Parameterized Complexity of Decision Problems in Non-Classical Logics

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades

Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation von
Dipl.-Math. Irena Schindler
geboren am 05.07.1984 in Alma-Ata

2017

Referent:	Heribert Vollmer, Leibniz Universität Hannover
Korreferent:	Till Tantau, Universität zu Lübeck
Tag der Promotion:	07.06.2018

Für Valentina und Waldemar

ACKNOWLEGMENTS

First of all I would like express my sincere gratitude to my supervisor Heribert Vollmer not only for the accommodating support but also for manifold conversations on a intellectual and personal level, which I will keep in remembrance. Furthermore I would like to thank Arne Meier for his support, encouragement, motivation and also for the joint research. For the outstanding patience during the proof reading of this thesis, as well as for the numerous scientific discussions and collaborative research, I would like to thank my colleagues Martin Lück, Anselm Haak and Maurice Chandoo.
Also I would like to thank my husband Andreas Schindler for the aid and support during my doctoral studies, as well as for the care of the children during my research trips.
My sincere thanks go to my father-in-law Alexander Schindler and also my parents Valentina and Waldemar Lange, who have always supported me in all situations, not only mentally but also physically with the care of the children. Without this help I would not have been able to write this thesis.

Zuerst möchte ich meinem Doktorvater Heribert Vollmer tiefe Dankbarkeit aussprechen, nicht nur für die hilfsbereite Unterstützung während der Promotion, sondern auch für die mannigfache Gespräche auf intellektueller und persönlicher Ebene, an die ich mich gern erinnern werde. Weiter möchte ich meinem Betreuer Arne Meier für die Unterstützung, Ermutigung, Motivation und nicht zuletzt für die gemeinsame Forschung herzlichen Dank aussprechen.
Für die herausragende Geduld bei dem mehrfachen Korrekturlesen dieser Arbeit, sowie für die zahlreichen fachlichen Diskussionen und der gemeinsamen Forschung danke ich meinen Kollegen Martin Lück, Anselm Haak und Maurice Chandoo.
Meinem Ehemann Andreas Schindler danke ich für den Beistand und die Unterstützung während der gesamten Promotionszeit, sowie für die Betreuung der Kinder während der Forschungsreisen.
Mein größter Dank gilt meinem Schwiegervater Alexander Schindler, sowie meinen Eltern Waldemar und Valentina Lange, die mich stets in jeder Lage, nicht nur mental, sondern auch physisch bei der Betreuung der Kinder Unterstützt haben. Ohne dieser Hilfe, währe es mir nicht möglich gewesen diese Arbeit anzufertigen.

"Not everything that can be counted counts, and not everything that counts can be counted."

Albert Einstein

CONTENTS

ZUSAMMENFASSUNG

Parametrisierte Komplexität ist ein Teilgebiet der klassischen Komplexitätstheorie und wurde von Downey und Fellows [DF99] im Jahre 1999 erstmals vorgestellt. Sie untersuchten die strukturellen Eigenschaften eines gegebenen Problems und schränkten die Eingabe durch einen Parameter ein. In dieser Arbeit befassen wir uns mit der parametrisierten Komplexität unterschiedlicher Probleme in der Default-Logik sowie in temporalen Logiken.
Im ersten Abschnitt von Kapitel 3 stellen wir einen dynamischen Algorithmus vor, welcher in fpt-Zeit entscheidet, ob eine gegebene Default-Theorie eine stabile Erweiterung besitzt. Darüber hinaus enumeriert dieser alle generierenden Default-Regeln, die zur stabilen Erweiterung führen, mit einer Vorberechnungszeit, die linear mit der Größe der gegebenen Theorie und dreifach exponentiell mit der Baumweite wächst, gefolgt von linearem Delay für die Ausgabe der Lösungen.
Im zweiten Abschnitt von Kapitel 3 führen wir das Konzept der Backdoors aus der klassischen Aussagenlogik in die Default-Logik ein. Dabei untersuchen wir die Komplexität zweier Probleme. Das erste Problem ist das Finden eines Backdoors für eine vorgegebene Formel-Zielklasse, wie etwa HORN, KROM, POSITIVE-UNIT und MONOTONE. Das zweite Problem evaluiert eine gegebene Formel anhand eines vorgegebenen Backdoors.
Im vierten Kapitel beschäftigen wir uns mit unterschiedlichen temporalen Logiken. Im ersten Abschnitt stellen wir unterschiedliche "Graphen-ähnliche" Strukturen vor, um eine Formel zu repräsentieren. Anschließend definieren wir die zugehörigen Begriffe wie Pfad- und Baumweite. Wir geben eine verallgemeinerte Version des Satzes von Courcelle an die für unbeschränkte Signaturen gilt. Außerdem beschäftigen wir uns mit den Booleschen Operatorfragmenten im Sinne von Post's Lattice.
Im letzten Abschnitt führen wir den Begriff Backdoor in das "globally" Fragment der linearen temporalen Logik ein. Auch in diesem Abschnitt betrachten wir die beiden in diesem Zusammenhang oben genannten Probleme: das Finden eines Backdoors und die anschließende Evaluation der vorgegebenen Formel mit einem gegebenen Backdoor für Formel-Zielklassen HORN und KROM.

Schlagwörter: Prarametrisierte Komplexität, Default Logik, Temporale Logik, Backdoor, Post's Lattice, Baum- und Pfadweite, temporale Tiefe

ABSTRACT

Parameterized complexity is a branch of a computational complexity. The pioneers of this new and promising research field are Downey and Fellows [DF99]. They suggest to examine the structural properties of a given problem and restrict the instance by a parameter. In this thesis we investigate the parameterized complexity of various problems in default logic and in temporal logics.
In the first section of Chapter 3 we introduce a dynamic programming algorithm which decides whether a given default theory has a consistent stable extension in fpt-time and enumerates all generating defaults that lead to a stable extension with a pre-computation step that is linear in the input theory and triple exponential in the tree-width followed by a linear delay to output solutions.
In the second part of this chapter we lift the notion of backdoors to the field of default logics. We consider two problems, first we are interested to detect a backdoor and then to evaluate it for the target formulae classes HORN, KROM, POSITIVE-UNITand MONOTONE.
In Chapter 4, we investigate the parameterized complexity of problems in various temporal logics. In the first section we introduce several graph-like structures for formula representation and the corresponding notion of tree-width and path-width. To obtain the fixed parameter tractability of different fragments, we generalize the prominent Courcelle's Theorem to work for infinite signatures. In this section, we also consider Boolean operator fragments in the sense of Post's lattice.
In the second part of Chapter 4 we introduce the notion of backdoors for the globally fragment of linear temporal logic. Again, our problems of interest are to detect a backdoor and to evaluate it, this time, for the target formulae classes HORN and KROM.

Keywords: Parameterized complexity, default logic, temporal logic, backdoor, Post's Lattice, tree- and pathwidth, temporal depth

CHAPTER 1

INTRODUCTION

1.1 Parameterized Complexity

In classical complexity theory problems are investigated and evaluated with respect to the time and space needed to solve them. Doing so the structural properties are neglected and a problem may appears more difficult than it actually is. In contrast to classical complexity theory, in parameterized complexity we take a closer look at the structural properties and try to find an appropriate parametrization on them [FG06]. One of the main complexity classes in parameterized complexity is **FPT**, which denotes the *fixed parameter tractability*. On a high level of abstraction this means that all problems which do not have an algorithm with polynomial runtime or space respectively, finally obtain a polynomial behaviour by parameterizing the instance by some key property, such that all non-polynomial parts in the runtime or space depend on a parameter only.

Parameterized complexity theory has received much attention over the last two decades and has become increasingly important, as it has many applications in various fields of research such as automated verification, artificial intelligence, and computational biology to discover and to count the number of motifs in protein-protein interaction networks [AN08], to mention just a few.

Many general algorithmic techniques were developed for solving parameterized problems efficiently. Among these techniques are the following: The method of *bounded search trees* where the search space is bounded by a function of the parameter. *Kernelization* is a technique for the design of efficient algorithms. The idea of kernelization is to reduce the size of the input via certain rules and to obtain in this way a smaller instance, the so-called kernel. *Color coding* is a technique to design randomized fpt-algorithms.

Another approach to achieve a tractable algorithm is *dynamic programming* on the tree-decompositions. Here, the problem is solved in several small steps with bounded search space. Usually, this search space is restricted to the bags of a tree-decomposition.

Tree-decompositions are used in many algorithmic techniques in parameterized complexity. This approach was originally suggested by Rudolf Halin in 1976 [Hal76] and was subsequently developed further by Neil Robertson and Paul Seymor in 1984 [RN84], who are more prominent in the literature. The tree-decomposition of a graph or arbitrary relational structure is not unique. The concept of tree-decomposition came primarily from the context of graph minor theory [FG06] and offers a very strong parameter, the *treewidth*. This parameter has applications in many varied fields ([RN84],[KS93], [Sch94][FMR08]). The parameter treewidth measures the similarity of a relational structure to a tree: Smaller treewidth mans that the structure more closely resembles a tree. The treewidth of an acyclic graph is 1. A fundamental and important tool in this context is Courcelle's Theorem [CE12], which is applicable to problems definable in monadic second order (MSO) logic. It allows to solve MSO-definable problems that are known to be **NP**-hard, efficiently (all non polynomial parts depend on fixed parameter) [FG06].

1.2 Temporal Logics

Priors research laid a groundwork for temporal logics, as he published his book "Time and Modality" in 1957 [Pri57], where he introduced two modal operators "*sometime in the future*" F and "*sometime in the past*" P. It has received much attention and many remarkable publications followed. Kripke was concerned with the semantic clarification of modal logic [Kri63]. Pnueli took a step forward and formalized verification in linear temporal logic ($\mathcal{LTL}$) [Pnu77] and followed by computational tree logic ($\mathcal{CTL}$) almost equivalent formalism to computational tree logic was published by [AC81] and [AH85]. Nowadays, temporal logic has found an increasing number of application world-wide in the area of program verification and to express specifications of a program. Especially, the application of $\mathcal{CTL}$ enjoys a vast popularity since its model checking problem is polynomial time solvable, e.g. in artificial intelligence [CR14] or program verification [KP05]. By contrast, the satisfiability problem is **EXPTIME**-complete in $\mathcal{CTL}$ [FL79b] and even double-**EXPTIME**-complete for full branching time logic $\mathcal{CTL}^\star$ [VS85]. A possible approach to handle the intractable behaviour in temporal logics is the application of parametrization, viz. to examine restrictions of the problem with the aid of operators, Boolean connectives or bounded temporal depth.

1.3 Default Logic

To express a problem in propositional logic a complete description of the problem is required. Incomplete information cannot be expressed directly. In 1980, Raymond Reiter

[Rei80] reported a new and convenient procedure to make conjectures in the case of incomplete information possible. Default logic is tentative in nature. It follows the approach of *closed world assumption*, which states: "Everything that we do not know to be true is assumed to be false". Default logic is one of the most prominent approaches to non-monotonic reasoning where we can make tentative conclusions that can be retracted based on further evidence. Default logic contains an initial set of facts, the so-called *knowledge base*, and extends propositional logic by rules of default assumption, the so-called *default rules*. Roughly speaking, such default rule enunciate "*in the absence of contrary information, assume ...* ".
Default logic is a major area of interest within the fields of medical diagnosis [Kon01], legal reasoning[Pra93] and has a pivotal role in artificial intelligence [Rei87].

1.4 Publications

In the first section of Chapter 3, we establish a dynamic programming algorithm for default logic to decide whether a given default theory has a consistent stable extension (Ext) and to enumerate all sets of generating defaults (EnumSE) that lead to a stable extension. The algorithm runs in linear time in the input theory and triple exponential time in the treewidth to answer the (Ext) problem. To solve the (EnumSE) problem, the algorithm runs with a pre-computation step that is linear in the input theory and triple exponential in the treewidth followed by a linear delay to output the solutions. This section is based on [FHS17]. But in this thesis, we improved the worst-case runtime of the algorithm from fourfold exponential in the size of the input to triple exponential. This algorithm is not published yet.

In the second part of Chapter 3 we examine backdoors in default logic. The related problems are backdoor detection (Detect) as well as backdoor evaluation (Eval) for various kinds of target classes. Both problems are parameterized by the size of a backdoor set. We show that backdoor detection is fixed-parameter tractable for all considered target classes HORN, KROM, POSITIVE-UNIT, MONOTONE and backdoor evaluation is either fixed-parameter tractable, in **para-**Δ_2^{p} , or is **para-NP**-complete, depending on the target class. This section is based on [FMS16]. For the case of backdoor evaluation problem with target class HORN, only membership in **para-NP** was shown. As a new result, we show **para-NP**-hardness establishing **para-NP**-completness.

In Chapter 4, we consider the parameterized complexity of problems in temporal logics. In the first section, we examine various temporal logics, full branching time logic ($\mathcal{CTL}^\star$) and subsets of it, computation tree logic ($\mathcal{CTL}$) and linear temporal logic ($\mathcal{LTL}$). We will discuss several graph-like structures for formula representation and introduce the corresponding terms of treewidth and path-width. We also present a generalisation of Courcelle's Theorem for infinite signatures. Our classification shows a dichotomy between **W**[1]-hard and fixed-parameter tractable operator fragments. By exploring Boolean

operator fragments in the sense of Post's lattice we obtain the same complexity results as in classical computational complexity, if a set of available Boolean functions can express the function "negation of implication". In the reverse case we show the containment in **FPT** for almost all other clones. This section is based on [LMS17].

In the second section, we lift the notion of backdoors to the field of linear temporal logic for the globally fragment. Again, the problems of interest are backdoor detection (Detect) as well as backdoor evaluation (Eval). Here, we consider target classes HORN and KROM. We classify the operator fragments of globally-operators past, future and always and combinations of them. It turned out that the backdoor detection is fixed-parameter tractable, whereas backdoor evaluation is more challenging. For KROM formulae it is **para-NP**-complete and for HORN is it either in **FPT** or **para-NP**-complete, depending on the considered operator fragment. This section is based on [MOSS16].

CHAPTER 2

PRELIMINARIES

A *literal* l is a propositional variable or the negation of a variable and a *clause* c is a finite set of of literals. A *conjunctive normal form* (CNF) formula is a finite set of clauses of the form $\varphi = \bigwedge_i \bigvee_j (\neg) x_{ij}$, where x_{ij} is the j-th variable in the i-th clause. Additionally we want to define an truth-assignment $\vartheta \colon X \to \{0,1\}$ for a set of variables X. For $x \in X$ we set $\vartheta(\neg x) = 1 - \vartheta(x)$ and denote by $\mathbb{A}(X)$ the set of all truth-assignments $\vartheta \colon X \to \{0,1\}$. Furthermore, we will need from technical consideration the concept of the *truth assignment reduct* of a CNF formula φ with respect to $\vartheta \in \mathbb{A}(X)$, this is the CNF formula φ_ϑ obtained from φ by removing all clauses c that contains a literal l with $\vartheta(l) = 1$ first. Afterwards, we remove all occurrences $\vartheta(l) = 0$ from remaining clauses $c \in \varphi_\vartheta$. A formula φ is *satisfied* by ϑ if $\varphi_\vartheta = \varnothing$, and φ is *satisfiable* if it is satisfied by some ϑ. Let φ_1 and φ_2 be some CNF formulae and $X = \operatorname{var}(\varphi_1) \cup \operatorname{var}(\varphi_2)$. By $\varphi_1 \vDash \varphi_2$ we denote for all assignments $\vartheta \in \mathbb{A}(X)$ that satisfy φ_1 satisfy φ_2 also. We define the *deductive closure* of a formula φ as $Th(\varphi) := \{\varphi' \in \mathsf{CNF} \mid \varphi \vDash \varphi'\}$. In this thesis, we will consider several decision problems such as

Problem:	SAT
Input:	Propositional formula φ
Question:	Is the formula φ satisfiable?

Problem:	TAUT
Input:	Propositional formula φ
Question:	Is the formula φ satisfied by all possible assignments?

2.1 First Order and Second Order Logic

First Order Logic (FO) is applied for assigning certain properties for (not necessarily logical) variables by means of universal quantifier $\forall$ and existential quantifier $\exists$ and represents in this way a symbolized reasoning. The aim of this section is to get familiar with some basic terms in this field and to learn the syntax and afterwards the semantics of first and second order logic receptively. We will follow the notion of Immerman [Imm99].

Definition 2.1: Alphabet of FO

The *alphabet for first order logic* Σ_{FO} consists of

(1.) a set of variables $V_{\mathrm{FO}} = \{x_1, x_2, x_3, \dots x_n\}$
(2.) a set of constants $C_{\mathrm{FO}} = \{c_1, c_2, c_3, \dots c_m\}$
(3.) a set of functions $F_{\mathrm{FO}} = \{f_1^{a_1}, f_2^{a_2}, f_3^{a_3}, \dots f_\ell^{a_\ell}\}$, with the arity a_i of the function f_i,
(4.) a set relations $R_{\mathrm{FO}} = \{R_1^{a_1}, R_2^{a_2}, R_3^{a_3}, \dots R_k^{a_k}\}$, with the arity a_i of relation $R_i^{a_i}$,
(5.) a set of connectives $\{\neg, \wedge, \vee\}$
(6.) a set of quantifiers $\{\forall, \exists\}$
(7.) a set of brackets $\{(,)\}$

To formulate the syntax of FO we require the notion of *term:*

Definition 2.2: Term

We define a *set of terms* $\mathscr{T}_{\mathrm{FO}}$ inductively as follows:

(1.) $C_{\mathrm{FO}} \subseteq \mathscr{T}_{\mathrm{FO}}$
(2.) $V_{\mathrm{FO}} \subseteq \mathscr{T}_{\mathrm{FO}}$
(3.) if $f \in F_{\mathrm{FO}}$ and $t_1, \dots, t_n \in \mathscr{T}$, then $f(t_1, \dots, t_n) \in \mathscr{T}_{\mathrm{FO}}$, with $n = \mathrm{arity}(f)$

And no other strings are terms.

A syntax restricts the set of possible symbols for expressions in first-order logics.

Definition 2.3: Syntax of FO

We define a set of formulae of first order logic FO inductively as follows

(1.) $R^n \in \mathscr{R}_{\mathrm{FO}}$ and $t_1, \dots, t_n \in \mathscr{T}_{\mathrm{FO}}$, then $R(t_1, \dots, t_n) \in \mathrm{FO}$,
(2.) if $\varphi \in \mathrm{FO}$, then $(\neg\varphi) \in \mathrm{FO}$,
(3.) if $\varphi, \vartheta \in \mathrm{FO}$, then $\varphi \circ \vartheta \in \mathrm{FO}$ for each $\circ \in \{\wedge, \vee\}$ and finally

(4.) if $x \in V_{\mathrm{FO}}$ and $\varphi \in \mathrm{FO}$, then $(\forall\varphi(x)) \in \mathrm{FO}$ and $(\exists\varphi(x)) \in \mathrm{FO}$.

No other strings are elements of FO.

To interpret the symbols given in the syntax of FO, we need to explain the semantics, but first we require the following definition.

Definition 2.4: Vocabularies and Structures

By *vocabulary* we indentify a tuple of relation symbols R_i of arity a_i, constant symbols c_i, and function symbols f_i of arity k_i.

$$\tau := \langle R_1^{a_1}, \ldots, R_k^{a_k}, c_1, \ldots, c_s, f_1^{k_1}, \ldots, f_t^{k_t} \rangle$$

A *structure* with vocabulary τ is defined to be a tuple

$$\mathcal{A} := \langle |\mathcal{A}|, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_s^{\mathcal{A}}, f_1^{\mathcal{A}}, \ldots, f_t^{\mathcal{A}} \rangle.$$

We call the non empty set $|\mathcal{A}|$ a *universe*.

Vocabularies are collections of symbols, these are interpreted by the structure. Finally, we are able to construe the semantics of first order.

Definition 2.5: Semantics of FO, [FG06]

For each first-order formula $\varphi(x_1, \ldots, x_k)$ of vocabulary τ and each structure $\mathcal{A}$ we define a relation $\varphi(\mathcal{A}) \subseteq A^k$ inductively as follows:

- If $\varphi(x_1, \ldots, x_k) = R(x_{i_1}, \ldots, x_{i_r})$ with $i_1 \ldots, i_r \in [k]$, then

  $$\varphi(\mathcal{A}) := \{(a_1, \ldots, a_k) \in \mathcal{A}^k \mid (a_{i_1}, \ldots, a_{i_r}) \in R^{\mathcal{A}}\}.$$

  Equalities are treated similarly.

- If $\varphi(x_1, \ldots, x_k) = \psi(x_{i_1}, \ldots, x_{i_l}) \wedge \chi(x_{j_1}, \ldots, x_{j_m})$
  with $i_1, \ldots, i_l, j_1, \ldots, j_m \in [k]$ then

  $$\varphi(\mathcal{A}) := \{(a_1, \ldots, a_k) \in \mathcal{A}^k \mid (a_{i_1}, \ldots, a_{i_l}) \in \psi(\mathcal{A}) \text{ and } (a_{j_1}, \ldots, a_{j_m}) \in \chi(\mathcal{A})\}.$$

  The other connectives are treated similarly.

- If $\varphi(x_1, \ldots, x_k) = \exists x_{k+1}\psi(x_{i_1}, \ldots, x_{i_l})$ with $i_1, \ldots, i_l \in [k+1]$ then

  $$\varphi(\mathcal{A}) := \{(a_1, \ldots, a_k) \in \mathcal{A}^k \mid \text{ there exists an } a_{k+1} \in A \text{ s.t. } (a_{i_1}, \ldots, a_{i_l}) \in \psi(\mathcal{A})\}.$$

  Universal quantifiers are treated similarly.

Here we use the definition of [HDE95]. *Second order logic* (SO) is an extension of first order logic. It enables to quantify over subsets and relations of the universe $|\mathcal{A}|$ of the structure $\mathcal{A}$. Is augments the symbols of FO the alphabet containing for each $n \geqslant 1$, countably many n-ary *relation* (or *predicate*) variables $V_1^n, V_2^n, \ldots$. To denote relation variables we use big latin letters. A second order formula of vocabulary τ is the set generated by the rules for first-order formula extended by:

- If X is n-ary and $t_1, \ldots, t_n$ are terms then $Xt_1, \ldots, t_n$ is a formula.
- If φ is a formula and X is a relation variable then $\exists X\varphi$ is a formula.

The free occurrence of a variable or of a relation variable in a second-order formula is defined in the straightforward way and the notion of satisfaction is extended canonically. Then, given $\varphi = \varphi(x_1, \ldots, x_n, Y_1, \ldots, Y_k)$ with free (individual and relation) variables among $x_1, \ldots, x_n, Y_1, \ldots, Y_k$, a τ-structure $\mathcal{A}$, elements $a_1, \ldots, a_n \in A$ and relations $R_1, \ldots R_k$ over A of arity corresponding to $Y_1, \ldots, Y_k$, respectively,

$$\mathcal{A} \vDash \varphi[a_1, \ldots, a_n, R_1, \ldots, R_k]$$

means that $a_1, \ldots, a_n$ together with $R_1, \ldots, R_k$ satisfy φ in $\mathcal{A}$.

Monadic second order logic (MSO) is in some sense the intermediate level between FO and SO logic. Here, we are able to quantify over unary relations only. MSO is crucial for the application of Courcelle's Theorem. A variety of the graph problems can be expressed in monadic second order logic, like the classical problem of the three-colorability of a graph.

Example 2.1. Let a vocabulary $\tau = (V^G, E^G)$ be given, with V as the set of vertices and E as the set of edges of the instance graph G. Now, we want to describe the tree-colorability problem in terms of MSO

$$\begin{aligned}
\text{3-Color} := & \exists C_{\text{blue}} \exists C_{\text{red}} \exists C_{\text{green}} && (2.1)\\
& \forall v_1 \forall v_2 \Big(\big(C_{\text{blue}}(v_1) \vee C_{\text{red}}(v_1) \vee C_{\text{green}}(v_1)\big) && (2.2)\\
& \wedge \Big(\neg\big(C_{\text{blue}}(v_1) \wedge C_{\text{red}}(v_1)\big) \wedge \neg\big(C_{\text{blue}}(v_1) \wedge C_{\text{green}}(v_1)\big) && (2.3)\\
& \wedge \quad \neg\big(C_{\text{red}}(v_1) \wedge C_{\text{green}}(v_1)\big)\Big) && (2.4)\\
& \wedge E(v_1, v_2) \rightarrow \Big(\neg\big(C_{\text{blue}}(v_1) \wedge C_{\text{blue}}(v_2)\big) && (2.5)\\
& \wedge \neg\big(C_{\text{red}}(v_1) \wedge C_{\text{red}}(v_2)\big) \wedge \neg\big(C_{\text{green}}(v_1) \wedge C_{\text{green}}(v_2)\big)\Big)\Big) && (2.6)
\end{aligned}$$

In the first line (2.1) we demand tree varied colours, namely $C_{\text{blue}}, C_{\text{red}}, C_{\text{green}}$. In the following tree columns (2.2)–(2.4) we determine the vertex v_1 to be monochrome. And the two adjacent vertices v_1, v_2 do not have the same color (2.5)–(2.6).

2.2 Temporal Logics

Temporal logics deal with time-bounded statements, like “I am always freezing in winter”, “I am eventually freezing in autumn”, or “I will be freezing until I drink a hot beverage”. In this way, temporal logics enhance the common propositional logic by temporal operators next X, globally G, future F, until U and release R together with two path quantifiers exists E and all A. The syntax of temporal formula φ is defined through the following Backus Naur form:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid PT\varphi \mid P[\varphi \mathsf{U} \varphi] \mid P[\varphi \mathsf{R} \varphi],$$

where $T \in \{\mathsf{X}, \mathsf{G}, \mathsf{F}\}$ and $P \in \{\mathsf{E}, \mathsf{A}\}$.

Definition 2.6: Kripke Structure, [Kri63]

Let Φ denote a finite set of propositions. A *Kripke structure* is a triple

$$\mathcal{K} = (W, R, \varrho),$$

with a finite set of *worlds* W, a *successor relation* $R\colon W \to W$ and an *evaluation function* $\varrho\colon W \to 2^{\Phi}$ labelling sets of propositions to words.
A *path* π in a Kripke structure, is a infinite sequence of worlds $w_0, w_1, w_2, \ldots$, with $w_i R w_{i+1}$ for every $i \in \mathbb{N}$. Additionally, $\pi(i)$ represents the i-th world w_i in a path π and $\Pi(w)$ denotes the set of all paths beginning at world w.

Definition 2.7: Semantics for a Temporal Formula

Let $\mathcal{K} = (W, R, \varrho)$ a Kripke structure, a world $w \in W$, temporal formulae $\varphi, \varphi_1, \ldots, \varphi_n$, an path π, a proposition $p \in \Phi$ and a Boolean formula f be given. The truth value of a temporal formula according to the Kripke structure is defined as follows:

$$\begin{aligned}
&\mathcal{K}, w \vDash p && \Leftrightarrow && p \in \varrho(w),\\
&\mathcal{K}, w \vDash f(\varphi_1, \varphi_2, \ldots, \varphi_n) && \Leftrightarrow && \text{there is a propositional assignment } \vartheta \vDash f \text{ and}\\
& && && \text{for } 1 \leqslant i \leqslant n : \vartheta(i) = 1 \Leftrightarrow \mathcal{K}, w \vDash \varphi_i,\\
&\mathcal{K}, w \vDash \mathsf{A}\varphi && \Leftrightarrow && \text{for all } \pi \in \Pi(w) \text{ it holds } \mathcal{K}, \pi \vDash \varphi,\\
&\mathcal{K}, w \vDash \mathsf{E}\varphi && \Leftrightarrow && \text{there exists a } \pi \in \Pi(w) \text{ such that } K, \pi \vDash \varphi,\\
&\mathcal{K}, \pi \vDash p && \Leftrightarrow && p \in \varrho(\pi(0)),
\end{aligned}$$

$\mathcal{K}, \pi \vDash f(\varphi_1, \varphi_2, \ldots, \varphi_n)$	$\Leftrightarrow$	there is a propositional assignment $\vartheta \vDash f$ and for $1 \leqslant i \leqslant n : \vartheta(i) = 1 \Leftrightarrow \mathcal{K}, \pi \vDash \varphi_i$,
$\mathcal{K}, \pi \vDash \mathsf{A}\varphi$	$\Leftrightarrow$	$\mathcal{K}, \pi(0) \vDash \mathsf{A}\varphi$,
$\mathcal{K}, \pi \vDash \mathsf{E}\varphi$	$\Leftrightarrow$	$\mathcal{K}, \pi(0) \vDash \mathsf{E}\varphi$,
$\mathcal{K}, \pi \vDash \mathsf{X}\varphi$	$\Leftrightarrow$	$\mathcal{K}, \pi(1) \vDash \varphi$,
$\mathcal{K}, \pi \vDash \mathsf{F}\varphi$	$\Leftrightarrow$	there exists an $i \geqslant 0$ such that $\mathcal{K}, \pi(i) \vDash \varphi$,
$\mathcal{K}, \pi \vDash \mathsf{G}\varphi$	$\Leftrightarrow$	for all $i \geqslant 0 : \mathcal{K}, \pi(i) \vDash \varphi$,
$\mathcal{K}, \pi \vDash \varphi \mathsf{U} \psi$	$\Leftrightarrow$	$\exists\, i \geqslant 0 \,\forall\, j < i : \mathcal{K}, \pi(j) \vDash \varphi$ and $\mathcal{K}, \pi(i) \vDash \psi$,
$\mathcal{K}, \pi \vDash \varphi \mathsf{R} \psi$	$\Leftrightarrow$	$\forall\, i \geqslant 0 \,\exists\, j < i : \mathcal{K}, \pi(j) \vDash \varphi$ or $\mathcal{K}, \pi(i) \vDash \psi$.

Now, we lift the satisfiability problem to temporal logics and define:

Problem:	CTL-SAT
Input:	Temporal formula $\varphi \in \mathcal{CTL}$
Question:	Does there exist a Kripke structure $\mathcal{K}$, a sequence $w_i R w_{i+1}$ of worlds w_i with $i \in \mathbb{N}$, and a world w_n such that $\mathcal{K}, w_n \vDash \varphi$ holds?

An analogous problem definition is given for a given temporal formula $\varphi \in \mathcal{CTL}^\star$. Additionally we define:

Problem:	LTL-SAT
Input:	Temporal formula $\varphi \in \mathcal{LTL}$
Question:	Does there exist a Kripke structure $\mathcal{K}$, a sequence $w_i R w_{i+1}$ of worlds w_i with $i \in \mathbb{N}$, and a path π such that $\mathcal{K}, \pi \vDash \varphi$ holds?

For investigating the structural properties of temporal logics, we require the notion of *temporal depth*.

Definition 2.8: Temporal Depth

The *temporal depth* of a formula φ, denoted by $\mathrm{td}(\varphi)$ is inductively defined as follows:

$$
\begin{array}{llll}
\mathrm{td}(p) & := 0, & \mathrm{td}(f(\varphi_1, \ldots, \varphi_n)) & := \max\{\mathrm{td}(\varphi_1), \ldots, \mathrm{td}(\varphi_n), 0\}, \\
\mathrm{td}(\mathsf{P}\varphi) & := \mathrm{td}(\varphi), & \mathrm{td}(\varphi \mathsf{U} \psi) & := \max\{\mathrm{td}(\varphi), \mathrm{td}(\psi)\} + 1, \\
\mathrm{td}(\mathsf{T}\varphi) & := \mathrm{td}(\varphi) + 1, & \mathrm{td}(\varphi \mathsf{R} \psi) & := \max\{\mathrm{td}(\varphi), \mathrm{td}(\psi)\} + 1,
\end{array}
$$

where p is a propositional symbol, f is Boolean function, $\mathsf{P} \in \{\mathsf{A}, \mathsf{E}\}$ and $\mathsf{T} \in \{\mathsf{X}, \mathsf{F}, \mathsf{G}\}$.

2.3 Parameterized Complexity

Decision problems in classical complexity theory are expressed as languages over finite alphabets. In the next sections, we follow the notion of Flum and Grohe for the parameterized complexity theory and start with some central definitions [FG06].

Definition 2.9: Parametrization

Let Σ be a finite alphabet.

(1.) A *parametrization* of Σ^* is a polynomial time computable mapping $\kappa\colon \Sigma^* \to \mathbb{N}$.

(2.) A *parameterized problem* over an alphabet Σ is a tuple (Q, κ) consisting of a set of strings $Q \subseteq \Sigma^*$ and a parametrization κ of Σ^*.

Definition 2.10: κ-Bounded Function

Let κ be a parametrization. We say a function $f\colon \Sigma^* \to \Sigma^*$ is *κ-bounded*, if there is a computable function h, s.t. for all $x \in \Sigma^*$ holds $|f(x)| \leqslant h(\kappa(x))$.

Another key term in the context of parameterized complexity is *fixed parameter tractability*, roughly speaking it represents a complexity class with problems, which are computable in polynomial time and all non-polynomial parts depend on a parameter.

Definition 2.11: FPT

Let Σ be a finite alphabet and $\kappa\colon \Sigma^* \to \mathbb{N}$ be a parametrization.

(1.) An Algorithm $\mathscr{A}$ with a instance alphabet Σ is an *fpt-Algorithm with respect to* κ if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial $p \in \mathbb{N}_{\geqslant 0}[X]$ such that for every $x \in \Sigma^*$, $\mathscr{A}$ on input x runs in

$$f(\kappa(x)) \cdot p(|x|)$$

worst-case time.

(2.) A parameterized problem (Q, κ) is *fixed-parameter tractable* if there is an fpt-Algorithm with respect to κ that decides Q.

(3.) **FPT** denotes the class of all fixed-parameter tractable problems.

A further helpful tool for proving problems to be fixed-parameter tractable is the notion of a *slice*. The concept of slice is crucial for the generalization of Courcelles Theorem for infinite signutures, what we will see in Chapter 4.

Definition 2.12: Slice, [FG06]

Let (Q, κ) be a parameterized problem and $\ell \in \mathbb{N}$. The ℓth *slice* of (Q, κ) is the classical problem

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

Theorem 2.2 ([FG06])**.**
Let $\mathcal{C}$ be a complexity class in $\{\mathbf{NP}, \mathbf{PSPACE}, \mathbf{EXPTIME}\}$. Let (Q, κ) be a parameterized problem, $Q \subsetneq \Sigma^*$, $Q \neq \varnothing$. Then (Q, κ) is para-$\mathcal{C}$-hard if and only if a union of finitely many slices of (Q, κ) is $\mathcal{C}$-hard.

2.3.1 The W-Hierarchy

Parameterized complexity theory pertains to classical computational complexity theory. The complexity of problems is determined by an effort and subsequently classified to the corresponding complexity classes. We assume the familiarity with the concept of multitape Turing machines.

Definition 2.13: W[P]

Let Σ be an alphabet and $\kappa\colon \Sigma^* \to \mathbb{N}$ be a parametrization.

(1.) A nondeterministic Turing machine $\mathscr{M}$ with the input Σ is defined as κ-*restricted* if there exist computable functions $f_1\colon \mathbb{N} \to \mathbb{N}$, $f_2\colon \mathbb{N} \to \mathbb{N}$ and a polynomial $p \in \mathbb{N}_{\geqslant 0}[X]$ such that on every run with the input $x \in \Sigma^*$ the machine $\mathscr{M}$ performs at most $f_1(\kappa(x)) \cdot p(|x|)$ steps, and at most $f_2(\kappa(x)) \cdot \log(|x|)$ of them being nondeterministic.

(2.) $\mathbf{W}[\mathrm{P}]$ is the class of all parameterized problems (Q, κ) that can be decided by a κ-restricted nondeterministic Turing machine $\mathscr{M}$.

Definition 2.14: para-NP

A parameterized problem (Q, κ) over alphabet Σ is in *para*-**NP**, if there is a computable function $f : \mathbb{N} \to \mathbb{N}$, a polynomial $p \in \mathbb{N}_{\geqslant 0}[X]$, and a not deterministic algorithm that, given $x \in \Sigma^*$, decides if $x \in Q$ in at most $f(\kappa(x)) \cdot p(|x|)$ steps.

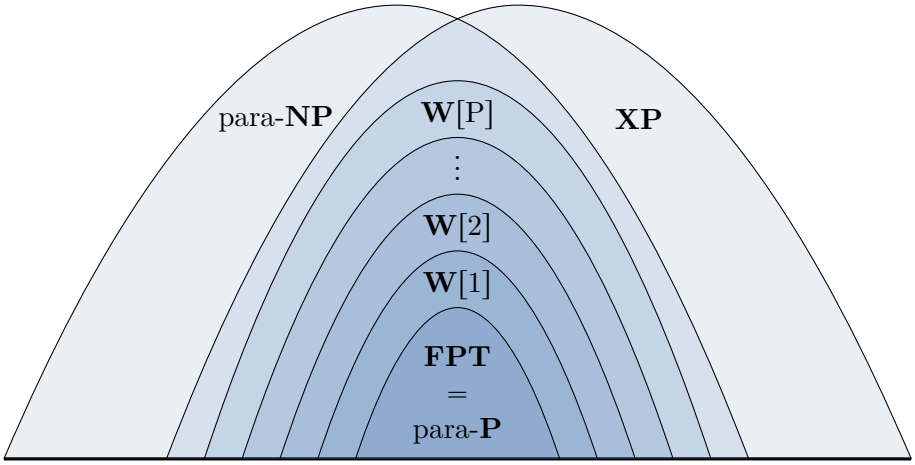


Figure 2.1: Parameterized complexity classes

Definition 2.15: XP

A parameterized problem (Q, κ) over alphabet Σ is in **XP**, if there is a computable function $f : \mathbb{N} \to \mathbb{N}$, and an algorithm that, given $x \in \Sigma^*$, decides if $x \in Q$ in at most $|x|^{f(\kappa(x))} + f(\kappa(x))$ steps.

Proposition 2.3 (The Relation among the Classes)**.** [FG06]

$$\mathbf{FPT} \subseteq \mathbf{W[P]} \subseteq \mathbf{XP} \cap \text{para-}\mathbf{NP}.$$

2.3.2 Tree-Decomposition

Tree-decomposition provide one of the most significant parameter *tree-width*, which attaches great importance in the theoretical considerations. Such as in Theorem of Courcelle. With an appropriate parameter we are able to solve efficient, some in computational complexity theory as **NP**-hard classified problems.

Definition 2.16: Tree-Decomposition

A *tree-decomposition* of a graph $G = (V, E)$ is a pair $\mathcal{T} = (T, \chi)$, with a rooted tree $T = (N, F, n)$ and a function $\chi : N \to 2^V$ that maps to each node $t \in T$ a set of vertices such that:

(1.) For every vertex $v \in V$ there is a node $t \in N$ with $v \in \chi(t)$.

