL-SAT}(\mathsf{AX}), \mathrm{tw})$ is para-**PSPACE**-hard.

The constant treewidth of $\mathcal{S}_\varphi$ can be understood with a simple consideration: A formula without variables always has a tree-like structure and treewidth one. Adding a single variable to the structure increases the size of each bag of an optimal tree decomposition by at most one. ☐

**Theorem 5.39.** *Let $\kappa(\varphi) := tw(\mathcal{S}_\varphi)$, i.e. the parameter is only the structural treewidth.*

- CTL-SAT($T$) *is in* **FPT** *if* $T = \emptyset$, para-**PSPACE**-*hard if* $\{\mathsf{AX}\} \sqsubseteq T$ *and otherwise* **W**[1]-*hard.*
- LTL-SAT($T$) *is in* **FPT** *if* $T \subseteq \{\mathsf{X}\}$ *and otherwise* **W**[1]-*hard.*
- CTL$^\star$-SAT($T$) *is in* **FPT** *if* $T \sqsubseteq \{\mathsf{A}\}$ *or* $\{\mathsf{A}\} \not\sqsubseteq T$, para-**PSPACE**-*hard if* $\{\mathsf{A}, \mathsf{X}\} \sqsubseteq T$ *and otherwise* **W**[1]-*hard.*

*Proof.* A temporal formula without temporal operators is satisfiable if and only if it is satisfiable in a model with a single world. Hence **FPT** membership for $T = \emptyset$ in the CTL and LTL cases follows from Theorem 5.12 and Theorem 5.31: The MSO formulas that are used can easily be replaced by such with constant length. CTL$^*$ formulas cannot contain path formulas as subformulas unless they use $\mathsf{A}, \mathsf{E}$ operators together with $\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}$ operators and hence have this property as well. The tractable case LTL-SAT($\mathsf{X}$) is proven in Lemma 5.35.

Obviously $\mathrm{td}(\varphi)+\mathrm{tw}(\mathcal{S}_\varphi)$ is an upper bound for $\mathrm{tw}(\mathcal{S}_\varphi)$, thus satisfiability for the fragments **CTL**($\mathsf{AG}$) and **CTL**($\mathsf{AU}$) is again **W**[1]-hard via an identity reduction from the parameterization by treewidth and temporal depth (see Lemma 5.17, Lemma 5.18 and Lemma 5.19). Analogously the satisfiability problems for **LTL**($\mathsf{F}$), **LTL**($\mathsf{U}$), **CTL**$^\star$($\mathsf{A}, \mathsf{F}$) and **CTL**$^\star$($\mathsf{A}, \mathsf{U}$) are **W**[1]-hard. The case **CTL**($\mathsf{AF}$) is **W**[1]-hard due to Lemma 5.36. ☐

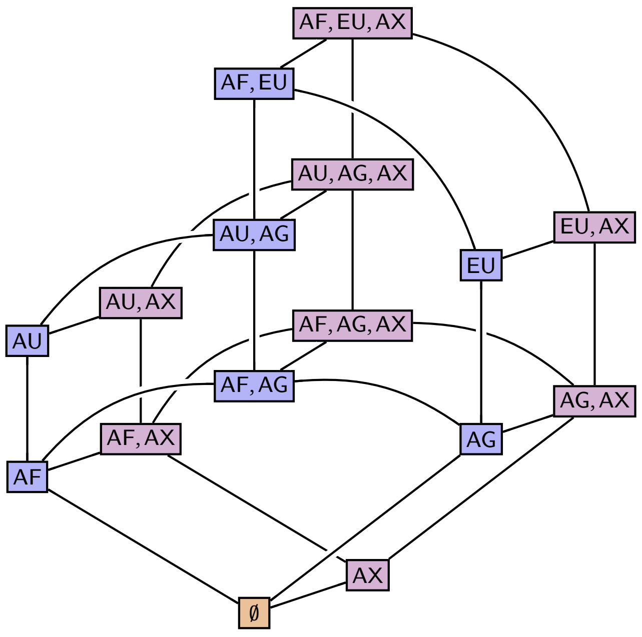
**Corollary 5.40.** CTL-SAT($T$) *is* para-**PSPACE**-*complete if* $\{\mathsf{AX}\} \sqsubseteq T \sqsubseteq \{\mathsf{AX}, \mathsf{AF}\}$.

**Theorem 5.41.** *Let $\kappa(\varphi) := td(\varphi)$, i.e. the parameter is only the temporal depth. Then for any $T$ the problems* CTL-SAT($T$)*,* LTL-SAT($T$) *and* CTL$^\star$-SAT($T$) *are* para-**NP**-*hard under $\leq^{fpt}$-reduction.*

*Proof.* Because propositional formulas have no temporal operators, it holds SAT $\leq^{\mathrm{P}}_{\mathrm{m}}$ (CTL-SAT($\emptyset$), td)$_0$ and thus the slice (CTL-SAT($T$), td)$_0$ is already **NP**-hard.

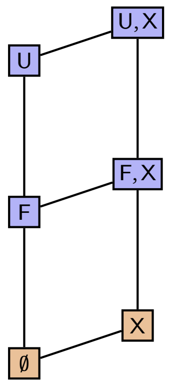
The same argument holds for LTL and CTL$^*$. □

**Corollary 5.42.** (CTL-SAT($\emptyset$), *td*)*,* (LTL-SAT($\emptyset$), *td*)*,* (CTL$^\star$-SAT($\emptyset$), *td*)*,* (LTL-SAT($\mathsf{X}$), *td*)*, and* (LTL-SAT($\mathsf{F}$), *td*) *are complete for* para-**NP** *under $\leq^{fpt}$-reduction.*



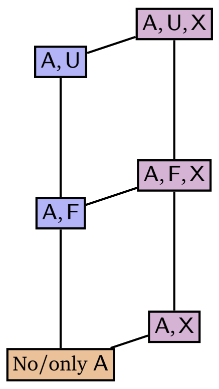
🟧 in **FPT**
🟦 **W**[1]-hard
🟪 para-**PSPACE**-hard

Figure 5.9: Complexity of CTL-SAT parameterized by $\kappa = \text{tw}$



🟧 in **FPT**
🟦 **W**[1]-hard

(a) Complexity of LTL satisfiability



🟧 in **FPT**
🟦 **W**[1]-hard
🟪 para-**PSPACE**-hard

(b) Complexity of CTL$^*$ satisfiability

Figure 5.10: Complexity of temporal logics parameterized by $\kappa = \mathrm{tw}$

# 6 Parameterizing temporal model checking of LTL and CTL$^*$

The model checking of temporal logics started to gain attention in theory as well as in practice in the last decades, whereas satisfiability (or equivalently validity) was the primarily interesting problem for logicians in the early days of temporal logics.

The success of verification with temporal logic in practice doubtlessly is due to the discovery of efficient algorithms, in particular models for CTL can be decided in polynomial time:

**Theorem 6.1** (Beyersdorff et al., 2011 [BMM+11]). CTL-MC$(T)$ *is* **P**-*complete if* $T \neq \emptyset$ *and* $\mathbf{NC}^1$-*complete if* $T = \emptyset$.

Here and in the following, $\mathcal{L}$-MC$(T)$ is the model checking problem of the logic $\mathcal{L}$ restricted to the operators that can be simulated by $T$. $\mathbf{NC}^1$ is the class of problems that can be decided by Boolean circuits of only logarithmic depth (see [Vol99] for a formal definition).

But also LTL grew important despite its PSPACE-complete validity and model checking problems.

**Corollary 6.2.** LTL-$\exists$MC$(\emptyset)$, LTL-$\forall$MC$(\emptyset)$ *and* CTL$^\star$-MC$(\emptyset)$ *are* $\mathbf{NC}^1$-*complete.*

**Theorem 6.3** (Sistla and Clarke, 1985 [SC85]). LTL-$\forall$MC$(T)$ *is* **coNP**-*complete if* $T = \{\mathsf{X}\}$ *or* $T = \{\mathsf{F}\}$ *and* **PSPACE**-*complete if* $\{\mathsf{X}, \mathsf{F}\} \subseteq T$ *or* $\mathsf{U} \in T$.

**Theorem 6.4** (Schnoebelen [Sch02]). CTL$^\star$-MC$(T)$ *is:*

- $\mathbf{NC}^1$-*complete if $T$ contains no or only path quantifiers,*
- $\mathbf{P}^{\mathbf{NP}}$-*complete if* $T \equiv \{\mathsf{A}, \mathsf{F}\}$,
- $\mathbf{P}^{\mathbf{NP}[\log^2 n]}$-*complete if* $T \equiv \{\mathsf{A}, \mathsf{X}\}$,
- **PSPACE**-*complete otherwise (namely if* LTL-$\forall$MC$(T)$ *also is* **PSPACE**-*complete)*

Here, $\mathbf{P}^{\mathbf{NP}}$ is the class of problems that are decidable by such deterministic polynomial time Turing machines that have access to an **NP** *oracle*. $\mathbf{P}^{\mathbf{NP}[\log^2 n]}$ is defined similar, but only $\mathcal{O}(\log^2 n)$ oracle accesses are allowed in a computation of the Turing machine.

In automata theory it is a known result that LTL model checking is a special case of satisfiability. The rough idea is as follows: A given LTL formula $\varphi$ can equivalently described by a Büchi automaton $\mathcal{B}(\varphi)$.[1] But for every finite model $\mathcal{A}$ there is an automaton $\mathcal{B}(\mathcal{A})$ which accepts exactly this model. Now it holds that $\varphi$ is satisfiable if $L(\mathcal{B}(\varphi))$, the language accepted by $\mathcal{B}(\varphi)$, is non-empty and that $\mathcal{A} \models \varphi$ if the language $L(\mathcal{B}(\neg\varphi)) \cap L(\mathcal{B}(\mathcal{A}))$ is empty.