(2.) For every edge $e \in E$ there is a node $t \in N$ with $e \subseteq \chi(t)$.

(3.) For any three nodes $t_1, t_2, t_3 \in N$ holds, if t_2 lies on the unique path from t_1 to t_3, then $\chi(t_1) \cap \chi(t_3) \subseteq \chi(x_2)$.

The *width* of the decomposition $\mathcal{T} = (T, \chi)$ is the number

$$\max\{|\chi(t)| - 1 \mid t \in T\}.$$

The *tree-width* $\mathrm{tw}(G)$ of the graph G is the minimum width over all possible tree-decompositions of G.

The decision problem of a graph G, that needs to decide whether G has a given tree-width $k \in \mathbb{N}$ is **NP**-complete. Bodlaender and Koster have shown that for arbitrary but fixed k a tree-decomposition of a Graph G of a width that equals its tree-width is computable in time $2^{\mathcal{O}(\kappa^3)} \cdot |V|$ [BK08].

Based on Bodlaender's Theorem we know that the parameterized decision problem Tree-Width is fixed-parameter tractable.

Problem:	Tree-Width
Input:	A graph G
Parameter:	$\kappa \in \mathbb{N}$
Question:	Decide whether $\mathrm{tw}(G) = \kappa$.

2.3.3 The Theorem of Courcelle

Founded on the Theorem of Bodlaender, Courcelle has published in 1990 a theorem [Cou90], which has found increasing application and is one of the central and one of the most cited theorems in the field of parameterized complexity theory. In the following we directly deal with the extended version of Courcelle's Theorem, which was introduce by M. Elberfeld, A. Jacoby and T. Tantau [EJT10], where they investigated additionally the space requirements.

Theorem 2.4 (Courcelle 1990; Elberfeld, Jakoby and Tantau 2010)**.**
Let (Q, κ) be a MSO-definable parameterized problem, and $\mathcal{A}_x$ be a structure associated with an instance x. Further let the treewidth of the structure $\mathcal{A}_x$ be bounded by the parameter $\kappa \in \mathbb{N}$, $\mathrm{tw}(\mathcal{A}_x) = \kappa$. Then (Q, κ) is solvable in time

$$\mathcal{O}\big(f(\kappa) \cdot |x|\big)$$

and in space

$$\mathcal{O}(\log(f(\kappa)) + \log|x|).$$

2.4 Post's Lattice

Emil Post investigated arbitrary restrictions of Boolean connectives in a formula and published in 1941 a lattice demonstrating the inclusion relations among all existing sets of Boolean functions [Pos41], so-called *Post's lattice*, which is illustrated in Figure 2.2. His work yielded a number of promising new avenues of research in various types of logics, like non-monotonic logics [BMTV10, CMVT12], temporal logics [MMTV09, BMM$^+$11], modal logic [HSS10], hybrid logics [MMS$^+$10], circuits [BCG$^+$12], description logics [MS13], or constraints [BBC$^+$10]. But the first step was done by Lewis in 1979 [Lew79], he proved the **NP**-completeness for $\textsc{Sat}(\mathsf{B})$ iff $\twoheadrightarrow$ is an element of the smallest clone of the base B which is defined as follows :

Definition 2.17: Clone and Base

Let B be a finite set of Boolean functions. A superset $\mathsf{B}' \supseteq \mathsf{B}$ containing all projects and is closed under arbitrary compositions of functions from B' is called *clone*.
We denote by $[\mathsf{B}]$ the smallest clone of the set B and we call the set B its *base*.

We summarize all Boolean clones in Table 2.1, where a Boolean function is:

- *c-reproducing* if $f(c, \ldots, c) = c$,
- *monotone* if for $a_1 \leqslant b_1, \ldots, a_n \leqslant b_n$ follows $f(a_1, \ldots, a_n) \leqslant f(b_1, \ldots, b_n)$,
- *c-seperating* if there is an $i \in \{1, \ldots, n\}$ with $f(a_1, \ldots, a_n) = 1$ implies $a_i = c$,
- *c-separating of degree n* if all $A \subseteq f^{-1}(c)$ with $|A| = n$ are c-separating,
- *self-dual* if $f \equiv \text{dual}(f)$, with $\text{dual}(f)(x_1, \ldots, x_n) = \neg f(\neg x_1, \ldots, \neg x_n)$, an finally
- *linear* or *affine* if $f(x_1, \ldots, x_n) \equiv x_1 \oplus \cdots \oplus x_n \oplus c$.

Further we have we have a identity function id, defined as $f(x) = x$ and a threshold function $T_n^{n+1} := \bigvee_{i=0}^{n}(x_0 \wedge \cdots \wedge x_{i-1} \wedge x_{i+1} \cdots \wedge x_n)$, where n of $n+1$ bits are necessarily set to "*true*".

Formula Class	Definition	Base
BF	All Boolean functions	$\{x \wedge y, \neg x\}$
R_0	$\{f \mid f$ is $\bot$-reproducing$\}$	$\{x \wedge y, x \oplus y\}$
R_1	$\{f \mid f$ is $\top$-reproducing$\}$	$\{x \vee y, x \leftrightarrow y\}$
R_2	$\mathsf{R}_0 \cap \mathsf{R}_1$	$\{\vee, x \wedge (y \leftrightarrow z)\}$
M	$\{f \mid f$ is monotone$\}$	$\{x \vee y, x \wedge y, \bot, \top\}$
M_0	$\mathsf{M} \cap \mathsf{R}_0$	$\{x \vee y, x \wedge y, \bot\}$
M_1	$\mathsf{M} \cap \mathsf{R}_1$	$\{x \vee y, x \wedge y, \top\}$
M_2	$\mathsf{M} \cap \mathsf{R}_2$	$\{x \vee y, x \wedge y\}$
S_0	$\{f \mid f$ is $\bot$-separating$\}$	$\{x \rightarrow y\}$
S_1	$\{f \mid f$ is $\top$-separating$\}$	$\{x \nrightarrow y\}$
S_0^n	$\{f \mid f$ is $\bot$-separating of degree $n\}$	$\{x \rightarrow y, \mathrm{dual}(T_n^{n+1})\}$
S_1^n	$\{f \mid f$ is $\top$-separating of degree $n\}$	$\{x \nrightarrow y, T_n^{n+1}\}$
S_{00}	$\mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z)\}$
S_{00}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \mathrm{dual}(T_n^{n+1})\}$
S_{01}	$\mathsf{S}_0 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \top\}$
S_{01}^n	$\mathsf{S}_0^n \cap \mathsf{M}$	$\{\mathrm{dual}(T_n^{n+1}), \top\}$
S_{02}	$\mathsf{S}_0 \cap \mathsf{R}_2$	$\{x \vee (y \nrightarrow z)\}$
S_{02}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2$	$\{x \vee (y \nrightarrow z), \mathrm{dual}(T_n^{n+1})\}$
S_{10}	$\mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z)\}$
S_{10}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z), T_n^{n+1}\}$
S_{11}	$\mathsf{S}_1 \cap \mathsf{M}$	$\{x \wedge (y \vee z), \bot\}$
S_{11}^n	$\mathsf{S}_1^n \cap \mathsf{M}$	$\{T_n^{n+1}, \bot\}$
S_{12}	$\mathsf{S}_1 \cap \mathsf{R}_2$	$\{x \wedge (y \rightarrow z)\}$
S_{12}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2$	$\{x \wedge (y \rightarrow z), T_n^{n+1}\}$
D	$\{f \mid f$ is self-dual$\}$	$\{(x \nrightarrow y) \vee (x \nrightarrow z) \vee (\overline{y} \nrightarrow z)\}$
D_1	$\mathsf{D} \cap \mathsf{R}_2$	$\{(x \nrightarrow \overline{y}) \vee (x \nrightarrow z) \vee (y \nrightarrow z)\}$
D_2	$\mathsf{D} \cap \mathsf{M}$	$\{(x \nrightarrow \overline{y}) \vee (x \nrightarrow \overline{z}) \vee (y \nrightarrow \overline{z})\}$
L	$\{f \mid f$ is linear$\}$	$\{x \oplus y, \top\}$
L_0	$\mathsf{L} \cap \mathsf{R}_0$	$\{x \oplus y\}$
L_1	$\mathsf{L} \cap \mathsf{R}_1$	$\{x \leftrightarrow y\}$
L_2	$\mathsf{L} \cap \mathsf{R}_2$	$\{x \oplus y \oplus z\}$
L_3	$\mathsf{L} \cap \mathsf{D}$	$\{x \oplus y \oplus z \oplus \top\}$
V	$\{f \mid f$ is a disjunction or constant$\}$	$\{x \vee y, \bot, \top\}$
V_0	$\mathsf{M}_0 \cap \mathsf{V}$	$\{x \vee y, \bot\}$
V_1	$\mathsf{M}_1 \cap \mathsf{V}$	$\{x \vee y, \top\}$
V_2	$\mathsf{M}_2 \cap \mathsf{V}$	$\{x \vee y\}$
E	$\{f \mid f$ is a conjunction or constant$\}$	$\{x \wedge y, \bot, \top\}$
E_0	$\mathsf{M}_0 \cap \mathsf{E}$	$\{x \wedge y, \bot\}$
E_1	$\mathsf{M}_1 \cap \mathsf{E}$	$\{x \wedge y, \top\}$
E_2	$\mathsf{M}_2 \cap \mathsf{E}$	$\{x \wedge y\}$
N	$\{f \mid f$ depends on at most one variable$\}$	$\{\neg x, \bot, \top\}$
N_2	$\mathsf{L}_3 \cap \mathsf{N}$	$\{\neg x\}$
I	$\{f \mid f$ is a projection or a constant$\}$	$\{\mathrm{id}, \bot, \top\}$
I_0	$\mathsf{R}_0 \cap \mathsf{I}$	$\{\mathrm{id}, \bot\}$
I_1	$\mathsf{R}_1 \cap \mathsf{I}$	$\{\mathrm{id}, \top\}$
I_2	$\mathsf{R}_2 \cap \mathsf{I}$	$\{\mathrm{id}\}$

Table 2.1: A list of all Boolean clones with definitions and bases.

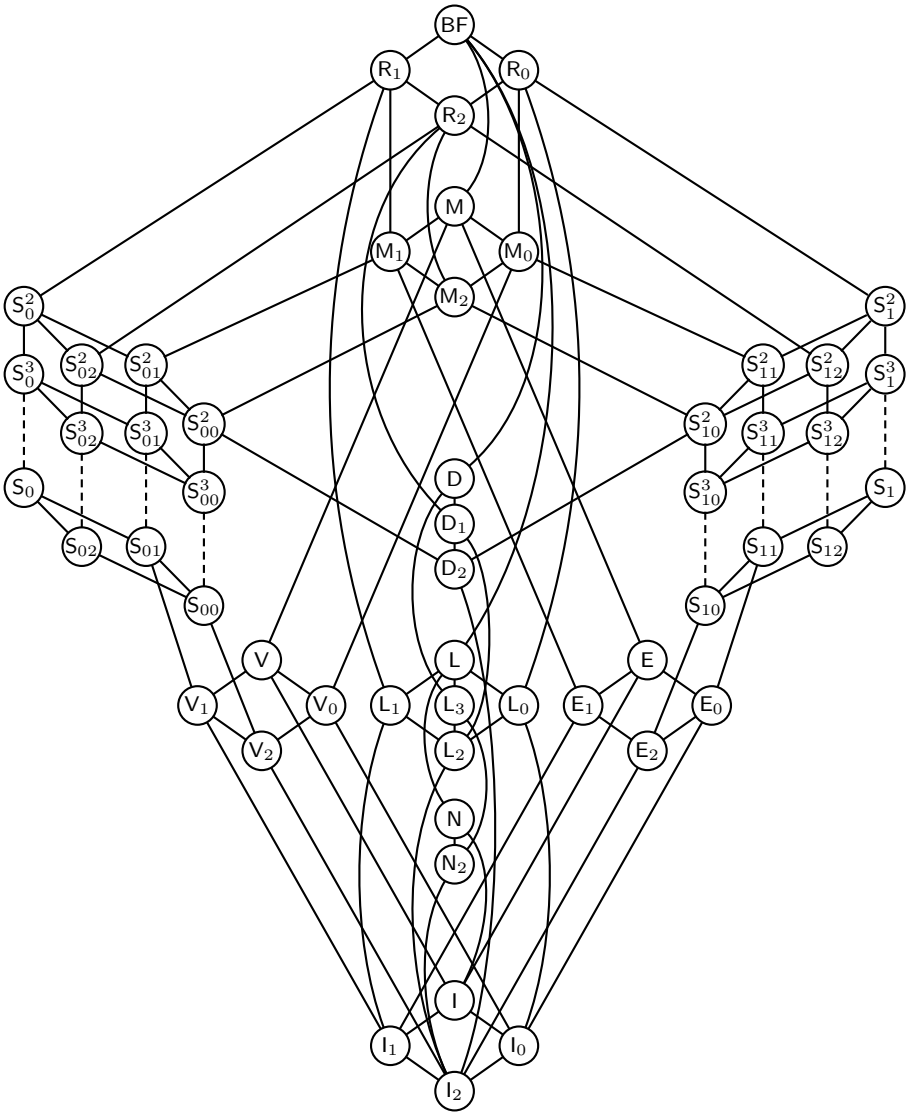


Figure 2.2: Post's lattice.

2.5 Default Logic

In this thesis we follow the notion for default logic (DL) of Reiter [Rei80]. A *default rule* δ is a triple $\frac{P:J}{C}$, where a formula P is so-called *prerequisite*, a formula J is the *justification* and a formula C is the *conclusion* of a default rule δ. A *default theory* $T_{\mathsf{DL}} = \langle W, D \rangle$ is a tuple containing the knowledge base W, which is a set of propositional CNF formulae and D a set of default rules. The aim in DL is to find stable extension:

Problem:	Stable Extension Existence Ext
Input:	Default theory $T_{\mathsf{DL}} = \langle W, D \rangle$
Question:	Does a stable extension for T_{DL} exist?

A fundamental principal for investigation of the existence of stable extension Reiter [Rei80] has formulated as follows:

Definition 2.18: Fixed Point Semantics

Let $T_{\mathsf{DL}} = \langle W, D \rangle$ be a default theory and E be a set of formulae. Then $\Gamma(E)$ is the smallest set of formulae such that:

(1.) $W \subseteq \Gamma(E)$,

(2.) $\Gamma(E) = Th(\Gamma(E))$,

(3.) for each $\frac{P:J}{C} \in D$ with $P \in \Gamma(E)$ and $\neg J \notin E$, we have $C \in \Gamma(E)$.

E is a stable extension of $T_{\mathsf{DL}} = \langle W, D \rangle$, if $E = \Gamma(E)$. An extension is inconsistent if it contains $\bot$, otherwise it is called consistent.

The stage construction is an alternativ way to determine the existence of stable extension, which was also formulated by Reiter.

Lemma 2.5 (Stage Construction, [Rei80])**.** Let $T_{\mathsf{DL}} = \langle W, D \rangle$ be a default theory and E be a set of formulae. Then we define $E_0 := W$ and

$$E_{i+1} := Th(E_i) \cup \left\{ C \;\middle|\; \frac{P:J}{C} \in D, P \in E_i \text{ and } \neg J \notin E \right\}.$$

E is a stable extension of $T_{\mathsf{DL}} = \langle W, D \rangle$ if and only if

$$E = \bigcup_{i \in \mathbb{N}} E_i.$$

In addition we define the set of generating defaults

$$G = \left\{ \frac{P:J}{C} \in D \;\middle|\; P \in E \text{ and } \neg J \notin E \right\}$$

If E is a stable extension of $T_{\mathsf{DL}} = \langle W, D \rangle$, then $E = Th(W \cup C(\delta) \mid \delta \in G)$.

Example 2.6. Stable Extension I
Let a default theory be given with a knowledge base $W_1 := \{x_1\}$ and a set of default rules $D_1 := \{\delta_1 = \frac{x_1 : x_2}{x_2}, \delta_2 = \frac{x_1 : \neg x_2}{x_1 \wedge x_4}, \}$. Now we need to examine the existence of a stable extension. In δ_1 prerequisite is fulfilled with $Th(W)$ and the justification is not violated, as we cannot infer $\neg x_2$ from $Th(W)$. Consequently, we conclude x_2 and obtain $E_1 = Th(W \cup x_2)$. In the next step, we examine the default rule δ_2. Here, the prerequisite is fulfilled again but we can infer x_2 from E_1, in consequence the justification is violated. Overall holds W_1 does not have any stable extension.

Example 2.7. Stable Extension II
Now let $T_{\mathsf{DL}} = \langle W, D \rangle$ be given as $W = \varnothing$ and $D := \{\delta_1 = \frac{\top : x_2}{x_1}, \delta_2 = \frac{\top : \neg x_1}{\neg x_2}\}$, this default theory has two stable extensions $E_1 = Th(x_1)$ and $E_2 = Th(\neg x_2)$.

We call SE a set of all stable extensions of a default theory $T_{\mathsf{DL}} = \langle W, D \rangle$.

CHAPTER 3

PARAMETRIZATION IN DEFAULT LOGIC

In this chapter we will be concerned with parametrizations in default logic. It is divided into two sections. In the first part we investigate the existence of stable extensions via a dynamic programming algorithm. In the second part we proceed with the study of so-called backdoors in default logic.

3.1 Default Logic and Bounded Treewidth

In this section, we present a dynamic programming algorithm which examines the existence of a stable extension in Reiter's propositional default logic. It runs in linear time in the input theory and triple exponential time in the treewidth. Additionally we enumerate all generating defaults with a pre-computation step that needs linear time in the input theory and triple exponential time in the treewidth followed by a linear delay to output the solutions. We represent default theories by their semi-primal graph with bounded treewidth of its tree-decomposition. Let $\alpha(\delta)$ denote the prerequisite P of the default rule δ, analogously let $\beta(\delta)$ denote the justification J and $\gamma(\delta)$ denote the conclusion C of the default rule δ. For a default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ with consistent knowledge base W, we are able to transform every formula in W into a default rule. This can be done in linear time by utilising so-called *normal* default rules. For all $w_i \in W$ we insert a rule $\frac{\top : w_i}{w_i}$, where $i \in \mathbb{N}$ and $1 \leqslant i \leqslant |W|$. Finally we get $T'_{\mathsf{DL}} = \langle \varnothing, D \cup \{\frac{\top : w_1}{w_1}; \cdots ; \frac{\top : w_{|W|}}{w_{|W|}}\} \rangle$. In the following, we assume that any default theory has an empty knowledge base, unless stated otherwise. Such a default theory $T_{\mathsf{DL}} = \langle \varnothing, D \rangle$ is also denoted simply by D.

For the examination of the existence of a stable extension we formulate an alternative characterisation of stable extension beyond fixed point semantics, which is inspired by Reiter's stage construction [Rei80].

Definition 3.1

Let D be a default theory and $S \subseteq D$. Further, let $E(S) := \{\gamma(\delta) \mid \delta \in S\}$. We call a default $\delta \in D$

- *p-satisfiable* in S, if $E(S) \cup \neg\alpha(\delta)$ is satisfiable
- *j-satisfiable* in S, if $E(S) \cup \beta(\delta)$ is unsatisfiable
- *c-satisfiable* in S, if $\delta \in S$.

The set S is a *satisfying default set*, if each default $\delta \in D$ is p-satisfiable in S, or j-satisfiable in S, or c-satisfiable in S.
The set S is a *stable default set*, if

(1.) S is a satisfying default set and

(2.) there is no S' where $S' \subsetneq S$ such that for each default δ it holds that δ is p-satisfiable in S', or j-satisfiable in S, or c-satisfiable in S'

We refer to the set of all stable default sets of D by $\mathrm{SD}(D)$.

In the next lemma we show that we can use stable default sets to obtain stable extensions of a default theory.

Lemma 3.1.
Let D be a default theory. Then,

$$\mathrm{SE}(D) = \bigcup_{S \in \mathrm{SD}(D)} Th(\{\gamma(\delta) \mid \delta \in S\}).$$

In particular, $S \in \mathrm{SD}(D)$ is the set of generating defaults of extension $Th(\{\gamma(\delta) \mid \delta \in S\})$.

Proof. Let a default theory D be given. For the forward direction let $E \in \mathrm{SE}(D)$ and observe that $E = Th(E)$. Next we construct a set $S := \{\delta \in D \mid \gamma(\delta) \in E, \alpha(\delta) \in E, \neg\beta(\delta) \notin E\}$ from E. For a sake of contradiction assume that S is not a stable default set. Consequently, S is either not subset-minimal, or it dissatisfies at least one default, which immediately leads to a contradiction since E is a stable extension. If S is not subset-minimal, there is a smaller set $S' \subsetneq S$, which is a satisfying default set. Observe that there is at least one $\delta \in S \backslash S'$ where $\gamma(\delta) \notin Th(\{\gamma(\delta') \mid \delta' \in S'\})$, since otherwise S' can not be a satisfying default set due to $Th(\{\gamma(\delta) \mid \delta \in S\}) = Th(\{\gamma(\delta') \mid \delta' \in S'\})$ and $S' \subsetneq S$, which results in at least one default in $S \backslash S'$ that is dissatisfied by construction of S, c.f. Definition 3.1(1.). As a result, there is a smaller extension $E' \subsetneq E$, where $E' := Th(\{\gamma(\delta') \mid \delta' \in S'\})$, which contradicts, once again, that E is a stable extension.
For the reverse direction let S be any stable default set. We define $E := Th(\{\gamma(\delta) \mid \delta \in S\})$. Assume towards a contradiction that E is not stable. Obviously, by construction of E, $\Gamma(E) := E$ satisfies the requirements of stable extension. It remains to show, that there

is no smaller $\Gamma'(E) \subsetneq \Gamma(E)$ which also satisfies the three conditions of stable extension. Assume towards a contradiction, that such a set $\Gamma'(E)$ with $\Gamma'(E) = Th(\Gamma'(E))$ exists. Then there is at least one default $\delta \in D$, such that $\gamma(\delta) \in \Gamma(E) \backslash \Gamma'(E)$. As a result, by construction of E, S can not be a stable default set, which contradicts the assumption. □

For our dynamic programming algorithm we need a slightly modified form of tree-decomposition. The so-called *nice tree-decompositions* have convenient properties that are useful for the bottom-up approach of our algorithm.

Definition 3.2: Nice Tree-Decomposition

Let a tree-decomposition $\mathcal{T} = (T, \chi)$ be given with a tree $T = (N, \cdot, \cdot)$. Then for a node $t \in N$ its type type(t) is specified as:

$$
\begin{aligned}
\textit{leaf}&: \text{if } t \text{ has no children} \\
\textit{join}&: \text{if } t \text{ has two children nodes } t_1 \text{ and } t_2, \\
&\quad \text{where } t_1 \neq t_2 \text{ and } \chi(t) = \chi(t_1) = \chi(t_2) \\
\textit{int}&: \text{if } t \text{ has a single child } t_1 \text{ with } \chi(t_1) \subseteq \chi(t) \text{ and } |\chi(t)| = |\chi(t_1)| + 1 \\
\textit{rem}&: \text{if } t \text{ has a single child } t_1 \text{ with } \chi(t) \subseteq \chi(t_1) \text{ and } |\chi(t_1)| = |\chi(t)| + 1
\end{aligned}
$$

If every node $t \in N$ has at most two children, is of a type(t) $\in \{\textit{leaf}, \textit{int}, \textit{rem}, \textit{join}\}$ and the bags χ of the leaf nodes as well as the root are empty, then we call $\mathcal{T} = (T, \chi)$ *nice*.

Note that we are able to compute from a given tree-decomposition a *nice* tree-decomposition in linear time, without increasing the width.

3.1.1 Graph Representations of Default Theories

For the investigation of a given default theory via a dynamic programming algorithm, we require an appropriate graph representation.

Definition 3.3: Semi-Primal Graph of a Default Theory

For a default theory D the *semi-primal graph* $S(D)$ is a graph, where the vertices are variables from $\mathsf{Var}(D)$ and defaults δ of D.
The edges $\mathcal{E}$ of $S(D)$ are

- (a, δ) if variable $a \in \mathsf{Var}(\delta)$, for each $\delta \in D$
- (a_1, a_2) if either $a_1, a_2 \in \mathsf{Var}(\alpha(\delta))$, or $a_1, a_2 \in \mathsf{Var}(\beta(\delta))$, or $a_1, a_2 \in \mathsf{Var}(\gamma(\delta))$.

Note that the formulae $\alpha(\delta), \beta(\delta)$ or $\gamma(\delta)$ for a default δ may be $\top$ or $\bot$. This can be transformed to $x \vee \neg x$ or $x \wedge \neg x$ for a fresh variable x.
In our dynamic programming algorithm for default logic we need to remember when we can evaluate a formula (prerequisite, justification, or conclusion) for a default, i.e., we have a default and all the variables of the formula in a bag. To that end, we introduce labels of nodes. Since we work along the tree-decomposition and want a unique point where to evaluate, we restrict a label to the first occurrence of a bag containing all appropriate parts of the default when working along the tree-decomposition TD.

Definition 3.4: Labeled Tree-Decomposition

A *labeled tree-decomposition (LTD)* $\mathcal{T}$ of a default theory D is a tuple $\mathcal{T} = (T, \chi, \omega)$ where (T, χ) is a TD of $S(D)$ and $\omega : N \to 2^{(\{\alpha,\beta,\gamma\} \times D)}$ is a mapping where for any (f, δ) in $\{\alpha, \beta, \gamma\} \times D$ it holds that

(1.) if $(f, \delta) \in \omega(t)$, then $\{\delta\} \cup f(\delta) \subseteq \chi(t)$ and

(2.) if $\{\delta\} \cup f(\delta) \subseteq \chi(t)$ and there is no descendent t' of t such that $(f, \delta) \in \omega(t')$, then $(f, \delta) \in \omega(t)$.

For our purpose we require the properties of both nice and labeled TDs and combine these in the following definition.

Definition 3.5: Pretty LTD

For a node $t \in N$ that has exactly one child t' where $\chi(t) = \chi(t')$ and $\omega(t) \neq \varnothing$, we say that type$(t)$ is *label*.
Further, we call the LTD *pretty* if:

(1.) every node $t \in N$ has at most two children

(2.) type$(t) \in \{$*leaf*, *join*, *int*, *label*, *rem*$\}$ for all nodes $t \in N$

(3.) bags of leaf nodes and the root are empty

(4.) and $\omega(t) = \varnothing$ for type$(t) \neq$ *label*.

Observe that we are able to construct a pretty LTD in linear time, without increasing the width from a nice TD. This can be done by traversing the TD and constructing the labels and duplicating nodes t where $\omega(t) \neq \varnothing$. Unless mentioned otherwise, in the following we will use pretty LTDs.

Example 3.2. Let the default theory D be given as

$$D = \left\langle \left\{ \delta_1 = \frac{\top : \neg x_2}{x_1}, \delta_2 = \frac{\top : x_2}{x_1 \vee x_2} \right\} \right\rangle$$

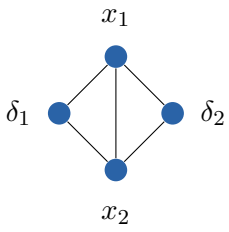


Figure 3.1: Semi-Primal Graph $S(D)$ of Example 3.2

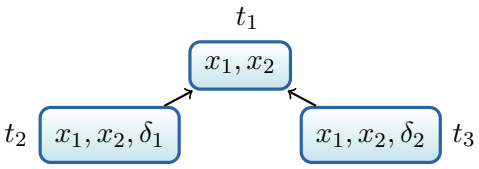


Figure 3.2: tree-decomposition of a Graph $S(D)$ of Example 3.2

Figure 3.1 exemplifies a semi-primal graph of this default theory with its TD.

3.1.2 The Idea of Dynamic Programming for DL

We now sketch the methodology of our DP algorithm on tree-decompositions. The fundament of our algorithm is shown in Algorithm 1. The algorithm $\mathcal{DP}$ traverses the

Algorithm 1: Dynamic Programming Algorithm $\mathcal{DP}(\mathcal{T})$ for DL on TD $\mathcal{T}$, cf. [FHMW17].

In: Pretty LTD $\mathcal{T} = (T, \chi, \omega)$ $T = (N, \cdot, n)$ of the semi-primal graph $S(D)$,
Out: A table for each node $t \in T$ stored in a mapping Tables[t]

```
1 for iterate t in post-order(T,n) do
2     Child-Tabs := {Tables[t'] | t' is a child of t in T}
      Tables[t] ← SPRIM(t, χ(t), ω(t), D_t, Child-Tabs)
3 return Tables[·];
```

given LTD $\mathcal{T} = (T, \chi, \omega)$ in post-order and runs at each node $t \in T$ the algorithm SPRIM. The core algorithm SPRIM computes a new table τ_t according to the tables of the children of t. The evaluation of SPRIM is restricted to the *bag-defaults*, i.e. $D_t := D \cap \chi(t)$, and exhibits only a " local view". Roughly speaking, we garner in each table τ_t information that is required for the local decision. For this, only information concerning variables that belong to bag $\chi(t)$. Additionally, we require the notion of *default theory below* t, which is denoted by $D_{\leqslant t} := \{\delta \mid \delta \in D_{t'}, t' \in \text{post-order}(T, t)\}$ and *default theory strictly below* t denoted by $D_{<t} := D_{\leqslant t} \backslash D_t$. For the root n of T, it holds $D_{\leqslant n} = D_{<n} = D$.
Before we talk about SPRIM in detail we have to talk about a notion for sequences of results of a computation. The algorithm SPRIM garners tuples in a table τ_t results from a computation that depends on "*originating tuples*" that are stored in the τ_s of the child nodes. To describe properties of these tuples or properties of parts of these tuples we need a similar notion "the default theory below t" for parts of tuples. Ignoring

this detail, assume for now that our tuples in tables are only tuples of sets. Then, we garner recursively in pre-order along the induced subtree T' of T rooted at t a sequence s of originating tuples $(\boldsymbol{u}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_m})$. If the set T occurs in position i of tuple $\boldsymbol{u}$, our notion $T^{\leqslant t}(s)$ takes the union over all sets $T, T_1, \ldots, T_m$ at position i in the tuples $\boldsymbol{u_1}, \ldots, \boldsymbol{u_m}$. Since a node of type *rem* will typically result in multiple originating tuples, we have multiple sequences $s_1, \ldots, s_m$ of originating tuples in general. This results in a family $\mathcal{T}^{\leqslant t} := \{\, T^{\leqslant t}(s) \mid s \in \{s_1, \ldots, s_m\}\}$ of such sets. However, when stating properties we are usually only interested in the fact that each $S \in \mathcal{T}^{\leqslant t}$ satisfies the property. Therefore, we refer to $T^{\leqslant t}$ as any arbitrary $S \in \mathcal{T}^{\leqslant t}$. Additionally, let $T^{<t} := T^{\leqslant t} \setminus T$. The definition trivially extends to nested tuples and families of sets. For detailed information we refer the reader to so-called extension pointers [BCHW16].

3.1.3 Computing Stable Default Sets

In this section we solve the problems Ext and EnumSE which enumerates all stable extensions via a dynamic programming algorithm. Our algorithm is inspired by the previous work of J. Fichte et al. [FHMW17] about answer set programming, but substantial adjustments are required due to more complex semantics in default logic. Let the default theory D be a given with the corresponding pretty LTD $\mathcal{T} = (T, \chi, \omega)$ of the semi-primal graph $S(D)$.
Our table algorithm follows Definition 3.1 and is split into two parts:

(1.) finding satisfying default sets of D and

(2.) evidence of the subset minimality of these satisfying default sets.

On an abstract level, the SPRIM algorithm follows the approach of "divide and conquer". We restrict the search space to the bags of a given LTD and compute the default theory D in parts. since the SPRIM algorithm's vision is limited to the current bag, we are only able to compute the sets of satisfying defaults in parts. Nevertheless we vouch for, if the "visible" part Z of satisfying defaults set for any node $t \in T$ there is no smaller set of satisfying defaults, then Z can be augmented to a stable default set of $D_{\leqslant t}$.
In general it is not sufficient to consider only Z. We require some auxiliary information to decide the satisfiability of defaults. Additionally, we warrant to prove that Z witnesses a satisfying default set $Z^{\leqslant t}$. Although each $\delta \in Z^{\leqslant t}$ is vacuously c-satisfiable, we need to ensure that each default $\delta \in D \setminus Z^{\leqslant t}$ is also p-satisfiable or j-satisfiable. For this, we need a set $\mathcal{M}$ of (partial) assignments of $Z^{\leqslant t}$.
On that score, we garner in table τ_t tuples that are of the form $\langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C} \rangle$, comprising $Z \subseteq D_t$ and $\mathcal{M} \subseteq 2^X$ for $X = \chi(t) \cap \mathsf{Var}(D)$. The first three components $Z, \mathcal{M}, \mathcal{P}$ are responsible for finding sets of satisfying default sets of D. The last position $\mathcal{C}$ is itself a set of tuples $\langle \rho, \mathcal{AC}, \mathcal{BC} \rangle$ and can be seen as a *counter-witness* part, which is responsible for an evidence for the subset minimality of the satisfying default sets.

Now we add more details in into the tuples. We call Z the *witness set*, since Z *witnesses* the existence of a satisfying default set $Z^{\leqslant t}$ for a *sub-theory* S. Each element M in the set $\mathcal{M}$ of *witness models* witnesses the existence of a model of $F_{\leqslant t} := \bigwedge_{\delta \in Z^{\leqslant t}} \gamma(\delta)$. For our assumed witness set Z, we require a set $\mathcal{P}$ of *witness proofs*. The set $\mathcal{P}$ consists of tuples of the form $\langle \sigma, \mathcal{A}, \mathcal{B} \rangle$, where $\sigma : D_t \to \{p, j, c\}$ and $\mathcal{A}, \mathcal{B} \subseteq 2^X$ for $X = \chi(t) \cap \mathrm{at}(D)$. The function σ, which we call *state function*, maps each default $\delta \in D_t$ to a decision state $v \in \{p, j, c\}$ representing the case where δ is v-satisfiable. The set $\mathcal{A}$, which we call the *required p-assignments*, contains an assignment $A \in 2^X$ for *each* default δ that is claimed to be p-satisfiable. More formally, there is an assignment $A \in \mathcal{A}$ for each default $\delta \in \sigma^{-1}(p) \cup D_{<t}$ where $\sigma^{\leqslant t}(\delta) = p$ such that there is an assignment $A^{\leqslant t}$ that satisfies $F_{\leqslant t} \wedge \neg\alpha(\delta)$.
The set $\mathcal{B}$, which we call the *refuting j-assignments*, contains an assignment $B \in 2^X$ for certain defaults. Intuitively, for each $B \in \mathcal{B}$ there is a default δ in the current bag $\chi(t)$ or was in a bag below t such that there is an assignment $B^{\leqslant t}$ where the justification is not fulfilled. More formally, there is a $B \in \mathcal{B}$ if there is an assignment $B^{\leqslant t}$ that satisfies $F_{\leqslant t} \wedge \beta(\delta)$ for some default $\delta \in \sigma^{-1}(j) \cup D_{<t}$ where $\sigma^{\leqslant t}(\delta) = j$.
In the end, if Z proves the existence of a satisfying default set $Z^{\leqslant t}$ of theory $D_{<t}$, then there is at least one tuple $\langle \cdot, \cdot, \mathcal{B} \rangle \in \mathcal{P}$ with $\mathcal{B} = \varnothing$. Consequently, we require that $\mathcal{B} = \varnothing$ in order to guarantee that each default $\delta \in D_{<t}$ is j-satisfiable where $\sigma^{\leqslant t}(\delta) = j$. To conclude, if table τ_n for (empty) root n contains $\boldsymbol{u} = \langle Z, \cdot, \mathcal{P}, \mathcal{C} \rangle$ and $\mathcal{P}$ contains $\langle \cdot, \cdot, \varnothing \rangle$, then $Z^{\leqslant t}$ is a satisfying default set of the default theory D.
The purpose of $\mathcal{C}$ is to invalidate the subset-minimality of $Z^{\leqslant t}$. This will be covered later, as it works similar to the witness-part.

In lieu of this we finally present our SPRIM algorithm and discuss the main cases of it. SPRIM uses the following abbreviations:

Algorithm 2: Table algorithm SPRIM$(t, \chi_t, \omega_t, D_t,$ Child-Tabs$)$.

In: Bag χ_t, label mapping ω_t, bag-theory D_t, and child tables Child-Tabs of t.
Out: Table τ_t.