Even if general model checking for LTL and CTL* is PSPACE-complete, statements as early as from Lichtenstein and Pnueli already distinguished between *program complexity*, the runtime dependence on the length of the formula $\varphi$, and *structure complexity*, the runtime dependence on the length of the structure $\mathcal{A}$ to be checked. They stated that the runtime factor $2^{|\varphi|}$ does not prohibit efficient model checking as the size of the structure is clearly dominant in practice.

**Theorem 6.5** (Lichtenstein and Pnueli, 1985 [LP85]). *There is an algorithm that decides if* $(\mathcal{A}, w, \varphi) \in$ LTL-$\forall$MC *in time* $2^{\mathcal{O}(|\varphi|)} \cdot \mathcal{O}(|\mathcal{A}|)$.

**Theorem 6.6** (Emerson and Lei, 1987 [EL87a]). *There is an algorithm that decides if* $(\mathcal{A}, w, \varphi) \in$ CTL$^\star$-MC *in time* $2^{\mathcal{O}(|\varphi|)} \cdot \mathcal{O}(|\mathcal{A}|)$.

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

**Corollary 6.7.** *Let* $\kappa(\varphi, \mathcal{A}, w) := |\varphi|$. *For* $Q \in \{$LTL-$\exists$MC, LTL-$\forall$MC, CTL$^\star$-MC$\}$ *we have* $(Q, \kappa) \in$ **FPT**.

In the rest of this chapter the influence of several parameterizations on the model checking complexity is studied. Starting with temporal depth as parameter we can get hardness results similar as for the satisfiability problems.

## 6.1 Temporal depth

**Definition 6.8.** Define $\mathbf{LTL}_c(T)$ as the fragment of $\mathbf{LTL}(T)$ which has temporal depth at most $c$, and similar $\mathbf{CTL}^\star_c(T)$ as the corresponding subset of $\mathbf{CTL}^\star(T)$.

Define LTL$_c$-$\forall$MC$(T)$ and CTL$^\star_c$-MC$(T)$ as the model checking problems restricted to $\mathbf{LTL}_c(T)$ resp. $\mathbf{CTL}^\star_c(T)$ formulas.

**Lemma 6.9.** (LTL-$\forall$MC$(\mathsf{F})$, $td$) *is complete for* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

*Proof.* We follow Sistla and Clarke and show that LTL$_1$-$\forall$MC$(\mathsf{F})$ (i.e. only $\mathsf{F}$-operators without nesting) is **coNP**-hard. This is done by a reduction from the **NP**-complete *3SAT* problem: Given a propositional formula $\varphi$ in 3CNF[2], is it satisfiable?

[1]In fact every LTL formula defines a *ω-regular* language, but not vice versa.

[2]Conjunctive normal form with exactly three literals per clause

For this we construct a formula $\psi \in \mathbf{LTL}_1$ and a structure $\mathcal{S}$ such that $(\mathcal{S}, w_0) \models \psi$ if and only if $\varphi$ is unsatisfiable. First assume $\varphi = \bigwedge_{i=1}^{m} \left(L_{i,1} \vee L_{i,2} \vee L_{i,3}\right)$ where $L_{i,j}$ is a literal, i.e. a propositional variable or its negation. Then simply define $\psi := \bigvee_{i=1}^{m} \left(\mathsf{F}\neg L_{i,1} \wedge \mathsf{F}\neg L_{i,2} \wedge \mathsf{F}\neg L_{i,3}\right)$, so $\psi$ is basically the negation of $\varphi$ supplemented with $\mathsf{F}$ operators in front of the literals.

Assume that $\varphi$ contains variables $\{ x_1, \ldots, x_n \}$. For a correct reduction the structure $\mathcal{S}$ is now required to allow either $\mathsf{F}x_i$ or $\mathsf{F}\neg x_i$ 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 $\mathcal{S}$ and vice versa. The structure depicted in Figure 6.1 has these property and therefore models propositional assignments as runs from its initial world $w_0$. This means that all runs in $\mathcal{S}$ fulfill the path formula $\psi$ if and only if $\neg\varphi$ is satisfied by all Boolean assignments. Hence $\varphi \notin 3\mathrm{SAT} \Leftrightarrow (\psi, \mathcal{S}, w_0) \in \mathrm{LTL}_1\text{-}\forall\mathrm{MC}(\mathsf{F})$. $\psi$ and $\mathcal{S}$ are both constructible in linear time.

Due to a small model property also $\mathrm{LTL}\text{-}\forall\mathrm{MC}(F) \in \mathbf{coNP}$ holds [SC85] and the para-**coNP** completeness follows from Theorem 2.17. □

**Corollary 6.10.** *Let* $\{\mathsf{F}\} \sqsubseteq T$. *Then the problem* $(\mathrm{LTL}\text{-}\exists\mathrm{MC}(T), td)$ *is* para-**NP**-*hard and the dual problem* $(\mathrm{LTL}\text{-}\forall\mathrm{MC}(T), td)$ *is* para-**coNP**-*hard under* $\leq^{fpt}$-*reduction.*

**Corollary 6.11.** *Let* $\{\mathsf{A}, \mathsf{F}\} \sqsubseteq T$. *Then* $(\mathrm{CTL}^\star\text{-}\mathrm{MC}(T), td)$ *is* para-**NP**-*hard and* para-**coNP**-*hard under* $\leq^{fpt}$-*reduction.*

The twofold hardness of $\mathrm{CTL}^\star$-MC results from the path quantifiers available in $\mathbf{CTL}^\star$; $\mathrm{CTL}^\star$-MC incorporates LTL-$\forall$MC as well as LTL-$\exists$MC.

When replacing $\mathsf{F}\alpha$ by $\top\mathsf{U}\alpha$ in Lemma 6.9 the intractability automatically carries over to the corresponding model checking problems with $\mathsf{U}$-operators.

**Lemma 6.12.** *Let $T$ be a non-empty set of temporal operators. Then* $\mathrm{LTL}\text{-}\forall\mathrm{MC}(T)$ *is* **coNP**-*hard.*

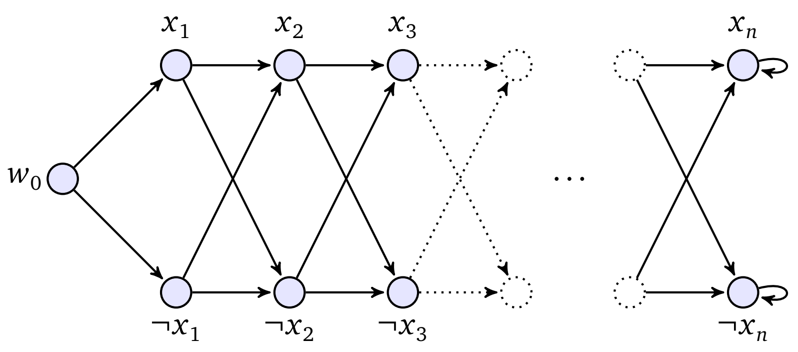


Figure 6.1: Structure that models propositional assignment as runs

*Proof.* For $\mathsf{X} \in T$ we modify the reduction given in Lemma 6.9. Simply replace the subformula $\mathsf{F}x_i$ by $\mathsf{X}^i x_i$ in $\psi$ and the reduction stays valid. This substitution leads only to polynomial blowup. The cases $\mathsf{F} \in T$ and $\mathsf{U} \in T$ follow from Lemma 6.9. □

**Theorem 6.13.** *Let* $T \sqsubseteq \{\mathsf{A}, \mathsf{X}\}$. *Then* $(\mathrm{CTL}^\star\text{-MC}(T), td) \in \mathbf{XP}$.

*Proof.* Algorithm 6.1 is based on the *depth-bounded search tree*[1] method and decides $\mathrm{CTL}^\star\text{-MC}(\mathsf{A}, \mathsf{X})$.

The main idea is similar to Theorem 5.31: We recursively traverse the given formula $\varphi$, only in this case we are not quantifying potential quasi-labels but instead check the labels of the model. Due to the lack of $\mathsf{F}$ and $\mathsf{U}$ operators it is sufficient to follow paths in the model only up to depth $\mathrm{td}(\varphi)$. This leads to a recursion tree with depth roughly $\mathrm{td}(\varphi)$ and maximal branching degree $|W|$. Note that between two such **foreach**-branchings there is at most a polynomial sized subtree of *checkState*, *checkAllPaths* and *checkOnePath* calls which do the processing of propositional connectives.

The resulting runtime is thus:

$$|\varphi|^{\mathcal{O}(1)} \cdot |W|^{\mathrm{td}(\varphi)}$$

□

*Remark:* The term $\mathrm{td}(\varphi)$ that appears as exponent of $|W|$ is typical for **XP**-runtime; even for **LTL**(X) it is unlikely to find such an approach by exhaustive search that works in FPT-runtime. Note that on the other hand the hardness of LTL model checking arises *only* from the problem of searching the correct path in a possibly branching structure! In fact, the LTL model checking problem on non-branching structures is in **P** due to a dynamic programming algorithm (but it is unknown if it is **P**-hard like CTL-MC) [Gor09].

**Theorem 6.14.** *Let* $T \subseteq \{\mathsf{X}\}$. *Then* $(\mathrm{LTL}\text{-}\forall\mathrm{MC}(T), td) \in \mathbf{coW}[P]$.

*Proof.* Let a formula $\varphi \in \mathbf{LTL}(\mathsf{X})$ and a structure $\mathcal{A}$ with root $w$ be given. Now it holds that $(\mathcal{A}, w) \not\models \varphi$ if and only if there is a path $\pi \in \Pi(w)$ s. t. $(\mathcal{A}, \pi) \not\models \varphi$. But this depends only on a finite prefix of $\pi$ that has length $\mathrm{td}(\varphi)$. This can be proven similar to Theorem 5.28.

So to determine if the given formula is not satisfied by the structure, simply guess a finite path of length $\mathrm{td}(\varphi)$ through $\mathcal{A}$ and verify the formula. This requires at most $\mathcal{O}(\mathrm{td}(\varphi) \cdot \log |\mathcal{A}|)$ non-deterministic steps (using pointers to denote worlds) and leads to **coW**[**P**] according to Definition 2.13. □

[1]Further information on this method is presented e.g. by Niedermeier [Nie06].

**Algorithm 6.1:** Algorithm for deciding CTL$^\star$-MC$(\mathsf{A},\mathsf{X})$

```
Input  : $\varphi \in \mathbf{CTL}^\star(\mathsf{A},\mathsf{X})$
         $\mathcal{A} = (W,R,L)$ Kripke structure
         $w_0 \in W$ initial world
Output : $(\mathcal{A},w_0) \models \varphi$ ?
Procedure checkState($\varphi, w$)
    if $\varphi = x, x \in \mathbf{PS}$ then return $x \in L(w)$
    if $\varphi = \neg x, x \in \mathbf{PS}$ then return $x \notin L(w)$
    if $\varphi = \alpha \wedge \beta$ then return checkState($\alpha, w$) $\wedge$ checkState($\beta, w$)
    if $\varphi = \alpha \vee \beta$ then return checkState($\alpha, w$) $\vee$ checkState($\beta, w$)
    if $\varphi = \mathsf{A}\alpha$ then return checkAllPaths($\alpha, w$)
    if $\varphi = \mathsf{E}\alpha$ then return checkOnePath($\alpha, w$)
Procedure checkAllPaths($\varphi, w$)
    if $\varphi = \mathsf{X}\alpha$ then
        foreach R-successor $v$ of $w$ do
            if $\neg$ checkAllPaths($\alpha, v$) then return false
        return true
    if $\varphi = \alpha \wedge \beta$ then return checkAllPaths($\alpha, w$) $\wedge$ checkAllPaths($\beta, w$)
    if $\varphi = \alpha \vee \beta$ then return checkAllPaths($\alpha, w$) $\vee$ checkAllPaths($\beta, w$)
    return checkState($\varphi, w$)                    /* state formula */
Procedure checkOnePath($\varphi, w$)
    if $\varphi = \mathsf{X}\alpha$ then
        foreach R-successor $v$ of $w$ do
            if checkOnePaths($\alpha, v$) then return true
        return false
    if $\varphi = \alpha \wedge \beta$ then return checkOnePath($\alpha, w$) $\wedge$ checkOnePath($\beta, w$)
    if $\varphi = \alpha \vee \beta$ then return checkOnePath($\alpha, w$) $\vee$ checkOnePath($\beta, w$)
    return checkState($\varphi, w$)                    /* state formula */
Transform $\varphi$ into NNF
return checkAllPaths($\varphi, w_0$)
```

**Definition 6.15** (Maximum branching degree)**.** Let $\mathcal{A}$ be a Kripke structure. Then write $\Delta(\mathcal{A})$ for the maximum branching degree in $\mathcal{A}$, i.e. the smallest number $\Delta$ s. t. every world in $\mathcal{A}$ has at most $\Delta$ successors.

**Theorem 6.16.**

1. $(\text{LTL-}\forall\text{MC}(T), td + \Delta)$ *is in* **FPT** *for* $T \sqsubseteq \{\mathsf{X}\}$*,* para-**coNP**-*complete for* $T \equiv \{\mathsf{F}\}$ *and* para-**coNP**-*hard otherwise,*
2. $(\text{CTL}^\star\text{-MC}(T), td + \Delta)$ *is in* **FPT** *for* $\{\mathsf{A}\} \not\sqsubseteq T$ *or* $T \sqsubseteq \{\mathsf{A}, \mathsf{X}\}$ *and hard for* para-**NP** *and* para-**coNP** *otherwise.*

*Proof.* The runtime of Algorithm 6.1 is at most $|\varphi|^{\mathcal{O}(1)} \cdot \Delta(\mathcal{A})^{\text{td}(\varphi)}$ because the branching of its recursion tree is bounded by the branching of the Kripke structure. This proves the **FPT** cases. For the para-**coNP**-hard cases it is obvious that the Kripke structures in Theorem 6.17 in fact have only branching of degree two. □

## 6.2 Treewidth and pathwidth

Define $\text{tw}_{\mathcal{A}}$ as the treewidth of the input structure, i.e. $\text{tw}_{\mathcal{A}}(\varphi, \mathcal{A}, w) := \text{tw}(\mathcal{A})$. Define $\text{tw}_{\varphi}$ as the structural treewidth of the input formula, i.e. $\text{tw}_{\varphi}(\varphi, \mathcal{A}, w) := \text{tw}(\mathcal{S}_{\varphi})$. Similarly define $\text{pw}_{\mathcal{A}}$ and $\text{pw}_{\varphi}$.

**Theorem 6.17.** *For every non-empty set $T$ of temporal operators* $(\text{LTL-}\forall\text{MC}(T), pw_{\mathcal{A}})$ *is hard for* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

*Proof.* This is shown using the same argument as in Lemma 6.9 and Lemma 6.12. The structure $\mathcal{S}$ used there has a constant pathwidth of at most three: One bag $B_0$ contains the worlds $w_0, w_1^+, w_1^-$, where $w_i^+$ ($w_i^-$) is the unique world in $\mathcal{S}$ that has $x_i$ ($\neg x_i$) labeled. Further bags $B_i$ contain $w_i^+, w_{i+1}^+, w_i^-$ and $w_{i+1}^-$ for $1 \leq i < n$. Connecting the bags $B_i$ and $B_{i+1}$ for $0 \leq i < n$ results in a path-decomposition of $\mathcal{S}$ with width at most three. □

**Corollary 6.18.** *For every non-empty set $T$ of temporal operators* $(\text{LTL-}\forall\text{MC}(T), tw_{\mathcal{A}})$ *is hard for* para-**coNP**.

**Corollary 6.19.** *For* $T \equiv \{\mathsf{F}\}$ *or* $T \equiv \{\mathsf{X}\}$*,* $(\text{LTL-}\forall\text{MC}(T), tw_{\mathcal{A}})$ *is complete for* para-**coNP**.

**Corollary 6.20.** *For every set $T$ that contains at least one path quantifier and one temporal operator,* $(\text{CTL}^{\star}\text{-MC}(T), pw_{\mathcal{A}})$ *and* $(\text{CTL}^{\star}\text{-MC}(T), tw_{\mathcal{A}})$ *are hard for* para-**NP** *and* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

This hardness result is not surprising when considering usual LTL model checking algorithms since they already have runtime exponential in $|\varphi|$ but only linear in $|\mathcal{A}|$.

As mentioned above, model checking of LTL can be done by intersecting an $\omega$-language defined by the input formula with an $\omega$-language defined by the input structure. The satisfiability problem LTL-SAT(X) is in **FPT** w. r. t. structural treewidth of the formula (see Lemma 5.35). There is however a problem when considering this parameterization for model checking: The Büchi automaton $\mathcal{B}(\neg\varphi)$ can be constructed on-the-fly with its state depth bounded by $\text{td}(\varphi)$ [Muk97]. But the Büchi automaton $\mathcal{B}(\mathcal{A})$ that represents the input structure cannot be bounded by the structural treewidth of $\varphi$ or its temporal depth (it is completely independent of $\varphi$). Hence we cannot use Courcelle's theorem for LTL model checking due to the unbounded size of the model.

In fact we get the following hardness result in constrast to the tractable satisfiability problem.

**Lemma 6.21.** $(\text{LTL-}\forall\text{MC}(\mathsf{X}), tw_{\varphi})$ *is complete for* para-**coNP**.

*Proof.* The hardness proof can be done similar to Lemma 6.9 and Lemma 6.12. The crucial modification is to simulate the formulas $\mathsf{F}x_1, \dots, \mathsf{F}x_n$ by certain LTL formulas using only a constant number of variables. This leads to a formula with a constant structural treewidth.

First modify the structure as shown in Figure 6.2, i.e. replace every variable $x_i$ by $q$ and every variable $\neg x_i$ by $\neg q$. Then from the constructed formula $\psi$ obtain $\psi'$ by substituting $\mathsf{X}^i q$ for $\mathsf{F}x_i$ and $\mathsf{X}^i\neg q$ for $\mathsf{F}\neg x_i$. Since any path from $w_0$ through $\mathcal{S}$ can only visit either $w_i^+$ or $w_i^-$, fulfilling the formula $\mathsf{X}^i q$ is equivalent to choosing $x_i = 1$ in a Boolean assignment for $\varphi$. Thus we again get that $\varphi$ is valid if and only if $(\psi', \mathcal{S}, w_0) \in \text{LTL-}\forall\text{MC}$. □

**Corollary 6.22.** *Let* $\mathsf{X} \in T$*. Then* $(\text{LTL-}\exists\text{MC}(T), tw_{\varphi})$ *resp.* $(\text{LTL-}\forall\text{MC}(T), tw_{\varphi})$ *is hard for* para-**NP** *resp.* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

**Corollary 6.23.** *Let* $\{\mathsf{A}, \mathsf{X}\} \sqsubseteq T$*. Then* $(\text{CTL}^{\star}\text{-MC}(T), tw_{\varphi})$ *is hard for* para-**NP** *and* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

**Lemma 6.24.** $(\text{LTL-}\forall\text{MC}(\mathsf{F}), tw_{\varphi})$ *is hard for* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

*Proof.* We change the proof of Lemma 6.21 again such that the resulting structure $\mathcal{S}$ has only constantly many variables but the formula $\varphi$ uses only $\mathsf{F}$ operators. This proves the theorem similar to Lemma 6.21.