1 **if** type$(t) =$ *leaf* **then** $\tau_t \leftarrow \{\langle \varnothing, \{\varnothing\}, \{\langle \varnothing, \varnothing, \varnothing\rangle\}, \ \varnothing\rangle\}$;
2 **else if** type$(t) =$ *int*, $\delta \in D_t$ *is the introduced default, and* $\tau' \in$ *Child-Tabs* **then**
3 $\quad \tau_t \leftarrow \{\langle Z^+_\delta, \mathcal{M}, \mathrm{sub}_{\delta,\{c\}}(\mathcal{P}),\ \mathrm{sub}_{\delta,\{p,j,c\}}(\mathcal{C}) \cup \mathrm{sub}_{\delta,\{p,j\}}(\mathcal{P},\mathcal{M})\rangle,$
$\qquad\quad \langle Z, \mathcal{M}, \mathrm{sub}_{\delta,\{p,j\}}(\mathcal{P}),\ \mathrm{sub}_{\delta,\{p,j\}}(\mathcal{C})\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau'\}$
4 **else if** type$(t) =$ *label,* $\{(\gamma, \delta)\} = \omega_t$ *is the label of* t, $\delta \in D_t$, *and* $\tau' \in$ *Child-Tabs* **then**
5 $\quad \tau_t \leftarrow \{\langle Z, \mathrm{Mod}_{\mathcal{M}}(\gamma(\delta)), \mathrm{PCon}_\delta(\mathcal{P}),\ \mathrm{CW}_d(\mathcal{C})\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau', \delta \in Z\} \cup$
$\qquad\quad \{\langle Z, \mathcal{M}, \mathcal{P},\ \mathcal{C}\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau', \delta \notin Z\}$
6 **else if** type$(t) =$ *label,* $\{(\alpha, \delta)\} = \omega_t$ *is the label of* t, $\delta \in D_t$, *and* $\tau' \in$ *Child-Tabs* **then**
7 $\quad \tau_t \leftarrow \{\langle Z, \mathcal{M}, \mathrm{PPre}_\delta(\mathcal{P},\mathcal{M}),\ \mathrm{CPre}_\delta(\mathcal{C})\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau'\}$
8 **else if** type$(t) =$ *label,* $\{(\beta, \delta)\} = \omega_t$ *is the label of* t, $\delta \in D_t$, *and* $\tau' \in$ *Child-Tabs* **then**
9 $\quad \tau_t \leftarrow \{\langle Z, \mathcal{M}, \mathrm{PJust}_\delta(\mathcal{P},\mathcal{M}),\ \mathrm{PJust}_\delta(\mathcal{C},\mathcal{M})\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau'\}$
10 **else if** type$(t) =$ *int,* $a \in \chi_t$ *is the introduced variable, and* $\tau' \in$ *Child-Tabs* **then**
11 $\quad \tau_t \leftarrow \{\langle Z, \mathcal{M} \cup \mathcal{M}^{\uplus}_a, \mathrm{AGuess}_a(\mathcal{P}),\ \mathrm{AGuess}_a(\mathcal{C})\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau'\}$
12 **else if** type$(t) =$ *rem,* $\delta \notin D_t$ *is the removed default, and* $\tau' \in$ *Child-Tabs* **then**
13 $\quad \tau_t \leftarrow \{\langle Z^-_\delta, \mathcal{M}, \mathrm{SProj}_\delta(\mathcal{P}),\ \mathrm{SProj}_\delta(\mathcal{C})\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau'\}$
14 **else if** type$(t) =$ *rem,* $a \notin \chi_t$ *is the removed variable, and* $\tau' \in$ *Child-Tabs* **then**
15 $\quad \tau_t \leftarrow \{\langle Z, \mathcal{M}^{\sim}_a, \mathrm{AProj}_a(\mathcal{P}),\ \mathrm{AProj}_a(\mathcal{C})\rangle \mid \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle \in \tau'\}$
16 **else if** type$(t) =$ *join and* $\tau', \tau'' \in$ *Child-Tabs with* $\tau' \neq \tau''$ **then**
17 $\quad \tau_t \leftarrow \{\langle Z, \mathcal{M}' \cap \mathcal{M}'', \mathcal{P}' \hat{\bowtie}_{\mathcal{M}',\mathcal{M}''} \mathcal{P}'', (\mathcal{C}' \hat{\bowtie}_{\mathcal{M}',\mathcal{M}''} \mathcal{C}'') \cup (\mathcal{P}' \hat{\bowtie}_{\mathcal{M}',\mathcal{M}''} \mathcal{C}'') \cup$
$\qquad\quad (\mathcal{C}' \hat{\bowtie}_{\mathcal{M}',\mathcal{M}''} \mathcal{P}'')\rangle \mid \langle Z, \mathcal{M}', \mathcal{P}', \mathcal{C}'\rangle \in \tau', \langle Z, \mathcal{M}'', \mathcal{P}'', \mathcal{C}''\rangle \in \tau''\}$
18 **return** τ_t

$$
\begin{aligned}
\mathcal{S}_{MO} &:= \{S \mid S \in \mathcal{S}, mo \in S\} \\
S^-_e &:= S \backslash \{e\} \\
S^{\sim}_e &:= \{S^-_e \mid S \in \mathcal{S}\} \\
\varnothing^?_e &:= \{\varnothing\} \\
\mathcal{S}^?_e &:= \bigcup_{S \in \mathcal{S}, S' \in (\mathcal{S} \backslash S)^?_e} \{S' \cup \{S^+_e\}, S' \cup \{S\}\} \\
\mathrm{cpy}_\delta(\mathcal{P}, \pi) &:= \{\langle \sigma, \mathcal{A}, \mathcal{B}\rangle \mid \langle \sigma, \mathcal{A}, \mathcal{B}\rangle \in \mathcal{P}, \sigma(\delta) \neq \pi\} \\
\mathrm{sub}_{\delta,\mathcal{S}}(\mathcal{P}, \mathcal{M}) &:= \{\langle \sigma^+_{\delta \mapsto \pi}, \mathcal{A}, \mathcal{M}^+_{mo} \cup \mathcal{B}\rangle \mid \langle \sigma, \mathcal{A}, \mathcal{B}\rangle \in \mathcal{P}, \pi \in \mathcal{S}\} \\
\mathrm{sub}_{\delta,\mathcal{S}}(\mathcal{P}) &:= \mathrm{sub}_{\delta,\mathcal{S}}(\mathcal{P}, \varnothing) \\
\mathrm{PCon}_\delta(\mathcal{P}) &:= \{\langle \sigma, \mathcal{A}, \mathrm{Mod}_{\mathcal{B}}(\gamma(\delta))\rangle \mid \langle \sigma, \mathcal{A}, \mathcal{B}\rangle \in \mathcal{P}, \sigma(\delta) = c, \\
&\qquad \mathcal{A} = \mathrm{Mod}_{\mathcal{A}}(\gamma(\delta))\} \\
\mathrm{CW}_\delta(\mathcal{C}) &:= \mathrm{PCon}_\delta(\mathcal{C}) \cup \{\langle \rho, \mathcal{AC}, \mathcal{BC}_{MO} \cup \mathrm{Mod}_{\mathcal{BC}}(\gamma(\delta))\rangle \\
&\qquad \mid \langle \rho, \mathcal{AC}, \mathcal{BC}\rangle \in \mathcal{C}, \rho(\delta) \neq c\} \\
\mathrm{PPre}_\delta(\mathcal{P}, \mathcal{M}) &:= \mathrm{cpy}_\delta(\mathcal{P}, p) \cup \{\langle \sigma, \mathcal{A} \cup \mathcal{A}', \mathcal{B}\rangle \mid \langle \sigma, \mathcal{A}, \mathcal{B}\rangle \in \mathcal{P},
\end{aligned}
$$

$$
\begin{aligned}
&\quad \sigma(\delta) = p, \mathcal{A}' \in \mathrm{Mod}_{\mathcal{M} \cup \mathcal{B}^{\sim}_{mo}}(\neg\alpha(\delta))\} \\
\mathrm{CPre}_\delta(\mathcal{C}) &:= \mathrm{PPre}_\delta(\mathcal{C}, \varnothing) \\
\mathrm{PJust}_\delta(\mathcal{P}, \mathcal{M}) &:= \mathrm{cpy}_\delta(\mathcal{P}, j) \cup \{\langle \sigma, \mathcal{A}, \mathcal{B} \cup [\mathrm{Mod}_{\mathcal{M}}(\beta(\delta))]^{\sim}_{mo}) \rangle \\
&\quad \mid \langle \sigma, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{P}, \sigma(\delta) = j\} \\
\mathrm{AGuess}_a(\mathcal{P}) &:= \{\langle \sigma, \mathcal{A}', \mathcal{B} \cup \mathcal{B}^{\uplus}_a \rangle \mid \mathcal{A}' \in \mathcal{A}^{?}_a, \langle \sigma, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{P}\} \\
\mathrm{SProj}_\delta(\mathcal{P}) &:= \{\langle \sigma \backslash \{\delta \mapsto p, \delta \mapsto j, \delta \mapsto c\}, \mathcal{A}, \mathcal{B} \rangle \mid \langle \sigma, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{P}\} \\
\mathrm{AProj}_a(\mathcal{P}) &:= \{\langle \sigma, \mathcal{A}^{\sim}_a, \mathcal{B}^{\sim}_a \rangle \mid \langle \sigma, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{P}\} \\
\mathcal{M}' \bowtie \mathcal{M}'' &:= \{M' \cup M'' \mid M' \in \mathcal{M}', M'' \in \mathcal{M}'', M' \cap [\chi_t]^{+}_{mo} = \\
&\quad M'' \cap [\chi_t]^{+}_{mo}\} \\
\mathcal{B}' \bowtie_{\mathcal{M}', \mathcal{M}''} \mathcal{B}'' &:= [\mathcal{B}' \bowtie (\mathcal{B}'' \cup \mathcal{M}'')] \cup [(\mathcal{B}' \cup \mathcal{M}') \bowtie \mathcal{B}''] \\
\mathcal{P}' \hat{\bowtie}_{\mathcal{M}', \mathcal{M}''} \mathcal{P}'' &:= \{\langle \sigma, \mathcal{AR}, \mathcal{B}' \bowtie_{\mathcal{M}', \mathcal{M}''} \mathcal{B}'' \rangle \mid \langle \sigma, \mathcal{A}', \mathcal{B}' \rangle \in \mathcal{P}', \\
&\quad \langle \sigma, \mathcal{A}'', \mathcal{B}'' \rangle \in \mathcal{P}'', \mathcal{AR} = \mathcal{A}' \bowtie (\mathcal{A}'' \cup \mathcal{M}'' \cup [\mathcal{B}'']^{\sim}_{mo}), \\
&\quad \mathcal{A}' \cup \mathcal{A}'' \subseteq \mathcal{AR}\} \cup \{\langle \sigma, \mathcal{RA}, \mathcal{B}' \bowtie_{\mathcal{M}', \mathcal{M}''} \mathcal{B}'' \rangle \\
&\quad \mid \langle \sigma, \mathcal{A}', \mathcal{B}' \rangle \in \mathcal{P}', \langle \sigma, \mathcal{A}'', \mathcal{B}'' \rangle \in \mathcal{P}'', \\
&\quad \mathcal{RA} = \mathcal{A}'' \bowtie (\mathcal{A}' \cup \mathcal{M}' \cup [\mathcal{B}']^{\sim}_{mo}), \mathcal{A}' \cup \mathcal{A}'' \subseteq \mathcal{AR}\}
\end{aligned}
$$

As mentioned before, we first discuss the methodology of SPRIM for computing satisfying defaults set. For this purpose the first tree tuple positions, highlighted as red and green, are responsible. Let $t \in T$, $\boldsymbol{u}' = \langle Z, \mathcal{M}, \mathcal{P}, \cdot \rangle$ a tuple in table $\tau_{t'}$ for a child node t' of t and $\langle \sigma, \mathcal{A}, \mathcal{B} \rangle$ a tuple in $\mathcal{P}$. We describe informally how we transform $\boldsymbol{u}'$ into one or more tuples for the table τ_t.

The first case type(t)= *int* (line 2-4) introduces a default $\delta \in D_t$. Doing so, the algorithm guesses whether δ is p-satisfiable, j-satisfiable, or c-satisfiable. For this purpose, $\mathrm{sub}_{\delta,\mathcal{S}}(\mathcal{P})$ adds potential proofs to $\mathcal{P}$ where the satisfiability state of δ is within $\mathcal{S}$. The following three cases (line 4-9) cover the nodes of the type *label* in the following way: If (γ, δ) is the label and $\sigma(\delta) = c$, each $M \in \mathcal{M}$ has to be model of $\gamma(\delta)$. $\mathrm{PCon}_d(\mathcal{P})$ only keeps tuples in $\mathcal{P}$ where each $A \in \mathcal{A}$ is a model of $\gamma(\delta)$.
Line 6-7 cover the case where (α, δ) is the label and $\sigma(\delta) = p$. $\mathrm{PPre}_d(\mathcal{P}, \mathcal{M})$ enforces that each $A \in \mathcal{A}$ within $\mathcal{P}$ is a model of $\neg\alpha(\delta)$.
In line 8, (β, δ) is the label and $\sigma(\delta) = j$. $\mathrm{PJust}_d(\mathcal{P}, \mathcal{M})$ adds assignments of $\mathcal{M}$ to $\mathcal{B}$ that are also models of $\beta(\delta)$.

Next, we turn to the case where a variable a is introduced. Roughly speaking the algorithm guesses the assignment of the variable. In line 10, we augment the existing witness set $M \cup \{a\}$ for each $M \in \mathcal{M}$. $\mathrm{AGuess}_a(\mathcal{P})$ works analogously to $\mathcal{M}$ for $\mathcal{B}$ and computes all potential combinations of every $A \in \mathcal{A}$, where a is either set to true or to false.

The penultimate cases in line 12-15 are for nodes of type *rem*, which remove a default δ

from Z. Additionally, since δ is not considered anymore, $\mathrm{SProj}_\delta(\mathcal{P})$ removes δ from the domain of the mapping σ.
In line 14, we remove a variable a from each $M \in \mathcal{M}$. Further, $\mathrm{AProj}_a(\mathcal{P})$ removes a variable a from each $A \in \mathcal{A}$ as well as from each $B \in \mathcal{B}$.

The last case is for a node of type *join*. Here we have to evaluate two child nodes t' and t'' with corresponding tables τ' and τ'' and corresponding tuples $\boldsymbol{u}' \in \tau'$ and $\boldsymbol{u}'' \in \tau''$, respectively. Intuitively, tuples $\boldsymbol{u}'$ and $\boldsymbol{u}''$ represent intermediate results of two different branches in T, which we have to *join* i.e., to combine (with respect to the witness extension, witness states and witness models) to obtain the main result. The join operation $\bowtie$ can be seen as a combination of inner and outer joins used in database theory [AHV95]. Note that e.g. for an assignment $B \in \mathcal{B}$ to appear within the witness proof $\mathcal{P}$ of τ_t, it suffices that B is a corresponding witness model of one of the tuples $\boldsymbol{u}'$ or $\boldsymbol{u}''$.

Next we clarify the notion via the following example.

Example 3.3 (Computing Stable Default Sets)**.**
Let D be a given default theory $\big(\delta_1 = (\frac{\top : \neg x_2}{\neg x_1}), \delta_2 = (\frac{\top : x_2}{x_1 \vee x_2})\big)$ with the corresponding pretty LTD of the semi-primal graph $S(D)$, see Figure **??**. The SPRIM algorithm traverses the pretty LTD in post-order and computes the tables $\tau_1, \ldots, \tau_{18}$ in the following way. Here, we will only explain selected cases to avoid repetitions.
We call for presentation that each tuple in a table τ_t is defined by a number, i.e., the i-th tuple corresponds to $\boldsymbol{u}_{t.i} = \langle Z_{t.i}, \mathcal{M}_{t.i}, \mathcal{P}_{t.i}, \mathcal{C}_{t.i} \rangle$. The numbering naturally extends to sets in witness proofs and counter-witnesses. For the moment we ignore the counter witness part $\mathcal{C}$.
The type of t_1 is leaf. Therefore, table τ_1 contains empty sets only, i.e.
$\tau_1 = \{\langle \varnothing, \{\varnothing\}, \{\langle \varnothing, \varnothing, \varnothing \rangle\}, \cdot \rangle\}$.

The next node introduces a variable x_2 ($\mathrm{type}(t_2) = \mathit{int}$). We evaluate table τ_2 using table τ_1 by $\{\langle \sigma_{1.1}, \mathcal{M}_{2.1}, \mathcal{A}_{1.1} \rangle\}$, where $\mathcal{M}_{2.1}$ contains $M_{1.1.k}$ and $M_{1.1.k} \cup \{x_2\}$ for each $M_{1.1.k}$ ($k \leqslant 1$) in τ_1. This corresponds to a guess on x_2 and we obtain $\tau_2 = \{\langle \varnothing, \{\varnothing, x_2\}, \{\langle \varnothing, \varnothing, \varnothing \rangle\}, \cdot \rangle\}$.

Node t_3 introduces a default δ_2. This corresponds to a guess on δ_2. Consequently, we obtain two tuples, namely $\boldsymbol{u}_{3.1.k}$ ($1 \leqslant k \leqslant 2$) where δ_2 is guessed to be p-satisfiable or j-satisfiable. From this, we obtain $\mathcal{P}_{3.1.1} = \{\langle \{\delta_2 \mapsto p\}, \varnothing, \varnothing \rangle\}$ and $\mathcal{P}_{3.1.2} = \{\langle \{\delta_2 \mapsto j\}, \varnothing, \varnothing \rangle\}$ (cf. $\mathcal{P}_{3.1.k}$ and line 3 of SPRIM). In the second tuple $\boldsymbol{u}_{3.2}$ δ_2 is guessed to be c-satisfiable. This leads to $Z_{3.2} = \{\delta_2\}$ and $\mathcal{P}_{3.2} = \{\langle \{\delta_2 \mapsto c\}, \varnothing, \varnothing \rangle\}$.

The next node introduces a label (β, δ_2) and modifies $\mathcal{P}_{4.1.2}$, where δ_2 is guessed to be j-satisfiable. In particular, it chooses among $\mathcal{M}$ candidates which might contradict that δ_2 is j-satisfiable. This yields $\mathcal{B}_{4.1.2} = \{\{x_2\}\}$.

In table τ_5, we handle the case of δ_2 being p-satisfiable. In this case since $\alpha(\delta_2) = \top$, we do not find any model of $\bot$. In consequence, there is no corresponding successor of $\mathcal{P}_{4.1.1}$ in τ_5, i.e., in τ_5 it turns out that δ_2 can not be p-satisfiable.

Table τ_6 introduces the variable x_1 in the same vein as x_2 was introduced in table τ_2.

Table τ_7 is concerned with the conclusion $\gamma(\delta_2)$ of a default. It updates every assignment occurring in the table, such that the models satisfy $\gamma(\delta_1)$ if δ_2 is c-satisfiable.

In table τ_8 we remove the default δ_2 with the corresponding mapping and in τ_9 we eliminate the variable x_1.

The remaining cases of the left branch of the tree work similarly. In the end, join node t_{18} just combines witnesses agreeing on their content.
We already see that the given default theory has no stable extension, as the root node does not have a tuple of the form $\langle\varnothing, \{\varnothing\}, \langle\cdot,\cdot,\varnothing\rangle, \cdot\rangle$ with $\mathcal{P} \neq \varnothing$. Since the tuple $\mathcal{P}$ witnesses the existence of extension.

Assume that the root node in Example 3.3 contains a tuple which witnesses the existence of extension. In this case we require to witness the subset-minimality of this extension via the counter-witness part $\mathcal{C}$. We now consider the remaining part of the tuple $\boldsymbol{u}' = \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C}\rangle$, the counter-witness part $\mathcal{C}$ which is colored blue in SPRIM. The evaluation of $\mathcal{C}$ is similar to the witness proofs $\mathcal{P}$. The counter-witness tuple $\langle\rho, \mathcal{AC}, \mathcal{BC}\rangle \in \mathcal{C}$ consist of a state function $\rho : D_{\leqslant t} \mapsto \{p, j, c\}$, required p-assignments $\mathcal{AC} \subseteq 2^X$ and refuting j- assignments $\mathcal{BC} \subseteq 2^{(X \cup \{mo\})}$ for $X = \mathsf{Var}_{\leqslant t} \cap \chi(t)$. In contrast to the refuting j-assignments in $\mathcal{B}$, $\mathcal{BC}$ may in addition contain an assignment $B \in \mathcal{BC}$ with a marker mo. The marker indicates that $B^{\leqslant t}$ is actually not refuting, but only a model of $\gamma(\delta)$ for each default below t that is c-satisfiable, i.e., $\bigwedge_{\delta \in D_{\leqslant t}, \rho^{\leqslant t}(\delta)=c} \gamma(\delta)$. In other words, those assignments setting mo to true are the counter-witness assignments that do not refute c-assignments (comparable to witness assignments in $\mathcal{M}$).

The existence of a certain counter-witness tuple for a witness in a table τ_t establishes that the corresponding witness can *not* be extended to a stable default set of $D_{\leqslant t}$. In particular, there exists a stable extension for D if the table τ_n for root n contains a tuple of the form $\langle\varnothing, \{\varnothing\}, \mathcal{P}, \mathcal{C}\rangle$, where $\mathcal{P} \neq \varnothing$ and contains tuples of the form $\langle\cdot,\cdot,\varnothing\rangle$. Moreover, for *each* $\langle\rho, \mathcal{AC}, \mathcal{BC}\rangle \in \mathcal{C}$ there is $\varnothing \in \mathcal{BC}$ indicating a true refuting j-assignment for the empty root n. Intuitively, this establishes that there is no actual counter-witness, which contradicts that the corresponding satisfying default set $Z^{\leqslant t}$ is subset-minimal and hence indeed a stable default set.

A main difference of the counter-witness part to the witness part is that we require a special function $\mathrm{CW}_d(\mathcal{C})$ to establish that a default δ is j-satisfiable, which is defined with respect to a fixed set S, c.f. Case (2.) of Definition 3.1. $\mathrm{CW}_d(\mathcal{C})$ adds additional potential proofs involving counter-witnesses and mo models, where $\rho(\delta) \neq c$ but $\sigma(\delta) = c$.

In the remainder of this section we pursue the proofs of the correctness and completeness of Algorithm $\mathcal{DP}$. Our strategy for the proof of correctness is to provide evidence that a tuple for node t ensures the existence of a model for the subtheory $D_{\leqslant t}$. We need to consider each node type separatly. Afterwards we will prove the completeness of the $\mathcal{DP}$

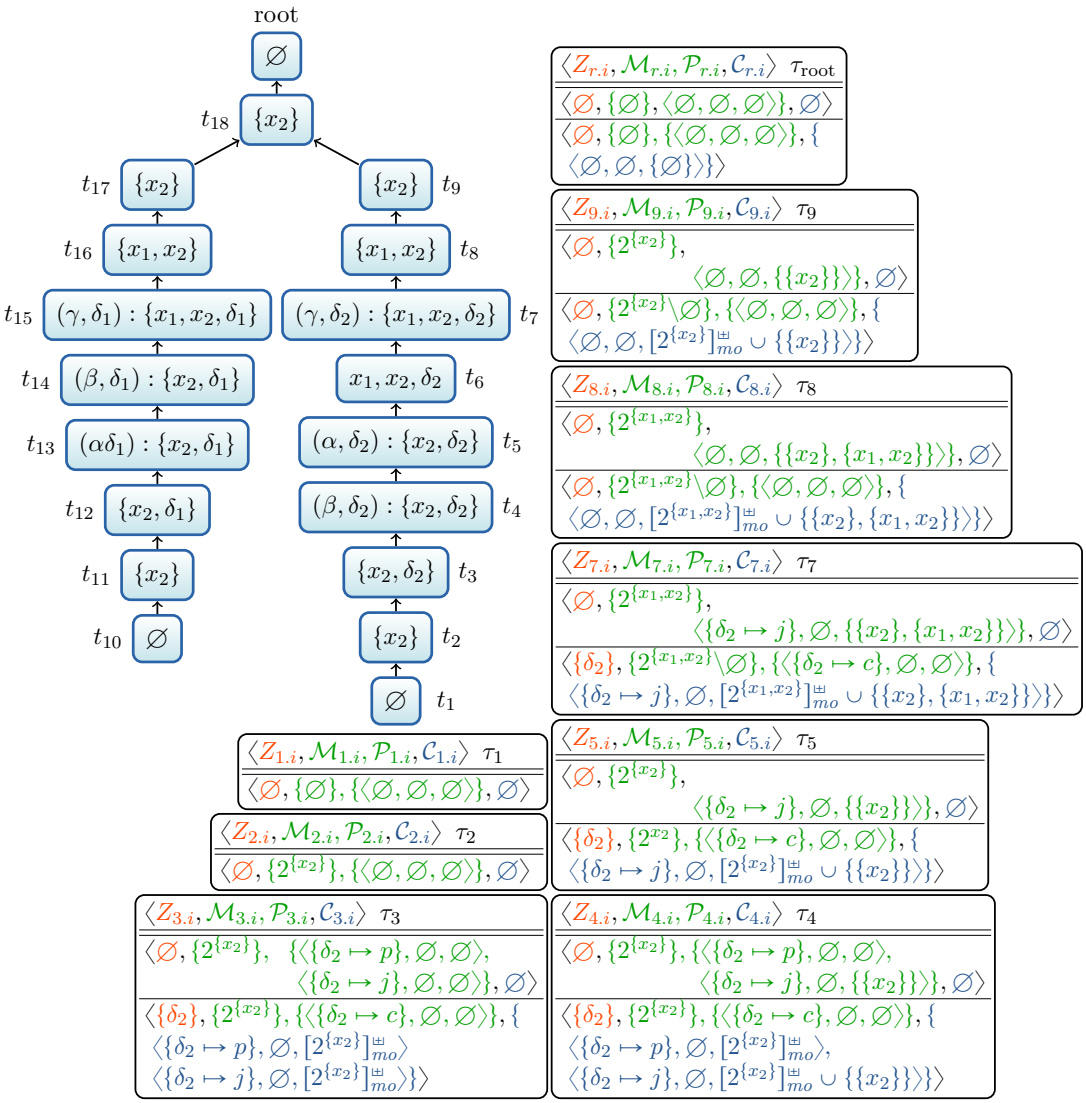


Figure 3.3: Pretty LTD of the $S(D)$ in Example 3.3

algorithm by showing that we get all possible sets of generating defaults when traversing the given tree-decomposition bottom-up. For this, the following additional notion is helpful.

Definition 3.6: Variables Below t and Bag-Default Parts

Let (T, χ, ω) be a pretty labeled TD of the semi-primal graph $S(D)$ of a given default theory D. The set $\mathsf{Var}_{\leqslant t} := \{ v \mid v \in \mathsf{Var}(D) \cap \chi(t'), t' \in \text{post-order}(T, t)\}$ is called *variables below* t.

Further, we define the *bag-default parts* for $f \in \{\alpha, \beta, \gamma\}$ (prerequisite, justification, or conclusion) $f_t := \{ f(\delta) \mid (f, \delta) \in \omega(t)\}$.
We naturally extend the definition of the bag-default parts to the respective default parts below t (analogously to our definitions for default theory below t), i.e., we also use $\alpha_{\leqslant t}, \beta_{\leqslant t}$, and $\gamma_{\leqslant t}$.

To clarify the previous notion, we give the following example.

Example 3.4.
Consider Fig. 3.3. Observe that we have introduced all variables of D already in t_6. Therefore, $\mathsf{Var}_{\leqslant t_6} = \mathsf{Var}(D)$ holds. Further, $\beta_{\leqslant t_9} = \{\beta(\delta_2)\}$ and $\gamma_{\leqslant t_{18}} = \{\gamma(\delta_1), \gamma(\delta_2)\}$.

Additionally, we abbreviate the mapping $\Gamma_t : 2^{\gamma(D_t)} \to 2^{\gamma_{\leqslant t}}$ by $\Gamma_t[E] := E \cap \gamma_{\leqslant t}$.

For the argumentation in the correctness proof we require some auxiliary notion.

Definition 3.7: Partial Extension under E for t

Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ be an LTD of the semi-primal graph $S(D)$ of D, where $T = (N, \cdot, \cdot)$, and $t \in N$ be a node. Further, let $\varnothing \subsetneq \mathcal{B} \subseteq 2^{\mathsf{Var}_{\leqslant t} \cup \{mo\}}$, $\mathcal{A} \subseteq 2^{\mathcal{B}\,\widetilde{}_{mo}}$, $\sigma : D_{\leqslant t} \to \{p, j, c\}$, $E \supseteq \gamma(Z)$, where $Z := \sigma^{-1}(c)$. The tuple $(\sigma, \mathcal{A}, \mathcal{B})$ is a *partial extension under E for t* if the following conditions hold:

(1.) Z is a set of satisfying defaults of $D_{<t} \backslash [\{\delta \in D_{<t} \mid \sigma(\delta) = j, \exists B \in \mathcal{B} : B \models \Gamma_t[E] \wedge \beta(\delta)\}]$,

(2.) $\mathcal{A}$ is a set such that:

  (1.) $|\mathcal{A}| \leqslant |\sigma^{-1}(p)| - 1$

  (2.) $\exists \delta \in D_{\leqslant t} : \sigma(\delta) = p, \alpha(\delta) \in \alpha_{\leqslant t}, A \models \Gamma_t[\gamma(Z)] \wedge \neg\alpha(\delta)$ for every $A \in \mathcal{A}$

  (3.) $\exists A \in \mathcal{A} : A \models \Gamma_t[\gamma(Z)] \wedge \neg\alpha(\delta) \Longleftarrow \sigma(\delta) = p$ for every $\delta \in D_{\leqslant t}$ such that $\alpha(\delta) \in \alpha_{\leqslant t}$

(3.) $\mathcal{B}$ is the largest set such that:

  (1.) $B \models \Gamma_t[\gamma(Z)]$ for every $B \in \mathcal{B}$

(2.) $\exists \delta \in D_{\leqslant t} : \sigma(\delta) = j, \beta(\delta) \in \beta_{\leqslant t}, B \vDash \Gamma_t[E] \wedge \beta(\delta)$ for every $B \in \mathcal{B}$ where $mo \notin B$

Definition 3.8: Partial Solution for t

Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ where $T = (N, \cdot, n)$ be an LTD of $S(D)$, and $t \in N$ be a node. A *partial solution for* t is a tuple $(Z, \mathcal{M}, \mathcal{P}, \mathcal{C})$ where $Z \subseteq D_{\leqslant t}$, and $\mathcal{P}$ is the largest set of tuples such that each $(\sigma, \mathcal{A}, \mathcal{B}) \in \mathcal{P}$ is a partial extension under $\gamma(Z)$ with $\mathcal{B}_{MO} = \varnothing$ and $Z = \sigma^{-1}(c)$. Moreover, $\mathcal{C}$ is the largest set of tuples such that for each $(\rho, \mathcal{AC}, \mathcal{BC}) \in \mathcal{C}$, we have that $(\rho, \mathcal{AC}, \mathcal{BC})$ is a partial extension under $\gamma(Z)$ with $\rho^{-1}(c) \subsetneq \sigma^{-1}(c)$. Finally, $\mathcal{M} \subseteq 2^{\mathsf{Var}_{\leqslant t}}$ is the largest set with $M \vDash \Gamma_t[\gamma(Z)]$ for each $M \in \mathcal{M}$.

We present now the link between partial extension and partial solution in the proceed lemma. It depicts that a partial solution is a sin qua non for a partial extension.

Lemma 3.5.
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ be an LTD of the semi-primal graph $S(D)$, where $T = (\cdot, \cdot, n)$, and $\chi(n) = \varnothing$. Then, there exists a stable default set Z' for D if and only if there exists a partial solution $\boldsymbol{u} = (Z', \mathcal{M}, \mathcal{P}, \mathcal{C})$ for root n with at least one tuple $\langle \sigma, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{P}$ where $\mathcal{B} = \varnothing$ and $\mathcal{C}$ is of the following form: For each $(\rho, \mathcal{AC}, \mathcal{BC}) \in \mathcal{C}$, $\mathcal{BC}_{MO} \neq \mathcal{BC}$.

Proof. Given a stable default set Z' of D we construct $\boldsymbol{u} = (Z', \mathcal{M}, \mathcal{P}, \mathcal{C})$ where we generate every potential $\sigma : D \to \{p, j, c\}$ such that $\sigma(\delta) = c$ for $\delta \in Z'$ as follows. For $\delta \in D \backslash Z'$, we are allowed to set $\sigma(\delta) := p$ if $\gamma(Z') \wedge \neg\alpha(\delta)$ is satisfiable and $\sigma(\delta) := j$ if $\gamma(Z') \wedge \beta(\delta)$ is unsatisfiable. For each of this functions σ, we require $\langle \sigma, \mathcal{A}, \varnothing \rangle \in \mathcal{P}$, where $\mathcal{A} \subseteq 2^{\mathsf{Var}(D)}$ is the smallest set with $|\mathcal{A}| \leqslant |\sigma^{-1}(\alpha)| - 1$ such that for all $\delta \in \sigma^{-1}(\alpha)$ there is at least one $A \in \mathcal{A}$ with $A \vDash \gamma(Z') \wedge \neg\alpha(\delta)$.

Moreover, we define set $\mathcal{M} := \mathrm{Mod}_{2^{\mathsf{Var}(D)}}(\bigwedge_{\delta \in Z'} \gamma(\delta))$, in order for $\boldsymbol{u}$ to be a partial solution for n (see Definition 3.8). We construct $\mathcal{C}$, consisting of partial solutions $(\rho, \mathcal{AC}, \mathcal{BC})$ where we use every potential state function ρ with $\rho^{-1}(c) \subsetneq \sigma^{-1}(c)$. For this, let $Z := \rho^{-1}(c)$. For the defaults δ with $\rho(\delta) \neq c$, i.e., defaults δ that are p-satisfiable or j-satisfiable, we also set their state $\rho(\delta)$ to α or β, respectively (analogous to above). Finally, we define set

$$\mathcal{BC} := [\mathrm{Mod}_{2^{\mathsf{Var}(D)}}(\bigwedge_{\delta \in Z} \gamma(\delta))^{\uplus}_{mo} \cup [\bigcup_{\delta : \rho(\delta) = j} \mathrm{Mod}_{2^{\mathsf{Var}(D)}}([\bigwedge_{\delta \in Z'} \gamma(\delta)] \wedge \beta(\delta))],$$

and $\mathcal{AC} \subseteq 2^{\mathsf{Var}(D)}$ as the smallest set such that $|\mathcal{AC}| \leqslant |\rho^{-1}(p)| - 1$ and for all $\delta \in \rho^{-1}(p)$, there is at least one $AC \in \mathcal{AC}$ with $AC \vDash \gamma(Z) \wedge \neg\alpha(\delta)$ according to Definition 3.7.

For the reverse direction, Definitions 3.7 and 3.8 guarantee that Z' is a stable extension if there exists such a partial solution $\boldsymbol{u}$. In consequence, the lemma holds. □

Next, we require the notion of local partial solutions corresponding to the tuples obtained in Algorithm 2.

Definition 3.9: Local Partial Solution Part

Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ an LTD of the semi-primal graph $S(D)$, where $T = (N, \cdot, n)$, and $t \in N$ be a node. A tuple $(\sigma, \mathcal{A}, \mathcal{B})$ is a *local partial solution part* of partial solution $(\hat{\sigma}, \hat{\mathcal{A}}, \hat{\mathcal{B}})$ for t if

(1.) $\sigma = \hat{\sigma} \cap (\chi(t) \times \{p, j, c\})$,

(2.) $\mathcal{A} = \hat{\mathcal{A}}_t$, and

(3.) $\mathcal{B} = \hat{\mathcal{B}}_t$, where $\mathcal{S}_t := \{S \cap (\chi(t) \cup \{mo\}) \mid S \in \mathcal{S}\}$.

Definition 3.10: Local Partial Solution

Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ an LTD of the semi-primal graph $S(D)$, where $T = (N, \cdot, n)$, and $t \in N$ be a node. A tuple $\boldsymbol{u} = \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C} \rangle$ is a *local partial solution* for t if there exists a partial solution $\hat{\boldsymbol{u}} = (\hat{Z}, \hat{\mathcal{M}}, \hat{\mathcal{P}}, \hat{\mathcal{C}})$ for t such that the following conditions hold:

(1.) $Z = \hat{Z} \cap 2^{D_t}$,

(2.) $\mathcal{M} = \hat{\mathcal{M}}_t$,

(3.) $\mathcal{P}$ is the smallest set containing local partial solution part $(\sigma, \mathcal{A}, \mathcal{B})$ for each $(\hat{\sigma}, \hat{\mathcal{A}}, \hat{\mathcal{B}}) \in \hat{\mathcal{P}}$, and

(4.) $\mathcal{C}$ is the smallest set with local partial solution part $(\rho, \mathcal{AC}, \mathcal{BC}) \in \mathcal{C}$ for each $(\hat{\rho}, \hat{\mathcal{AC}}, \hat{\mathcal{BC}}) \in \hat{\mathcal{C}}$.

We denote by $\hat{\boldsymbol{u}}^t$ the local partial solution $\boldsymbol{u}$ for t given partial solution $\hat{\boldsymbol{u}}$.

In the following we show, that it suffices to store local partial solution instead of partial solutions for a node $t \in N$.

Lemma 3.6.
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ an LTD of $S(D)$, where $T = (N, \cdot, n)$, and $\chi(n) = \varnothing$. Then, there exists a stable default set set for D if and only if there exists a local partial solution of the form $\langle \varnothing, \{\varnothing\}, \mathcal{P}, \mathcal{C} \rangle$ for the root $n \in N$ with at least one tuple of the form $\langle \sigma, \mathcal{A}, \varnothing \rangle \in \mathcal{P}$. Moreover, for each $\langle \rho, \mathcal{AC}, \mathcal{BC} \rangle$ in $\mathcal{C}$, $\mathcal{BC}_{MO} \neq \mathcal{BC}$.

Proof. According to Definition 3.10, every partial solution for the root n is an extension of the local partial solution $\boldsymbol{u}$ for the root $n \in N$, with a bag $\chi(n) = \varnothing$. In combination with Lemma 3.5, we obtain that the lemma is true.

□

For simplicity of notation we continue to write Var_t for variables occurring in the bag $\chi(t)$, i.e., $\mathsf{Var}_t := \chi(t) \setminus D_t$.

Building on the previous results we are able to prove the soundness and correctness of the SPRIM algorithm.

Proposition 3.7 (Soundness)**.**
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ an LTD of the semi-primal graph $S(D)$, where $T = (N, \cdot, \cdot)$, and $t \in N$ a node. Given a local partial solution $\boldsymbol{u}'$ of child table τ' (or local partial solution $\boldsymbol{u}'$ of table τ' and local partial solution $\boldsymbol{u}''$ of table τ''), each tuple $\boldsymbol{u}$ of table τ_t constructed using table algorithm SPRIM is also a local partial solution.

Proof. Let $\boldsymbol{u}'$ be a local partial solution for $t' \in N$ and $\boldsymbol{u}$ a tuple for node $t \in N$ such that $\boldsymbol{u}$ was derived from $\boldsymbol{u}'$ using table algorithm SPRIM. Hence, node t' is the only child of t and t is either removal or introduce node.

Assume that t is a removal node and $\delta \in D_{t'} \setminus D_t$ for some default δ. Observe that for $\boldsymbol{u} = \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C} \rangle$ and $\boldsymbol{u}' = \langle Z', \mathcal{M}, \mathcal{P}', \mathcal{C}' \rangle$, sets $\mathcal{A}$ and $\mathcal{B}$ are equal, i.e., $\langle \cdot, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{P} \iff \langle \cdot, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{P}'$ and $\langle \cdot, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{C} \iff \langle \cdot, \mathcal{A}, \mathcal{B} \rangle \in \mathcal{C}'$. Since $\boldsymbol{u}'$ is a local partial solution, there exists a partial solution $\hat{\boldsymbol{u}}'$ of t', satisfying the conditions of Definition 3.10. Then, $\hat{\boldsymbol{u}}'$ is also a partial solution for node t, since it satisfies all conditions of Definitions 3.7 and 3.8. Finally, note that $\boldsymbol{u} = (\hat{\boldsymbol{u}}')^t$ since the projection of $\hat{\boldsymbol{u}}'$ to the bag $\chi(t)$ is $\boldsymbol{u}$ itself. In consequence, the tuple $\boldsymbol{u}$ is a local partial solution.

For $a \in \mathsf{Var}_{t'} \setminus \mathsf{Var}_t$ as well as for introduce nodes, we can analogously check the proposition.