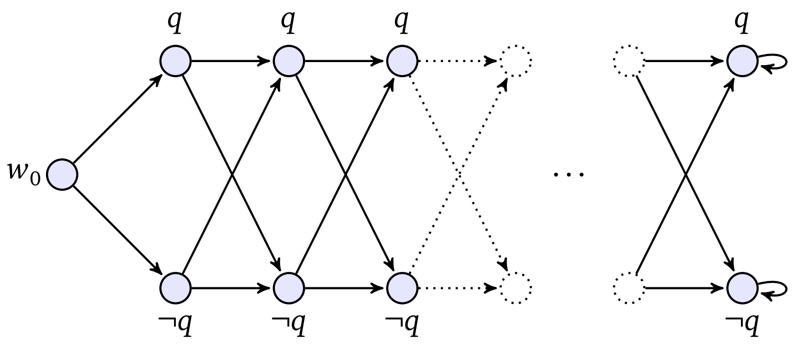


Figure 6.2: A one-variable structure with hard LTL(X) model checking

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 $\pi$ and $\pi'$ say that $\pi'$ is a *j-suffix* of $\pi$ if there is an $i \geq j$ s. t. $\pi' = \pi_{\geq i}$.

Consider the structure $\mathcal{S}$ with only one variable that was defined in Lemma 6.21. Modify it such that every world $w_i^+$,$w_i^-$ and $w_i$ has the variable $p_{\text{even}}$ labeled if $i$ is even, and otherwise the variable $p_{\text{odd}}$. The resulting $\mathcal{S}$ is illustrated in Figure 6.3.

To create a matching LTL formula, first inductively define a shortcut operator $\mathsf{F}_{i,j}$. $\mathsf{F}_{i,j}(\alpha)$ holds on a path $\pi \in \Pi(w_i^+) \cup \Pi(w_i^-)$ if a $j$-suffix of $\pi$ fulfills $\alpha$. For this, we "skip" at least $j$ worlds using the odd- and even-literals:

$$\mathsf{F}_{i,0}(\alpha) := \mathsf{F}\alpha$$

$$\mathsf{F}_{i,j+1}(\alpha) := \begin{cases} \mathsf{F}(p_{\text{even}} \land \neg p_{\text{odd}} \land \mathsf{F}_{i+1,j}(\alpha)) & \text{if } i \text{ is odd} \\ \mathsf{F}(p_{\text{odd}} \land \neg p_{\text{even}} \land \mathsf{F}_{i+1,j}(\alpha)) & \text{if } i \text{ is even} \end{cases}$$

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 $\mathcal{S}$ starts at $w_n^+$ or $w_n^-$ exactly when it fulfills $\mathsf{G}p_{\text{even}}$ ($n$ is even) resp. $\mathsf{G}p_{\text{odd}}$ ($n$ is odd). In the rest of the proof, write $p_{\text{end}}$ for the matching literal depending on $n$. Then a path $\pi \in \Pi(w_i^+) \cup \Pi(w_i^-)$ fulfills $\mathsf{F}_{i,j}(\mathsf{F}\neg p_{\text{end}})$ if after skipping $j$ or more worlds it gets to a world which still has a proper successor in $\mathcal{S}$. Thus $\mathsf{F}_{i,j}(\mathsf{F}\neg p_{\text{end}})$ means that there are at least $j+1$ more steps between $\pi[1]$ and a world with $p_{\text{end}}$ labeled.

We aggregate the subexpressions into

$$\mathsf{F}_{=j}(\alpha) := \mathsf{F}_{0,j}\left(\alpha \land \mathsf{F}_{j,n-j-1}(\neg p_{\text{end}})\right) \text{ for } n > j \text{ and}$$
$$\mathsf{F}_{=n}(\alpha) := \mathsf{F}_{0,n}(\alpha).$$

Due to the construction $\mathsf{F}_{=j}(\alpha)$ is the desired formula; it states that for a path $\pi \in$

$\Pi(w_0)$ the expression $\alpha$ should hold in world $\pi[j]$: This is the only world on $\pi$ which is reachable from $w_0$ with at least $j$ steps but which still has distance at least $n-j$ from the world $\pi[n]$.

The rest of the proof follows Lemma 6.21 but replaces $\mathsf{X}^i q$ by $\mathsf{F}_{=i}(q)$. Then again every path starting in $w_0$ fulfills the constructed $\psi$ if and only if it visits worlds $W = \{w^+_{i_1}, w^+_{i_2}, \ldots, w^+_{i_m}\}$ s. t. $X = \{x_{i_1}, x_{i_2}, \ldots, x_{i_m}\}$ is a satisfying assignment for the original $\varphi$. Also the structure $\mathcal{S}$ and the formula $\psi$ are again constructible in polynomial time. $\square$

**Corollary 6.25.** *Let* $\{\mathsf{F}\} \sqsubseteq T$ *or* $\{\mathsf{U}\} \sqsubseteq T$*. Then the problem* $(\text{LTL-}\exists\text{MC}(T), tw_\varphi)$ *resp.* $(\text{LTL-}\forall\text{MC}(T), tw_\varphi)$ *is hard for* para-**NP** *resp.* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

**Corollary 6.26.** *Let* $\{\mathsf{A}, \mathsf{F}\} \sqsubseteq T$ *or* $\{\mathsf{A}, \mathsf{U}\} \sqsubseteq T$*. Then* $(\text{CTL}^\star\text{-MC}(T), tw_\varphi)$ *is hard for* para-**NP** *and* para-**coNP** *under* $\leq^{fpt}$*-reduction.*

The proof of Lemma 6.24 would also work with only one "even" variable $p$ and its negated literal $\neg p$, together with $q$ resulting in only two variables, but using $p_{\text{even}}$ and $p_{\text{odd}}$ makes it clearer.

The given reduction from 3SAT would likely not work with only one variable. The used "even-odd" trick is necessary since $\mathsf{F}$-formulas are "compressible" in the sense that future in most temporal logics is reflexive (the present being a part of the future). Also it seems that the linear nesting depth of $\mathsf{F}$ or $\mathsf{X}$ operators cannot be avoided if only constantly many variables are used. Therefore it is unlikely that 3SAT can offer para-**coNP**-hardness for model checking w. r. t. the parameterization $\text{td} + \text{tw}_\varphi$. We can however use the **W**[1]-complete SquareTiling problem.

**Definition 6.27** (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: $d = \langle c_u, c_d, c_l, c_r \rangle$. A *D-tiling* for a discrete area $R \subseteq \mathbb{N} \times \mathbb{N}$ is a function $\gamma : R \to D$.

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:

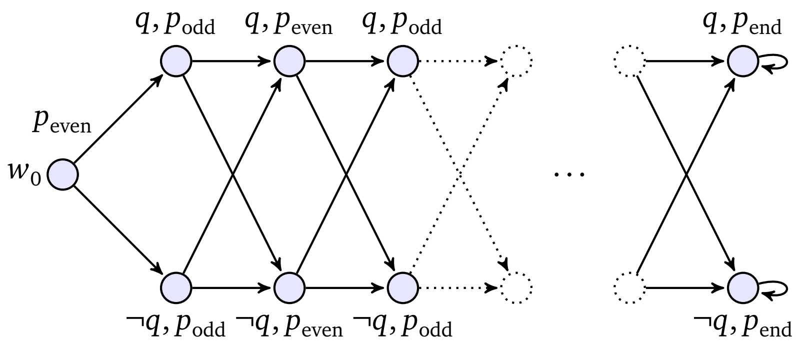


Figure 6.3: A three-variable structure with hard LTL(F) model checking

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

Tiling problems usually ask if a certain area has a valid tiling.

**Definition 6.28.** The problem SQUARETILING contains the instances $(C, D, \langle k\rangle_1)$ for which the $k \times k$-grid has a valid $D$-tiling:

$$\textsc{SquareTiling} := \left\{ (C, D, \langle k\rangle_1) \mid D \subseteq C^4, k \in \mathbb{N} \text{ and } [k] \times [k] \text{ has a valid } D\text{-tiling} \right\}$$

Here, $\langle\cdot\rangle_1$ denotes a unary encoding.

**Theorem 6.29** (Chlebus, 1986 [Chl86])**.** SQUARETILING *is* **NP**-*complete.*

**Theorem 6.30** (Cai, Chen, Downey and Fellows, 1997 [CCDF97])**.** *Let* $\kappa(C, D, \langle k\rangle_1) := k$. *Then the parameterized problem* (SQUARETILING, $\kappa$) *is* **W**[1]-*complete.*

**Theorem 6.31.** *Let* $\kappa(\varphi, \mathcal{A}, w) := td(\varphi) + pw(\mathcal{S}_\varphi)$. *Then* (LTL-$\forall$MC(X), $\kappa$) *is* **coW**[1]-*hard.*

*Proof.* The idea of the proof, a reduction from SQUARETILING, is to use the path semantics of LTL to describe a valid tiling of the $k \times k$ grid: For every SQUARETILING 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 pathwidth. The Kripke structure $\mathcal{S}$ however will have unbounded branching degree $\Delta$ (which is unlikely to be avoided as LTL-$\forall$MC 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 $w_d^i$.
- Connect $w_{\text{start}}$ to $w_d^1$ for every $d \in D$.
- Connect $w_d^{k^2}$ to $w_{\text{end}}$ for every $d \in D$.
- Connect $w_d^i$ to $w_{d'}^{i+1}$ for every $d, d' \in D$ and $1 \le i < k^2$.
- Connect $w_{\text{end}}$ to itself.
- In every world $w_d^i$ with $d = \langle c_u, c_d, c_l, c_r\rangle$ label propositional variables $c_u^i$, $c_d^i$, $c_l^i$, $c_r^i$.
- In every world $w_d^i$ where $i = k \cdot j$ for $j \in [k]$ label a propositional variable $q_{\text{border}}$.