Next, assume that t is a join node. Therefore, let $\boldsymbol{u}'$ and $\boldsymbol{u}''$ be local partial solutions for $t', t'' \in N$, respectively, and $\boldsymbol{u}$ be a tuple for node $t \in N$ such that $\boldsymbol{u}$ can be derived using both $\boldsymbol{u}'$ and $\boldsymbol{u}''$ in accordance with the SPRIM algorithm. Since $\boldsymbol{u}'$ and $\boldsymbol{u}''$ are local partial solutions, there exists partial solution $\hat{\boldsymbol{u}}' = (\hat{Z}', \hat{\mathcal{M}}', \hat{\mathcal{P}}', \hat{\mathcal{C}}')$ for node t' and partial solution $\hat{\boldsymbol{u}}'' = (\hat{Z}'', \hat{\mathcal{M}}'', \hat{\mathcal{P}}'', \hat{\mathcal{C}}'')$ for node t''. Using these two partial solutions, we can construct $\hat{\boldsymbol{u}} = (\hat{Z}' \cup \hat{Z}'', \hat{\mathcal{M}}' \bowtie \hat{\mathcal{M}}'', \hat{\mathcal{P}}' \mathbin{\hat{\bowtie}}_{\hat{\mathcal{M}}', \hat{\mathcal{M}}''} \hat{\mathcal{P}}'', (\hat{\mathcal{C}}' \mathbin{\hat{\bowtie}}_{\hat{\mathcal{M}}', \hat{\mathcal{M}}''} \hat{\mathcal{C}}'') \cup (\hat{\mathcal{P}}' \mathbin{\hat{\bowtie}}_{\hat{\mathcal{M}}', \hat{\mathcal{M}}''} \hat{\mathcal{C}}'') \cup (\hat{\mathcal{C}}' \mathbin{\hat{\bowtie}}_{\hat{\mathcal{M}}', \hat{\mathcal{M}}''} \hat{\mathcal{P}}''))$ where for $\bowtie(\cdot, \cdot)$ and $\hat{\bowtie}(\cdot, \cdot)$ we refer to Algorithm 2. Then, we check all conditions of Definitions 3.7 and 3.8 in order to verify that $\hat{\boldsymbol{u}}$ is a partial solution for t. Moreover, the projection $\hat{\boldsymbol{u}}^t$ of $\hat{\boldsymbol{u}}$ to the bag $\chi(t)$ is exactly $\boldsymbol{u}$ by construction and hence, $\boldsymbol{u} = \hat{\boldsymbol{u}}^t$ is a local partial solution.

Since one can provide similar arguments for each node type, we established soundness in terms of the statement of the proposition. □

Proposition 3.8 (Completeness)**.**
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ where $T = (N, \cdot, \cdot)$ be an LTD of $S(D)$ and $t \in N$ be a node. Given a local partial solution $\boldsymbol{u}$ of table τ_t, either t is a leaf node, or there exists a local partial solution $\boldsymbol{u}'$ of child table τ' (or local partial solution $\boldsymbol{u}'$ of table τ' and local partial solution $\boldsymbol{u}''$ of table τ'') such that $\boldsymbol{u}$ can be constructed by $\boldsymbol{u}'$ (or $\boldsymbol{u}'$ and $\boldsymbol{u}''$, respectively) and using table algorithm SPRIM.

Proof. Let $t \in N$ be a removal node and $\delta \in D_{t'} \setminus D_t$ with child node $t' \in N$. We show that there exists a tuple $\boldsymbol{u}'$ in table $\tau_{t'}$ for node t' such that $\boldsymbol{u}$ can be constructed using $\boldsymbol{u}'$ by SPRIM (Listing 2). Since $\boldsymbol{u}$ is a local partial solution, there exists a partial solution $\hat{\boldsymbol{u}} = (\hat{Z}, \hat{\mathcal{M}}, \hat{\mathcal{P}}, \hat{\mathcal{C}})$ for node t, satisfying the conditions of Definition 3.10. It is easy to see that $\hat{\boldsymbol{u}}$ is also a partial solution for t' and we define $\boldsymbol{u}' := \hat{\boldsymbol{u}}^{t'}$, which is the projection of $\hat{\boldsymbol{u}}$ onto the bag of t'. Apparently, the tuple $\boldsymbol{u}'$ is a local partial solution for node t' according to Definition 3.10. Then, $\boldsymbol{u}$ can be derived using SPRIM algorithm and $\boldsymbol{u}'$. By similar arguments, we establish the proposition for $a \in \mathsf{Var}_{t'} \setminus \mathsf{Var}_t$ and the remaining node types. Hence, the propositions holds. □

Finally we are in the situation to prove that the algorithm $\mathcal{DP}$ decides the Ext problem.

Theorem 3.9.
Given a default theory D, the algorithm $\mathcal{DP}$ correctly solves Ext.

Proof. We first show soundness. Let $\mathcal{T} = (T, \chi, \omega)$ be the given LTD, where $T = (N, \cdot, n)$. By Lemma 3.6 we know that there is a stable default set if and only if there exists a local partial solution for the root n. Note that the tuple is by construction of the form $\langle \varnothing, \{\varnothing\}, \mathcal{P}, \mathcal{C} \rangle$, where $\mathcal{P} \neq \varnothing$ can contain a combination of the following tuples $\langle \varnothing, \varnothing, \varnothing \rangle$, $\langle \varnothing, \{\varnothing\}, \varnothing \rangle$. For each $\langle \rho, \mathcal{AC}, \mathcal{BC} \rangle \in \mathcal{C}$, we have $\mathcal{BC}_{MO} \neq \mathcal{BC}$. In total, this results in 16 possible tuples, since $\mathcal{C} \subseteq 2^C$ can contain any combination (4 many) of C, where $C = \{\langle \varnothing, \varnothing, \{\varnothing, \{mo\}\} \rangle, \langle \varnothing, \{\varnothing\}, \{\varnothing, \{mo\}\} \rangle\}$.

We proceed by induction starting from the leaf nodes in order to end up with such a tuple at the root node n. In fact, the tuple $\langle \varnothing, \{\varnothing\}, \{\langle \varnothing, \varnothing, \varnothing \rangle\}, \varnothing \rangle$ is trivially a partial solution for (empty) leaf nodes by Definitions 3.7 and 3.8 and also a local partial solution of $\langle \varnothing, \{\varnothing\}, \{\langle \varnothing, \varnothing, \varnothing \rangle\}, \varnothing \rangle$ by Definition 3.10. We already established the induction step in Proposition 3.7. Consequently, when we reach the root n, when traversing the TD in post-order by Algorithm $\mathcal{DP}$, we obtain only valid tuples inbetween and a tuple of the form discussed above in the table of the root n witnesses an answer set.

Next, we establish completeness by induction starting from the root n. Let therefore, $\hat{Z}$ be an arbitrary stable default set of D. By Lemma 3.6, we know that for the root n there exists a local partial solution of the discussed form $\langle \varnothing, \{\varnothing\}, \mathcal{P}, \mathcal{C} \rangle$ for some partial solution $\langle \hat{Z}, \hat{\mathcal{M}}, \hat{\mathcal{P}}, \hat{\mathcal{C}} \rangle$. We already established the induction step in Proposition 3.8. Therefore, we obtain some (corresponding) tuples for every node t. Finally, stopping at the leaves n. In consequence, we have shown both soundness and completeness resulting in the fact that Theorem 3.9 is true. □

Proposition 3.10 (Completeness for Enumeration)**.**
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ where $T = (N, \cdot, \cdot)$ be an LTD of $S(D)$ and $t \in N$ be a node. Given a partial solution $\hat{\boldsymbol{u}}$ and the corresponding local partial solution $\boldsymbol{u} = \hat{\boldsymbol{u}}^t$ for table τ_t, either t is a leaf node, or there exists a local partial solution $\boldsymbol{u}'$ of child table τ' (or local partial solution $\boldsymbol{u}'$ of table τ' and local partial solution $\boldsymbol{u}''$ of table

Algorithm 3: Algorithm $\mathcal{NGD}_{\prec}(\mathcal{T}, \mathcal{S})$ for computing the next stable default set of $\mathcal{S}$.

In: TD $\mathcal{T} = (T, \cdot, \cdot)$ with $T = (N, \cdot, n)$, solution tuples $\mathcal{S}$, total ordering $\prec$ of orig.$(\cdot)$.
Out: The next solution tuples of $\mathcal{S}$ using $\prec$.

```
1  Tables[·] ← DP(T)
2  for iterate t in post-order(T,n) do
3  |  Child-Tabs := { Tables[t'] | t' is a child of t in T};
4  |  t̂ := parent of t
5  |  S[t] ← direct successor s' ≻ S[t] in orig_t̂(S[t̂])
6  |  if S[t] defined then
7  |  |  for iterate t' in Child-Tabs do
8  |  |  |  for iterate t'' in pre-order(T,t') do
9  |  |  |  |  t̂'' := parent of t''
10 |  |  |  |  S[t''] ← ≺-smallest element in orig_t̂''(S[t̂''])
11 |  |  return S;
12 return undefined;
```

τ'') such that $\boldsymbol{u}$ can be constructed by $\boldsymbol{u}'$ (or $\boldsymbol{u}'$ and $\boldsymbol{u}''$, respectively) and using table algorithm SPRIM.

Proof. The correctness proof requires to extend the previous results to establish a one-to-one correspondence when traversing the tree of the TD and such that we can reconstruct each solution as well as we do not get duplicates. According to the proof for completeness in Proposition 3.8, the result follows. □

We require the following three auxiliary results to prove that the algorithm $\mathcal{DP}$ can be used as a preprocessing step to construct tables from which we can correctly solve the problem EnumSE. More precisely, this is solved by first running Algorithm $\mathcal{DP}$, constructing the $\prec$-smallest solution $\mathcal{S}$, and then running Algorithm $\mathcal{NGD}_{\prec}(\mathcal{T}, \mathcal{S})$ on the resulting tables of Algorithm $\mathcal{DP}$ until $\mathcal{NGD}_{\prec}(\mathcal{T}, \mathcal{S})$ returns "undefined".

The next observation states that we are able to compute one unique partial solution, which is a consequence of Definition 3.8.

Observation 3.11.
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ where $T = (N, \cdot, \cdot)$ be an LTD of $S(D)$ and $t \in N$ be a node. Then, for each partial solution $\boldsymbol{u} = \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C} \rangle$ for t, $\mathcal{M}, \mathcal{P}$ and $\mathcal{C}$ are functional dependent from Z, i.e., for any partial solution $\boldsymbol{u}' = \langle Z, \mathcal{M}', \mathcal{P}', \mathcal{C}' \rangle$ for t, we have $\boldsymbol{u} = \boldsymbol{u}'$.

Proof. The claim immediately follows from Definition 3.8. □

Lemma 3.12.
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ with $T = (N, \cdot, \cdot)$ be an LTD of $S(D)$, and Z be a stable default set. Then, there is a unique set of tuples S, containing exactly one tuple per node $t \in N$ containing only local partial solutions of the unique partial solution for Z.

Proof. By Observation 3.11, given Z, we can construct one unique partial solution $\hat{\boldsymbol{u}} = \langle Z, \mathcal{M}, \mathcal{P}, \mathcal{C} \rangle$ for n. We then define the set S by $S := \bigcup_{t \in N} \{\hat{\boldsymbol{u}}^t\}$. Assume that there is a different set $S' \neq S$ containing also exactly one tuple per node $t \in N$. Then there is at least one node $t \in N$, for which the corresponding tuples $\boldsymbol{u} \in S, \boldsymbol{u}' \in S'$ differ ($\boldsymbol{u} \neq \boldsymbol{u}'$), since $\hat{\boldsymbol{u}}$ is unique and the computation $\hat{\boldsymbol{u}}^t$ is defined in a deterministic, functional way (see Definition 3.10). Consequently, either $\hat{\boldsymbol{u}}^t \neq \boldsymbol{u}$ or $\hat{\boldsymbol{u}}^t \neq \boldsymbol{u}'$, leading to the claim. □

Proposition 3.13.
Let D be a default theory, $\mathcal{T} = (T, \chi, \omega)$ with $T = (N, \cdot, \cdot)$ be an LTD of $S(D)$, and Z be a stable default set. Moreover, let S be the unique set of tuples, containing exactly one tuple per node $t \in N$ and containing only local partial solutions of the unique partial solution for Z. Given S, and tables of Algorithm SPRIM, one can compute in time $\mathcal{O}(||D||)$ a stable default set Z' with $Z' \neq Z$, assuming one can get for a specific tuple $\boldsymbol{u}$ for node t its corresponding $\prec$-ordered predecessor tuple set $\text{orig}_t(\boldsymbol{u})$ of tuples in the child node(s) of t in constant time.

Proof. Note that with Z, it is easy to determine, which element of S belongs to which node t in T. Consequently, we can construct a mapping $\mathcal{S} : N \to S$. With $\mathcal{S}$, we can easily apply algorithm $\mathcal{NGD}$, which is given in Algorithm 3, in order to construct a different solution $\mathcal{S}'$ in a systematic way with linear time delay, since $\mathcal{T}$ is pretty. □

Finally we prove next, that using resulting tables of $\mathcal{DP}$, the algorithm $\mathcal{NGD}$ solves the problem EnumSE correctly.

Theorem 3.14.
Given a default theory D, the combination of Algorithms $\mathcal{DP}$ and $\mathcal{NGD}$ correctly solves EnumSE.

Proof. Let LTD $\mathcal{T} = (T, \chi, \omega)$ with $T = (N, n)$ for graph $S(D)$ be given. Then we run our algorithm $\mathcal{DP}$ and get tables for each TD node. In order to enumerate all the stable default sets, we investigate each of these tuple, which lead to a valid stable default set (cf. proof of Theorem 3.9). For each of these tuples (if exist), we construct a first solution S, if exist, (as done in Lines 7 to 10 of Listing 3, for the root n) using $\text{orig}_t(\cdot)$, and total order $\prec$. Thereby, we keep track of which tuple in S belongs to which node, resulting in the mapping $\mathcal{S}$ (see proof of Proposition 3.13). Note that $\text{orig}_t(\cdot)$ and $\prec$ can easily be provided by remembering for each tuple an ordered set of predecessor tuple sets during

construction (using table algorithm SPRIM). Now, we call algorithm $\mathcal{NGD}_{\prec}(\mathcal{T}, \mathcal{S})$ multiple times, by outputting and passing the result again as argument, until the return value is undefined, enumerating solutions in a systematic way. Using correctness results by Theorem 3.9, and completeness result for enumeration by Proposition 3.10, we obtain only valid solution sets, which directly represent stable default sets and, in particular, we do not miss a single one. Observe, that by Lemma 3.12 we do not get duplicates.

□

In the last part of this section we investigate the runtime upper bounds. For this issue, we compute first the worst-case space requirements in tables for the nodes of our algorithm.

Proposition 3.15.
Given a default theory D, an LTD $\mathcal{T} = (T, \chi, \omega)$ with $T = (N, \cdot, \cdot)$ of the semi-primal graph $S(D)$, and a node $t \in N$. Then, there are at most $2^{k+1} \cdot 2^{2^{k+1}} \cdot 2^{2\cdot(3^{k+1}\cdot 2^{2^{k+2}})}$ tuples in τ_t using algorithm $\mathcal{DP}$, where $k := \mathrm{tw}S(D)$ the treewidth of the semi-primal graph $S(D)$.

Proof. Let D be the given default theory, $\mathcal{T} = (T, \chi, \omega)$ an LTD of the semi-primal graph $S(D)$, where $T = (N, \cdot, \cdot)$, and $t \in N$ a node of the TD. Then, by definition of a decomposition of the semi-primal graph for each node $t \in N$, we have $|\chi(t)| - 1 \leqslant k$. In consequence, we can have at most 2^{k+1} many witness defaults and $2^{2^{k+1}}$ many witnesses models. Each set $\mathcal{P}$ may contain a set of witness proof tuples of the form $\langle \sigma, \mathcal{A}, \mathcal{B} \rangle$, with at most 3^{k+1} many witness state σ mappings, $2^{2^{k+1}}$ many backfire witness models $\mathcal{B}$, and $2^{2^{k+1}}$ many required witnesses model sets. In the end, we need to distinguish $2^{k+1} \cdot 2^{2^{k+1}} \cdot 2^{(3^{k+1}\cdot 2^{2^{k+2}})}$ different witnesses of a tuple in the table τ_t for node t. For each witness, we can have at most $2^{(3^{k+1}\cdot 2^{2^{k+2}})}$ many counter-witnesses per witness default, witness models, and required witness model sets. Therefore, there are at most $2^{k+1} \cdot 2^{2^{k+1}} \cdot 2^{2\cdot(3^{k+1}\cdot 2^{2^{k+2}})}$ tuples in table τ_t for node t. In consequence, we established the proposition. □

Theorem 3.16.
Given a default theory D, the algorithm $\mathcal{DP}$ and runs in time $O(2^{2^{2^{k+4}}} \cdot ||S(D)||)$, where $k := \mathrm{tw}S(D)$ is the treewidth of the semi-primal graph $S(D)$.

Proof. Let D be a default theory, $S(D) = (V, \cdot)$ its semi-primal graph, and k be the treewidth of $S(D)$. Then, we can compute in time $2^{O(k^3)} \cdot |V|$ an LTD of width at most k [**?**]. We take such a TD and compute in linear time a nice TD [**?**]. Let $\mathcal{T} = (T, \chi, \delta)$ be such a pretty LTD with $T = (N, \cdot, \cdot)$. Since the number of nodes in N is linear in the graph size and since for every node $t \in N$ the table τ_t is bounded by $2^{k+1} \cdot 2^{2^{k+1}} \cdot 2^{2\cdot(3^{k+1}\cdot 2^{2^{k+2}})}$ according to Proposition 3.15, we obtain a running time of $O(2^{2^{2^{k+4}}} \cdot ||S(D)||)$. Consequently, the theorem applies. □

3.2 Solving Default Logic using Backdoors

The purpose of this section is to examine Reiter's propositional default logic and its decision problems, but with the new approach of so-called strong backdoors. The utilisation of backdoors is already recognized as a helpful tool in propositional logic .
We will depict the approach for default logic to transform a given CNF formula φ of the complexity class $\mathfrak{C}$ by the application of backdoors to a formula in a fragment in a lower complexity class $\mathfrak{C}'$.

Definition 3.11: Strong Backdoors in Propositional Logic, [WGS03]

Let $\mathfrak{F}$ be a a class of formulae. A *strong* $\mathfrak{F}$*-backdoor* is a set $BV \subseteq \mathsf{Var}(\varphi)$ of variables of a formula φ s.t. for all assignments $\vartheta \in \mathbb{A}(V)$ we obtain $\varphi[\vartheta] \in \mathfrak{F}$.

In this way, given a CNF formula we are interested to modify it to a more restrictive formula class, providing a lower complexity. We summarize the most common classes of formulae in the Table 3.1.

Formula Classes	Clause Constraints	Clause Form
CNF	No restrains	$\{\ell_1^+, \ldots, \ell_n^+, \ell_1^-, \ldots, \ell_m^-\}$
HORN	At most one positive literal	$\{\ell_1^+, \ell_1^-, \ldots, \ell_n^-\}, \{\ell_1^-, \ldots, \ell_m^-\}$
KROM	Binary clauses	$\{\ell_1^+, \ell_2^+\}, \{\ell_1^+, \ell_2^-\}, \{\ell_1^-, \ell_2^-\}$
MONOTONE	Positive literals only	$\{\ell_1^+, \ldots, \ell_n^+\}$
POSITIVE-UNIT	Positive unit clauses only	$\{\ell_1^+\}$

Table 3.1: Formulae Classes with positive ℓ_i^+ and negative literals ℓ_i^-, where $n, m \in \mathbb{N}_{\geqslant 0}$

To illustrate the idea of backdoors, let us to examine following example:

Example 3.17 (Backdoors in Propositional Logic)**.**
Let the formula $\varphi_{\mathsf{CNF}} = (x_1 \vee x_2 \vee x_3) \wedge (x_3 \vee x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_5)$ be given. The question now is: "Does a KROM-strong backdoor exist?" We see at once that the answer is "*yes*". Choosing $BV = \{x_3\}$ yields:

$$\varphi[x_3 = 0] = (x_1 \vee x_2) \wedge (x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_5)$$
$$\varphi[x_3 = 1] = (\neg x_1 \vee x_5)$$

In consequence of choosing $BV = \{x_3\}$, we obtain for all assignments of $\varphi[x_3] \in \mathsf{KROM}$.

The most exciting results for the complexity of the stable extension problem, with regard to restricted classes of formulae, are conflated in the following proposition.

Proposition 3.18.

(1.) Ext(CNF) is Σ_2^P-complete [Got92].

(2.) Ext(HORN) is **NP**-complete [Sti90b, Sti90a].

(3.) Ext(POSITIVE-UNIT) $\in$ **P** [BTV12].

Their work has yielded a number of promising new avenues of research for default logic in combination with backdoors. The challenge now is to adapt backdoors to default logic.

While elements of backdoors in common propositional satisfiability are propositional variables and consequently can only have truth values of a variable "true" and "false", this does not hold for backdoors on default logic. Here we require a trinity of values of a formula φ concerning to an extension E of the default theory $T_{\mathsf{DL}} = \langle W, D \rangle$, namely:

(1.) φ is included in the stable extension E,

(2.) $\neg\varphi$ is included in the stable extension E,

(3.) neither φ nor $\neg\varphi$ is included in the stable extension E.

Now we want to illustrate this fact with a simple example.

Example 3.19. Let a default theory be given as $T_{\mathsf{DL}} = \langle \varnothing, \{\frac{\top : x_2}{x_3}\} \rangle$. A stable extension of the T_{DL} is $E = Th(x_3)$. In consequence, neither x_2 nor $\neg x_2$ is a part of the stable extension.

Due to the complex semantics of default logic it is not possible to transfer the required trinity directly from the satisfiability settings of propositional logic to DL. For this reason we need to take a detour by starting with some auxiliary definitions.

Definition 3.12: Extended Literal and Reducts

An *extended literal* is a literal or fresh variable x_ν. The negation of a literal ℓ, in symbols $\sim\ell$, is defined as $\neg x$ in case $\ell = x$ is a variable, resp. as x in case $\ell = \neg x$. Given a formula φ and an extended literal ℓ, the *reduct* $\rho_\ell(\varphi)$ is obtained from φ as follows:

(1.) If ℓ is a literal, then all clauses containing ℓ are deleted and additionally the literal $\sim\ell$ is deleted from all clauses.

(2.) If ℓ is x_ν, then we delete all occurrences of literals $\neg x, x$ from all clauses.

Let $T_{\mathsf{DL}} = \langle W, D \rangle$ be a default theory and ℓ an extended literal, then

$$\rho_\ell(W, D) := \left(\rho_\ell(W), \left\{ \frac{\rho_\ell(P) : \rho_x(J)}{\rho_\ell(C) \wedge y_i} \mid \delta_i = \frac{P : J}{C} \in D \right\} \right),$$

with a fresh proposition y_i according to default δ_i and $\rho_\ell(W) := \bigcup_{\omega \in W} \rho_\ell(\omega)$.

Next we lift the notion of extended literals and reducts to the sets of assignments. In this way we obtain so-called *trivalent assignment sets*.

Definition 3.13: Trivalent Assignment Sets

Let X be a set of variables, we denote the *Trivalent assignment sets* by

$$\mathbb{T}(X) := \left\{\{a_1, \ldots, a_{|X|}\} \mid x \in X \text{ and } a_i \in \{x, \neg x, x_\nu\}\right\}.$$

In comparison with the common assignment set $\mathbb{A}$, the trivalent assignment set is extended by the variable x_ν which has a "do not care" character. For $X' \in \mathbb{T}(X)$ we proceed for the reduct $\rho_{X'}(W, D)$ analogously to Definition 3.12. It is noteworthy that the order of application of reducts is irrelevant. Additionally, we show in the following proposition, that the implication of formulae is invariant under extending conjuncts of new variables to the premise.

Proposition 3.20.
Let two formulae $\varphi, \psi \in \mathsf{CNF}$ be given and let y be a fresh variable with $y \notin \mathsf{Var}(\varphi) \cup \mathsf{Var}(\psi)$. Then it holds $\varphi \models \psi$ if and only if $\varphi \wedge y \models \psi$.

Furthermore, in the following lemma we point out the invariance of CNF formulae without tautological clauses under the application of so-called "*deletion reducts*" $\rho_{x_\nu}(\cdot)$.

Lemma 3.21.
Let two formulae $\varphi, \psi \in \mathsf{CNF}$, without including tautological clauses be given. If $\varphi \models \psi$, then $\rho_{x_\nu}(\varphi) \models \rho_{x_\nu}(\psi)$, for all variables $x \in \mathsf{Var}(\varphi) \cup \mathsf{Var}(\psi)$.

Proof. Suppose the assertion of our lemma is false, viz. $\rho_{x_\nu}(\varphi) \not\models \rho_{x_\nu}(\psi)$. Then we could find an assignment $\vartheta : \mathsf{Var}(\rho_{x_\nu}(\varphi)) \cup \mathsf{Var}(\rho_{x_\nu}(\psi)) \to \{0, 1\}$ with $\vartheta \models \rho_{x_\nu}(\varphi)$ but $\vartheta \not\models \rho_{x_\nu}(\psi)$. Since $\vartheta \models \rho_{x_\nu}(\varphi)$ yields, according to Proposition 3.20, that every arbitrary extension of the assignment ϑ satisfies φ, and as a consequence, also any extension of the form $\{x\} \cup \mathsf{Var}(\rho_{x_\nu}(\varphi)) \cup \mathsf{Var}(\rho_{x_\nu}(\psi))$, which will be denoted by ϑ_x. Consequently, from $\varphi \models \psi$ follows $\vartheta_x \models \psi$, for any extension ϑ_x. Therefore the satisfiability of ψ is independent of setting x, in this way $\vartheta_x \models \rho_{x_\nu}(\psi)$ remains valid, as ψ does not contain tautological clauses. Since $x \notin \mathsf{Var}(\rho_{x_\nu}(\psi))$ holds, yields the assignment $\vartheta \models \rho_{x_\nu}(\psi)$, this contradicts our assumption of $\rho_{x_\nu}(\varphi) \not\models \rho_{x_\nu}(\psi)$ and the proof is complete. □

Now we follow the next step and prove the invariance of CNF formulae under the application of reducts over the assignment set $\mathbb{A}$.

Lemma 3.22.
Let two formulae $\varphi, \psi \in \mathsf{CNF}$ and $X \subseteq \mathsf{Var}(\varphi) \cup \mathsf{Var}(\psi)$ be given. If $\varphi \models \psi$, then $\rho_{X'}(\varphi) \models \rho_{X'}(\psi)$ holds for every set $X' \in \mathbb{A}(X)$.

Proof. Let the formulae $\varphi, \psi \in \mathsf{CNF}$ and $X \subseteq \mathsf{Var}(\varphi) \cup \mathsf{Var}(\psi)$ be given and assume that $\varphi \models \psi$ holds. Let $X' \in \mathbb{A}(X)$ be fixed. In the next step we examine every assignment $\omega_{\widetilde{X'}} : \mathsf{Var}(\rho_{X'}(\varphi)) \cup \mathsf{Var}(\rho_{X'}(\psi)) \to \{0, 1\}$. Note that $\omega_{\widetilde{X'}}$ is defined on the

set $\{\mathsf{Var}(\varphi) \cup \mathsf{Var}(\psi)\} \backslash X'$.
Now we abbreviate the assignment ω, defined by $\omega(x) = 1$ if $x \in X'$ and $\omega(x) = 0$ if $\neg x \in X'$ to $\omega \restriction X'$. In this way $\omega \restriction X'$ agrees with ω on the variables in X'.
Accordingly, since $\varphi \models \psi$, it follows by assumption $\omega \restriction X' \models \neg\varphi \wedge \psi$. By induction we obtain $\omega \restriction X' \models \psi$ if and only if $\omega \models \rho_{X'}(\psi)$, and $\omega \restriction X' \models \varphi$ if and only if $\omega \models \rho_{X'}(\varphi)$. As a consequence the following correlation arises:

$$\omega \models \rho_{X'}(\varphi) \Leftrightarrow \omega \restriction X' \models \varphi \Rightarrow \omega \restriction X' \models \psi \Leftrightarrow \omega \models \rho_{X'}(\psi),$$

which completes the proof. □

Based on Lemma 3.21 and 3.22 we are finally in the position to formulate CNF formulae on trivalent assignment set $\mathbb{T}$. We say a formula is tautological-free, if it does not contain tautological clauses.

Corollary 3.23.
Let two tautological-free formulae $\varphi, \psi \in \mathsf{CNF}$ be given. Furthermore let $X \subseteq \mathsf{Var}(E) \cup \mathsf{Var}(\psi)$ be a set of variables. For every set $X' \in \mathbb{T}(X)$ it holds $\rho_{X'}(\varphi) \models \rho_{X'}(\psi)$ if $\varphi \models \psi$.

For the examination of stable extensions in combination with strong backdoors it is crucial to prove that we preserve any stable extension by application of strong backdoors. For this issue we require further notion.

We denote by y-concl(D, E) the set of conclusions of default rules δ_i, such that y_i is implied by all formulae in E:

$$\text{y-concl}(D, E) := \{C(\delta_i) \mid 1 \leqslant i \leqslant n, \delta_i \in D, E \models y_i\},$$

with the default rule set $D = \{\delta_1, \ldots, \delta_n\}$ and a set of formulae E. Additionally, we extend the set of stable extension $\mathrm{SE}(T_{\mathsf{DL}})$ of the default theory $T_{\mathsf{DL}} = \langle W, D\rangle$ by the set X in following terms:

$$\mathrm{SE}(\langle W, D\rangle, X) := \bigcup_{Y \in \mathbb{T}(X)} \{\mathit{Th}(W \cup \text{y-concl}(D, E)) \mid E \in \mathrm{SE}(\rho_Y(W, D))\}.$$

Finally we are able to prove that we do not lose any stable extension due to the application of strong backdoors.

Lemma 3.24.
Let $T_{\mathsf{DL}} = \langle W, D\rangle$ be a CNF default theory with tautological-free formulae and X be a subset of $\mathsf{Var}(W, D)$. Then $\mathrm{SE}(T_{\mathsf{DL}}) \subseteq \mathrm{SE}(T_{\mathsf{DL}}, X)$.

Proof. Let the default theory $T_{\mathsf{DL}} = \langle W, D\rangle$ be given as stated, with $X \subseteq \mathsf{Var}(W, D)$ and a stable extension E of T_{DL}.
Assume contrary to our claim that there exists a stable extension $E \in \mathrm{SE}(T_{\mathsf{DL}}) \backslash \mathrm{SE}(T_{\mathsf{DL}}, X)$.

We mean by G the set of all generating defaults of E obtained from the stage construction. Now assume w.l.o.g. $G := \{\delta_1, \ldots, \delta_k\}$ and fix the order of applying defaults. In consequence we obtain $E = Th(W \cup \{C(\delta) \mid \delta \in G\})$. In this way $W \models P(\delta_1)$ holds, as $\delta_1 \in G$.
Now we obtain a set $Y \in \mathbb{T}(X)$ agreeing with the set E applying on literals of $\mathsf{Var}(W, D)$, to be more precise $x \in Y$ if $E \models x$ for $x \in \mathsf{Var}(W, D)$ and $\neg x \in Y$ if $E \models \neg x$ and $x_\nu \in Y$ otherwise.
According to Corollary 3.23:

$$\bigwedge_{w \in W} \rho_Y(w) \models \rho_Y(P(\delta_1)).$$

Moreover, we get

$$\bigwedge_{w \in W} \rho_Y(w) \wedge \bigwedge_{1 \leqslant j \leqslant i} \rho_Y(C(\delta_j)) \models \rho_Y(P(\delta_{i+1})),$$

for $i < k$.
Additionally, by Definition 3.12 of $\rho_Y(W, D)$, both the reducts of the knowledge base W, as well as the derived conclusions imply the y_is. From this we obtain

$$\bigwedge_{w \in W} \rho_Y(w) \wedge \bigwedge_{1 \leqslant i \leqslant k} \rho_Y(C(\delta_i)) \models \bigwedge_{1 \leqslant i \leqslant k} y_i.$$

Since neither $E \models P(\delta)$ holds for some default $\delta \in D \backslash G$, nor $E \cup \{C(\delta) \mid \delta \in G\} \models \delta'$ is true for some default $\delta' \in D \backslash G$, it follows E is consistent.
As Y agrees with E on the implied variables of $\mathsf{Var}(W, D)$, we conclude that no additional default rule δ is induced by the reduct of the knowledge base $\rho_Y(W)$ or by $\rho_Y(W \cup \{C(\delta) \mid \delta \in D \backslash G\})$.
Moreover, assume $E \models \neg\beta$ for some $\beta \in \bigcup_{\delta \in G} J(\delta)$, then by Corrollary 3.23 this would also imply that $\rho_Y(E) \models \neg\rho_Y(J)$, so it is legitimate to conclude that no justification is violated. Now assume

$$E' = Th\left(\rho_Y(W) \cup \{\rho_Y(C(\delta)) \mid \delta \in G\}\right)$$

is a stable extension according to the reduct $\rho_Y(W, D)$. But, the set of conclusions of generating defaults G is tied in with y-concl(D, E'), ergo we obtain

$$\begin{aligned} E &= Th(W \cup \{C(\delta) \mid \delta \in G\}) \\ E &= Th(W \cup \text{y-concl}(D, E')) \in \mathrm{SE}(\langle W, D\rangle, X), \end{aligned}$$

contrary to our initial assumption $E \notin \mathrm{SE}(T_{\mathsf{DL}}, X)$, consequently the proof is complete. □

Since we have proven the legitimacy of our concept of backdoors for default logic, we

want to give a formal definition. Let us emphasise here that the set $\mathrm{SE}(T_{\mathsf{DL}}, X)$ is a set of stable extensions *candidates*. We assume for the rest of the chapter, that the formulae are tautological-free.

Definition 3.14: Strong Backdoors for Default Logic

Let a CNF default theory $T_{\mathsf{DL}} = \langle W, D \rangle$, as well as $B \subseteq \mathit{Var}(W, D)$ a set of variables of T_{DL} and a class of formulae $\mathfrak{F}$ be given. We call B as *strong* $\mathfrak{F}$*-backdoor*, where for each $Y \in \mathbb{T}(B)$ the reduct $\rho_Y(W, D)$ is a $\mathfrak{F}$-default theory.

We conclude this subsection with an example, which illustrates the terms defined in this part.

Example 3.25. Let $T_{\mathsf{DL}} := \left\{\{x_1\}, \{\frac{x_1 : x_3}{x_2 \wedge (x_3 \vee x_1)}\}\right\}$ be given.
It is obvious that T_{DL} has a stable extension, namely $E := \mathit{Th}(x_1, x_2 \wedge (x_3 \vee x_1))$.
With the backdoor $B = \{x_1\}$ we obtain the following reduct of the default theory T_{DL}: $\rho_{x_1}(W, D) = \langle \{\top\}, \{\frac{\top : x_3}{y_1}\} \rangle$ and in this way

$$\mathrm{SE}(\rho_{x_1}(W, D)) = \{\mathit{Th}(y_1)\},$$

$$\mathrm{SE}(\rho_{\neg x_1}(W, D)) = \varnothing,$$

$$\mathrm{SE}(\rho_{x_{1_\nu}}(W, D)) = \varnothing.$$

Accordingly, y-concl$(D, \mathit{Th}(y_1)) = \{x_2 \wedge (x_3 \vee x_1)\}$, yields $\mathit{Th}(\{x_2 \wedge (x_3 \vee x_1)\} \cup \{x_1\})$ which is equivalent to the stable extension E of T_{DL}.

3.2.1 The Implication Problem

Before we can investigate the backdoor evaluation problem in default logic, we require some auxiliary results of a subproblem of default logic, namely the implication problem of specific formula classes $\mathfrak{F} \in \{\mathsf{HORN}, \mathsf{KROM}\}$.

Problem:	$\textsc{Imp}(\mathfrak{F})$
Input:	A set Φ of $\mathfrak{F}$-formulae and a formula $\varphi \in \mathfrak{F}$
Question:	Does $\Phi \models \varphi$?

We built upon the important work of Beyersdorff et al. [BTV12], where the authors investigated all Boolean fragments of the implication problem $\textsc{Imp}(\mathfrak{F})$, classifing the computational complexity according to Post's lattice. The formulae in the previous work were restricted to use only some Boolean functions like "$\wedge$", "$\vee$" and "$\neg$". In contrast, for HORN and KROM, the Boolean connectives "$\wedge$", "$\vee$" and "$\neg$" are all allowed, but can only be used in a certain way.

Lemma 3.26.
The implication problem for KROM-formulae IMP(KROM) is in **P**.

Proof. Let a KROM-formulae set Φ and a formula $\varphi \in \mathsf{KROM}$ be given. Suppose w.l.o.g.

$$\bigwedge_{\psi \in \Phi} \varphi = \bigwedge_{i=1}^{m} C_i, \text{ and}$$
$$\varphi = \bigwedge_{i=1}^{n} C'_i.$$

In this way it follows:

$$\langle \Phi, \varphi \rangle \in \textsc{Imp}(\mathsf{KROM}) \Leftrightarrow \left(\bigwedge_{i=1}^{m} C_i, \bigwedge_{i=1}^{n} C'_i \right) \in \textsc{Imp}(\mathsf{KROM}) \tag{3.1}$$
$$\Leftrightarrow \left(\bigwedge_{i=1}^{m} C_i \right) \rightarrow \left(\bigwedge_{i=1}^{n} C'_i \right) \in \textsc{Taut} \tag{3.2}$$
$$\Leftrightarrow \bigwedge_{i=1}^{n} \left(\bigwedge_{j=1}^{m} C_j \rightarrow C'_i \right) \in \textsc{Taut} \tag{3.3}$$
$$\Leftrightarrow \forall\, i \in [n] \left(\bigwedge_{j=1}^{m} C_j \rightarrow C'_i \right) \in \textsc{Taut} \tag{3.4}$$
$$\Leftrightarrow \neg \exists\, i \in [n] \left(\bigwedge_{j=1}^{m} C_j \rightarrow C'_i \right) \notin \textsc{Taut} \tag{3.5}$$

The line (3.1) is the definition of the implication problem IMP. In the next line (3.2) we make use of "$\rightarrow$" the implication function. In line (3.3) we transform the equivalence via $\psi_1 \rightarrow (\psi_2 \wedge \psi_3) \in \textsc{Taut} \Leftrightarrow (\psi_1 \rightarrow \psi_2) \wedge (\psi_1 \rightarrow \psi_3) \in \textsc{Taut}$. Afterwards we separate the TAUT-problem (3.4). And finally the line (3.5) is a double negated statement of line (3.4) it says there does not exist $(\psi_1 \wedge \psi_2) \in \textsc{Taut} \Leftrightarrow \psi_1 \in \textsc{Taut}$ nor $\psi_2 \in \textsc{Taut}$. Now we are in the position to examine the n problems of line (3.5) separately for each $i \in [n]$ by

$$\left(\bigwedge_{i=1}^{m} C_i \rightarrow (\ell \vee \ell') \right) \notin \textsc{Taut} \Leftrightarrow \left(\bigwedge_{i=1}^{m} C_i \right) [\vartheta_0] \in \textsc{Sat}(\mathsf{KROM}),$$

with the assignment ϑ_0, that sets both literals ℓ and ℓ' to zero, viz. $\vartheta_0(\ell) = 0$ and $\vartheta_0(\ell') = 0$. Now it only remains to note that if $\ell \equiv\sim \ell'$ then the implication on the left hand side of the equivalence is always tautological, which completes the proof. □

In the same vein it is possible to show the membership of the implication problem for HORN-formulae in **P**. This complexity result was proved in 1990 by Stillman [Sti90a].