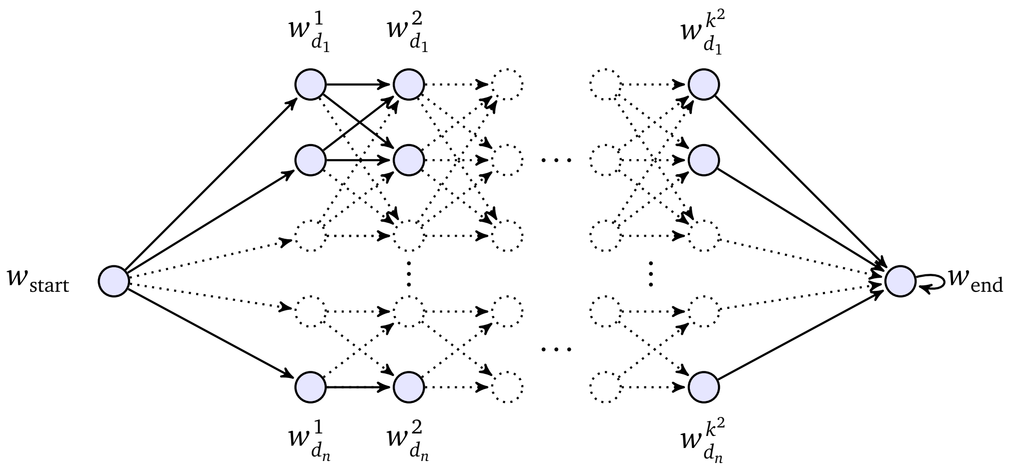


Figure 6.4: Structure that models square tilings as runs

The structure $\mathcal{S}$ is illustrated in Figure 6.4. 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 $w_d^{(i-1)\cdot k+j}$.

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

$$\psi^i_{c,r} := \left[q_{\text{border}} \vee \left(c^i_r \rightarrow \mathsf{X}(q_{\text{end}} \vee c^{i+1}_l)\right)\right]$$
$$\psi^i_{c,d} := \left[c^i_d \rightarrow \mathsf{X}^k \left(q_{\text{end}} \vee c^{i+k}_u\right)\right]$$

The formula $\psi^i_{c,r}$ is true in a path $\pi$ starting in a world $w^i_d$ (which has color $c$ to the right) if $\pi$ chooses a matching successor: Either $w^i_d$ is a *border* and the color to the right is not relevant, or $w^i_d$ 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 $\psi^i_{c,d}$ 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 $i$:

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

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

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

**Claim.** *Let $\pi = (w_{start}, w^1_{d_1}, w^2_{d_2}, \ldots, w^{k^2}_{d_{k^2}}, w_{end}, \ldots)$ be a run through $\mathcal{S}$. Then $\pi \models \psi$ if and only if $d_1, d_2, \ldots, d_{k^2}$ 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 $\pi[i+1] = w^i_{d_i}$. But as $w^i_{d_i}$ is not a border and $\pi_{\geq i+1} \models \psi^i_{c,r}$, so the successor has $c$ as left color. But this means that $c^i_r$ is labeled in $w^i_{d_i}$ and $c^{i+1}_l$ is labeled in $w^{i+1}_{d_{i+1}}$, contradiction to $c' \neq c$.
- Case 2: $x' = x$ and $y' = y+1$ which is similar proven as Case 1.

"$\Leftarrow$": Let $d_1, d_2, \ldots, d_{k^2}$ be a valid tiling $\gamma$ of $[k] \times [k]$. Assume that $\neg\psi$ holds, i.e. there is a color $c$ and an $i$ s. t. $\pi_{\geq i+1}$ does not satisfy $\psi^i_{c,r} \wedge \psi^i_{c,d}$. If $\psi^i_{c,r}$ is false, then $w^i_{d_i}$ is not a border but also has a different *right* color than its successor on $\pi$ has as *left* color. But then $\gamma$ would not be a valid tiling. The case that $\psi^i_{c,d}$ is false can be handled analogously. ■

It is easy to see that every run $\pi$ through $\mathcal{S}$ from $w_{\text{start}}$ has the form as in the above claim, i.e. $\pi = (w_{\text{start}}, w^1_{d_1}, w^2_{d_2}, \ldots, w^{k^2}_{d_{k^2}}, w_{\text{end}}, \ldots)$. Hence we get

$$(C,D,k) \in \text{SQUARETILING} \Leftrightarrow \exists \pi \in \Pi(w_{\text{start}}) : (\mathcal{S}, \pi) \models \psi$$

and converse

$$\begin{aligned}(C,D,k) \notin \text{SQUARETILING} &\Leftrightarrow \neg\exists \pi \in \Pi(w_{\text{start}}) : (\mathcal{S}, \pi) \models \psi \\ &\Leftrightarrow \forall \pi \in \Pi(w_{\text{start}}) : (\mathcal{S}, \pi) \models \neg\psi \\ &\Leftrightarrow (\neg\psi, \mathcal{S}, w_{\text{start}}) \in \text{LTL-}\forall\text{MC}(\mathsf{X})\end{aligned}$$

**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 $\mathsf{X}$ operators in $\psi$.

For the pathwidth we construct a path-decomposition $\mathcal{P}$ 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 $\vee, \rightarrow$ and $\vee$ in $\psi^i_{c,r}$,
- the Boolean connectives $\rightarrow$ and $\vee$ in $\psi^i_{c,d}$,
- the variables $q_{\text{border}}$, $q_{\text{end}}$, $c^i_r$, $c^{i+1}_l$, $c^i_d$ and $c^{i+k}_u$,
- the single X-operator in $\psi^i_{c,r}$,
- the $k$ X-operators in $\psi^i_{c,d}$.

The isolated bag covers every edge between nodes representing subformulas of $\psi^i_{c,r}$ and $\psi^i_{c,d}$ with a width of $|B^i_c| = 3+2+6+1+k = k+12$. Also every subformula of $\psi^i_{c,r}$ and $\psi^i_{c,d}$ 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 $\mathcal{P}$. 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: First for every formula $\xi^i_c := \left(\psi^i_{c,r} \wedge \psi^i_{c,d}\right)$, 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 $((((\xi_1 \wedge \xi_2) \wedge \xi_3) \ldots) \wedge \xi_{|C|})$. 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 $\mathcal{P}$, and add the $j$-th $\wedge$-node into both bags, similar as in Lemma 5.15. Then after inserting the last conjunction, add the $i$ nodes for $\mathsf{X}^i$ 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 $\mathcal{P}$ now consists of $k^2$ disconnected sequences of $|C|$ bags each, concatenate them into a path in arbitrary order. This leads to $\mathcal{P}$ 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. ■

As the above claims show

$$(\textsc{SquareTiling}, k) \leq^{fpt} (\text{LTL-}\exists\text{MC}(\mathsf{X}), \text{td} + \text{pw}_\varphi),$$

the theorem follows. □

**Corollary 6.32.** *Let* $\mathsf{X} \in T$*. Then* LTL-∃MC *is* **W**[1]*-hard and* LTL-∀MC($T$) *is* **coW**[1]*-hard when parameterized by temporal depth and structural treewidth resp. pathwidth.*

**Corollary 6.33.** *Let* $\{\mathsf{A}, \mathsf{X}\} \sqsubseteq T$*. Then* CTL*-MC($T$) *is* **W**[1]*-hard and* **coW**[1]*-hard when parameterized by temporal depth and structural treewidth resp. pathwidth.*

The **W**[1]-hard case **LTL**(X) is special because it is the only case of LTL model checking where a non-trivial (read: actually using the parameter) upper bound (**W**[**P**]) has been found. It would be very interesting to improve the result to a completeness result for some class $\mathbf{W}[\cdot]$. But such a result seems to be much harder to find for the following reason.

The class $\mathbf{W}[1]$ can be characterized as the class of parameterized problems which are reducible to the $p$-SHORT-NSTM-HALT problem. $p$-SHORT-NSTM-HALT contains the pairs $(M, k)$ s. t. $M$ is a single-tape NTM halting on the empty word in at most $k$ steps. It is plausible that the $\mathbf{W}[1]$-hard LTL model checking can simulate this problem when $\mathsf{X}$ operators of depth $k$ are available. On the other hand we have already seen that LTL-$\exists$MC($\mathsf{X}$) $\in \mathbf{W}[\mathbf{P}]$, $k \cdot \log n$ non-deterministic bits allow to guess the fulfilling path. To have $\mathbf{W}[1]$ also as an upper bound, we would have to reduce the process of choosing $k$ worlds (forming a path) in a model to a Turing machine making $k$ non-deterministic transitions. At first sight it is perfectly possible to do this for a model $\mathcal{M}$ by having $|\mathcal{M}|$-fold branching of the Turing machine. But a correct reduction would also require that the propositional formulas at temporal depth $i$ are evaluated on the $i$-th chosen world in constant time, but checking propositional assignments is already $\mathbf{NC}^1$-hard and therefore impossible in constant time. It is also not clear how the evaluation could be "sufficiently precomputed" during the reduction (due to the exponential number of possible propositional assignments).

**Theorem 6.34.** *Let* $\{\mathsf{F}\} \sqsubseteq T$ *or* $\{\mathsf{U}\} \sqsubseteq T$. *Then* (LTL-$\forall$MC($T$), $\kappa$) *is* $\mathbf{coW}[1]$*-hard when parameterized by temporal depth and structural treewidth resp. pathwidth.*

*Proof.* We adapt the reduction given in Theorem 6.31. First label a new depth proposition $d_i$ in every world $w_d^i$ for $d \in D$, $1 \leq i \leq k^2$. Then change the formulas as follows:

$$\psi^i_{c,r} := \left[q_{\text{border}} \vee \left(c^i_r \to \mathsf{F}(c^{i+1}_l)\right)\right]$$

$$\psi^i_{c,d} := \begin{cases} \left[c^i_d \to \mathsf{F}\left(c^{i+k}_u\right)\right] & \text{if } i+k \leq k^2 \\ \top & \text{otherwise} \end{cases}$$

$$\psi := \bigwedge_{i=1}^{k^2-1} \left[\mathsf{F}\left(d_i \wedge \bigwedge_{c \in C} \left(\psi^i_{c,r} \wedge \psi^i_{c,d}\right)\right)\right]$$

This does increase the pathwidth at most by $k^2$ for $d_1, \ldots, d_{k^2}$. Also $\mathsf{F}\alpha$ can be replaced by $\top\mathsf{U}\alpha$ as usual. □

**Corollary 6.35.** *For* $\{\mathsf{A}, \mathsf{F}\} \sqsubseteq T$, CTL*-MC($T$) *is* $\mathbf{W}[1]$*-hard and* $\mathbf{coW}[1]$*-hard when parameterized by temporal depth and structural treewidth resp. pathwidth.*

**Definition 6.36.** 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:

$$\textsc{RectangleTiling} := \left\{ (C, c_0, c_1, D) \;\middle|\; \begin{array}{l} D \subseteq C^4, \text{ and for an } m \in \mathbb{N} \text{ the} \\ [|D|] \times [m] \text{ plane has a valid} \\ D\text{-tiling with } c_0 \in C \text{ at the top edge} \\ \text{and } c_1 \in C \text{ at the bottom edge} \end{array} \right\}$$

**Theorem 6.37** (Chlebus, 1986 [Chl86])**.** *The problem* RECTANGLETILING *is* **PSPACE**-*complete.*

**Theorem 6.38.** *Let* $\kappa(\varphi, \mathcal{A}, w) := pw(\mathcal{S}_\varphi)$*. Then* $(\text{LTL-}\forall\text{MC}(\mathsf{X}, \mathsf{F}), \kappa)$ *is* para-**PSPACE**-*complete.*

*Proof.* We consider a $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from RECTANGLETILING to LTL-∀MC(X, F) such that only LTL formulas with constant structural pathwidth are produced. As LTL-∀MC is in **PSPACE**, this proves the theorem according to Theorem 2.17. This reduction originally is from Sistla and Clarke 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 Theorem 6.31 we construct a Kripke structure $\mathcal{S}$ that models $n \times m$-tilings as runs:

- Add worlds $w_{\text{left}}$, $w_{\text{right}}$ and $w_{\text{end}}$ which have each only one proposition labeled, namely $q_{\text{left}}$, $q_{\text{right}}$ and $q_{\text{end}}$.
- For every tile $d \in D$ and for every $i \in [n]$ add a world $w^i_d$.
- Connect $w_{\text{left}}$ to $w^1_d$ for every $d \in D$.
- Connect $w^k_d$ to $w_{\text{right}}$ for every $d \in D$.
- Connect $w^i_d$ to $w^{i+1}_{d'}$ 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 $w^i_d$ with $d = \langle c_u, c_d, c_l, c_r \rangle$ label propositional variables $c_u$, $c_d$, $c_l$, $c_r$.

The structure $\mathcal{S}$ is shown in Figure 6.5 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.

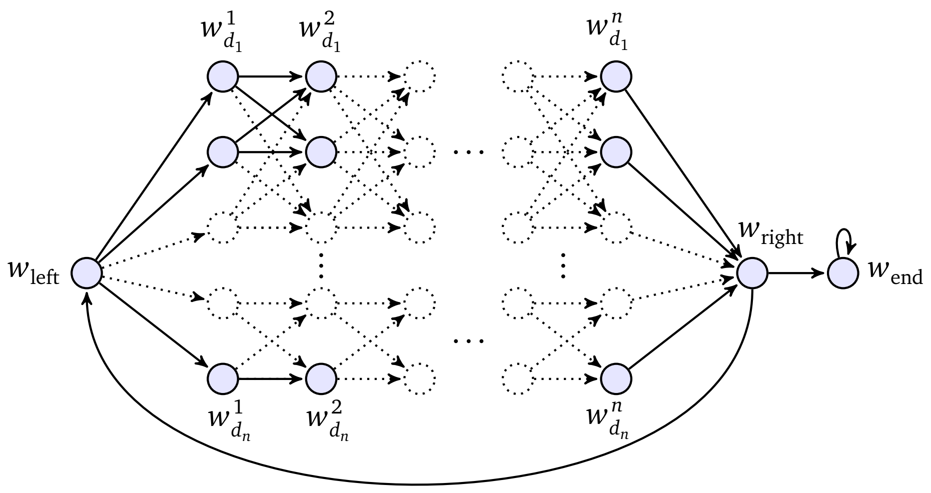


Figure 6.5: Structure that models rectangle tilings as runs

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

$$\psi_{\text{first}} := \bigwedge_{i=1}^{n} \mathsf{X}^i (c_0)_u$$

Check the neighbor to the right and below (if it is not the border):

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

The last row must exist and have *down* color $c_1$:

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

The whole tiling is expressed by $\psi$:

$$\psi := \psi_{\text{first}} \wedge \psi_{\text{last}} \wedge \neg\mathsf{F}\neg \bigwedge_{c \in C} \left(\psi_{c,r} \wedge \psi_{c,d}\right)$$

Similar to Theorem 6.31 is 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 $\psi_{\text{first}}$. It contains $n$ "chains" of $\mathsf{X}$-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 $\mathsf{X}$-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 Theorem 6.31, 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 $n+2$ $\mathsf{X}$-operators can be decomposed like in $\psi_{\text{first}}$, leading to $n+2$ new bags with each constant size.

The length of $\psi_{last}$ is again not constant; it contains a big conjunction of chains of $\mathsf{X}$-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 $\wedge$-nodes and the $\mathsf{F}$ 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 $\wedge$-operators, negations and the $\mathsf{F}$ 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.

Due to the claim and the **PSPACE**-hardness of RECTANGLETILING we get that the problem LTL-$\forall$MC($\mathsf{X},\mathsf{F}$) is para-**PSPACE**-hard for the structural pathwidth parameter. $\square$

**Corollary 6.39.** *For* $\{\mathsf{X},\mathsf{F}\} \sqsubseteq T$*, the problems* LTL-$\forall$MC($T$) *and* CTL$^\star$-MC($\{\mathsf{A}\} \cup T$) *are* para-**PSPACE**-*complete when parameterized by structural pathwidth or treewidth.*

**Theorem 6.40.** *Let* $\kappa(\varphi, \mathcal{A}, w) := td(\varphi) + pw(\mathcal{S}_\varphi)$*. Then* (LTL-$\forall$MC($\mathsf{U}$), $\kappa$) *is* para-**PSPACE**-*complete.*

*Proof.* For the Until operator the reduction from Theorem 6.38 is possible in constant temporal depth. Similar to Theorem 6.34 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. $c_u^i, c_d^i, c_l^i$ and $c_r^i$ for $1 \leq i \leq n$.

Modify the formulas as follows:

$$\begin{aligned}
\psi_{\text{first}} &:= [q_{\text{left}} \vee (c_0)_u]\,\mathsf{U} q_{\text{right}} \\
\psi_{\text{last}} &:= \top \mathsf{U}\left[q_{\text{left}} \wedge [(q_{\text{left}} \vee (c_1)_d)\,\mathsf{U} q_{\text{right}}]\right] \\
\psi^i_{c,r} &:= c^i_r \to \left[c^i_r \mathsf{U} c^{i+1}_l\right] \\
\psi^i_{c,d} &:= c^i_d \to \left[c^i_d \mathsf{U}\left(\neg c^i_d \mathsf{U}(q_{\text{end}} \vee c^i_u)\right)\right] \\
\psi &:= \psi_{\text{first}} \wedge \psi_{\text{last}} \wedge \neg\left[\top \mathsf{U} \neg \bigwedge_{i=1}^{n} \bigwedge_{c \in C} \left(\psi^i_{c,r} \wedge \psi^i_{c,d}\right)\right]
\end{aligned}$$

The variables $(c_0)_u, (c_1)_d, q_{\text{left}}$ and $q_{\text{right}}$ 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 $\psi^i_{c,r}$ and $\psi^i_{c,d}$. But each such subformula $\psi^i_{c,r}$ resp. $\psi^i_{c,d}$ can be covered by a single isolated bag: It has only a constant number of nodes and every occuring 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. □

**Corollary 6.41.** *For* $\mathsf{U} \in T$*,* LTL-$\exists$MC($T$)*,* LTL-$\forall$MC($T$) *and* CTL$^\star$-MC($\{\mathsf{A}\} \cup T$) *are* para-**PSPACE***-complete when parameterized by temporal depth and structural pathwidth resp. treewidth.*

# 7 Summary and conclusion

In 2013 Praveen asked: *“Does Treewidth Help in Modal Satisfiability?”*

He saw that the otherwise **PSPACE**-complete modal satisfiability problem becomes fixed-parameter tractable when the modal depth and the structural treewidth are chosen as parameter. If we modify the question to

*“Does Treewidth Help in Temporal Satisfiability?”*

then the correct answer should be: *“Only if it is Modal Logic.”* This might be surprising at first sight, but this thesis shows that the recent results on applying Courcelle’s theorem to modal logic likely cannot be transferred to temporal fragments with “future” or “global” operators; they are $\mathbf{W}[1]$-hard under the given parameterization. Praveen already observed that this is the case for modal logic restricted to transitive frames. Also treewidth alone is not a sufficient parameterization, a combination with modal depth is required. Still, for some kinds of logics the modal or temporal depth exactly fails to help as a parameter. The results imply that such logics are those with fixpoint operators like $\mathsf{F}$. It is plausible that the hardness is inherent to all logics which can express “deep” properties of structures with low operator depth.