With Lemma 3.26 we are in the position to examine the backdoor evaluation problem in default logic.

3.2.2 Backdoor Set Evaluation in Default Logic

In this section we determine the complexity of the evaluating strong backdoors for the existing stable extension problem into certain fragments, namely either the HORN, KROM, MONOTONE or POSITIVE-UNIT-fragment.

Problem:	EVALEXT($\mathsf{CNF} \to \mathfrak{F}$)
Input:	A CNF-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$, a strong $\mathfrak{F}$-backdoor B where, $B \subseteq \mathsf{Var}(W) \cup \mathsf{Var}(D)$.
Parameter:	$\kappa = \lvert B \rvert$
Question:	Does a stable extension for the CNF-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ exist?

By Lemma 3.24 we obtain candidates for stable extensions. In this way we require further examination to verify those candidates. This leads to some kind of "*extension checking problem*", we abbreviate it with EC. To be more precise, given a default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ and a finite set of formulae Φ, we are interested in whether $Th(\Phi) \in \mathrm{SE}(T_{\mathsf{DL}})$ holds.

The complexity of EC was initially investigated by Rosati in 1999, [Ros99]. He proved EC is $\Theta_2^P = \Delta_2^P[\log]$-complete, i.e., complete for the class of problems decidable in polynomial deterministic time with logarithmically many queries to an **NP** oracle. For detailed information about the complexity class Θ_2^P we refer the reader to the work of Eiter and Gottlob [EG97].

Modifying the algorithm of Rosati [Ros99, Figure 1] to our notion yields the containment in Δ_2^P, see Algorithm 4.

Algorithm 4: Extension checking algorithm [Ros99, Theorem 4]

Input: Set E of formulae and a default theory $T_{\mathsf{DL}} = \langle W, D \rangle$
Output: True if E is a stable extension of T_{DL}, and False otherwise

```
1 D' := ∅
2 forall (P:J)/C ∈ D do
3     if E ⊭ ¬J then D' := D' ∪ {P:/C}
4 E' := W
5 while E' did change in the last iteration do
6     forall P:/C ∈ D' do
7         if E' ⊨ P then E' := E' ∧ C
8 if E ⊨ E' and E' ⊨ E then return true else return false
```

Proposition 3.27 ([Ros99, Figure 1, Theorem 4])**.**
The problem EC is in Δ_2^{p}.

We explain the result in comparison with the extension existence problem. For the Ext-problem we require first to apply the stage construction. In this way we have a non-deterministic guess of generating defaults (with ordering) for the stage construction and afterwards we need to answer a quadratic number of implication questions. In consequence the problem is in $\mathbf{NP}^{\mathbf{NP}}$. In case of EC-problem, we merely need to check the given formulae set and omit the non-deterministic guess of the stage construction. In doing so we obtain a membership in $\mathrm{P}^{\mathbf{NP}}$, which is also known as Δ_2^{p}.

In the following we transfer the complexity result of the upper bound from Σ_2^{p} to para-**NP** via using a strong HORN-backdoor on a CNF-default theory with the size of the backdoor as parameter.

Theorem 3.28.
The problem EvalExt(CNF $\to$ HORN) is in para-**NP**.

Proof. Let the CNF-default theory $T_{\mathsf{DL}} = \langle W, D\rangle$ and a HORN-backdoor set B be given. First we need to compute the reducts of a given HORN-backdoor on the CNF-default theory according to the Trivalent assignment set $\mathbb{T}(X)$, to transform the CNF-default theory to a HORN-default theory. Subsequently, we need to guess non-deterministically the set of generating defaults G and by this the candidates for stable extensions, for all of the $|\mathbb{T}(X)| = 3^{|B|}$ reducts. Ensuing we verify the candidates for stable extension $W \wedge \bigwedge_{g\in G} g$ via Algorithm 4. We know from Stillman [Sti90a], that the implication problem for HORN-formulae is tractable. Accordingly the verification of the given stable extension candidates is tractable as well. Since we have verified that E, is a stable extension of the reduced default theory $\rho_Y(T_{\mathsf{DL}})$, it is obligatory to compute the according set E' of stable extensions for the initial CNF-T_{DL} default theory. This is done by computing $E \vDash y_i$, for $1 \leqslant i \leqslant |D|$. Finally we need to confirm the validity of E' via Algorithm 4. By Lemma 3.26 we know that the implication problem parameterized on the size of a backdoor is fixed-parameter tractable. As the length of formulae is bounded by the input size and the parameter is $\kappa = |B|$, yields fixed parameter tractability for this computation. Overall we obtain a para-**NP** algorithm and the theorem applies.

Algorithm 5: Generic algorithm for EvalExt($\mathfrak{F} \to \mathfrak{F}'$)

Input: $\mathfrak{F}$-default theory $T_{\mathsf{DL}} = \langle W, D\rangle$, backdoor $B \subseteq \mathsf{Var}(W, D)$
1 **for** $Y \in \mathbb{T}(X)$ **do**
2 construct set of generating defaults G for $\mathfrak{F}'$ default theory $\rho_Y(T_{\mathsf{DL}})$
3 **if** $E := \bigwedge_{w\in\rho_Y(W)} w \wedge \bigwedge_{\frac{P:J}{C}\in G} C$ *is extension for* $\rho_Y(T_{\mathsf{DL}})$ **then**
4 $E' := \bigwedge_{w\in W} \omega \wedge \bigwedge_{c\in \text{y-concl}(D,E')} c$ **if** E' *is extension for* T_{DL} **then**
  return true
5 **return** false

We recap the proof steps with a generic algorithm in pseudocode, see Algorithm 5.

□

The same conclusion can be drawn for the EvalExt problem from CNF to KROM formulae.

Theorem 3.29.
The problem EvalExt(CNF → KROM) is in para-**NP**.

Proof. Let the CNF-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ be given. We know from Lemma 3.26 that Imp(KROM) $\in$ **P**. The rest of the proof can be handled the same way as in Theorem 3.28 by constructing a para-**NP** algorithm.

□

Theorem 3.30.
The problem EvalExt(CNF → MONOTONE) is in para-Δ_2^{p}.

Proof. Let the CNF-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ be given. The next preliminary consideration is crucial for a sine qua non of the justification due to the stage construction, viz. $\neg J \notin Th(W \cup E)$. Observe that the formulae class MONOTONE allows positive literals only, in this way the negation of a formula φ is not monotone, unless $\varphi \in \{\top, \bot\}$. In the case of $\varphi \notin \{\top, \bot\}$ we remove the corresponding justification, as it could not be inferred anyway. On the other hand, in case of $\varphi \in \{\top, \bot\}$, we obtain that φ is always applicable in an inconsistent way only. Consequently, this case discrimination is done in polynomial time. Now bear in mind that if any stable extension exists, then it is a unique stable extension, due to the reasoning above. Accordingly we have to compute quadratically many implication questions and as Imp(MONOTONE) $\in$ **coNP**, it follows that the construction of the set of generation defaults G, and also the extension is done in para-Δ_2^{p}. The verification of an extension from Algorithm 5, line 4 can be used, as we construct a para-Δ_2^{p} algorithm, completing the proof.

□

Theorem 3.31.
The problem EvalExt(CNF → POSITIVE-UNIT) is in **FPT**.

Proof. Let the CNF-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ and a POSITIVE-UNIT backdoor B be given. Assume Algorithm 4 runs in polynomial time, for the given default theory T_{DL} and the backdoor B, as described in the theorem. For this reason we can apply similar arguments as in Theorem 3.28 and as the backdoor size is bounded by κ we obtain precisely the assertion of the theorem. Consequently, it remains to show Algorithm 4 runs in polynomial time. The authors Beyersdorff et al. [BMTV09] investigated the implication problem for POSITIVE-UNIT formulae using AND-gates only. They proved that this implication problem is in the circuit complexity class $\mathbf{AC^0}$, corresponding to constant-depth, unbounded fan-in, polynomial-size circuits with AND-gates.[Vol99]. In

this way we can adapt this result to the implication query of Algorithm 4, and achieve a polynomial time running algorithm, and the theorem applies.

□

We proceed to show for that general extension existence problem for KROM formulae, without using backdoors is **NP**-complete.

Theorem 3.32.
The Ext(KROM)-problem is **NP**-complete.

Proof. Let the KROM-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ be given. As we already know from Lemma 3.26, the implication problem for KROM-formulae is in **P**. In combination with the result of Rosati, [Ros99] we obtain that the extension checking problem of KROM-default theories is in **P** as well. For the proof of membership in **NP** observe that Algorithm 4 guesses the set of generating defaults $G \subseteq D$ and afterwards verifies whether $W \wedge \bigwedge_{\frac{P:J}{C} \in G} C$ is a valid extension of a given KROM-T_{DL}. This is done in **NP**.
The default theory constructed by Beyersdorff et al. [BTV12, Lemma 5.6] uses KROM-formulae only. With the reduction from 3 Sat, which is known to be **NP**-complete, we obtain the desired lower bound and consequently the assertion of the theorem.

□

3.2.3 Backdoor Set Detection in Default Logic

The aim of this section is to examine the problem of detection of backdoors of size $|B| \leqslant \kappa$ for a given CNF-default theory.

Problem:	BdDetection($\mathsf{CNF} \to \mathfrak{F}$)
Input:	A CNF-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$
Parameter:	$\kappa \in \mathbb{N}_{\geqslant 0}$
Question:	Does T_{DL} have a strong $\mathfrak{F}$-backdoor B of a size $\lvert B \rvert \leqslant \kappa$?

From [DF13] and [Sch81] we can assume for clause-induced target classes $\mathfrak{F}$, that we can utilise a decision algorithm for BdDetection($\mathsf{CNF} \to \mathfrak{F}$) to determine a backdoor B using self-reduction. Therefore we can formulate the following lemma.

Lemma 3.33.
BdDetection($\mathsf{CNF} \to \mathfrak{F}$) $\in$ **FPT**, yields that computing of a strong $\mathfrak{F}$-backdoor B with $|B| \leqslant \kappa$ is fixed-parameter tractable.

Proof. We perform the proof via induction on κ. Let the conditions of the lemma be fulfilled. We start with the basis $\kappa = 0$, i.e. we need a backdoor of a size $|B| = 0$. Here we immediately see that the statement holds. Now for the inductive step let $\kappa > 0$. We proceed to verify for all $v \in \mathsf{Var}(W \cup D)$ according to the trivalent assignment $\mathbb{T}$ for

the given T_{DL} and the parameter κ, whether $\rho_v(W, D)$, $\rho_{\neg v}(W, D)$ and $\rho_{v_\nu}(W, D)$ have a strong $\mathfrak{F}$-backdoor B of size $|B| \leqslant \kappa - 1$. If the default theory T_{DL} has no strong $\mathfrak{F}$-backdoor B of size $|B| \leqslant \kappa - 1$ for all $v \in \mathsf{Var}(W \cup D)$, then we can conclude that there is no $\mathfrak{F}$-backdoor B of size $|B| = \kappa$. On the other hand, if we have found a $\mathfrak{F}$-backdoor B of size $|B| \leqslant \kappa - 1$ for some $v \in \mathsf{Var}(W \cup D)$, then by induction hypothesis we can compute $\rho_v(W, D)$, $\rho_{\neg v}(W, D)$ and $\rho_{v_\nu}(W, D)$ and $|B| \cup v$ is the desired strong $\mathfrak{F}$-backdoor of the default theory $T_{\mathsf{DL}} = \langle W, D \rangle$. This finishes the proof.

□

Now we want to demostrate how to find a strong $\mathfrak{F}$-backdoor B for $\mathfrak{F} \in \{\mathsf{HORN}, \mathsf{KROM}, \mathsf{MONOTONE}, \mathsf{POSITIVE\text{-}UNIT}\}$ in fpt-time.

Theorem 3.34.
Let $\mathfrak{F} \in \{\mathsf{HORN}, \mathsf{KROM}, \mathsf{MONOTONE}, \mathsf{POSITIVE\text{-}UNIT}\}$, then BdDetection($\mathsf{CNF} \to \mathfrak{F}$) is fixed-parameter tractable.

Proof. Let the conditions of the theorem be fulfilled. Let us denote by F the set $W \cup \{P(\delta), J(\delta), C(\delta) \mid \delta \in D\}$. For the construction of a strong $\mathfrak{F}$-backdoor it is sufficient to consider the case $Y = \{x_\nu \mid x \in X\}$, as each class of $\mathfrak{F} \in \{\mathsf{HORN}, \mathsf{KROM}, \mathsf{MONOTONE}, \mathsf{POSITIVE\text{-}UNIT}\}$ is clause-induced and $\rho_Z(\varphi) \subseteq \rho_Y(\varphi)$ holds for any $Z \in \mathbb{T}(X)$. From now on we make the assumption $Y = \{x_\nu \mid x \in X\}$ and start with the case BdDetection($\mathsf{CNF} \to \mathsf{MONOTONE}$):
A monotone formula φ contains positive literals only. Therefore it is straightforward to take all negative literals of the formula $\varphi \in F$ to the strong backdoor B, to obtain the desired $\mathsf{MONOTONE}$ backdoor set. This is possible in linear time.
For the remaining cases $\mathsf{HORN}, \mathsf{KROM}$, and $\mathsf{POSITIVE\text{-}UNIT}$ we tie up on the constructions inspired by [SS09] for the propositional logic and adapt it to default logic.

For HORN let $G^{\mathsf{HORN}}_{T_{\mathsf{DL}}}(V, E)$ denote a graph with a set of variables $V = \mathsf{Var}(F)$ as nodes. The nodes v_1 and v_2 with $v_1 \neq v_2$ are connected by an edge iff v_1 and v_2 occur in a same clause $\{v_1, v_2\} \subseteq \mathsf{Var}(C)$, for $C \in \varphi$ and $\varphi \in F$.
For $\mathsf{POSITIVE\text{-}UNIT}$ let $G^{\mathsf{PU}}_{T_{\mathsf{DL}}}(V, E)$ denote a graph with a set of variables $V = \mathsf{Var}(F)$. The variables v_1 and v_2 with $v_1 \neq v_2$ are connected by an edge iff the literals ℓ_{v_1} and ℓ_{v_2} occur in a same clause C, where $\ell_{v_1} = \{v_1, \neg v_1\}$ and $\ell_{v_2} = \{v_2, \neg v_2\}$ for $C \in \varphi$ and $\varphi \in F$.
And finally for the formulae class KROM let $H^{\mathsf{KROM}}_{T_{\mathsf{DL}}}(V, E)$ denote a hypergraph with a set of variables $V = \mathsf{Var}(F)$ as vertices. The variables v_1, v_2 and v_2, with $v_1 \neq v_2 \neq v_3$ are connected by a hyperedge if v_1, v_2 and v_3 occur in the same clause $\{v_1, v_2, v_3\} \subseteq \mathsf{Var}(C)$, for $C \in \varphi$ and $\varphi \in F$.
Accordingly, we represent the given default theory by an appropriate graph and claim that set $B \subseteq \mathsf{Var}(F)$ is the desired strong $\mathfrak{F}$-backdoor of T_{DL}, if and only if B represents a λ-hitting set, with $\lambda \in \{2, 3\}$, of the appropriate graph representation of the default theory T_{DL}. In doing so λ is depending on the corresponding formulae class, $\lambda = 2$ for the formulae classes HORN und $\mathsf{POSITIVE\text{-}UNIT}$ and $\lambda = 3$ for KROM.

For the case $\lambda = 2$ we introduce the vertex cover problem of the graphs $G^{\mathsf{HORN}}_{T_{\mathsf{DL}}}(V, E)$ and $G^{\mathsf{PU}}_{T_{\mathsf{DL}}}(V, E)$. The graph $G^{\mathsf{HORN}}_{T_{\mathsf{DL}}}(V, E)$, or $G^{\mathsf{PU}}_{T_{\mathsf{DL}}}(V, E)$ respectively, has a vertex cover $S \subseteq V$, if for every edge $v_1v_2 \in E$ we have $\{v_1, v_2\} \cap S \neq \varnothing$. We know from [CKX10a] that we can find such vertex cover set of a size $|S| \leqslant \kappa$, if exists, in time $\mathcal{O}(1.2738^{\kappa} + \kappa \cdot n)$. For KROM we introduce the 3-hitting set problem of the hyper graph $H^{\mathsf{KROM}}_{T_{\mathsf{DL}}}(V, E)$, with $|E| \leqslant 3$. The hyper graph $H^{\mathsf{KROM}}_{T_{\mathsf{DL}}}(V, E)$ has a 3-hitting set $S \subseteq V$, if for every hyperedge we have $S \cap e \neq \varnothing$. By [Fer10] we can find such a 3-hitting set of a size $|S| \leqslant \kappa$, if exists, in time $\mathcal{O}(2.179^{\kappa} + n^3)$.

For the correctness proof we demonstrate the straightforward direction first. For this let B be a λ-hitting set of an appropriate graph representation, as discussed above. Let C be a clause in $\rho_Y(\varphi)$ for $\varphi \in F$. For the sake of contradiction, assume that the clause C be $C \notin \mathsf{HORN}$ or $C \notin \mathsf{POSITIVE\text{-}UNIT}$ or $C \notin \mathsf{KROM}$, in correspondence to λ. Consequently, there is a set $S \subseteq C$ and an edge S of the appropriate graph with $S \cap X = \varnothing$, which contradicts the assumption of B being a λ-hitting set of appropriate graph representation. Now let $B \subseteq \mathsf{Var}(F)$ be a strong $\mathfrak{F}$-backdoor of $T_{\mathsf{DL}} = \langle W, D \rangle$. We consider an edge $v_1v_2 \in E$ (or $v_1v_2v_3 \in E$) of the appropriate graph. The graphs $G^{\mathsf{HORN}}_{T_{\mathsf{DL}}}(V, E)$, $G^{\mathsf{PU}}_{T_{\mathsf{DL}}}(V, E)$ and $H^{\mathsf{KROM}}_{T_{\mathsf{DL}}}(V, E)$ have per design a corresponding clause $C \in \varphi$, for $\varphi \in F$ and $v_1, v_2 \in C$ (or $\{v_1, v_2, v_3\} \subseteq C$ for KROM). Since we construct the reduct $\rho_Y(\varphi)$ of φ by removing all occurrences of literals $\neg x$ and x from clauses in φ and each clause contains at most one positive literal in case of HORN, exactly one positive literal in case of $\mathsf{POSITIVE\text{-}UNIT}$ or two literals in case of KROM, we establish $\{v_1, v_2\} \cap X = \varnothing$ (or $\{v_1, v_2, v_3\} \cap X = \varnothing$, respectively). This yields that B is a λ-hitting set and the theorem applies.

□

By combining Theorem 3.34 with Theorem 3.28 and Theorems 3.29 and 3.30 we are finally able to enhance the results by the premise of a given strong backdoor.

Corollary 3.35.
Let a CNF-default theory $T_{\mathsf{DL}} = \langle W, D \rangle$ be given, further let $\mathfrak{F} = \{\mathsf{HORN}, \mathsf{KROM}\}$, then

$$\textsc{EvalExt}(\mathsf{CNF} \to \mathfrak{F}) \in \textbf{para-NP}$$

$$\textsc{EvalExt}(\mathsf{CNF} \to \mathsf{MONOTONE}) \in \textbf{para-}\Delta_2^{\mathrm{p}}$$

$$\textsc{EvalExt}(\mathsf{CNF} \to \mathsf{POSITIVE\text{-}UNIT}) \in \textbf{FPT}$$

Theorem 3.36.
The problem $\textsc{EvalExt}(\mathsf{CNF} \to \mathsf{HORN})$ is **para-NP**-complete.

Proof. By Corollary 3.35 we know that the upper bound of $\textsc{EvalExt}(\mathsf{CNF} \to \mathsf{HORN})$ is **para-NP**. To show the lower bound we make use of a reduction from $\textsc{Ext}(\mathsf{HORN})$, which is known from Stillman to be **NP**-complete [Sti90a]. Therefore let the reduction function f be given as

$$f : (\textsc{Ext}(\mathsf{HORN}), |B| = 0) \to \textsc{EvalExt}(\mathsf{HORN} \to \mathsf{HORN}, |B| = 0).$$

Formulae Class $\mathfrak{F}$	EvalExt(CNF $\to \mathfrak{F}$)	BdDetection(CNF $\to \mathfrak{F}$)
HORN	para-**NP**-complete	in **FPT**
KROM	in para-**NP**	in **FPT**
MONOTONE	in para-Δ_2^{p}	in **FPT**
POSITIVE-UNIT	in **FPT**	in **FPT**

Table 3.2: Computational Complexity Results for EvalExt(CNF $\to \mathfrak{F}$) and BdDetection(CNF $\to \mathfrak{F}$)

Now it holds there exists a stable extension of a given HORN default theory if and only if there exists a stable extension for EvalExt(HORN $\to$ HORN, $|B| = 0$). Obviously the function f is computable in polynomial time and the theorem applies.

□

We summerize our results of this section in Table 3.2.

CHAPTER 4

PARAMETRIZATION IN TEMPORAL LOGICS

4.1 Temporal Logics on Syntax graph Representations with Bounded Treewidth

Here we want to classify full branching time tree logic ($\mathcal{CTL}^\star$) and its fragments computational tree logic ($\mathcal{CTL}$) and linear temporal logic ($\mathcal{LTL}$) in terms of parameterized complexity of the satisfiability problem. Therefore we will introduce two graph-like representations of formulae, namely syntax circuit and syntax tree. To show the fixed-parameter tractability results we will generalize the prominent theorem of Courcelle to work on infinite vocabularies. For the parametrization we will use temporal depth and treewidth or path-width. We will see a dichotomy between $\mathbf{W}[1]$-hard and fixed-parameter tractable operator fragments. At the end of this section we will investigate Boolean operator fragments according to the Post's lattice.

4.1.1 Structural Representations of Formulae

By $\mathrm{SF}(\varphi)$ we mean the set of all subformulae of φ including φ itself.

Definition 4.1: Syntax Circuit

Let B denote a finite set of Boolean functions and T denote a set of temporal operators. The *syntax circuit* of a formula φ is denoted by $\mathcal{C}_\varphi$ is defined as the structure $(\mathrm{SF}(\varphi), \tau^{\mathcal{C}_\varphi})$, with the set of subformulae of φ and the vocabulary

$$\tau := \{\mathrm{repr}^1_{\mathsf{O}}, \mathrm{body}^{n+1}_{\mathsf{O}} \mid \mathsf{O} \in T, \mathrm{ar}(\mathsf{O}) = n\} \cup \{\mathrm{repr}^1_f, \mathrm{body}^{n+1}_f \mid f \in B, \mathrm{ar}(f) = n\} \cup \{\mathrm{var}^1\},$$

where the relations of τ are interpreted as follows:

$\mathrm{var}^1(x)$	$\Leftrightarrow$	x represents a variable,
$\mathrm{repr}^1_{\mathsf{O}}(x)$	$\Leftrightarrow$	x represents a formula $\mathsf{O}(\psi_1, \ldots, \psi_n)$ for some n and $\mathsf{O} \in T$,
$\mathrm{body}^{n+1}_{\mathsf{O}}(x, y_1, \ldots, y_n)$	$\Leftrightarrow$	x represents the formula $\mathsf{O}(\psi_1, \ldots, \psi_n)$, with $\psi_1, \ldots, \psi_n$ are represented by $y_1, \ldots, y_n$ and $\mathsf{O} \in T$ is a temporal operator of arity n,
$\mathrm{repr}^1_f(x)$	$\Leftrightarrow$	x represents a formula $f(\psi_1, \ldots, \psi_n)$ for some n and $f \in B$,
$\mathrm{body}^{n+1}_f(x, y_1, \ldots, y_n)$	$\Leftrightarrow$	x represents the formula $f(\psi_1, \ldots, \psi_n)$ where $\psi_1, \ldots, \psi_n$ are represented by $y_1, \ldots, y_n$ and $f \in B$ is a Boolean function of arity n.

For simplicity of notion we want to formulate some specific properties of syntax circuits in the following way:

- y is the i-th argument of x:
  $\mathrm{body}^2_{R,i}(x, y) := \exists y_1 \ldots \exists y_{\mathrm{ar}(R)} \mathrm{body}^{\mathrm{ar}(R)+1}_R(x, y_1, \ldots, y_n) \wedge y_i = y$
- y is some argument of x:
  $\mathrm{child}^2(x, y) := \bigvee_{R \in T \cup B} \bigvee_{1 \leqslant i \leqslant \mathrm{ar}(R)} \mathrm{body}^2_{R,i}(x, y)$
- x is the root of the structure:
  $\mathrm{repr}^1(x) := \forall y \neg \mathrm{child}^2(x, y)$

To take a concrete example, we present in Figure 4.1 a illustration of a $\mathcal{CTL}^\star$ formula $\varphi = \mathsf{AG}(x_1 \mathsf{U}(\mathsf{EX} x_2)) \wedge (x_2 \mathsf{R}(\mathsf{AF}(x_3)))$ as a syntax circuit $\mathcal{C}_\varphi$.

Similar to the syntax circuit $\mathcal{C}_\varphi$, we define a *syntax tree* $\mathcal{S}_\varphi$, but in lieu of a set of subformulae $\mathrm{SF}(\varphi)$ we will use a set of *ordered subformulae* $\mathrm{SF}_O(\varphi)$ to generate individuals. In contrast to subformulae we make the multiple occurrences of a same non-atomic subformula identifiable, by marking each symbol $\mathsf{O} \in T \cup B$ as $\mathsf{O}', \mathsf{O}'', \mathsf{O}''', \ldots$.

As no subformula occurs multiple times in φ from Figure 4.1, in this case it holds that $\mathcal{S}_\varphi = \mathcal{C}_\varphi$.

Definition 4.2: Formula Treewidth and Formula Pathwidth

For a temporal formula φ, we say the *circuit treewidth* denoted by $\mathrm{tw}_{\mathcal{C}}(\varphi)$ is the treewidth of the syntax circuit of formula φ and the *syntax treewidth* denoted by $\mathrm{tw}_{\mathcal{S}}(\varphi)$ is the treewidth of the syntax tree of φ.

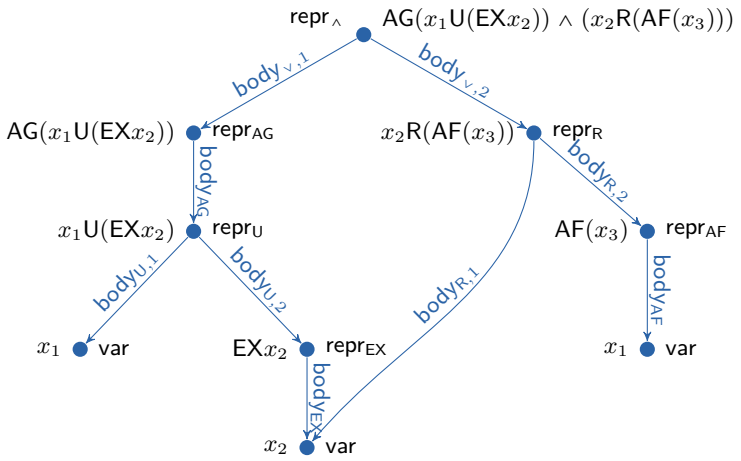


Figure 4.1: Example syntactical circuit $\mathcal{C}_\varphi$ as relational structure.

The *path-width* of the appropriate graph representation of the temporal formula φ is defined analogously.

In the next section we want to examine the parameterized satisfiability problem.

4.1.2 Fixed-Paremeter Tractable Fragments

In this section we want to prove that the satisfiability of $\mathcal{CTL}$-formulae with an appropriate parametrization is fixed-parameter tractable. For this use the well-known result: Courcelle's Theorem [CE12, Thm. 6.3 (1)]. To apply the theorem, we need to formulate the satisfiability of $\mathcal{CTL}$-formulae in MSO. Unfortunately we require a family of MSO formulae, which depend on the instance. For this reason we want to present a generalized version of Courcelle's Theorem for infinite sized signature under appropriate constraints and offer in this way new approaches for research.

4.1.2.1 Courcelle's Theorem with Infinite Signature

The initial version of the theorem states, that every MSO-definable decision problem (Q, κ) parameterized by the treewidth $\kappa = \mathrm{tw}(\mathcal{A}_x)$ over some relational structure $\mathcal{A}_x$, with an instance $x \in Q$, is fixed-parameter tractable. This theorem is restricted to finite signatures. The next theorem outline how to deploy Courcelle's Theorem even for infinite structures, when the problem is formulated over an infinite but uniform family of MSO-formulae $(\varphi_n)_{n\in\mathbb{N}}$.
Consider, for describing the set of all structures $\mathfrak{A}$ resulting to the appropriate family of MSO-formulae $(\varphi_n)_{n\in\mathbb{N}}$ expect a infinitely sized signature τ. Every subset $\mathcal{A} \subset \mathfrak{A}$ of structures corresponding to each φ_i have a finite signature.

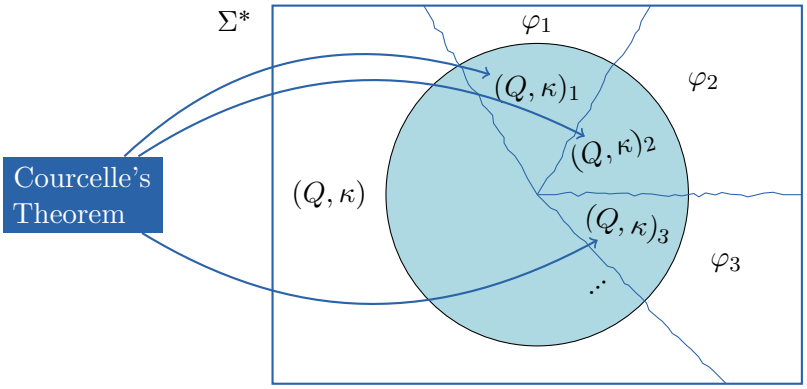


Figure 4.2: Infinite application of Courcelle's Theorem to each slice $(Q,\kappa)_{i\in\mathbb{N}}$, viz. $(Q,\kappa)_i := \{x \in \Sigma^* | x \in Q \text{ and } \kappa(x) = i\}$.

Theorem 4.1.
Let (Q,κ) be a parameterized problem such that instances $x \in \Sigma^*$ can be efficiently transformed to relational structures $\mathcal{A}_x$ over a (possibly infinite) signature τ and $\text{tw}(\mathcal{A}_x)$ is κ-bounded. If there exists a uniform MSO-formula family $(\varphi_n)_{n\in\mathbb{N}}$ and an fpt-computable, κ-bounded function f such that for all $x \in \Sigma^*$ it holds $x \in Q \Leftrightarrow \mathcal{A}_x \vDash \varphi_{|f(x)|}$ then (Q,κ) is fixed-parameter tractable.

Proof. Let the conditions of the theorem be fulfilled. We prove the theorem by presenting an algorithm $\mathscr{A}$, which runs in fpt-time and decides the parameterized problem (Q,κ) correctly. For a given instance $x \in Q^*$ we compute the size $i := |f(x)|$ in **FPT**. As the family of MSO-formulae $(\varphi_n)_{n\in\mathbb{N}}$ is uniform and with a κ- bounded function f we compute φ_i in time $g(i) = g(f(x)) \leqslant g(h(\kappa(x)))$, consequently in **FPT** where g is recursive and non-decreasing function. In the next step, we can solve the model checking problem instance $(\mathcal{A}_x, \varphi_i)$ in time $f'(\text{tw}(\mathcal{A}_x), |\varphi_i|) \cdot |\mathcal{A}_x|$ with respect to Courcelle's Theorem, where f' is a recursive function. For this reason we can conclude $\mathscr{A} \in$ **FPT**, as both the treewidth tw and $|\varphi_i|$ are κ-bounded.

□

Figure 4.2 illustrates the main idea of Theorem 4.1.

Praveen [Pra13] utilised a kind of a special case of Theorem 4.1, while he used a **P**-uniform MSO-family to show the fixed parameter tractability of ML-SAT parameterized by modal depth and path-width. He formulated each modal formula φ as an MSO-formula, whose length is linear in the modal depth of φ.

4.1.2.2 Design of MSO-Temporal Formulae

A classical model in temporal logic requires one world where the formula is true. It is quite challenging to built an MSO-temporal formula with the approach of the classical models. For this reason we will use the prominent procedure, which is used for computation of upper-bounds for model size, so-called *quasi-models*. The intuition of quasi-models is roughly speaking, that it transforms the statement "it holds" in "it is labeled". As we will see later, the inductive conditions of the quasi-labels reproduce exactly the semantics of truth for $\mathcal{CTL}^\star$. In consequence, we do not need to discourse about the *truth* of a subformula.

For the definition of the subformula we require first the definition of *closure* related to the Ladner-Fischer closure denoted by $\mathrm{cl}(\varphi)$ which was designed for propositional dynamic logic [FL79a]. It is crucial to know that $\mathrm{cl}(\varphi)$ is not necessarily a subset of $\mathcal{CTL}^\star(B,T)$ and consequently we do not care about wether $\neg \in B$.

Definition 4.3: Closure

Let B be a finite set of Boolean functions and T a set of temporal operators. Let φ be a $\mathcal{CTL}^\star(B,T)$ formula. Then define $\sim\psi := \xi$ if $\psi = \neg\xi$ for some ξ, and $\sim\psi := \neg\psi$ otherwise. Further define $\overline{\mathsf{A}} := \mathsf{E}$, $\overline{\mathsf{E}} := \mathsf{A}$, $\overline{\mathsf{F}} := \mathsf{G}$, $\overline{\mathsf{G}} := \mathsf{F}$, $\overline{\mathsf{U}} := \mathsf{R}$, $\overline{\mathsf{R}} := \mathsf{U}$, and $\overline{\mathsf{X}} := \mathsf{X}$. Set $P := \{\mathsf{A}, \mathsf{E}\}$. Now the *closure* $\mathrm{cl}(\varphi)$ of φ is the smallest set with the following properties:

- $\varphi \in \mathrm{cl}(\varphi)$.
- If $O \in T \cup P$, $O\psi \in \mathrm{cl}(\varphi)$, then $\overline{O}\sim\psi, \psi \in \mathrm{cl}(\varphi)$.
- If $O \in T$, $\psi O \xi \in \mathrm{cl}(\varphi)$, then $\sim\psi\overline{O}\sim\xi, \psi, \xi \in \mathrm{cl}(\varphi)$.
- If $f(\psi_1, \ldots, \psi_n) \in \mathrm{cl}(\varphi)$, $f \in C$, then $\psi_1, \ldots, \psi_n \in \mathrm{cl}(\varphi)$.
- $\psi \in \mathrm{cl}(\varphi)$ iff $\sim\psi \in \mathrm{cl}(\varphi)$.

Finally we are in the position to define *quasi-models*.

Definition 4.4: Quasi-Models

Let $\varphi \in \mathcal{CTL}^\star(B, \{\mathsf{A}, \mathsf{E}, \mathsf{X}\})$. A *quasi-model* of φ is a tuple $K = (W, R, L, L_\mathsf{A}, L_\mathsf{E})$ where (W, R) is a Kripke frame and $L, L_\ominus \colon W \to 2^{\mathrm{cl}(\varphi)}$ for $\ominus \in \{\mathsf{A}, \mathsf{E}\}$ are *extended labelling functions* with the following *quasi-label conditions*:

(1.) If $L(w)$ ($L_\ominus(w)$,resp.) contains $f(\psi_1, \ldots, \psi_n)$, resp. $\sim f(\psi_1, \ldots, \psi_n)$ for some n-ary $f \in B$, then there is a Boolean assignment ϑ s.t. $\vartheta \vDash f$, resp., $\vartheta \nvDash f$ and f.a. $1 \leqslant i \leqslant n$ holds $\vartheta(i) = 1 \to \psi_i \in L(w)$ (resp., in $L_\ominus(w)$) and $\vartheta(i) = 0 \to \sim\psi_i \in L(w)$ (resp., in $L_\ominus(w)$).

(2.) If $\psi \in L_{\bigcirc}(w)$ is a state formula, then $\psi \in L(w)$.

(3.) If $\psi, \neg\xi \in L(w)$ ($L_{\bigcirc}(w)$), then $\psi \neq \xi$.

(4.) If $\neg \mathsf{X}\psi \in L_{\bigcirc}(w)$, then $\mathsf{X}{\sim}\psi \in L_{\bigcirc}(w)$.

(5.) If $\mathsf{E}\psi \in L(w)$ ($\mathsf{A}\psi \in L(w)$,resp.), then $\psi \in L_{\mathsf{E}}(w)$ ($\psi \in L_{\mathsf{A}}(w)$,resp.).

(6.) If $\mathsf{X}\psi \in L_{\mathsf{A}}(w)$ (resp., in $L_{\mathsf{E}}(w)$), then for every (resp., some) successor w' of w it is $\psi \in L_{\mathsf{A}}(w')$ (resp., in $L_{\mathsf{E}}(w')$).

(7.) $\varphi \in L(w)$ for some $w \in W$.

$L_{\mathsf{A}}(w)$ is the *universal quasi-label* of a world w and $L_{\mathsf{E}}(w)$ the *existential quasi-label*. The numbers (1.) to (5.) are the *local* quasi-label conditions. L is the *local quasi-label* which contains only state formulae.

To show the coherence of a temporal formula and a quasi-model, we first need to define the notion of *depth in quasi-models*.

Definition 4.5: Depth of a Quasi-Model

Let $K = (W, R, L, L_{\mathsf{A}}, L_{\mathsf{E}})$ be a quasi-model, with $w_0 \in W$. Then the (K, w_0) is a rooted quasi-model with a root w_0. We call the maximal distance of a world from the root w_0 the *depth of a quasi-model*.

We say that a quasi-model is *tree-like*, if its root has no predecessor, every other node has exactly one predecessor and a *leaf* is a node with itself as the only successor.

The next lemma describes the correspondence of a temporal formula and tree-like model.

Lemma 4.2.
Let $\varphi \in \mathcal{CTL}^{\star}(B, \{\mathsf{A}, \mathsf{E}, \mathsf{X}\})$ for a finite set B of Boolean functions. Then φ is satisfiable if and only if it has a serial tree-like quasi-model of depth $\mathrm{td}(\varphi)$.