The $\mathbf{W}[1]$-hardness for a combination of treewidth and temporal depth is however not yet proven for the pure $\{\mathsf{AF}, \mathsf{EG}\}$ CTL fragment, but the long model property (see Theorem 5.20) forbids that satisfiability can be handled with a similar approach as for $\{\mathsf{AX}\}$. Expressing $\{\mathsf{AF}, \mathsf{EG}\}$ satisfiability with Courcelle’s theorem is presumably only possible with much more technial work, if at all.

Similar, the used parameterizations have no to little effect on the intractable temporal model checking cases. Merely the $\{\mathsf{X}\}$ resp. $\{\mathsf{A}, \mathsf{X}\}$ fragment falls down to $\mathbf{W}[\mathbf{P}]$ resp. **XP** with the temporal depth as parameter; but in contrast to satisfiability no fragment becomes fixed-parameter tractable, not even together with treewidth as parameter. Therefore the model checking algorithm by Lichtenstein and Pnueli which runs in time $2^{|\varphi|} \cdot |\mathcal{A}|$ stays the best known FPT result for LTL ($|\varphi|$ is the parameter).

Future research possibilities are the determination of a non-trivial parameterization for model checking that suffices for fixed-parameter tractability. Such parameterizations have potentially big relevance in practice where model checking tasks are dominant for temporal logics. Different approaches of LTL model checking, e.g. in terms of Büchi automata, may help here. Also the $\{\mathsf{AF}, \mathsf{EG}\}$ fragment is not yet fully understood in terms of complexity: Does the fact that both operators are composed of “mixed” quantifiers lead to fixed-parameter tractability?

| Problem $Q$ | Parameter $\kappa$ | | | |
|---|---|---|---|---|
| CTL-SAT$(\cdot)$ | - | td | $\mathrm{tw}_\varphi$ | $\mathrm{td}+\mathrm{tw}_\varphi$ / $\mathrm{td}+\mathrm{pw}_\varphi$ |
| $\emptyset$ | **NP**-c. | para-**NP**-c. | **FPT** | **FPT** |
| AF | ? | para-**NP**-c. | **W**[1]-h. | ? |
| AX | **PSPACE**-c. | para-**NP**-h. | para-**PSPACE**-c. | **FPT** |
| AF, AX | **PSPACE**-c. | para-**NP**-h. | para-**PSPACE**-c. | **W**[1]-h. |
| AG | **PSPACE**-c. | para-**NP**-h. | **W**[1]-h. | **W**[1]-h. |
| AG, AF | **PSPACE**-c. | para-**NP**-h. | **W**[1]-h. | **W**[1]-h. |
| other | **EXP**-c. | para-**NP**-h. | **W**[1]-h. | **W**[1]-h. |
| other (with AX) | **EXP**-c. | para-**NP**-h. | para-**PSPACE**-h. | **W**[1]-h. |
| CTL-MC$(\cdot)$ | | | | |
| $\emptyset$ | **NC**$^1$-c. | | | |
| other | **P**-c. | | | |

| Problem $Q$ | Parameter $\kappa$ | | | |
|---|---|---|---|---|
| LTL-SAT$(\cdot)$ | - | td | $\mathrm{tw}_\varphi$ / $\mathrm{pw}_\varphi$ | $\mathrm{td}+\mathrm{tw}_\varphi$ / $\mathrm{td}+\mathrm{pw}_\varphi$ |
| $\emptyset$ | **NP**-c. | para-**NP**-c. | **FPT** | **FPT** |
| X | **NP**-c. | para-**NP**-c. | **FPT** | **FPT** |
| F | **NP**-c. | para-**NP**-c. | **W**[1]-h. | **W**[1]-h. |
| other | **PSPACE**-c. | para-**NP**-h. | **W**[1]-h. | **W**[1]-h. |
| LTL-$\forall$MC$(\cdot)$ | - | td | $\mathrm{td}+\Delta$ | $\mathrm{tw}_\varphi$ |
| $\emptyset$ | **NC**$^1$-c. | | | |
| X | **coNP**-c. | **coW**[**P**], **coW**[1]-h. | **FPT** | para-**coNP**-c. |
| F | **coNP**-c. | para-**coNP**-c. | para-**coNP**-c. | para-**coNP**-c. |
| F, X | **PSPACE**-c. | para-**coNP**-h. | para-**coNP**-h. | para-**PSPACE**-c. |
| other | **PSPACE**-c. | para-**PSPACE**-c. | para-**coNP**-h. | para-**PSPACE**-c. |
| | | $\mathrm{pw}_{\mathcal{A}}$ / $\mathrm{tw}_{\mathcal{A}}$ | $\lvert\varphi\rvert$ | $\mathrm{td}+\mathrm{tw}_\varphi$ / $\mathrm{td}+\mathrm{pw}_\varphi$ |
| X | | para-**coNP**-c. | **FPT** | **coW**[**P**], **coW**[1]-h. |
| F | | para-**coNP**-c. | **FPT** | **coW**[1]-h. |
| F, X | | para-**coNP**-h. | **FPT** | **coW**[1]-h. |
| other | | para-**coNP**-h. | **FPT** | para-**PSPACE**-c. |

| Problem $Q$ | Parameter $\kappa$ | | | |
|---|---|---|---|---|
| CTL$^\star$-SAT$(\cdot)$ | - | td | $\mathrm{tw}_\varphi$ | $\mathrm{td} + \mathrm{tw}_\varphi$ / $\mathrm{td} + \mathrm{pw}_\varphi$ |
| no / only A | **NP**-c. | para-**NP**-c. | **FPT** | **FPT** |
| A, X | **PSPACE**-c. | para-**NP**-h. | para-**PSPACE**-c. | **FPT** |
| A, F | **PSPACE**-c. | para-**NP**-h. | **W**[1]-h. | **W**[1]-h. |
| other | **2EXP**-c. | para-**NP**-h. | **W**[1]-h. | **W**[1]-h. |
| CTL$^\star$-MC$(\cdot)$ | - | td | $\mathrm{td} + \Delta$ | $\mathrm{tw}_\varphi$ |
| no / only A | **NC**$^1$-c. | | | |
| A, X | $\mathbf{P}^{\mathbf{NP}[\log^2 n]}$-c. | **XP**, (**co**)**W**[1]-h. | **FPT** | para-(**co**)**NP**-h. |
| A, F | $\mathbf{P}^{\mathbf{NP}}$-c. | para-(**co**)**NP**-h. | para-(**co**)**NP**-h. | para-(**co**)**NP**-h. |
| A, F, X | **PSPACE**-c. | para-(**co**)**NP**-h. | para-(**co**)**NP**-h. | para-**PSPACE**-c. |
| other with A | **PSPACE**-c. | para-**PSPACE**-c. | para-(**co**)**NP**-h. | para-**PSPACE**-c. |
| | | $\mathrm{tw}_\mathcal{A}$ / $\mathrm{pw}_\mathcal{A}$ | $\lvert\varphi\rvert$ | $\mathrm{td} + \mathrm{tw}_\varphi$ / $\mathrm{td} + \mathrm{pw}_\varphi$ |
| A, X | | para-(**co**)**NP**-h. | **FPT** | **XP**, (**co**)**W**[1]-h. |
| A, F | | para-(**co**)**NP**-h. | **FPT** | (**co**)**W**[1]-h. |
| A, F, X | | para-(**co**)**NP**-h. | **FPT** | (**co**)**W**[1]-h. |
| other with A | | para-(**co**)**NP**-h. | **FPT** | para-**PSPACE**-c. |

# Bibliography

[BdV01] Patrick Blackburn, Maarten de Rijke, Yde Venema. *Modal logic*. New York, NY, USA: Cambridge University Press, 2001.

[BK91] Hans L. Bodlaender, Ton Kloks. *Better algorithms for the pathwidth and treewidth of graphs*. Automata, Languages and Programming. Springer, 1991, pp. 544–555.

[BMM+11] Olaf Beyersdorff, Arne Meier, Martin Mundhenk, Thomas Schneider, Michael Thomas, Heribert Vollmer. *Model Checking CTL is Almost Always Inherently Sequential*. Logical Methods in Computer Science **7** (2011), no. 2. DOI: 10.2168/LMCS-7(2:12)2011.

[Bod93a] Hans L. Bodlaender. *A linear time algorithm for finding tree-decompositions of small treewidth*. Proceedings of the twenty-fifth annual ACM symposium on Theory of computing. ACM. 1993, pp. 226–234.

[Bod93b] Hans L. Bodlaender. *A Tourist Guide through Treewidth*. Acta Cybern. **11** (1993), no. 1-2, pp. 1–21.

[BSS+07] Michael Bauland, Thomas Schneider, Henning Schnoor, Ilka Schnoor, Heribert Vollmer. *The complexity of generalized satisfiability for linear temporal logic*. of Lecture Notes in Computer Science. Springer, 2007, pp. 48–62.

[Büc90] J.Richard Büchi. *On a Decision Method in Restricted Second Order Arithmetic*. The Collected Works of J. Richard Büchi. Ed. by Saunders Mac Lane, Dirk Siefkes. Springer New York, 1990, pp. 425–435. DOI: 10.1007/978-1-4613-8928-6_23.

[CCDF97] Liming Cai, Jianer Chen, Rodney G. Downey, Michael R. Fellows. *On the parameterized complexity of short computation and factorization*. Archive for Mathematical Logic (1997).

[CD89] Edmund M. Clarke, I. A. Draghicescu. *Expressibility Results for Linear-time and Branching-time Logics*. Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop. London, UK, UK: Springer-Verlag, 1989, pp. 428–437.

[CE12] Bruno Courcelle, Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*. Vol. 138. Encyclopedia of mathematics and its applications. Cambridge University Press, 2012, pp. I–XIV, 1–728.

[CE82] Edmund Clarke, E. Allen Emerson. *Design and synthesis of synchronization skeletons using branching time temporal logic*. Logics of Programs. **131**. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1982. Chap. 5, pp. 52–71. DOI: 10.1007/bfb0025774.

[CES86] E. M. Clarke, E. A. Emerson, A. P. Sistla. *Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications*. ACM Trans. Program. Lang. Syst. **8** (Apr. 1986), no. 2, pp. 244–263. DOI: 10.1145/5397.5399.

[Chl86] Bogdan S. Chlebus. *Domino-tiling games*. Journal of Computer and System Sciences **32** (1986), no. 3, pp. 374–392. DOI: 10.1016/0022-0000(86)90036-X.

[Cou90] Bruno Courcelle. *The monadic second-order logic of graphs. I. Recognizable sets of finite graphs*. Information and Computation **85** (Mar. 1990), no. 1, pp. 12–75. DOI: 10.1016/0890-5401(90)90043-h.

[DF99] Rodney G. Downey, Michael R. Fellows. *Parameterized Complexity*. 530 pp. Springer-Verlag, 1999.

[EH85] E. Allen Emerson, Joseph Y. Halpern. *Decision procedures and expressiveness in the temporal logic of branching time*. Journal of Computer and System Sciences **30** (Feb. 1985), no. 1, pp. 1–24. DOI: 10.1016/0022-0000(85)90001-7.

[EH86] E. Allen Emerson, Joseph Y. Halpern. *"Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time Temporal Logic*. J. ACM **33** (Jan. 1986), no. 1, pp. 151–178. DOI: 10.1145/4904.4999.

[EL87a] E. Allen Emerson, Chin-Laung Lei. *Modalities for Model Checking: Branching Time Logic Strikes Back*. Sci. Comput. Program. **8** (June 1987), no. 3, pp. 275–306. DOI: 10.1016/0167-6423(87)90036-0.

[EL87b] E. Allen Emerson, Chin-Laung Lei. *Modalities for model checking: branching time logic strikes back*. Science of Computer Programming **8** (1987), no. 3, pp. 275–306. DOI: 10.1016/0167-6423(87)90036-0.

[Eme90] E. Allen Emerson. *Temporal and Modal Logic*. Handbook of Theoretical Computer Science (Vol. B). Ed. by Jan van Leeuwen. Cambridge, MA, USA: MIT Press, 1990, pp. 995–1072.

[FFL+07] Michael Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, Stefan Szeider, Carsten Thomassen. *On the Complexity of Some Colorful Problems Parameterized by Treewidth*. Combinatorial Optimization and Applications. Ed. by Andreas Dress, Yinfeng Xu, Binhai Zhu. **4616**. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 366–377. DOI: 10.1007/978-3-540-73556-4_38.

[FG02] Jörg Flum, Martin Grohe. *Describing Parameterized Complexity Classes*. STACS 2002. Ed. by Helmut Alt, Afonso Ferreira. **2285**. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, pp. 359–371. DOI: 10.1007/3-540-45841-7_29.

[FG04] Markus Frick, Martin Grohe. *The complexity of first-order and monadic second-order logic revisited.* Ann. Pure Appl. Logic **130** (2004), no. 1-3, pp. 3–31.

[FG06] Jörg Flum, Martin Grohe. *Parameterized Complexity Theory*. Springer Verlag, 2006.

[GO07] Valentin Goranko, Martin Otto. *5 Model theory of modal logic*. Handbook of Modal Logic. Ed. by Johan Van Benthem Patrick Blackburn, Frank Wolter. **3**. Studies in Logic and Practical Reasoning. Elsevier, 2007, pp. 249–329. DOI: 10.1016/S1570-2464(07)80008-5.

[Gor09] Valentin Goranko. *Tutorial notes on Introduction to Temporal Logics for Specification and Verification*. First PhD Autumn School on Modal Logic, IT University of Copenhagen. 2009.

[Hal76] Rudolf Halin. *S-functions for graphs*. Journal of Geometry **8** (1976), no. 1-2, pp. 171–186. DOI: 10.1007/BF01917434.

[Hal95] Joseph Y. Halpern. *The Effect Of Bounding The Number Of Primitive Propositions And The Depth Of Nesting On The Complexity Of Modal Logic*. Artificial Intelligence **75** (1995), pp. 361–372.

[Kam68] Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis. Ucla, 1968.

[Lad77] Richard E. Ladner. *The Computational Complexity of Provability in Systems of Modal Propositional Logic*. SIAM Journal on Computing **6** (1977), no. 3, pp. 467–480. DOI: 10.1137/0206033.

[Lar95] Francois Laroussinie. *About the Expressive Power of CTL Combinators*. Inf. Process. Lett. **54** (June 1995), no. 6, pp. 343–345. DOI: 10.1016/0020-0190(95)00053-F.

[LMS15] Martin Lück, Arne Meier, Irena Schindler. *Parameterized Complexity of CTL: A Generalization of Courcelle's Theorem*. Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France. Proceedings. **8977**. Lecture Notes in Computer Science. Springer, 2015, pp. 549–560.

[LP85] Orna Lichtenstein, Amir Pnueli. *Checking That Finite State Concurrent Programs Satisfy Their Linear Specification*. Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. POPL '85. New Orleans, Louisiana, USA: ACM, 1985, pp. 97–107. DOI: 10.1145/318593.318622.

[Mar03] Nicolas Markey. *Temporal logic with past is exponentially more succinct, Concurrency Column*. Bulletin of the EATCS **79** (2003), pp. 122–128.

[Mei11] Arne Meier. *On the Complexity of Modal Logic Variants and their Fragments*. Cuvillier, 2011.

[MTVM09] Arne Meier, Michael Thomas, Heribert Vollmer, Martin Mundhenk. *The complexity of satisfiability for fragments of CTL and CTL**. International Journal of Foundations of Computer Science **20** (2009), no. 05, pp. 901–918.

[Muk97] Madhavan Mukund. *Linear-Time Temporal Logic and Büchi Automata*. Winter School on Logic and Computer Science, Indian Statistical Institute, Calcutta. 1997.

[Nie06] Rolf Niedermeier. *Invitation to Fixed Parameter Algorithms*. Oxford University Press, USA, Mar. 2006.

[Pnu77] Amir Pnueli. *The temporal logic of programs*. Foundations of Computer Science, 18th IEEE Annual Symposium on. 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32.

[Pnu79] Amir Pnueli. *The temporal semantics of concurrent programs*. Semantics of Concurrent Computation. Ed. by Gilles Kahn. **70**. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1979, pp. 1–20. DOI: 10.1007/BFb0022460.

[Pra13] Manjunatha Praveen. *Does Treewidth Help in Modal Satisfiability?* ACM Trans. Comput. Logic **14** (Aug. 2013), no. 3, 18:1–18:32. DOI: 10.1145/2499937.2499939.

[Pri57] Arthur N. Prior. *Time and Modality*. Oxford, 1957.

[Ram97] Siddharthan Ramachandramurthi. *The Structure and Number of Obstructions to Treewidth*. SIAM J. Discret. Math. **10** (Feb. 1997), no. 1, pp. 146–157. DOI: 10.1137/S0895480195280010.

[Rey11] Mark Reynolds. *A tableau-based decision procedure for CTL**. Formal Aspects of Computing **23** (2011), no. 6, pp. 739–779. DOI: 10.1007/s00165-011-0193-4.

[RS84] Neil Robertson, Paul D. Seymour. *Graph minors. III. Planar tree-width*. J. Comb. Theory, Ser. B **36** (1984), no. 1, pp. 49–64.

[SC85] A. P. Sistla, E. M. Clarke. *The Complexity of Propositional Linear Temporal Logics*. J. ACM **32** (July 1985), no. 3, pp. 733–749. DOI: 10.1145/3828.3837.

[Sch02] Philippe Schnoebelen. *The Complexity of Temporal Logic Model Checking*. Advances in Modal Logic. Ed. by Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, Michael Zakharyaschev. King's College Publications, 2002, pp. 393–436.

[SS06] Marko Samer, Stefan Szeider. *A Fixed-Parameter Algorithm for #SAT with Parameter Incidence Treewidth*. CoRR **abs/cs/0610174** (2006).

[SS10] Marko Samer, Stefan Szeider. *Constraint satisfaction with bounded tree-width revisited*. Journal of Computer and System Sciences **76** (2010), no. 2, pp. 103–114. DOI: 10.1016/j.jcss.2009.04.003.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999.

# Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 9. März 2015

Martin Lück