Proof. The lemma can be proven similar to the approach of Blackburn et al. [BdV01]. They showed for the tree model property of a modal logic, that every model can be successively unwinded to an infinite tree. The approach for the transformation is described below.
As we mentioned before the inductive conditions for quasi-labels replicate the semantics of truth for $\mathcal{CTL}^{\star}$. To be more concrete, the label set $L_{\mathsf{A}}(w)$ corresponds to the formulae that should hold in all paths π starting at w, and the label set $L_{\mathsf{E}}(w)$ corresponds to the formulae that have to hold in at least one path starting at w.
If such a formula starts with a path-operator $P \in \{\mathsf{A}, \mathsf{E}\}$, it is a state formula and consequently contained in $L(w)$. Then it is also contained in the proper set L_A, resp., L_{E}, but has to fullfil no further (truth-)conditions. If a formula starts with X, then it is inherited to one or more successors of w, and its type $P \in \{\mathsf{A}, \mathsf{E}\}$ does not change.

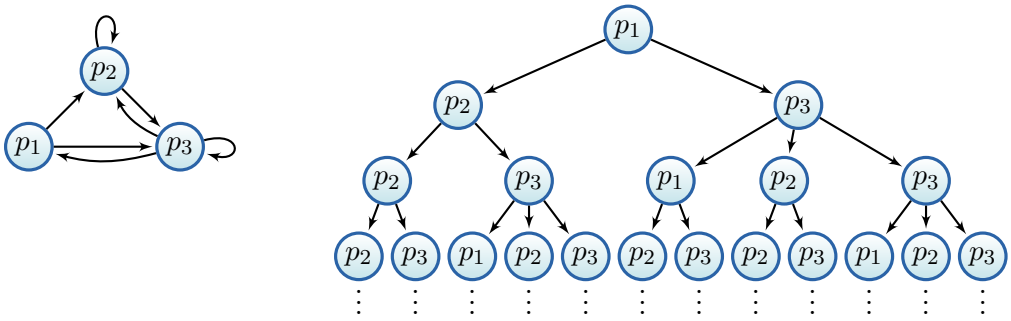


Figure 4.3: Transformation of a model to a treelike quasi-model

This property holds for Boolean functions as well. They are parsed without changing its type. Finally, for the satisfiability of φ the existence of some quasi label containing φ is required. It can be proven by induction over the nesting of X, that in minimal quasi-models the sets $L(w)$, $L_{\mathsf{A}}(w)$, and $L_{\mathsf{E}}(w)$ of worlds w of depth $\mathrm{td}(\varphi)$ can only contain X-free formulae and therefore the successors can be replaced by a self-loop. The transformation from a quasi-model to a model and vice versa is straightforward: If the labels L_{A} and L_{E} are removed and from L all labeled formulae except propositions are dropped, then we obtain a model from a quasi-model. The other way around, to enrich a model K to a quasi-model, set $L_{\ominus}(w) := \{\ \psi \mid \psi \in \mathrm{cl}(\varphi) \text{ and } K, w \vDash \ominus\psi\ \}$ for $\ominus \in \{\mathsf{A}, \mathsf{E}\}$ and $L(w) := \{\ \psi \mid \psi \in \mathrm{cl}(\varphi) \text{ and } K, w \vDash \psi\ \}$. □

Figure 4.3 illustrates the transformation of a model to a tree-like quasi-model.

Finally under the application of quasi-models, we want to express the satisfiability of $\mathcal{CTL}^\star$-formula in MSO, which is crucial for the application of Courcelle's Theorem, and consequently to show fixed parameter tractability.

Lemma 4.3.
Let B be a finite set of Boolean functions. There is a computable function $f_B : n \to \theta_n$, where $n \in \mathbb{N}$ and θ_n is an MSO-formula, s.t. the following holds: For all $\varphi \in \mathcal{CTL}^\star(B, \{\mathsf{A}, \mathsf{E}, \mathsf{X}\})$ with $\mathrm{td}(\varphi) \leqslant n$, φ is satisfiable iff $\mathcal{S}_\varphi \vDash \theta_n$ iff $\mathcal{C}_\varphi \vDash \theta_n$.

Proof. Let $f \in B$, $k = \mathrm{ar}(f)$. In the following we will construct MSO-formulae with constant depth, which simulate the conditions of local quasi-models according to Definition 4.4. The monadic relations are $L, \overline{L} \subseteq \mathrm{SF}(\varphi)$. They represent the formulae that must be verified, resp., falsified in the current world. The set $\overline{L}$ can be interpreted as the set of $\neg$-prefixed formulae of L.

$$\theta_f(L, \overline{L}) := \forall x \forall y_1 \ldots \forall y_k\ \mathrm{body}_f(x, y_1, \ldots, y_k) \to \Big(L(x) \to \bigvee_{\alpha \vDash f} \bigwedge_{\substack{1 \leqslant i \leqslant k, \\ \alpha(i)=1}} L(y_i) \wedge \bigwedge_{\substack{1 \leqslant i \leqslant k, \\ \alpha(i)=0}} \overline{L}(y_i)\Big)$$

$$\wedge \left(\overline{L}(x) \rightarrow \bigvee_{\alpha \not\models f} \bigwedge_{\substack{1 \leqslant i \leqslant k, \\ \alpha(i)=1}} L(y_i) \wedge \bigwedge_{\substack{1 \leqslant i \leqslant k, \\ \alpha(i)=0}} \overline{L}(y_i)\right)$$

$$\theta_{\text{local}}(L, \overline{L}) := \bigwedge_{f \in B} \theta_f(L, \overline{L}) \wedge \forall x(\neg L(x) \vee \neg \overline{L}(x))$$

The next formulae state that there is some $L(w)$ such that all state formulae in L_A, resp., L_E are also in L, and that formulae in L with leading path quantifiers are also in the correct set of path formulae. Note that technically all formulae from $L_A \cup L_E$ are placed in L, not only state formulae, since we cannot easily detect state formulae in the structure. But except being consistent in the Boolean sense this does not impose further constraints as X-preceded formulae are ignored here. In the following formula it holds that $O \in \{\mathsf{A}, \mathsf{E}, \mathsf{X}\}$.

$$\begin{aligned}
\theta_{\text{inherit } O}(L, L') &:= \forall x \forall y\ L'(x) \wedge \text{body}_O(x, y) \rightarrow L(y) \\
\theta_{\text{state}}(L_A, \overline{L}_A, L_E, \overline{L}_E) &:= \exists L \exists \overline{L}\ \theta_{\text{local}}(L, \overline{L}) \\
&\quad \wedge \forall x((L_A(x) \vee L_E(x)) \rightarrow L(x)) \\
&\quad \wedge \forall x((\overline{L}_A(x) \vee \overline{L}_E(x)) \rightarrow \overline{L}(x)) \\
&\quad \wedge \theta_{\text{inherit } \mathsf{A}}(L, L_A) \wedge \theta_{\text{inherit } \mathsf{E}}(L, L_E) \\
&\quad \wedge \theta_{\text{inherit } \mathsf{E}}(\overline{L}, \overline{L}_A) \wedge \theta_{\text{inherit } \mathsf{A}}(\overline{L}, \overline{L}_E)
\end{aligned}$$

The last formula propagates the sets L_A and L_E correctly to the successors of the current world. It ensures the existence of at least one successor at all, and also of one successor for every E-formula that must be witnessed by some path. We provide an inductive definition of this formula beginning with the base case:

$$\begin{aligned}
\theta^0_{\text{path}}(L_A, \overline{L}_A, L_E, \overline{L}_E) := \theta_{\text{local}}(L_A, \overline{L}_A) \ \wedge\ \theta_{\text{local}}(L_E, \overline{L}_E) \ \wedge \\
\theta_{\text{state}}(L_A, \overline{L}_A, L_E, \overline{L}_E)
\end{aligned}$$

and the inductive case for the formula is defined as

$$\begin{aligned}
\theta^n_{\text{path}}(L_A, \overline{L}_A, L_E, \overline{L}_E) :=\ & \theta^0_{\text{path}}(L_A, \overline{L}_A, L_E, \overline{L}_E) \\
& \wedge \exists L'_A\, \exists \overline{L}'_A\, \exists L'_E\, \exists \overline{L}'_E \left(\theta_{\text{inherit } \mathsf{X}}(L_A, L'_A) \wedge \theta_{\text{inherit } \mathsf{X}}(\overline{L}_A, \overline{L}'_A) \wedge \right. \\
& \qquad \left. \theta^{n-1}_{\text{path}}(L_A, \overline{L}_A, L_E, \overline{L}_E)\right)
\end{aligned}$$

$$
\begin{aligned}
\wedge\, \forall x \forall y \Big(& L_E(x) \wedge \mathrm{body}_{\mathsf{X}}(x,y) \rightarrow \\
& \exists L'_A\, \exists \overline{L}'_A\, \exists L'_E\, \exists \overline{L}'_E \, \Big(\theta_{\text{inherit } \mathsf{X}}(L_A, L'_A) \wedge L'_E(y) \\
& \qquad \wedge\, \theta_{\text{inherit } \mathsf{X}}(\overline{L}_A, \overline{L}'_A) \\
& \qquad \wedge\, \theta^{n-1}_{\text{path}}(L_A, \overline{L}_A, L_E, \overline{L}_E) \Big) \Big) \\
\wedge\, \forall x \forall y \Big(& \overline{L}_E(x) \wedge \mathrm{body}_{\mathsf{X}}(x,y) \rightarrow \\
& \exists L'_A\, \exists \overline{L}'_A\, \exists L'_E\, \exists \overline{L}'_E \, \Big(\theta_{\text{inherit } \mathsf{X}}(L_A, L'_A) \wedge \overline{L}'_E(y) \\
& \qquad \wedge\, \theta_{\text{inherit } \mathsf{X}}(\overline{L}_A, \overline{L}'_A) \\
& \qquad \wedge\, \theta^{n-1}_{\text{path}}(L_A, \overline{L}_A, L_E, \overline{L}_E) \Big) \Big).
\end{aligned}
$$

Finally let

$$\theta_n := \exists L_A\, \exists \overline{L}_A,\, \exists L_E,\, \exists \overline{L}_E\, \exists x\, \mathrm{repr}(x) \;\wedge\; L_A(x) \;\wedge\; \theta^n_{\text{path}}(L_A, \overline{L}_A, L_E, \overline{L}_E).$$

Overall, it holds for all formulae $\varphi \in \mathcal{CTL}^\star(B, \{\mathsf{A}, \mathsf{E}, \mathsf{X}\})$ with $\mathrm{td}(\varphi) \leqslant n$ that φ has a tree-like quasi-model of depth n iff $\mathcal{S}_\varphi \vDash \theta_n$ iff $\mathcal{C}_\varphi \vDash \theta_n$, and the lemma applies.

□

Finally we progressed to the main core of this section, as we are now able to show the fixed parameter tractability for satisfiability of $\mathcal{CTL}^\star$-formulae with $T \subseteq \mathsf{A}, \mathsf{E}, \mathsf{X}$ parameterized by temporal-depth and treewidth, or path width respectively.

Theorem 4.4.
Let B be a finite set of Boolean functions, $T \subseteq \{\mathsf{A}, \mathsf{E}, \mathsf{X}\}$. Then the problem CTL*-Sat(B, T) parameterized by $\mathrm{td} + \kappa$ is fixed-parameter tractable if $\kappa \in \{\mathrm{tw}_{\mathcal{C}}, \mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{S}}\}$.

Proof. Korach and Solel showed in 1993 that $\mathrm{pw} = \mathcal{O}(\log n \cdot \mathrm{tw}(G))$ for some graph G [KS93]. For this reason it suffices to consider only treewidth as parameter in combination with temporal depth td. Building on Theorem 4.1 for infinite signatures, we define an fpt-computable and κ-bounded function f as follows: $f : \varphi \rightarrow 1^{\mathrm{td}(\varphi)}$ and with a combination of the uniform MSO-formula θ_n of Lemma 4.3 , we finally obtain the desires properties of Theorem 4.1. Consequently, a given formula $\varphi \in \mathcal{CTL}^\star(B, T)$ is satisfiable, if and only if $\mathcal{S}_\varphi \vDash \theta_{|f(\varphi)|}$ if and only if $\mathcal{C}_\varphi \vDash \theta_{|f(\varphi)|}$, and $\mathcal{C}_\varphi$ can also be computed in fpt-time which is precisely the assertion of the theorem. □

Since $\mathcal{CTL}$ and $\mathcal{LTL}$ are fragments of $\mathcal{CTL}^\star$, the variations of Theorem 4.4 for $\varphi \in \mathcal{CTL}(B, T)$ and $\varphi \in \mathcal{LTL}(B, T)$, can be proven straightforward by reduction from CTL-Sat(B, T) and LTL-Sat(B, T), respectively, to CTL*-Sat(B, T).

Corollary 4.5.
Let B be a finite set of Boolean functions. The problems CTL-Sat(B,T), LTL-Sat(B,T') parameterized by td + κ are fixed-parameter tractable if $\kappa \in \{\mathrm{tw}_{\mathcal{C}}, \mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{S}}\}$ and $T \subseteq \{\mathsf{AX}\}$, $T' \subseteq \{\mathsf{X}\}$.

Now we want to show the fixed parameter tractability for $\mathcal{LTL}$-formulae parameterized only by $\mathrm{tw}_{\mathcal{S}}$ or $\mathrm{pw}_{\mathcal{S}}$.

Theorem 4.6.
Let B be a finite set of Boolean functions. For $T \subseteq \{\mathsf{X}\}$, the problem LTL-Sat$(B,T)$ is in **FPT** when parameterized by $\mathrm{tw}_{\mathcal{S}}$ or $\mathrm{pw}_{\mathcal{S}}$.

Proof. The main idea of the proof is to give a reduction

$$(\text{LTL-Sat}(B,X),\kappa) \leqslant^{fpt} (\text{Sat}(B),\kappa),$$

by putting the X-operators inside the brackets and consequently collecting them at the leaf nodes of $\mathcal{S}$ and to show that the treewidth or path-width respectively, does not increase to much.
Due to the path semantics of LTL, follows X distributes over arbitrary Boolean functions. To be more concrete, $\mathsf{X}f(\varphi_1,\ldots,\varphi_n) \equiv f(\mathsf{X}\varphi_1,\ldots,\mathsf{X}\varphi_n)$ holds for $f \in \mathcal{B}, \varphi_1,\ldots,\varphi_n \in \mathcal{LTL}$. Consequently, we can transform every $\mathcal{LTL}$ formula containing only X-operators to an equivalent Boolean combination β of X-preceded variables:

$$\varphi \equiv \beta(\mathsf{X}^{n_1}q_1,\ldots,\mathsf{X}^{n_m}q_m), \quad \mathsf{X}^{n_i} := \underbrace{\mathsf{X}\ldots\mathsf{X}}_{n_i \text{ times}}$$

with propositional variables q_i and $n_i \geqslant 0$. Note that inconsistent literals can only occur inside the same world and therefore at the same nesting depth of X. For this reason the above formula φ is satisfiable if and only if it is satisfiable as a purely propositional formula where each expression $\mathsf{X}^{n_i}q_i$ is interpreted as an atomic proposition. This is illustrated in the following example:

$$\begin{aligned}\varphi &= \mathsf{X}(a_1 \vee \mathsf{X}(a_5 \wedge a_2)) \vee \mathsf{X}(a_3 \wedge \mathsf{X}(a_1 \wedge a_2)) \\ &\equiv (\underbrace{\mathsf{X}a_1}_{p_1} \vee \underbrace{\mathsf{XX}a_5}_{p_2} \wedge \underbrace{\mathsf{XX}a_2}_{p_3}) \vee (\underbrace{\mathsf{X}a_3}_{p_4} \wedge \underbrace{\mathsf{XX}a_1}_{p_5} \wedge \underbrace{\mathsf{XX}a_2}_{p_3}) \\ &\equiv (p_1 \vee p_2 \wedge p_3) \vee (p_4 \wedge p_5 \wedge p_3).\end{aligned}$$

For an fpt-reduction to $(\text{Sat}(B),\kappa)$ with $\kappa \in \{\mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{S}}\}$. We know from Lemma 4.3 the formulae are MSO-definable. In the next step we need to show, that the parameter does not increase too much when we distribute the temporal X-operator over the Boolean functions as shown in the example above. Consider the substructure in the syntax tree $\mathcal{S}$ with nodes $\{\mathsf{X}, f, \varphi_1,\ldots,\varphi_n\}$ and edges $\{(f,\mathsf{X}_1),(f,\varphi_1),\ldots,(f,\varphi_n)\}$ is locally

replaced after distributing of X by a substructure with nodes $\{f, \mathsf{X}_1, \ldots, \mathsf{X}_n, \varphi_1, \ldots, \varphi_n\}$ and edges $\{(f, \mathsf{X}_1), (\mathsf{X}_1, \varphi_1), \ldots, (f, \mathsf{X}_n), (\mathsf{X}_n, \varphi_n)\}$. Consequently, we have to adapt the resulting substructure to the tree-, resp. path-decomposition. We append to each bag $\mathcal{B}$ containing X or f the additional node set $\{\mathsf{X}, f, \mathsf{X}_1, \ldots, \mathsf{X}_n\}$, with a constant $n = \mathrm{ar}(f)$ depending on $f \in B$ only. This approach ensures that the edges to former parents of X which are now parents of f are covered, and that the edges to former children of f which are now children of $\mathsf{X}_1, \ldots, \mathsf{X}_n$ are covered. The edges inside the new substructure are covered as well. If the path-, resp., treewidth previously was k, then every bag $\mathcal{B}$ contained at most $k + 1$ nodes representing some f or X. Overall the new width of the bag is at most $(k + 1) \cdot (c + 2)$, where c is the maximum arity of some $f \in B$. Now $(\textsc{Sat}(B), \mathrm{tw}_{\mathcal{S}}) \in \mathbf{FPT}$ and $(\textsc{Sat}(B), \mathrm{pw}_{\mathcal{S}}) \in \mathbf{FPT}$ hold as a special case of $\mathcal{CTL}^\star$ with $\mathrm{td} = 0$ according to Theorem 4.4. Hence the above reduction yields $(\text{LTL-}\textsc{Sat}(B, \{\mathsf{X}\}), \kappa) \in \mathbf{FPT}$ for $\kappa \in \{\mathrm{pw}_{\mathcal{S}}, \mathrm{tw}_{\mathcal{S}}\}$. □

Note that the proof of Theorem 4.6 does not work for circuit representation $\mathcal{C}$ as the resulting treewidth $\mathrm{tw}_{\mathcal{C}}$ and path-width $\mathrm{pw}_{\mathcal{C}}$ cannot be bounded during the transformation. To provide a better understanding of what we mean, we illustrate it in Figure 4.4.
Let $p_1, \ldots, p_n$ be parents of a node f with a stack of i-X operators between f and $p_1, \ldots, p_n$. The circuit contains n nodes of X, viz. $\mathsf{X}_1, \ldots, \mathsf{X}_n \in \mathcal{C}$, connected by an edge, as subformulae are reused. Now suppose that the whole formula with a root ψ is formed by $p_1, \ldots, p_n$, as illustrated on the left side of Figure 4.4. Function f has arguments $a_1, \ldots, a_n$. We use the sequence of bags $B_1, \ldots, B_n$ with $B_i := \{\psi, p_i, p_{i-1}, \mathsf{X}_i, \mathsf{X}_{i-1}, f, a_i\}$. In consequence, we obtain a bounded path-width as well as bounded treewidth. In the next step we have to distribute X over f. By distributing X the resulting circuit has to contain one root ψ but also has to contain n copies f_i of f, where each f_i has n arguments $a_1, \ldots, a_n$ and additionally a stack of i X-operators between f_i and the arguments. Each f_i has one parent node p_i connected with the root ψ. Now assume as a worst case scenario for each pair (i, j) with $1 \leqslant i, j \leqslant n$ the node f_i has a disjoint path to each a_j, always passing i X nodes. Obtain a minor of the circuit by merging each node a_j with the stack of n X-operators above it to a node A_j. Then for each (i, j) as above we have an edge (f_i, A_j). Therefore, the circuit contains $K_{n,n}$ the (n, n)-biclique, as a minor and consequently has treewidth—and hence path-width—at least n.
In the next section we examine, inter alia, exactly this case of syntax circuit and we show the $\mathbf{W}[1]$-hardness by using the idea from the example for a reduction. The framework of the temporal X operator does not work when we consider fragments which do not have models bounded by the temporal depth of a formula.

4.1.3 Fixed-Parameter Intractable Fragments

Here we will prove $\mathbf{W}[1]$-hardness for fragments of $\mathcal{CTL}^\star$ which do not have models bounded by temporal depth of a formula. In the first part of this section we will use two parameters, namely temporal depth and treewidth. In the second part of the section we

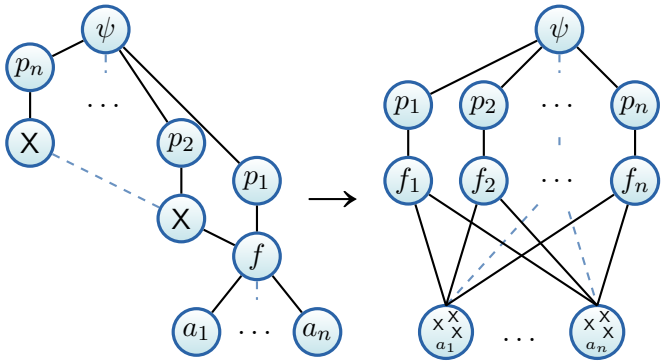


Figure 4.4: From constant path-width to unbounded treewidth by distributing X

will explore the computational complexity of the satisfiability problem in $\mathcal{CTL}^\star$ using only one of the parameters temporal depth and treewidth.

4.1.3.1 Parametrization by Temporal Depth and Treewidth

We take up the previous work of Praveen, where he examines the problem of *partitioned weighted satisfiability* in transitive modal logic to prove intractability.

The parameterized problem of partitioned weighted satisfiability in short p-PW-Sat, has as instance a propositional CNF-formula φ over variables $q_1, \ldots, q_n$, where the variables partitioned into disjoint sets $Q_1, \ldots, Q_k$. With $[k]$ we abbreviate the set $\{1, \ldots, k\}$. Each partition Q_i has an assigned *capacity* $C_i \in \mathbb{N}$. The parameter is the sum of primal path-width of φ and the number of partitions. An assignment ϑ is called *saturated* for an instance $I = (\varphi, k \in \mathbb{N}, (Q_i)_{i\in[k]}, (C_i)_{i\in[k]}, \kappa)$ if in every partition Q_i there are exactly C_i variables set to true by ϑ.

Problem:	p-PW-Sat
Input:	Propositional CNF φ, $(Q_i)_{i\in[k]}, (C_i)_{i\in[k]}$, $k \in \mathbb{N}$
Parameter:	$\kappa := \mathrm{pw}^*(\varphi) + k$
Question:	Has φ a satisfying saturated assignment?

The idea of Praveen to show intractability of modal satisfiability in transitive frames was to formulate the problem PW-Sat in modal logic. He constructed a formula which enforces the existence of a chain of worlds $w_0 \to w_1 \to \cdots \to w_n$. Each world w_i with $i \geqslant 1$ has to assign to each q_i variables either to "*true*" or to "*false*". At the end in the world w_n it is required to verify the number of variables assigned to "*true*" in each partition, which has to corresponds to the respective capacity. A great care needs to be payed to constructing such formula relating to parameterized reduction to keep the path-width low.
We will cover the different CTL fragments in two 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. In the second step the result is transferred to the fragments of CTL.

Lemma 4.7.
CTL-SAT(T) is $\mathbf{W}[1]$-hard for $T \subseteq \{\mathsf{AX}, \mathsf{AG}\}$ when parameterized by $\kappa := \mathrm{td} + f$, where $f \in \{\mathrm{tw}_{\mathcal{C}}, \mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{S}}\}$.

Proof. We prove $\mathbf{W}[1]$-hardness by giving a reduction showing CTL-SAT(T) $\leqslant^{fpt}$ p-PW-SAT. The reduction function transforms the instance $I = (\varphi, k \in \mathbb{N}, (Q_i)_{i\in[k]}, (C_i)_{i\in[k]}, \kappa)$ of p-PW-SAT to a CTL formula $\psi(I)$. We will construct the formula $\psi(I)$ corresponding to the properties of the instance of p-PW-SAT.

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

The formula ψ[formula] forces φ to be true in the starting world w_0, φ to be true to guarantee its satisfiability.

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

The formula ψ[depth] forces the desired chain $w_0 \rightarrow w_1 \rightarrow \cdots \rightarrow w_n$ of worlds in the model via labelling of a variables encoding the depth $d_0, d_1, \ldots$.

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

$Q = \{q_1, \ldots, q_n\}$ denotes the set of variables that are labeled in the starting world w_0. The formula ψ[fixed-Q] forces the variables of Q also be labeled in the successive worlds to ensure a valid saturation of Q.

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

As the name suggests, the formula ψ[signal] indicates that the partition number of q_i, denoted by $p(i) \in [k]$ has increased. For this issue we introduce new propositional variables $\top_{p_i}^{\uparrow}$.

$$\psi[\text{count}] := \mathsf{AG} \bigwedge_{i=1}^{n} \bigwedge_{j=0}^{|Q_p|} \left[\left(\top_p^{\uparrow} \rightarrow \left(\top_p^{j} \leftrightarrow \mathsf{AX} \top_p^{j+1}\right)\right) \wedge \left(\neg \top_p^{\uparrow} \rightarrow \left(\top_p^{j} \leftrightarrow \mathsf{AX} \top_p^{j}\right)\right)\right]$$

The formula ψ[count] forces whenever an increment signal for partition p is encountered, to increment the counter from j to the next integer $j+1$. To count the total amount of labeled variables per partition we required several variables named $\top_p^j$ here.

$$\psi[\text{monotone}] := \mathsf{AG}\left[\bigwedge_{i=1}^{n}(d_i \to d_{i-1}) \wedge \bigwedge_{p=1}^{k}\bigwedge_{j=1}^{|Q_p|+1}\left(\top_p^j \to \top_p^{j-1}\right)\right]$$

The formula ψ[monotone] ensures the consistent counting of all the used variables.

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

The formula ψ[init] forces the counter to be 0 in the initial world.

$$\psi[\text{target}] := \mathsf{AG}\bigwedge_{p=1}^{k}\left[d_{n+1} \to \left(\top_p^{C_p} \wedge \neg\top_p^{C_p+1}\right)\right]$$

Finally, the formula ψ[target] ensures that the number of positive variables per partition corresponds to the capacity.
Now we have to prove the correctness of the reduction. For this, suppose $\left(\varphi,\, k,\, (Q_i)_{i\in[k]},\, (C_i)_{i\in[k]}\right) \in$ p-PW-Sat. To show this we have to design a valid model for $\psi(I)$. We start in the world w_0. So φ is satisfied if the number of saturated variables in Q corresponds to the capacity. For this reason we label Q in w_0. Consequently, φ is satisfied in w_0.
Next we construct the successor worlds $w_1, \ldots, w_{n+1}$, with a self-loop on w_{n+1}. Then we label a maximum amount of variables in those world to satisfy formulae ψ[depth], ψ[fixed-Q], ψ[init], as this is always possible.
Moreover we label the variables $\top_{p(i)}^{\uparrow}$ corresponding to ψ[signal], doing so we obtain exactly $C_{p(i)}$ occurrences of $\top_{p(i)}^{\uparrow}$, as Q is chosen saturated. For this reason the formula ψ[count] allows for every partition p that its counter is incremented exactly C_p times. This construction does not violate the ψ[monotone] condition and allows to satisfy ψ[target] in the world w_{n+1}.
By this construction the formula ψ[monotone] is fulfilled and additionally we satisfy ψ[target] at the world w_{n+1}. Consequently, the constructed model is correct.
Now the other way around let $\mathcal{M}$ be a model of $\psi(I)$. This model $\mathcal{M}$ has to contain a tree of worlds which fulfil the conditions ψ[init], ψ[depth] and ψ[fixed-Q]. This tree of worlds can be transformed to a path $w_0, \ldots, w_{n+1}$ by deleting needless labels and afterwards identifying worlds at the same depth, which have consistent labels only due to the A-operators. As ψ[target] and ψ[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 $\top_p^{\uparrow}$ signal is labeled exactly C_p times for every partition p. But ψ[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 saturated assign-

ment ϑ for φ. Additionally, ϑ is also a satisfying assignment for φ since w_0 was labeled consistent with ψ[formula].

In the next step we have to prove the following claim.

Claim 4.8.
$\mathrm{td}(\psi(I)) + \kappa(\psi(I))$ is bounded by $\mathrm{pw}^*(\varphi) + k$.

Proof of claim.
Note that, we obtain a constant temporal depth from the reduction. Now we have to focus on the proof of bounded $\mathrm{pw}_{\mathcal{S}}$ and bounded $\mathrm{pw}_{\mathcal{C}}$, as $\mathrm{tw} \leqslant \mathrm{pw}$ holds. We abbreviate the syntax tree $\mathcal{S}_{\psi(i)}$ of $\psi(i)$ by $\mathcal{S}$. Further, let $\mathcal{P}$ be an optimal path decomposition of the primal graph of φ. In the following we need to show how to extend the optimal path decomposition to a path decomposition of $\mathcal{S}$, denoted by $\mathcal{P}'$. For $\mathcal{P}$ we require a special structured property, the so-called *one-step addition property*. It says that the bag $\mathcal{B}_i$ in $\mathcal{P}$ introduces exactly one new variable q, viz. $\mathcal{B}_i = \mathcal{B}_{i-1} \cup \{q\}$ and $\mathcal{B}_i \backslash \mathcal{B}_{i-1} = \{q\}$. To achieve this property we split bags which introduces multiple variables into multiple bags and remove bags which do not introduce variables. Additionally, we rename bags $\mathcal{B}_i$ and variables q_i, such that the bags are ordered according to their indices and bag $\mathcal{B}_i$ introduces variable q_i. This can be done in linear time. Consequently, we can assume the *one-step addition property*. This property was also used by Praveen [Pra13].
By *augmenting a bag $\mathcal{B}$ with x* we mean inserting a copy $\mathcal{B}'$ of $\mathcal{B}$ between $\mathcal{B}$ and its successor bag and placing the additional element x there. It holds that $|\mathcal{B}'| = |\mathcal{B}| + 1$. 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 we pursue the following seven steps to construct $\mathcal{P}'$:

(1.) First we increase every bag size by the number of partitions k by adding the variable $\top_p^{\uparrow}$ to every bag for $1 \leqslant p \leqslant k$.

(2.) In the formula ψ[formula] $:= \varphi$, φ is in CNF. By construction, the primal graph of φ contains for each clause of φ a clique of the variables appearing in that clause. Consequently, $\mathcal{P}$ has to contain a bag $\mathcal{B}$ covering all those variables. Now consider a clause of size m of the following form: $((((\ell_1 \vee \ell_2) \vee \ell_3) \cdots) \vee \ell_m)$, where every literal ℓ_i is a variable q or its negation $\neg q$. Let $\mathcal{B}$ be the bag containing the variables of this clause. We augment $\mathcal{B}$ with $\vee$-nodes as follows (we illustrate this procedure in Figure 4.5):

- copy the bag $\mathcal{B}$ m-times
- add the j-th $\vee$-node to the j-th and $(j+1)$-th copy of $\mathcal{B}$. This results in the bag containing an $\vee$-node including a connected component
- refer to the outmost $\vee$-operators as the *primary $\vee$-nodes*.

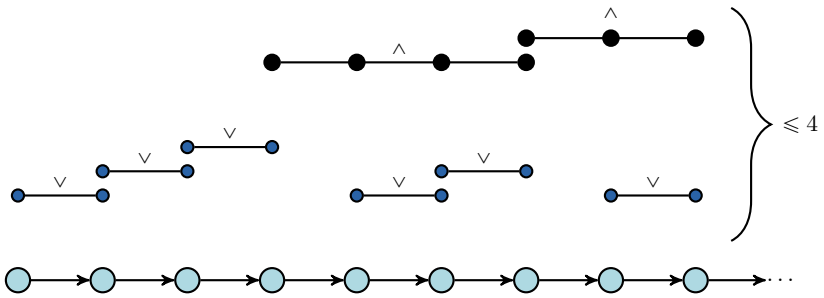


Figure 4.5: Bag augmentation

We proceed in similar vein with the $\wedge$-nodes, then select two adjacent primary $\vee$-clauses in the path decomposition and add $\wedge$-node to all bags that connect them.

From [FG06, Corollary 11.28] we know, that an optimal path decomposition can be computed in fpt-time. Consequently the structure $\mathcal{S}$ can be constructed in a way that allows the argumentation above *a priori*, and the placement of parentheses in $\psi(I)$ can always be chosen to associate literals in ascending order of variables in $\mathcal{P}$; and to 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.

(3.) Now we add the variables $d_{i-1}, d_i, d_{i+1}, d_{i+2}$ for $1 \leqslant i \leqslant n$ and their negation to the bag $\mathcal{B}$ that introduces q_i as well as the nodes representing $(d_i \wedge \neg d_{i+1})$, $(d_{i+1} \wedge \neg d_{i+2})$ and $(d_i \rightarrow \neg d_i)$. Consequently, due to the one step addition property concerning q_i's, every included node induces a connected component. Therefore we increase every bag by a constant number of items. To cover each conjunct of the ψ[depth] formula we need to add the necessary AX-nodes and $\rightarrow$-nodes to every bag, which also results in a constant number of additional items. To cover the ψ[depth] completely it remains to cover its $\bigwedge$-node. Here we proceed like before and add at most two items and its AG-node to every bag.

(4.) Here we cover the formula ψ[fixed-Q]. Therefore we need to add the AX-node and the $\leftrightarrow$-node for the i-th conjunct to the bag that introduces q_i and increase the size of the bag by at most two items. We proceed in similar vein with $\bigwedge$ and AG as before.

(5.) For the formula ψ[signal] we augment the bag introducing q_i by the nodes $\leftrightarrow$, $\rightarrow$ and $(d_i \wedge \neg d_{i+1})$. As the $\top^{\uparrow}_{p(i)}$ variables are already added in every bag we do not need to add them here. We proceed adding the nodes $\bigwedge$ and AG as before.

(6.) We augment the last bag by the remaining formulae. W.l.o.g. suppose this bag contains the variable d_{n+1}. Let the maximal capacity be denoted by C. We proceed for every $1 \leqslant j \leqslant C$ in the following way: We attach one bag that is a copy of

$\mathcal{B}$, but additionally the j-th attached bag further contains $\top_p^j, \top_p^{j+1}$ and $\top_p^{j-1}$ for every partition p. This increases every bag size by $3k$. Since d_{n+1} also is in these bags, the nodes representing subformulae of ψ[count], ψ[monotone], ψ[init] and ψ[target] containing $\top_p^j$'s can be added. Each increases the bag size by a constant number of items. Note that ψ[target] actually requires these nodes to be added after the last bag so the bags containing d_{n+1} are connected.

(7.) For the last step note that the remaining subformulae of $\psi(I)$ are either

- conjunctions of size C over variables $\top_p^j$'s, which are already covered by the bags introduced in the antecedent stage or
- conjunctions of size k and connectives of constant depth.

We obtain $\psi(I)$ by connecting all remaining subformulae together, by adding them to every bag.

This "seven-step" construction yields a path decomposition of the structure $\mathcal{S}_\varphi$ with κ-bounded width. The way we handle the signal variables $\top_p^\uparrow$ is the key factor for keeping the path-width low in this construction, as they are the only "link" between the variables q_i and the partition weight counters.
If $\mathcal{P}'$ should also be a path-decomposition of $\mathcal{C}_{\psi(I)}$, then identical subformulae of $\psi(I)$ actually have to induce connected subpaths in $\mathcal{P}'$—not only propositions—according to the definition of a path-decomposition. Obviously for every bag $\mathcal{B}$ that contains a formula ξ, the formulae $\neg\xi$ and $\mathsf{AX}\xi$ can be added to $\mathcal{B}$. The only other subformulae that occur multiple times in $\psi(I)$ are the $(d_i \wedge \neg d_{i+1})$ for $i \in [n]$, but for any i the respective node can be added to any bag which contains d_i or d_{i+1}. Then the occurrences of $(d_i \wedge \neg d_{i+1})$ are connected in $\mathcal{P}'$, as the occurrences of d_i as well as d_{i+1} are each connected and they are overlapping. Consequently the path-width is bounded by k also, which is precisely the assertion of the claim.

$\diamond$

Since we have shown the correctness of the reduction and that the there is a bound for the parameter in terms of the old parameter, we obtain the assertion of the lemma.

$\square$

Next we want to prove $\mathbf{W}[1]$-hardness of the CTL-Sat problem with operators AX and AF.

Lemma 4.9.
CTL-Sat(T) is $\mathbf{W}[1]$-hard for $\{\mathsf{AX}, \mathsf{AF}\} \subseteq T$ when parameterized by $\mathrm{td} + \kappa$ and $\kappa \in \{\mathrm{tw}_\mathcal{S}, \mathrm{pw}_\mathcal{S}, \mathrm{tw}_\mathcal{C}, \mathrm{pw}_\mathcal{C}\}$.

Proof. We use the formulae constructed in the proof of Lemma 4.7 with some ancillary considerations. In this formulae the AG-operator appears at $\mathrm{td} = 0$ and in conjunctions only. Consequently we achieve a formula $\varphi_\mathcal{F} = \psi \wedge \mathsf{AG}\chi$ with a propositional formulae

ψ and $\chi \in \mathcal{CTL}(\{\mathsf{AX}\})$. Now consider the conjunction of the AG-operators. Since $\mathsf{AG}(\alpha) \wedge \mathsf{AG}(\beta) \equiv \mathsf{AG}(\alpha \wedge \beta)$, we are able to transform the formula $\varphi_{\mathcal{F}}$ into a formula containing only one AG-operator. We can then substitute AG by EG, which can be use since $\neg\mathsf{AF} = \mathsf{EG}$ and $\mathsf{AG}(\alpha) \rightarrow \mathsf{EG}(\alpha)$ holds trivially. The reverse direction holds for models with just one path. Since only one path (the chain $w_0 \rightarrow \cdots \rightarrow w_n$) is required for the reduction from PW-Sat this suffices. Therefore we remove all paths where $\mathsf{EG}(\alpha)$ does not hold and obtain a model where $\mathsf{AG}(\alpha)$ holds.

Note, we are able to use this substituting step, when we have one AG-operator only appearing at $\mathrm{td} = 0$. □

Lemma 4.10.
CTL-Sat(T) is $\mathbf{W}[1]$-hard for $\mathsf{AG} \in T$ or $\mathsf{AR} \in T$ when parameterized by $\mathrm{td} + \kappa$ and $\kappa \in \{\mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{S}}, \mathrm{tw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{C}}\}$.

Proof. Here we do not have the X-operator. Consequently, we require more challenging consideration to prove the correctness of the reduction for several reasons, which we will explain below. Additionally note that $\mathsf{AG}(\alpha) \Leftrightarrow \mathsf{A}[\bot \mathsf{R} \alpha]$ holds, so it suffices to consider AG-operator only. Note that both operators G and F are *stutter-invariant*, viz. this operators cannot differentiate between a path π and another path π', where π' is constructed from π by duplicating arbitrary words on the path.
The first problem due to the absence of the X-operator is, that even if we are able to force a chain of worlds $w_0 \rightarrow \cdots \rightarrow w_n$ to appear, we can not prevent duplicates of the worlds. Consequently we are incapable of verifying the value of a counter exactly. Claiming a minimum value is not an option, as the reduction would not be correct in that case. The second problem is, due to the possibility of duplicated worlds, we also have an arbitrary number of occurrences of variables. in that case we would count one variable several times and get a satisfiable CTL-formula even if φ does not have a saturated assignment. To circumvent this problem we demand upper bounds for both the number of variables set to true and the number of variables set to false in a partition. Another problem is, that we are not able to force the counter to increase in the next world instead of the current, due to the *reflection property* of G and F. Therefore we label the depth properties together with their "parities", which requires two new variables and consequently increases the path width by at most two.
We now explain the desired reduction, emphasizing the differences to the construction in the proof of Lemma 4.7.

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

Here we replace the AX operator by $\neg\mathsf{AG} \equiv \mathsf{EF}$. We need to mention that we are not able to reach all paths starting at w_0, which form the desired chain of worlds $w_0 \rightarrow \cdots \rightarrow w_n$. Nevertheless, the branch of the model that satisfies one of the EF formulae again has to branch correctly at least once for the next depth level because of the nesting inside an

AG operator. Consequently, at least one path starting in w_0 contains the correct labels. Due to the duplicated worlds, 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 EF. Therefore, there is at least one path reachable from w_0 which has the desired form. In order to address the problem of irreflexivity, we enforce some kind of alternation in terms of variables. We require two new variables to label the parity of world, namely m_0 for parity 0 and m_1 for parity 1.

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

We adapt the formula $\psi[\text{fixed-}Q]'$ by using AG operators only.

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

In the next formula $\psi[\text{signal}]_2$ we have to deal with the problem with of duplicates of worlds with the same depth proposition labeled. It is not possible to count the signal with the former concept, as we cannot force that the labeled counter propositions may not change in the next world. To preserve the correctness of the reduction we require a second type of counters for variables set to zero, $\bot_p^{\uparrow}$. We introduce the following formulae to guarantee the correctness of the new counter.

$$\begin{aligned}
\psi[\text{signal}]_2 &:= \mathsf{AG} \bigwedge_{i=1}^{n} \left[(d_i \wedge \neg d_{i+1}) \rightarrow \left(\neg q_i \leftrightarrow \bot_{p(i)}^{\uparrow}\right)\right] \\
\psi[\text{init}]_2 &:= \bigwedge_{p=1}^{k} \left[\neg \bot_p^{\uparrow} \wedge \neg \bot_p^{1}\right] \wedge \mathsf{AG} \bigwedge_{p=1}^{k} \bot_p^{0} \\
\psi[\text{monotone}]_2 &:= \mathsf{AG} \bigwedge_{p=1}^{k} \bigwedge_{j=1}^{|Q_p|+1} \left(\bot_p^{j} \rightarrow \bot_p^{j-1}\right)
\end{aligned}$$

Next we verify whether the assignment for variables in partition p sets the correct number of variables to true and false with respect to the capacity C_p, viz. at most C_p variables have been set to true and at most $|Q_p| - C_p$ variables have been set to false.

$$\psi[\text{target}] := \mathsf{AG} \bigwedge_{p=1}^{k} \left[\neg \top_p^{C_p+1} \wedge \neg \bot_p^{|Q_p|-C_p+1}\right]$$

In the last step we split the one counter in two counters, namely one for counting variables

set to true and the second counting the variables set to false.

$$\psi[\text{count}]_1 := \mathsf{AG} \bigwedge_{p=1}^{k} \bigwedge_{j=0}^{|Q_p|} \bigwedge_{i=0}^{1} \Big[\big(\top_p^{\uparrow} \wedge \top_p^{j} \wedge m_i\big) \rightarrow \mathsf{AG}\big(m_{1-i} \rightarrow \mathsf{AG}\, \top_p^{j+1}\big) \Big]$$

$$\psi[\text{count}]_2 := \mathsf{AG} \bigwedge_{p=1}^{k} \bigwedge_{j=0}^{|Q_p|} \bigwedge_{i=0}^{1} \Big[\big(\bot_p^{\uparrow} \wedge \bot_p^{j} \wedge m_i\big) \rightarrow \mathsf{AG}\big(m_{1-i} \rightarrow \mathsf{AG}\, \bot_p^{j+1}\big) \Big]$$

We obtain that if a depth proposition d_i has a signal variable $\top_p^{\uparrow}$ or $\bot_p^{\uparrow}$ labeled, then the corresponding counter value increases during the next parity change of i. In consequence, 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 $|Q_p| - k$ parity changes with the proposition $\bot_p^{\uparrow}$ labeled. This yields that the counter $\top_p^{j}$ has value $j \geqslant k$ and the counter $\bot_p^{j}$ has a value $j \geqslant |Q_p| - k$ in world w_{n+1}. This contradicts $\psi[\text{target}]$ unless j is exactly k, resp. $|Q_p| - k$ and the partition is saturated.

The path-widths $\mathrm{pw}_{\mathcal{S}}$ and $\mathrm{pw}_{\mathcal{C}}$ increase only by a constant when considering the changes of the two formulae $\psi[\text{depth}]'$ 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 formulae the same procedure as in Lemma 4.9 can be applied: 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 κ-boundedness. □

Using the proof of Lemma 4.10 it is straightforward to proof $\mathbf{W}[1]$-hardness of CTL-Sat(AU).

Lemma 4.11.
CTL-Sat(T) parameterized by td+κ is $\mathbf{W}[1]$-hard if $\mathsf{AU} \in T$ and $\kappa \in \{\mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{S}}, \mathrm{tw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{C}}\}$.

Proof. The operator $\mathsf{AG}(\alpha)$ can be easily substituted by $\mathsf{A}(\alpha \mathsf{U} d_{n+2})$. Consequently, we need to introduce a new depth proposition d_{n+2}, which has to hold after d_{n+1}. This substitution can always be applied, as long as AG does not occur in negated form. Since it occurs negated in the formula $\psi[\text{depth}]$, we have to adjust it in the following way:

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

□

The upcoming **W**[1]-hardness result for CTL-SAT(AF) works for circuit formula representation only.

Lemma 4.12.
CTL-SAT(T) parameterized by $\mathrm{td} + \kappa$ is **W**[1]-hard if $\mathsf{AF} \in T$ and $\kappa \in \{\mathrm{tw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{C}}\}$.

Proof. Unfortunately, using the operator AF and its negation EG we are not able to count the worlds on a path, but only its "frontiers of reachability" about which the operators AF and EG can reason. In the following we will use the notion of [Eme90] for a quasi-model K. For a formula $\mathsf{AF}\beta$ labeled in a world w, write $\mathrm{DAG}[w, \beta]$ for the finite *Directed Acyclic Graph* (in short dag) that starts at w and contains all worlds reachable from w up to the first occurrence of the formula β in a quasi-label. Such a finite dag always has to exist due to the semantics of AF. Furthermore the dag is not only contained in K, but *embedded* in K, which means that every path through K that leads out of the dag has to go through its leaves. The leaves of $\mathrm{DAG}[w, \beta]$ are so-called *frontier worlds* and its non-leaves (including w if w is not already a leaf) are called *interior worlds*. Next we will construct a formula which witnesses the reduction. The formula we will construct is satisfiable iff $I \in$ P-PW-SAT. Consequently, there have to be dags in the sense of above which are "nested" in each other.
The key idea now is that every dag increments a counter value depending on the counter values of the dags reachable from its frontier, hence the total number of such dags is propagated to the root of the model.
Next, we will label a formula α in the frontier nodes of every such dag. By precomputation we can exclude any instance of P-PW-SAT where for some p it is $C_p = 0$ or $C_p = |Q_p|$, so both variables set to true and variables set to false have to occur at least once in every partition.

$$\begin{aligned}\alpha := \bigwedge_{j=0}^{n}\bigwedge_{p=1}^{k}(\top_p^{\uparrow} \wedge \mathsf{AF}(\neg\top_p^{\uparrow} \wedge \mathsf{AF}\top_p^{j}) \rightarrow \top_p^{j+1}) \\ \wedge (\bot_p^{\uparrow} \wedge \mathsf{AF}(\neg\bot_p^{\uparrow} \wedge \mathsf{AF}\bot_p^{j}) \rightarrow \bot_p^{j+1}) \\ \wedge (\top_p^{\uparrow} \rightarrow \top_p^{1}) \wedge (\bot_p^{\uparrow} \rightarrow \bot_p^{1})\end{aligned}$$

The formula α states for all partitions p, the counter for the variables set to true increments by one (with the counter value value $\top_p^{j+1}$), if α is enclosed by a frontier with $\neg\top_p^{\uparrow}$, which is itself enclosed by a frontier $\top_p^{\uparrow}$. We also want to count the variables set to zero. This can be done analogously: Simply substitute $\top$ by $\bot$ in argumentation. Due to the reflexive future in CTL, it is crucial to have the condition $\neg\top_p^{\uparrow}$, resp. $\neg\bot_p^{\uparrow}$. Additionally, the counter should not jump to the maximum value at the first occurrences of $\top_p^{\uparrow}$, resp. $\bot_p^{\uparrow}$. Furthermore, we have to initialise the respective counter to the value

one.

$$\beta_i^d := \left[q_i \to \mathsf{AF}\left(\top^{\uparrow}_{p(i)} \wedge d_i \wedge \mathsf{EG}(\neg e_{i-1}) \wedge \alpha\right)\right] \\ \wedge \left[\neg q_i \to \mathsf{AF}\left(\bot^{\uparrow}_{p(i)} \wedge d_i \wedge \mathsf{EG}(\neg e_{i-1}) \wedge \alpha\right)\right]$$

We force with the formulae β_i^d the existence of nested dags. Their frontier worlds have slightly different labels depending on wether q_i or $\neg q_i$ was chosen for the saturated, satisfying assignment.

$$\beta_i^e := \mathsf{AF}\left(e_i \wedge \mathsf{EG}(\neg d_i) \wedge \bigwedge_{p=1}^{k} \neg\top^{\uparrow}_p \wedge \neg\bot^{\uparrow}_p\right)$$

The formulae β_i^e target to enforce additional dags between the β_i^d-dags which are a kind of "graps". These gaps are required for α to work in an irreflexive way, as only alternation of variables can be distinguished by the stutter-invariant operators AF and EG. Finally, let

$$\psi(I) := \varphi \wedge \bigwedge_{i=1}^{n}\left(\beta_i^d \wedge \beta_i^e\right) \wedge \mathsf{EG}\bigwedge_{p=1}^{k} \neg\top_p^{C(p)+1} \wedge \neg\bot_p^{|Q_p|-C(p)+1}$$

This formula ensures the existence of aforementioned dags and a correct saturated assignment according to I. Consequently, $\psi(I)$ is satisfiable if and only if $I \in$ P-PW-Sat.

For the sake of clarity, we will split the proof in several claims. First, let (K, w_0) be a quasi-model of $\psi(I)$. For abbreviation let DAG$[i]$ stand for DAG$[w_0, \beta_i']$, where β' is the AF-proceeded formula implied by β_i^d depending on q_i. Additionally, we abbreviate DAG$[w_0, \beta_i'']$ with DAG$'[i]$, where $\beta_i^e = \mathsf{AF}\beta_i''$.

Claim 4.13.
DAG$[i]$ is contained in the interior worlds of DAG$'[i]$, and DAG$'[i]$ is contained in the interior worlds of DAG$[i+1]$.

Proof of claim.
We need only to consider that DAG$[i]$ is contained in DAG$'[i]$. The two dags cannot have common frontier worlds, as otherwise those worlds would have both d_i and $\neg d_i$ labeled. The same argumentation applies to DAG$'[i]$ and DAG$[i+1]$ via the proposition e_i.
Let w be a frontier world of DAG$'[i]$. Then $\beta_i'' \in L(w)$ which implies $\mathsf{EG}\neg d_i \in L(w)$. Further let $\pi \in \Pi(w)$ be the path that satisfies $\mathsf{G}\neg d_i$. Every path $\pi' \in \Pi(w_0)$ which runs through w has to visit a "shallower" world w' with $\beta_i' \in L(w')$ before w: Otherwise the

path

$$(w_0 = \pi'[0], \pi'[1], \ldots, w = \pi[0], \pi[1], \ldots)$$

would be a path starting in w_0 but not fulfilling $\mathsf{F}\beta_i'$. Consequently (K, w_0) would not be a quasi-model. This implies that on every path to a frontier node of DAG$'[i]$ there occurs a frontier node of DAG$[i]$. In the same vein we can prove the assertion for DAG$[i+1]$ and DAG$'[i]$.

◇

Claim 4.14.
If $\psi(I)$ has a quasi-model (K, w_0), then $I \in$ p-PW-SAT.

Proof of claim.
For clarity we illustrate our proof argumentation in Figure 4.6.

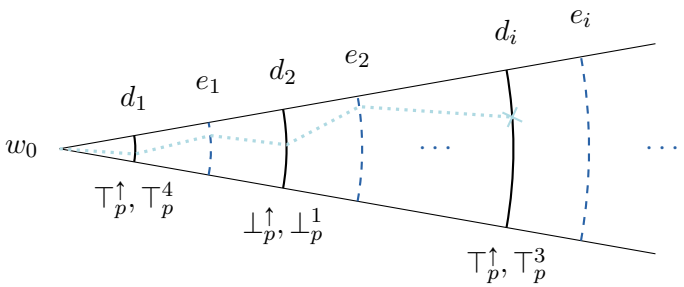


Figure 4.6: Example: $\mathsf{EG}\neg\top_p^3$ is false in w_0, Q_p has weight > 2.

By Claim 4.13 we know that K contains $2n$ nested dags, such that their frontier worlds have labeled the corresponding subformulas of $\beta_1^d, \beta_1^e, \beta_2^d, \beta_2^e, \ldots, \beta_n^d, \beta_n^e$ exactly in this order. The formula α expresses that the frontier of each β_i^d should, under the condition that it has $\top_p^{\uparrow}$ labeled, do the following: If $\top_p^j$ is labeled in some *reachable*, but *different* frontier, i.e., the counter for partition p was at j, then label $\top_p^{j+1}$, i.e., set the counter to $j+1$. Same argumentation applies to $\bot_p^j$.
We initialise the counting with $\top_p^1$ and $\bot_p^1$ if $\top_p^{\uparrow}$, resp., $\bot_p^{\uparrow}$ holds. Choosing a non-saturated assignment for φ contradicts the last part of $\psi(I)$. Consequently we obtain a satisfying and saturated assignment for φ from $L(w_0)$.
It is a simple matter to construct a model of $\psi(I)$ from a satisfying and saturated assignment of formula φ. A chain of length $2n+1$ satisfies the requirements for a model of $\psi(I)$ and consequently it has a constant temporal depth.

◇

Next we have to show that the path-width is bounded. First we consider the subformula α. We can easily split it into a path-decomposition of width $O(k)$ and length n. The path-decomposition of the residual subformulae can simply be appended to the path-decomposition of α, as α has only $O(k)$ subformulae in common with the other formulae, including α itself. Now we need to show that the remaining subformulae β_i^d, β_i^e and $\psi(I)$ also have a path-decomposition of bounded width. Starting from a decomposition of the primal graph of φ, the other formulae can be placed by bag augmentation as in Lemma 4.9, again leading to a total syntax circuit path-width of $O(k + \mathrm{pw}^*(\varphi))$. Consequently, the lemma applies. □

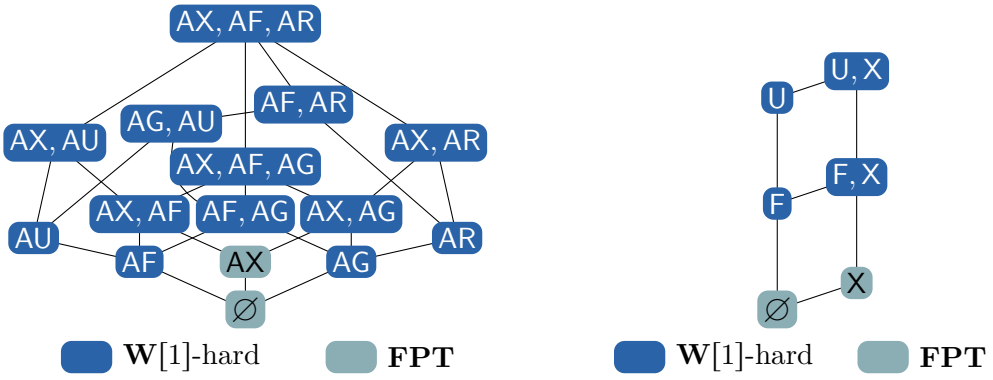
Pay attention, in the sense described above the subformula α has an unbounded path-width according to the syntax tree representation. Therefore, Lemma 4.12 only refers to the treewidth for the circuit representation.

We now turn to the satisfiability of linear temporal logic without the X-operator.

Lemma 4.15.
If $\mathsf{X} \notin T$, then LTL-Sat(T) is **W**[1]-hard when parameterized by $\mathrm{td} + \kappa$ where $\kappa \in \{\mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{S}}, \mathrm{tw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{C}}\}$.

Proof. We prove **W**[1]-hardness for the parameterized LTL-Sat(T) problem in a similar vein as Lemma 4.10. For this approach we have to adapt the formulae from the proof of Lemma 4.10 to LTL. The formula $\psi(I) \in \mathcal{LTL}(\mathsf{G})$ is a conjunction of the following subformulae.

$$
\begin{aligned}
\psi[\text{formula}] \quad &:= \varphi \\
\psi[\text{depth}] \quad &:= \mathsf{G} \bigwedge_{i=0}^{n-1} \big[(d_i \wedge \neg d_{i+1}) \rightarrow (m_{i \bmod 2} \wedge \neg m_{1-(i \bmod 2)} \\
&\qquad\qquad \wedge \neg\mathsf{G}\neg(d_{i+1} \wedge \neg d_{i+2}))\big] \\
\psi[\text{fixed-}Q] \quad &:= \bigwedge_{i=1}^{n} [(q_i \rightarrow \mathsf{G}q_i) \wedge (\neg q_i \rightarrow \mathsf{G}\neg q_i)] \\
\psi[\text{signal}] \quad &:= \mathsf{G} \bigwedge_{i=1}^{n} \Big[(d_i \wedge \neg d_{i+1}) \rightarrow \Big(\left(q_i \leftrightarrow \top_{p(i)}^{\uparrow}\right) \\
&\qquad\qquad \wedge \left(\neg q_i \leftrightarrow \bot_{p(i)}^{\uparrow}\right) \Big) \Big] \\
\psi[\text{count}] \quad &:= \mathsf{G} \bigwedge_{p=1}^{k} \bigwedge_{j=0}^{|Q_p|} \bigwedge_{i=0}^{1} \Big[\left(\top_p^{\uparrow} \wedge \top_p^{j} \wedge m_i\right) \rightarrow \mathsf{G}\left(m_{1-i} \rightarrow \mathsf{G}\top_p^{j+1}\right) \Big] \\
&\qquad\qquad \wedge \Big[\left(\bot_p^{\uparrow} \wedge \bot_p^{j} \wedge m_i\right) \rightarrow \mathsf{G}\left(m_{1-i} \rightarrow \mathsf{G}\bot_p^{j+1}\right) \Big]
\end{aligned}
$$



(a) CTL-SAT($\mathcal{T}$) (b) LTL-SAT($\mathcal{T}$)

Figure 4.7: Parameterized complexity of the satisfiability problem of CTL and LTL, parameterized by circuit path-width or treewidth, and temporal depth.

$$\psi[\text{monotone}] := \mathsf{G}\left[\bigwedge_{i=1}^{n}(d_i \rightarrow d_{i-1}) \wedge \bigwedge_{p=1}^{k}\bigwedge_{j=1}^{|Q_p|+1}\left[\left(\top_p^j \rightarrow \top_p^{j-1}\right) \wedge \left(\bot_p^j \rightarrow \bot_p^{j-1}\right)\right]\right]$$

$$\psi[\text{target}] := \mathsf{G}\bigwedge_{p=1}^{k}\left[d_{n+1} \rightarrow \left(\top^{C_p} \wedge \neg\top^{C_p+1} \wedge \bot_p^{|Q_p|-C_p} \wedge \neg\bot_p^{|Q_p|-C_p+1}\right)\right]$$

$$\psi[\text{init}] := d_0 \wedge \neg d_1 \wedge \mathsf{G}\bigwedge_{p=1}^{k}\left[\top_p^0 \wedge \bot_p^0\right]$$

In Lemma 4.7 we have shown that the path-width of $\psi(I)$ is bounded. The path-width remains bounded when replacing G by F, U, or R in the given formulae for the other cases. In Lemma 4.10 we have shown the correctness by the argumentation that there is at least one path with correct counting of variables in the respective partitions. Obviously, the given LTL formula is such a path formula, which ensures the same behaviour on a single path.

□

We summarize the results in Figure 4.7. In the next section we want to examine the satisfiability problem in temporal logics parameterized by either temporal depth or treewidth.

4.1.3.2 Parametrization only by Temporal Depth or Treewidth

In this section we will see, that even if we have the *next* operator, we are not able to obtain fixed parameter tractability for the satisfiability problem in temporal logics if we do not have both parameters temporal depth and treewidth.
First we consider the parametrization by temporal depth only, and establish the following results.

Theorem 4.16.
When parameterized by temporal depth, CTL-Sat(T) is para-**NP**-complete for $T = \varnothing$, para-**PSPACE**-complete if $T = \{\mathsf{AG}\}$, or $\{\mathsf{AF}\} \subseteq T \subseteq \{\mathsf{AF}, \mathsf{AX}\}$, and para-**EXPTIME**-complete if T contains AR, AU, $\{\mathsf{AG}, \mathsf{AF}\}$, or $\{\mathsf{AG}, \mathsf{AX}\}$.

Proof. [LM15] have shown for all fragments except the fragment $\{\mathsf{AX}\}$ the classical hardness results. Applying those results in combination with Theorem 2.2 yields the assertion of the lemma. □

Theorem 4.17.
When parameterized by temporal depth, LTL-Sat(T) is para-**NP**-complete if $T \subseteq \{\mathsf{X}\}$ and $T \subseteq \{\mathsf{F}\}$ and para-**PSPACE**-complete otherwise.

Proof. [DS02] have shown that LTL-Sat(T) is **PSPACE**-complete for $T \subseteq \{\mathsf{F}\}$ or **NP**-complete if $T \subseteq \{\mathsf{X}\}$. Again we apply Theorem 2.2 to the results of [DS02] establishing the assertion of the lemma. □

We turn to the examination of the second case, where we use path-width, or treewidth as the only parameter. We will use the reduction from p-PW-Sat again. Nested AX-operators will compensate for missing temporal depth. At this juncture, we again want to underline that the choice of representing structure leads to different results in terms of computational complexity when we use AX, resp. X. The satisfiability problem for those fragments is easier if we use the circuit representation.

Lemma 4.18.
CTL-Sat($\{\mathsf{AX}\}$) and LTL-Sat($\{\mathsf{X}\}$) are $\mathbf{W}[1]$-hard when parameterized by $\mathrm{tw}_{\mathcal{C}}$ or $\mathrm{pw}_{\mathcal{C}}$.

Proof. For the proof we modify the formulae trom the proof of Lemma 4.7 as follows: we substitute $\mathsf{AG}\alpha$ in each formula by $\bigwedge_{j=0}^{n} \mathsf{AX}^j\alpha$, where the AX^j is the j-fold nesting of AX operators. In the next step we need to show that the path-width of the syntax circuit does not increase too much. We illustrate the idea in Figure 4.8.

We delete the node AG in the syntax circuit and introduce $2n$ new vertices, $v_1, \ldots, v_n, u_0, \ldots, u_{n-1}$ instead, where each v_i represents an AX and each u_i represents a binary $\wedge$. u_0 has as parents every parent of $\mathsf{AG}\alpha$, and each u_i with $1 \leqslant i < n-1$ has as a child v_i and u_{i+1}. u_0 has α and u_1 as its children, while u_{n-1} has v_{n-1} and v_n. Furthermore v_0 has α as only child, and each v_i with $i > 0$ has v_{i-1} as child.

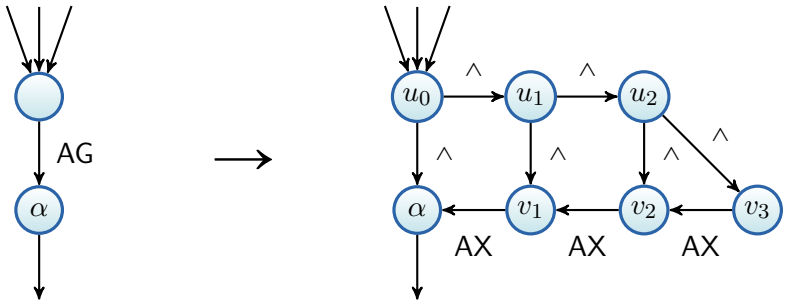


Figure 4.8: Circuit transformation from AG to nested AX for $n = 3$.

To see that the path-width of the resulting syntax circuit structure is still low, proceed as follows: As seen in Figure 4.8, the $2n$ vertices have a path-decomposition of width four and length n. Append this decomposition to $\mathcal{P}$ to obtain $\mathcal{P}'$, where $\mathcal{P}$ is an optimal path-decomposition of the formula before AG is replaced. To avoid the connectedness condition of path-decompositions being violated in $\mathcal{P}'$, add u_0 and α to each bag of $\mathcal{P}'$. The resulting path-width is still low as only constantly many AG operators have to be replaced in Lemma 4.7.

For LTL-Sat({X}), proceed exactly like for the CTL case but replace AX by X. As the branching semantics of CTL is not required in the reduction, and in fact AX occurs only positively, the reduction stays correct. □

To show the computational complexity of CTL-Sat({AX}) parameterized by $\mathrm{tw}_{\mathcal{S}}$, we first require one interim result about complexity of modal logic. Therefore we define KD-ML$_1$-Sat as the set of modal formulae φ that have at most one propositional variable and are satisfied by a serial Kripke structure.

Lemma 4.19 ([Hal95])**.**
KD-ML$_1$-Sat is **PSPACE**-complete.

Lemma 4.20.
CTL-Sat({AX}) parameterized by $\mathrm{tw}_{\mathcal{S}}$ is para-**PSPACE**-complete.

Proof. The idea of the proof is a reduction from KD-ML$_1$-Sat. The reduction function reduce the number of propositional variables by replacing them by so-called *primitive-proposition-like* (pp-like) formulae. These formulae are "sufficiently independent" from each other, consequently they can be used as an immediate replacement for propositional variables occurring in a model formula. The **PSPACE**-hardness proof can easily be modified for serial Kripke frames corresponding to the family $\varphi_1, \varphi_2, \ldots$ of pp-like formulae used by Halpern stays pp-like in such frames.
Now we need to show that the treewidth does not increase too much.
Note that a syntax tree with only one proposition has treewidth at most two. Consequently

the problem CTL-SAT($\{\mathsf{AX}\}$) is **PSPACE**-hard. As KD-ML-SAT which is equivalent to CTL-SAT($\{\mathsf{AX}\}$) is in **PSPACE**, the completeness follows. □

4.1.4 Parameterized Complexity of Satisfiability in Post's Lattice

Here we want to examine the temporal satisfiability problem restricted to Boolean fragments in Post's lattice with the parametrization over temporal depth in combination with treewidth or path-width respectively. We get a dichotomy similar to the one in propositional logic. Except for two open cases we get a complete classification: The temporal satisfiability problem is equivalent to the unrestricted case and therefore **W**[1]-hard as soon as the basis B is a superclone of S_1, otherwise the temporal satisfiability problem is fixed-parameter tractable. The cases that remains open are the clones $\mathsf{L} = [\oplus, \top]$ and $\mathsf{L}_0 = [\oplus]$.

Lemma 4.21.
Let $\kappa \in \{\mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{S}}, \mathrm{tw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{C}}\}$, T be a set of CTL-operators, T' be a set of LTL-operators, and B be a finite set of Boolean functions. If $\mathsf{S}_1 \subseteq [B]$, then

$$\text{CTL-SAT}(\{\wedge, \vee, \neg\}, T, \kappa + \mathrm{td}) \leqslant^{fpt} \text{CTL-SAT}(B, T, \kappa + \mathrm{td}),$$
$$\text{LTL-SAT}(\{\wedge, \vee, \neg\}, T', \kappa + \mathrm{td}) \leqslant^{fpt} \text{LTL-SAT}(B, T', \kappa + \mathrm{td}).$$

Proof. The idea of the proof is a reduction similar to one from [MMTV09] and the additional consideration that $\mathsf{BF} = [\{\wedge, \vee, \neg\}] = [\mathsf{S}_1 \cup \{\top\}] = [B \cup \{\top\}]$. We know that $\mathsf{E}_0 \subseteq \mathsf{S}_1$ and therefore $\wedge \in [B]$. Now we need to simulate $\top$ in all Boolean functions. Therefore we will use the concept from propositional logic which is known by the name "*Lewis knack*" [Lew79]. Using Lewis knack we need to add to every subformula $\alpha \in \mathrm{SF}(\varphi)$ of a given formula φ a new variable t, to be more concrete $\alpha \wedge t$. In the next step we substitute $\top$ by t. This substitution is legitimate, as t has to hold in every relevant world of a model of φ. We add $\wedge t$ to a subformulae, which start with a temporal operator. To be more precise we transform each $O(\alpha) \in \mathrm{SF}(\varphi)$ to $O(\alpha \wedge t)$, for $O \in T$, or $O \in T'$ respectively. As a result we obtain about twice as many subformulae in the transformed formula φ'. Using the base B, we represent $\alpha \wedge t$ as some fixed function

$$f(\underbrace{\alpha, \ldots, \alpha}_{c \text{ times}}, \underbrace{t, \ldots, t}_{c' \text{ times}}),$$

where f is a function composed of symbols of B and c, c' are constants depending on the base B. In general $c, c' > 1$ as a short representation of $\wedge$ in the base B does not necessarily exists. The blowup factor of the formula φ' is c^{td} and therefore exponential in general, but only FPT with parameter td, viz. $|\varphi'| = |\varphi| \cdot c^{\mathrm{td}}$. The size of φ is also a bound for the runtime of the reduction within a polynomial factor. An illustration of the transformation is shown in the Figure 4.9.

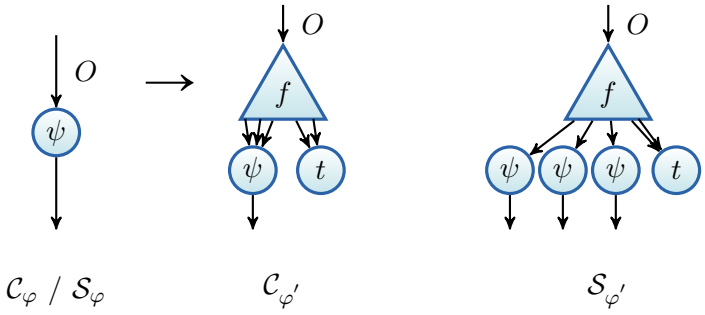


Figure 4.9: Transformation of the syntax tree and syntax circuit for $c = 3$.

It remains to prove that the the parameter $\kappa + \mathrm{td}$ does not increase too much. It is obvious that td does not change, as we do not add temporal operators to the subformulae. For the parameter path-width and treewidth assume the case where we represent φ' as a syntax tree $\mathcal{S}_{\varphi'}$. For each $\alpha \wedge t$, $\alpha \in \mathrm{SF}(\varphi)$ we obtain a subformula

$$f(\underbrace{\alpha, \ldots, \alpha}_{c \text{ times}}, \underbrace{t, \ldots, t}_{c' \text{ times}}).$$

Let F be the set of nodes of the local substructure which represents f. We construct an optimal tree-decomposition or path-decomposition of $\mathcal{S}_{\varphi'}$ in the following way: For each bag $\mathcal{B}$ and every node $u \in \mathcal{B}$ in the substructure representing $\wedge$ we require only constantly many more variables. Formally, define the new bag as $\mathcal{B}' := \mathcal{B} \cup \{t\} \cup \{u \mid u \in F \text{ for some } F \text{ simulating some } \wedge \text{ in } \mathcal{B}\}$. As the size of F is a constant δ and depends only on B, we have $|\mathcal{B}'| \leqslant \delta \cdot |\mathcal{B}| + 1$. Consequently, the reduction is bounded in the parameter. The case of the syntax circuit representation is proven analogously and consequently the proof is complete.

□

Now we are in the position to summarize our results in the following theorem.

Theorem 4.22.
Let B be a finite set of Boolean functions s.t. $[B] \notin \{\mathsf{L}, \mathsf{L}_0\}$. Then for $\kappa \in \{\mathrm{tw}_{\mathcal{S}}, \mathrm{pw}_{\mathcal{S}}, \mathrm{tw}_{\mathcal{C}}, \mathrm{pw}_{\mathcal{C}}\}$ the problems CTL-Sat(B), LTL-Sat(B) and CTL*-Sat(B) parameterized by $\kappa + \mathrm{td}$ are

(1.) **W**[1]-hard if $\mathsf{S}_1 \subseteq [\mathcal{B}]$,

(2.) in **FPT** otherwise.

Proof. With the combination of Lemma 4.7 and Theorem 4.15 we obtain the intractability for the case $\mathsf{S}_1 \subseteq [\mathcal{B}]$. The fixed-parameter tractable cases directly follow from the

Problem Q	Parameter κ		
CTL-SAT($\cdot$)	td	tw / pw	td + tw / td + pw
AX	para-**NP**-h.[4.16]	(v.i.)[b]	**FPT**[4.5]
AF	para-**PSPACE**-c.[4.16]	**W**[1]-h.[4.12]	**W**[1]-h.[4.12,a]
AF, AX	para-**PSPACE**-c.[4.16]	**W**[1]-h.[4.9,c]	**W**[1]-h.[4.9]
AG	para-**PSPACE**-c.[4.16]	**W**[1]-h.[4.10]	**W**[1]-h.[4.10]
other	para-**EXPTIME**-c.[4.16]	**W**[1]-h.[4.7,4.4,4.11]	**W**[1]-h.[4.7,4.4,4.11]
LTL-SAT($\cdot$)	td	tw / pw	td + tw / td + pw
X	para-**NP**-c.[4.17]	(v.i.)[δ]	**FPT**[4.5]
F	para-**NP**-c.[4.17]	**W**[1]-h.[4.15]	**W**[1]-h.[4.15]
other	para-**PSPACE**-c.[4.17]	**W**[1]-h.[4.15]	**W**[1]-h.[4.15]
CTL*-SAT($\cdot$)	td	tw / pw	td + tw / td + pw
A, X	para-**NP**-h.[4.17]	(v.i.)[b]	**FPT**[4.4]
other	para-**EXPTIME**-h.[4.16]	**W**[1]-h.[4.15]	**W**[1]-h.[4.15]

[a]: Only for $\mathcal{C}$, open for $\mathcal{S}$.
[b]: para-**PSPACE**-c. for $\mathrm{tw}_{\mathcal{S}}$ [4.20], **W**[1]-h. for $\mathrm{tw}_{\mathcal{C}}$ and $\mathrm{pw}_{\mathcal{C}}$ [4.18], open for $\mathrm{pw}_{\mathcal{S}}$.
[c]: para-**PSPACE**-c. for $\mathrm{tw}_{\mathcal{S}}$ [LM15].
[δ]: **FPT** for $\mathrm{tw}_{\mathcal{S}}$ and $\mathrm{pw}_{\mathcal{S}}$ [4.6], **W**[1]-h. for $\mathrm{tw}_{\mathcal{C}}$ and $\mathrm{pw}_{\mathcal{C}}$ [4.18].

Table 4.1: Classification of parameterized Temporal Logic

classification of the non-parameterized version by Meier et al.: They showed that for $[B] \notin \{\mathsf{L}, \mathsf{L}_0\}$ and $\mathsf{S}_1 \subseteq [\mathcal{B}]$ the problem is in **P**. □

Finally we want to summarize all result of this section in Table 4.1. Numbers in the exponent refer to the corresponding lemma, theorem, or corollary in this section. Unless stated otherwise the notion tw, resp., pw indicates both syntax circuits and trees.

4.2 Backdoors for Linear Temporal Logic

In this section, investigate the approach of Backdoors in linear temporal logic. In contrast to propositional logic we need to ensure that, whenever a propositional variable is in the backdoor set, then also all of its temporal literals are required to be in the backdoor set as well. Consequently, we need to consider assignments that are consistent between propositional variables and their temporal literals. Whereas in the propositional logic all of the assignments have to be considered. We will introduce the already known problems of backdoor detection and backdoor evaluation into the fragments of HORN and KROM formulae. Moreover, we classify the operator fragments of globally-operators for future $\square_F$, past $\square_P$ and always $\circledast$, as well as their combinations. As a parametrization we use the backdoor size. Let PROP be a finite set of propositions. First we want to recap the syntax of the globally fragment of LTL:

$$\varphi ::= \bot \mid \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \square_{\mathrm{P}}\,\varphi \mid \square_{\mathrm{F}}\,\varphi \mid \circledast\,\varphi,$$

where $p \in \mathsf{PROP}$. By $\square_{\mathrm{P}}\,\varphi$ we mean "φ holds in every point in the past", $\square_{\mathrm{F}}\,\varphi$ we mean "φ holds in every point in the future", and $\circledast\,\varphi$ states for "φ holds always". In the following, we will use abbreviations $\alpha \to \beta$ for $\neg\alpha \vee \beta$ and $\alpha \leftrightarrow \beta$ for $(\alpha \wedge \beta) \vee (\alpha \wedge \beta)$. In this section, we interpret $\mathcal{LTL}$ formulae over the structure $(\mathbb{Z}, <)$, which can be interpreted as a flow of time. For detail information on this approach, we refer to Gabbay et al. [GHR94]. It should be noted that all our results will also apply unchanged if the formulae are evaluated over the set of natural numbers instead of the set of all integers. In the next step we define the semantics for those temporal formulae.

Definition 4.6: Temporal Semantics

Let PROP be a finite set of propositions. A *temporal interpretation* $\mathcal{M} = (\mathbb{Z}, <, V)$ is a mapping from propositions to moments of time, viz. $V\colon \mathsf{PROP} \to \mathcal{P}(\mathbb{Z})$. The satisfaction relation $\models$ is then defined as follows, where $n \in \mathbb{Z}$, $\varphi, \psi \in \mathcal{LTL}$:

$\mathcal{M}, n \models \top$		always,
$\mathcal{M}, n \models \bot$		never,
$\mathcal{M}, n \models p$	iff	$n \in V(p)$,
$\mathcal{M}, n \models \neg\varphi$	iff	$\mathcal{M}, n \not\models \varphi$
$\mathcal{M}, n \models \varphi \vee \psi$	iff	$\mathcal{M}, n \models \varphi$ or $\mathcal{M}, n \models \psi$
$\mathcal{M}, n \models \varphi \wedge \psi$	iff	$\mathcal{M}, n \models \varphi$ and $\mathcal{M}, n \models \psi$
$\mathcal{M}, n \models \square_{\mathrm{F}}\,\varphi$	iff	for all $k > n$ it holds $\mathcal{M}, k \models \varphi$
$\mathcal{M}, n \models \square_{\mathrm{P}}\,\varphi$	iff	for all $k < n$ it holds $\mathcal{M}, k \models \varphi$
$\mathcal{M}, n \models \circledast\,\varphi$	iff	for all $k \in \mathbb{Z}$ it holds $\mathcal{M}, k \models \varphi$

We say that φ is *satisfiable* if there is a temporal interpretation $\mathcal{M}$ such that $\mathcal{M}, 0 \models \varphi$.

Then $\mathcal{M}$ is also referred to as a *(temporal) model (of φ)*. Sometimes we also directly write $\mathcal{M}(p)$ instead of explicitly defining V.

Every LTL formula which is evalueted over the frame $(\mathbb{Z}, <)$ has a satisfiability-equivalent formula in the separated normal form, SNF in short [FDP01]. We follow the notion of SNF formulae by Artale et al. [AKRZ13] and directly restrict them to the relevant globally fragment:

$$\lambda ::= \bot \mid p \mid \Box_{\mathrm{F}} \lambda \mid \Box_{\mathrm{P}} \lambda \mid \circledast \lambda, \tag{4.1}$$

$$\varphi ::= \lambda \mid \neg\lambda \mid \varphi \wedge \varphi \mid \boxdot(\neg\lambda_1 \vee \cdots \vee \neg\lambda_n \vee \lambda_{n+1} \vee \cdots \lambda_{n+m}), \tag{4.2}$$

where λ is called a *temporal literal* and φ is said to be in *clausal normal form*.

Let us remark that the operator $\Box_{\mathrm{F}}$ is often denoted as G in literature. But in contrast to the $\mathcal{LTL}$-expression $\mathsf{G}\varphi$, for $\Box_{\mathrm{F}} \varphi$ it is not required that φ holds in the current world. We differentiate fragments of $\mathcal{LTL}$ by adding superscripts and subscripts as follows. If $O \subseteq \{\Box_{\mathrm{F}}, \Box_{\mathrm{P}}, \circledast\}$ is an operator subset then $\mathcal{LTL}^{O}$ is the fragment of $\mathcal{LTL}$ consisting of formulae that are allowed to only use temporal operators from O for temporal literals. We also consider restrictions of the clausal normal form in (4.2):

$$\boxdot(\neg\lambda_1 \vee \cdots \vee \neg\lambda_n \vee \lambda_{n+1} \vee \cdots \lambda_{n+m}).$$

If $\alpha \in \{\mathsf{CNF}, \mathsf{HORN}, \mathsf{KROM}\}$ then $\mathcal{LTL}_{\alpha}$ is the set of formulae obeying the normal form α.

Next, we show that from certain $\mathcal{LTL}$ formulae we can efficiently construct SNF formulae while preserving their satisfiability, avoiding deep nesting of temporal operators.

Lemma 4.23 ([AKRZ13, Lemma 2])**.**
Let $\mathcal{L} \in \{\mathcal{LTL}_{\alpha}^{\Box_{\mathrm{F}},\Box_{\mathrm{P}}}, \mathcal{LTL}_{\alpha}^{\boxdot}\}$ be a formula class for $\alpha \in \{\mathsf{CNF}, \mathsf{HORN}, \mathsf{KROM}\}$. For any formula $\varphi \in \mathcal{L}$, one can construct in log-space an satisfiability-equivalent $\mathcal{L}$-formula $\Psi \wedge \circledast\, \Phi$, where Ψ is a conjunction of propositional variables from Φ, and Φ is a conjunction of clauses of the form (2) containing only $\Box_{\mathrm{F}}, \Box_{\mathrm{P}}$ for $\mathcal{LTL}_{\alpha}^{\Box_{\mathrm{F}},\Box_{\mathrm{P}}}$, and only $\circledast$ for $\mathcal{LTL}_{\alpha}^{\boxdot}$, in which the temporal operators are not nested.

From now on we consider formulae in the normal form $\Psi \wedge \circledast\, \Phi$ only. Using this normal form, we are in the position to give a formal definition of backdoors in LTL, but first we require the notion of *consistent assignment*.

Definition 4.7: Consistent Assignment

Let $\mathscr{O}$ be a set of operators. An assignment $\vartheta\colon \mathsf{Var}(\varphi) \cup \{Ox \mid x \in \mathsf{Var}(\varphi) \wedge O \in \mathscr{O}\} \to \{0,1\}$ is *consistent* if for every $x \in \mathsf{Var}(\varphi)$ it holds that if $\vartheta(\circledast\, x) = 1$, then also

$\vartheta(\Box_P x) = \vartheta(\Box_F x) = \vartheta(x) = 1$.

Based on this, we define strong backdoors for LTL in the following way:

Definition 4.8: Backdoors in LTL

Let $\mathcal{C}$ be a class of CNF-formulae, $\mathscr{O}$ be a set of operators, and φ be an $\mathcal{LTL}^{\mathscr{O}}_{\mathsf{CNF}}$ formula. A set $X \subseteq \mathsf{Var}(\varphi)$ is a *(strong) $\mathcal{C}$-backdoor* if for every consistent assignment $\vartheta\colon X \cup \{Ox \mid x \in X, O \in \mathscr{O}\} \to \{0,1\}$ it holds that $\varphi[\theta]$ is in $\mathcal{C}$.

Here, $\varphi[\theta]$ is called *reduct* analogously to the propositional approach on CNF-formulae, i.e., we delete all clauses with at least one satisfied literal and delete all falsified literals. Therefore, empty clauses may arise, which we substitute by false. Moreover, an empty formula is replaced by the constant true. As already discussed in the previous section of default logic, we first need to find a small backdoor set and subsequent evaluate this backdoor set on a given LTL-formula. Therefore we define the following two problems:

Problem:	DETECT$^{\mathscr{O}}(\mathcal{C})$ - Backdoor detection to LTL$^{\mathscr{O}}$
Input:	LTL$^{\mathscr{O}}_{\mathsf{CNF}}$ formula φ
Parameter:	$\kappa = k$, with $k \in \mathbb{N}$
Task:	Find a strong $\mathcal{C}$-backdoor of a size $\leqslant k$ if one exists.

Problem:	EVAL$^{\mathscr{O}}(\mathcal{C})$ - Backdoor evaluation to LTL$^{\mathscr{O}}$
Input:	LTL$^{\mathscr{O}}_{\mathsf{CNF}}$ formula φ, strong $(\mathcal{C}, \mathscr{O})$-backdoor set X
Parameter:	$\kappa = \lvert X \rvert$
Question:	Is φ satisfiable?

4.2.1 Backdoor Set Detection in LTL

In this section we study the problem of finding a strong $\mathcal{C}$-backdoor set of a size at most k and show the fixed-parameter tractability of this problem for $\mathcal{C} \in \{\mathsf{HORN}, \mathsf{KROM}\}$. It is only meaningful to search for backdoors with a target class that has polynomial time solvable satisfiability problem. Artale et al. [AKRZ13] have shown that satisfiability for $\mathcal{LTL}^{\boxplus}_{\mathsf{HORN}}$ and $\mathcal{LTL}^{\boxplus}_{\mathsf{KROM}}$ is solvable in polynomial time. To detect a backdoor set in $\mathcal{LTL}^{\boxplus}_{\mathcal{C}}$ we use a similar approach as [GS12] for the detection of strong backdoors for propositional CNF formulae.

With the next observation we show that only consistent assignments are potential backdoor candidates.

Observation 4.24.
Let $\varphi := \Psi \wedge \circledast\, \Phi$ be an $\mathcal{LTL}^{\Box_P,\Box_F,\circledast}$ formula. Then any clause C of Φ containing $\neg \circledast\, x$ and (at least) one of $\Box_P x$, $\Box_F x$ or x for some variable $x \in \mathsf{Var}(\varphi)$ is tautological and consequently can be removed from φ without changing the satisfiability of φ.

The correctness of the observation is straightforward, as if one of $\Box_{\mathrm{P}}\, x, \Box_{\mathrm{F}}\, x, x$ does not hold then $\neg \circledast\, x$ is true. This observation is crucial for the backdoor detection algorithm. Recall, that tautological clauses from the observation above are those clauses which are satisfied by every consistent assignment. Consequently, once these clauses are removed from the formula, it holds that for every clause C of φ there is a consistent assignment ϑ such that C is not satisfied by ϑ.
Next we show how to detect a strong backdoor of size at most k to obtain a HORN formula. The main idea for the proof is to give a reduction to the parameterized problem VertexCover, which is defined as follows:

Problem:	VertexCover
Input:	Undirected graph $G = (V, E)$
Parameter:	$\kappa = k$, with $k \in \mathbb{N}$
Question:	Does a *vertex cover* $C \subseteq V(G)$ with $\lvert C\rvert \leqslant k$ exists s.t. $C \cap e \neq \varnothing$ for every $e \in E(G)$.

Theorem 4.25.
For every $\mathscr{O} \subseteq \{\circledast, \Box_P, \Box_F\}$, Detect$^{\mathscr{O}}$(HORN) is fixed-parameter tractable.

Proof. Let $\mathscr{O} \subseteq \{\circledast, \Box_P, \Box_F\}$. The core concept of the proof is to construct an undirected graph $G = (V, E)$ for a given an $\mathcal{LTL}^{\mathscr{O}}$- formula $\varphi = \Psi \wedge \circledast\, \Phi$ s.t. the given formula φ has a strong HORN-backdoor set X with $\lvert X\rvert \leqslant k$ if and only if the constructed graph G has a vertex cover C with $\lvert C\rvert \leqslant k$.
We know due to Chen et al.[CKX10b] that the VertexCover problem parameterized by the solution size is fixed-parameter tractable and can be solved in time $\mathcal{O}(1.2738^k + k \cdot n)$, where $k = \lvert C\rvert$ is the size of the vertex cover and $n = \lvert V\rvert$ is the number of vertices of the input graph G.
Now we proceed with the construction of the desired graph G. It has as vertices the variables of φ, i.e., $V = \mathsf{Var}(\varphi)$, and two vertices v_1 and v_2 are connected by an edge if and only if there is a clause that contains at least two literals from $\{v_1, v_2\} \cup \{Ov_1, Ov_2 \mid O \in \mathscr{O}\}$. Note that in the case $v_1 = v_2$ the graph G contains a self-loop.

Claim 4.26. A set $X \subseteq \mathsf{Var}(\varphi)$ is a strong HORN-backdoor if and only if X is a vertex cover of G.

To show the forward direction of the claim, let $X \subseteq \mathsf{Var}(\varphi)$ be a strong HORN-backdoor set of φ. We claim that X is also a vertex cover of G. Suppose for the sake of contradiction that X is not a vertex cover of G, i.e., there is an edge $\{x, y\} \in E(G)$ such that $X \cap \{x, y\} = \varnothing$. As $\{x, y\} \in E(G)$, we obtain that there is a clause C in Φ that contains

at least two literals from $\{x, y\} \cup \{Ox, Oy \mid O \in \mathscr{O}\}$. Moreover, by Observation 4.24 there is a consistent assignment $\theta\colon X \cup \{\, Ox \mid x \in X \wedge O \in \mathscr{O}\} \to \{0, 1\}$ that falsifies all literals of C over the variables in X. Consequently, $\varphi[\theta]$ contains a sub-clause of C that still contains at least two literals from $\{x, y\} \cup \{\, Ox, Oy \mid O \in \mathscr{O}\}$. For this reason, $\varphi[\theta] \notin \mathsf{HORN}$, contradicting our assumption that X is a strong HORN-backdoor set of φ. Now we prove the reverse direction. Therefore let $X \subseteq V(G)$ be a vertex cover of G. We claim that X is also a strong HORN-backdoor of φ. Suppose for the sake of contradiction that this is not the case. Then there is an (consistent) assignment $\theta\colon X \cup \{Ox \mid x \in X \wedge O \in \mathscr{O}\} \to \{0, 1\}$ and a clause C in $\varphi[\theta]$ containing two positive literals, say, over variables x and y. We obtain that C contains at least two positive literals from $\{x, y\} \cup \{\, Ox, Oy \mid O \in \mathscr{O}\}$ and consequently G contains the edge $\{x, y\}$, contradicting our assumption that X is a vertex cover of G. $\diamond$

This concludes the proof.

□

To show the fixed-parameter tractability of $\textsc{Detect}^{\mathscr{O}}(\mathsf{KROM})$ we will reduce it to the problem 3HittingSet, which is defined as follows:

Problem:	3HittingSet
Input:	A universe U, a family $\mathcal{F}$ of subsets of U
Question:	Does a *hitting set* $S \subseteq U$ with $\lvert S\rvert \leqslant 3$ exist s.t. $S \cap F \neq \varnothing$ for every $F \in \mathcal{F}$.

Theorem 4.27.
For every $\mathscr{O} \subseteq \{\circledast, \Box_P, \Box_F\}$, $\textsc{Detect}^{\mathscr{O}}(\mathsf{KROM})$ is fixed-parameter tractable.

Proof. Let $\mathscr{O} \subseteq \{\circledast, \Box_P, \Box_F\}$.
The main idea of this proof is to construct a family $\mathcal{F}$ of subsets of a size 3 of a universe U s.t. φ has a strong KROM-backdoor set X with $|X| \leqslant k$ if and only if $\mathcal{F}$ has a hitting set S with $|S| \leqslant k$.
We know due to Abu-Khzam[Abu10] that the 3HittingSet-problem parameterized by the solution size is fixed-parameter tractable.
Now we proceed with the construction of the family $\mathcal{F}$. Therefore let $U = \mathsf{Var}(\varphi)$ and $\mathcal{F}$ contains the set $\mathsf{Var}(C)$ for every set C of exactly three literals contained in some clause of Φ.

Claim 4.28. A set $X \subseteq \mathsf{Var}(\varphi)$ is a strong KROM-backdoor if and only if X is a hitting set of $\mathcal{F}$.

Proof of claim. For the direction from left to right, let $X \subseteq \mathsf{Var}(\varphi)$ be a strong KROM-backdoor set of φ. Now suppose for the sake of contradiction that there is a set $F \in \mathcal{F}$ such that $X \cap F = \varnothing$. It follows from the construction of $\mathcal{F}$ that Φ contains a clause C containing

at least three literals over the variables in F. Moreover, because of Observation 4.24 there is a consistent assignment $\theta\colon X \cup \{\, Ox \mid x \in X \wedge O \in \mathscr{O}\} \to \{0,1\}$ that falsifies all literals of C over the variables in X. Consequently, $\varphi[\theta]$ contains a sub-clause of C that still contains at least three literals over the variables in F. As a result, $\varphi[\theta] \notin \mathsf{KROM}$, contradicting our assumption that X is a strong KROM-backdoor set of φ.
Next we prove the direction from right to left. Therefore, let $X \subseteq U$ be a hitting set of $\mathcal{F}$ and suppose for a sake of contradiction that there is a consistent assignment $\theta\colon X \cup \{Ox \mid x \in X \wedge O \in \mathscr{O}\} \to \{0,1\}$ and a clause C in $\varphi[\theta]$ containing at least three literals. Let C' be a set of at exactly three literals from C. It follows from the construction of $\mathcal{F}$ that $\mathcal{F}$ contains the set $\mathsf{Var}(C')$, however, $\mathsf{Var}(C') \cap X = \varnothing$, contradicting our assumption that X is a hitting set of G. $\diamond$

□

In this section we have proved that a strong backdoor set to the formula classes HORN and KROM can be found in **FPT** time, independently of the considered temporal operators. We continue with the investigation of the backdoor evaluation problem.

4.2.2 Evaluation of a Backdoor Set in LTL

From the previous section we know the backdoor set detection is possible in **FPT** time. In this section we will examine how efficiently we can use the detected backdoor set to answer the satisfiability question for a given $\mathcal{LTL}^{\boxtimes}$ formula.
We will see that the evaluation problem for $\mathcal{LTL}^{\boxtimes}$-$\mathsf{HORN}$ formulae is in **FPT**, whereas for $\mathcal{LTL}^{\boxtimes}$-$\mathsf{KROM}$ formulae it is **para-NP**-complete with backdoor size as parameter.
Before we can prove these results we require to depict some properties for of such formulae.

Let $\mathcal{M} = (\mathbb{Z}, <, V)$ be a temporal interpretation. We call $\mathsf{Var}(\mathcal{M})$ the set of propositions, in the following referred to as variables, for which V is defined. For a set of variables $X \subseteq \mathsf{Var}(\mathcal{M})$, we write $\mathcal{M}_{|X}$ for the *projection* of $\mathcal{M}$ onto X, which is defined as the temporal interpretation $\mathcal{M}_{|X} = (\mathbb{Z}, <, V_{|X})$, where $V_{|X}$ is only defined for the variables in X, viz. $V_{|X}(x) = V(x)$ for every $x \in X$.

For an integer z, $\mathbf{A}(\mathcal{M}, z)$ denotes the assignment $\vartheta\colon \mathsf{Var}(\mathcal{M}) \to \{0,1\}$ holding at world z in $\mathcal{M}$, i.e. $\vartheta(v) = 1$ if and only if $z \in \mathcal{M}(v)$ for every $v \in \mathsf{Var}(\mathcal{M})$. Moreover, for a set of worlds $Z \subseteq \mathbb{Z}$ we denote by $\mathbf{A}(\mathcal{M}, Z)$ the set of all assignments occurring in some world in Z of $\mathcal{M}$, i.e. $\mathbf{A}(\mathcal{M}, Z) := \{\, \mathbf{A}(\mathcal{M}, z) \mid z \in Z\}$. We also write $\mathbf{A}(\mathcal{M})$ for $\mathbf{A}(\mathcal{M}, \mathbb{Z})$. For an assignment $\vartheta\colon X \to \{0,1\}$, $\mathbf{W}(\mathcal{M}, \vartheta)$ denotes the set of all worlds $z \in \mathbb{Z}$ of $\mathcal{M}$ such that $\mathbf{A}(\mathcal{M}, z)$ is equal to ϑ on all variables in X.

Let $\varphi := \Psi \wedge \circledast\, \Phi \in \mathcal{LTL}^{\boxtimes}_{\mathsf{CNF}}$. We write $\mathbf{CNF}(\Phi)$ for the propositional CNF formula obtained from Φ after replacing each occurrence of $\circledast\, x$ in Φ with a fresh propositional variable with the same name. For a set of variables V and a set of assignments $\mathbb{A}$ of the variables in V, we denote by $\mathbf{G}(\mathbb{A}, V)\colon \{\, \circledast\, v \mid v \in V\} \to \{0,1\}$ the assignment defined

by setting $\mathbf{G}(\mathbb{A}, V)(\circledast v) = 1$ if and only if $\alpha(v) = 1$ for every $\alpha \in \mathbb{A}$. Moreover, if $\vartheta\colon V \to \{0,1\}$ is an assignment of the variables in V, $\mathbf{G}(\mathbb{A}, V, \vartheta)$ denotes the assignment defined by setting $\mathbf{G}(\mathbb{A}, V, \vartheta)(v) = \vartheta(v)$ and $\mathbf{G}(\mathbb{A}, V, \vartheta)(\circledast v) = \mathbf{G}(\mathbb{A}, V)(\circledast v)$ for every $v \in V$. For a set $\mathbb{A}$ of assignments over V and an assignment $\vartheta\colon V' \to \{0,1\}$ with $V' \subseteq V$, we call $\mathbb{A}(\vartheta)$ the set of all assignments $\alpha \in \mathbb{A}$ such that $\alpha(v) = \vartheta(v)$ for every $v \in V'$.

For a set $\mathbb{A}$ of assignments over some variables V and a subset $V' \subseteq V$, we write $\mathbb{A}|_{V'}$ for the *projection* of $\mathbb{A}$ onto V', which is defined as the set of assignments $\alpha \in \mathbb{A}$ restricted to the variables in V'.

With the next lemma we show how to translate a temporal formula into separated satisfiability checks for propositional formulae.

Lemma 4.29.
Let $\varphi := \Psi \wedge \circledast \Phi \in \mathcal{LTL}^{\circledast}$. Then, φ is satisfiable if and only if there is a set $\mathbb{A}$ of assignments of the variables in φ and an assignment $\alpha_0 \in \mathbb{A}$ such that α_0 satisfies Ψ and for every assignment $\alpha \in \mathbb{A}$ it holds that $\mathbf{G}(\mathbb{A}, \mathsf{Var}(\varphi), \alpha)$ satisfies the propositional formula $\mathbf{CNF}(\Phi)$.

Proof. First we prove the forward direction and assume that $\varphi := \Psi \wedge \circledast \Phi$ is satisfiable. Furthermore let $\mathcal{M}$ be a corresponding temporal interpretation witnessing the satisfiability of φ. Then we set $\mathbb{A} := \mathbf{A}(\mathcal{M})$ and $\alpha_0 := \mathbf{A}(\mathcal{M}, 0)$, which satisfies the conditions of the lemma.
For the reverse direction, let $\mathbb{A} := \{\alpha_0, \ldots, \alpha_{|\mathbb{A}|}\}$ be a set of assignments satisfying the statement of the lemma. Then we claim that $\mathcal{M} = (\mathbb{Z}, <, V)$ is a temporal interpretation which satisfies the formula φ. Let $\mathbb{Z}_{<0}$ be a set of all integers smaller then 0 and let $\mathbb{Z}_{>\mathbb{A}}$ be a set of all integers greater then $|\mathbb{A}|$. Then for every variable $v \in \mathsf{Var}(\varphi)$, the set $\mathcal{M}(v)$ contains the set $\{\, z \mid \alpha_z(v) = 1 \wedge 0 \leqslant z \leqslant |\mathbb{A}| \}$. In addition, if $\alpha_0(v) = 1$, $\mathcal{M}(v)$ then also contains the set $\mathbb{Z}_{<0}$ and if $\alpha_{|\mathbb{A}|}(v) = 1$, then $\mathcal{M}(v)$ also contains the set $\mathbb{Z}_{>\mathbb{A}}$. It is obvious that $\mathcal{M}, 0 \models \varphi$ holds, which completes the proof.

□

Using this lemma we can prove the following statement.

Lemma 4.30.
Let $\varphi := \Psi \wedge \circledast \Phi \in \mathcal{LTL}^{\circledast}$ and $X \subseteq \mathsf{Var}(\varphi)$. Then φ is satisfiable if and only if there is a set Θ of assignments of the variables in X, an assignment $\vartheta_0 \in \Theta$, a set $\mathbb{A}$ of assignments of the variables in $\mathsf{Var}(\varphi)$, and an assignment $\alpha_0 \in \mathbb{A}$ satisfying following conditions:

- C_1) the set Θ is equal to $\mathbb{A}_{|X}$,
- C_2) the assignment ϑ_0 is equal to $\alpha_0|_X$,
- C_3) $\mathbb{A}$ and α_0 satisfy the conditions stated in Lemma 4.29, and
- C_4) $|\mathbb{A}(\vartheta)| \leqslant |\mathsf{Var}(\varphi) \backslash X| + 1$ for every $\vartheta \in \Theta$.

Proof. First we show the forward direction. Therefore we assume that φ is satisfiable. Due to Lemma 4.29 we know that there is a set $\mathbb{A}$ of assignments of $\mathsf{Var}(\varphi)$ and and an assignment $\alpha_0 \in \mathbb{A}$ that satisfies the conditions of Lemma 4.29. Now we set Θ to be equal to the assignment set $\mathbb{A}$ restricted to X and ϑ_0 to be equal to $\alpha_0|X$. Consequently, we satisfy the conditions C_1) to C_3). In the next step, we show that there is a subset $\mathbb{A}'$ of $\mathbb{A}$, that satisfies all conditions C_1) to C_4) of the lemma. Towards showing this consider any subset $\mathbb{A}'$ of $\mathbb{A}$ that satisfies the following three conditions:

(1.) $\alpha_0 \in \mathbb{A}'$

(2.) for every $\vartheta \in \Theta$ it holds that $\mathbb{A}'(\vartheta) \neq \varnothing$, and

(3.) for every variable v of φ and every $b \in \{0,1\}$ it holds that there is an assignment $\alpha \in \mathbb{A}$ with $\alpha(v) = i$ if and only if there is an assignment $\alpha' \in \mathbb{A}'$ with $\alpha'(v) = i$

Observe that conditions (1.) and (2.) ensure that $\mathbb{A}'$ satisfies conditions C_1) and C_2) and condition (3) ensures C_3). Consequently, any subset $\mathbb{A}'$ of $\mathbb{A}$ satisfying (1.) - (3.) satisfies also the conditions C_1) to C_3) of the lemma. Now we need to show that such subset $\mathbb{A}'$ also satisfies the last condition C_4) of the lemma. Therefore let $\mathbb{A}'_0$ be a subset of $\mathbb{A}$ containing α_0 as well as one arbitrary assignment $\alpha \in \mathbb{A}(\vartheta)$ for every $\vartheta \in \Theta$. Thereby $\mathbb{A}'_0$ satisfies the conditions (1.) - (3.) for every variable $v \in X$. Note that, if there is a variable v of φ such that the condition (3.) is violated by $\mathbb{A}'_0$, then it suffices to add at most one additional assignment to $\mathbb{A}'_0$ in order to satisfy the condition (3.) for the variable v. To ensure this third condition for every variable $v \in \mathsf{Var}(\varphi)\backslash X$ we add to $\mathbb{A}'_0$ at most $|\mathsf{Var}(\varphi)\backslash X|$ assignments and obtain a subset $\mathbb{A}'$ which satisfies the conditions of the lemma.
The reverse direction follows immediately from Lemma 4.29, due to the existence of the set of assignments $\mathbb{A}$ and the assignment α_0 satisfying condition C_3 imply the satisfiability of φ.

□

Roughly speaking, this lemma says that it suffices to consider only a set of assignments $\mathbb{A}$ of size linear in the number of variables, instead of exponential size, to decide the satisfiability of an $\mathcal{LTL}^{\boxplus}$ formula.
Finally we are in the position to prove the tractability of the evaluation problem of strong HORN-backdoor sets.

Theorem 4.31.
The evaluation problem $\textsc{Eval}^{\boxplus}(\mathsf{HORN})$ is in **FPT**.

Proof. Let $\varphi := \Psi \wedge \boxplus \Phi \in \mathcal{LTL}^{\boxplus}$ and let $X \subseteq \mathsf{Var}(\varphi)$ be a strong HORN-backdoor of φ. The general idea is to construct for every set of assignments Θ of the backdoor variables and for $\vartheta_0 \in \Theta$ a propositional HORN formula F_{Θ,ϑ_0}. Thereby F_{Θ,ϑ_0} is satisfiable if and only if there is a set of assignments $\mathbb{A}$ of the variables in $\mathsf{Var}(\varphi)$ and an assignment $\alpha_0 \in \mathbb{A}$ satisfying the conditions of Lemma 4.30. According to this Lemma 4.30 it holds that φ is satisfiable if and only if there is such a set Θ of assignments and a assignment $\vartheta_0 \in \Theta$

for which F_{Θ,ϑ_0} is satisfiable.
Note that there are at most $2^{2^{|X|}}$ such sets of assignments Θ and $2^{|X|}$ assignments ϑ_0, where for each of these sets the formula F_{Θ,ϑ_0} is a HORN-formula. Consequently, we can verify whether there are Θ and ϑ_0 where F_{Θ,ϑ_0} is satisfied, and therefore to verify the satisfiability of the formula φ, in $\mathcal{O}(2^{2^{|X|}} \cdot 2^{|X|} \cdot |F_{\Theta,\vartheta_0}|)$ time.
To obtain the tractability of Eval$^{\boxplus}$(HORN), we prove that the length of the formula F_{Θ,ϑ_0} is bounded, which implies that the length of φ is bounded as well.
Now we construct for a fixed set of assignments Θ and a fixed assignment $\vartheta_0 \in \Theta$ the formula F_{Θ,ϑ_0} and prove that it meets the conditions of Lemma 4.30.
Let $R := \mathsf{Var}(\varphi)\backslash X$ and $r := |R|+1$. For a propositional formula F, a subset $V \subseteq \mathsf{Var}(F)$, an integer i and a label s, we call $\mathbf{copy}(F, V, i, s)$ the propositional formula obtained from F after replacing each occurrence of a variable $v \in V$ with a new variable v^i_s. We require the following auxiliary formulae.
For every $\vartheta \in \Theta\backslash\vartheta_0$, let $F^{\vartheta}_{\Theta,\vartheta_0}$ be the formula:

$$\bigwedge_{1 \leqslant i \leqslant r} \mathbf{copy}(\mathbf{CNF}(\Phi[\mathbf{G}(\Theta, X, \vartheta)]), R, i, \vartheta).$$

Moreover, let $F^{\vartheta_0}_{\Theta,\vartheta_0}$ be the formula:

$$\mathbf{copy}(\Psi[\vartheta_0] \wedge \mathbf{CNF}(\Phi[\mathbf{G}(\Theta, X, \vartheta_0)]), R, 1, \vartheta_0) \wedge \bigwedge_{2 \leqslant i \leqslant r} \mathbf{copy}(\mathbf{CNF}(\Phi[\mathbf{G}(\Theta, X, \vartheta_0)]), R, i, \vartheta_0).$$

Note that the formula $F^{\vartheta}_{\Theta,\vartheta_0}$ is a HORN-formula for every $\vartheta \in \Theta$, as X is a strong HORN-backdoor set and the formula Ψ consists of unit clauses only.
In addition we require the propositional formula F_{const}, which ensures the consistency between the propositional variables $\boxplus x$ and the variables in $\{ x^i_\vartheta \mid \vartheta \in \Theta \wedge 1 \leqslant i \leqslant r\}$ for every $x \in \mathsf{Var}(\varphi)\backslash X$. The formula F_{const} consists of the following clauses:

- For every $\vartheta \in \Theta$, i with $1 \leqslant i \leqslant r$, and $v \in R$, the clause $\boxplus v \rightarrow v^i_\vartheta = \neg \boxplus v \vee v^i_\vartheta$ and
- for every $v \in R$ the clause

$$\neg \boxplus v \rightarrow \bigvee_{\vartheta \in \Theta \wedge 1 \leqslant i \leqslant r} \neg v^i_\vartheta = \boxplus v \vee \bigvee_{\vartheta \in \Theta \wedge 1 \leqslant i \leqslant r} \neg v^i_\vartheta.$$

Due to the form of these clauses, F_{const} is a HORN formula.
Finally, the formula F_{Θ,ϑ_0} is defined as:

$$F_{\Theta,\vartheta_0} := \bigwedge_{\vartheta \in \Theta} F^{\vartheta}_{\Theta,\vartheta_0} \wedge F_{\text{const}}.$$

Note that F_{Θ,ϑ_0} is HORN and the length of F_{Θ,ϑ_0} is at most

$$\begin{aligned}|F_{\Theta,\vartheta_0}| &\leqslant \sum_{\vartheta\in\Theta} |F^{\vartheta}_{\Theta,\vartheta_0}| + |F_{\text{const}}| \\ &\leqslant 2^{|X|}(|\mathsf{Var}(\varphi)\backslash X| + 1)(|\Phi| + |\Psi|) + 2 \cdot 2^{|X|} \cdot (|\mathsf{Var}(\varphi)\backslash X| + 1)^2\end{aligned}$$

and consequently bounded by a function of $|X|$ times a polynomial in the input size.
Now it remains to prove that $F_{\Theta,\vartheta}$ is satisfiable if and only if there is a set $\mathbb{A}$ of assignments of the variables in $\mathsf{Var}(\varphi)$ and an assignment $\alpha_0 \in \mathbb{A}$ satisfying the conditions of Lemma 4.30. Observe that for every $\vartheta \in \Theta$, each of the r copies of the formula $\mathbf{CNF}(\Phi[\mathbf{G}(\Theta, X, \vartheta)])$ represent one of the at most r assignments in $\mathbb{A}(\vartheta)$, the formula $F^{\vartheta_0}_{\Theta,\vartheta_0}$ ensures that the assignment chosen for α_0 satisfies Ψ and the formula F_{const} ensures that the "global assignments" represented by the propositional variables $\circledast\, x$ are consistent with the set of local assignments in $\mathbb{A}$ represented by the variables in $\{\, x^i_\vartheta \mid \vartheta \in \Theta \wedge 1 \leqslant i \leqslant r\}$ for every $x \in \mathsf{Var}(\varphi)\backslash X$. As a consequence, we obtain the tractability of Eval$^{\circledast}$(HORN).

□

With the last result we have shown the tractability for the backdoor set evaluation problem of HORN-formulae. HORN-formulae are characterized by containing only one positive literal in each clause, where the clause size is not limited. Surprisingly, the backdoor set evaluation problem for KROM-formulae, which clause size is limited to two literals, is more intricate.

Theorem 4.32.
Eval$^{\circledast}$(KROM) is **paraNP**-complete.

Proof. Artale et al. [AKRZ13, Table 1] proved the **NP** membership of the satisfiability of $\mathcal{LTL}^{\circledast}_{\mathsf{CNF}}$. Consequently Eval$^{\circledast}$(KROM) $\in$ **paraNP**.
For the hardness result we give a polynomial time reduction from the **NP**-hard problem 3Col to Eval$^{\circledast}$(KROM) for a backdoor set X, with $|X| = 2$. In 3Col we ask whether a given input graph $G = (V, E)$ has a coloring $f\colon V(G) \to \{1, 2, 3\}$ of its vertices such that $f(v) \neq f(u)$ for every edge $\{u, v\}$ of G. Given such a graph $G = (V, E)$, we will construct an $\mathcal{LTL}^{\circledast}_{\mathsf{CNF}}$ formula $\varphi := \Psi \wedge \circledast\, \Phi$, which has a strong KROM-backdoor B of size two, such that the graph G has a 3-coloring if and only if φ is satisfiable.
We assume for the remainder of the proof that there exists an arbitrary but fixed ordering of the vertices $V(G) = \{v_1, \ldots, v_n\}$. Further for the construction we assume w.l.o.g. that any undirected edge $e = \{v_i, v_j\} \in E$ follows this ordering, i.e., $i < j$. The formula φ contains the following variables:

(V1) The variables b_1 and b_2. These variables make up the backdoor set B, i.e., $B := \{b_1, b_2\}$.

(V2) For every i with $1 \leqslant i \leqslant n$, the variable v_i.

(V3) For every $e = \{v_i, v_j\} \in E(G)$ with $1 \leqslant i, j \leqslant n$ the variables $e^{b_1 b_2}_{v_i v_j}$, $e^{\bar{b}_1 b_2}_{v_i v_j}$, and $e^{b_1 \bar{b}_2}_{v_i v_j}$.

We set Ψ to be the empty formula and the formula Φ contains the following clauses:

(C1) For every i with $1 \leqslant i \leqslant n$, the clause $\neg \circledast v_i$. Informally, this clause ensures that v_i has to be false at least at one world, which will later be used to assign a color to the vertex v_i of G. Observe that the clause is KROM.

(C2) For every $e = \{v_i, v_j\} \in E(G)$ with $1 \leqslant i, j \leqslant n$, the clauses $v_i \vee \circledast e^{b_1 b_2}_{v_i v_j} \vee b_1 \vee b_2$, $v_i \vee \circledast e^{\bar{b}_1 b_2}_{v_i v_j} \vee \neg b_1 \vee b_2$, and $v_i \vee \circledast e^{b_1 \bar{b}_2}_{v_i v_j} \vee b_1 \vee \neg b_2$ as well as the clauses $v_j \vee \neg \circledast e^{b_1 b_2}_{v_i v_j} \vee b_1 \vee b_2$, $v_j \vee \neg \circledast e^{\bar{b}_1 b_2}_{v_i v_j} \vee \neg b_1 \vee b_2$, and $v_j \vee \neg \circledast e^{b_1 \bar{b}_2}_{v_i v_j} \vee b_1 \vee \neg b_2$. Observe that all of these clauses are KROM after deleting the variables in B.

(C3) The clause $\neg b_1 \vee \neg b_2$. Informally, this clause excludes the color represented by setting b_1 and b_2 to true. Observe that the clause is KROM.

It follows from the definition of φ that $\varphi[\vartheta] \in \mathcal{LTL}^{\circledast}_{\mathsf{KROM}}$ for every assignment ϑ of the variables in B. As a consequence, B is a strong KROM-backdoor of size two of φ as required. Moreover, since φ can be constructed in polynomial time, it only remains to show that G has a 3-coloring if and only if φ is satisfiable.
We start with the forward direction. Therefore we assume that G has a 3-coloring and let $f\colon V(G) \to \{1, 2, 3\}$ be such a 3-coloring for G. We will show that φ is satisfiable by constructing a temporal interpretation $\mathcal{M}$ such that $\mathcal{M} \models \varphi$. $\mathcal{M}$ is defined as follows:

- For every i with $1 \leqslant i \leqslant n$, we set $\mathcal{M}(v_i) = \mathbb{Z} \backslash \{f(v_i)\}$.
- We set $\mathcal{M}(b_1) = \{2\}$ and $\mathcal{M}(b_2) = \{3\}$.
- For every $e = \{v_i, v_j\} \in E(G)$:
  - if $f(v_i) = 1$ set $\mathcal{M}(e^{b_1 b_2}_{v_i v_j}) = \mathbb{Z}$, else set $\mathcal{M}(e^{b_1 b_2}_{v_i v_j}) = \varnothing$.
  - if $f(v_i) = 2$ set $\mathcal{M}(e^{\bar{b}_1 b_2}_{v_i v_j}) = \mathbb{Z}$, else set $\mathcal{M}(e^{\bar{b}_1 b_2}_{v_i v_j}) = \varnothing$.
  - if $f(v_i) = 3$ set $\mathcal{M}(e^{b_1 \bar{b}_2}_{v_i v_j}) = \mathbb{Z}$, else set $\mathcal{M}(e^{b_1 \bar{b}_2}_{v_i v_j}) = \varnothing$.

Next we prove that $\mathcal{M} \models \varphi$. Therefore we consider the different types of clauses given in (C1)–(C3).

- The clauses in (C1) hold because $\mathcal{M}, f(v_i) \not\models v_i$ for every i with $1 \leqslant i \leqslant n$.
- For every $e = \{v_i, v_j\} \in E(G)$, we have to show that the clauses given in (C2) are satisfied for every world. Because f is a 3-coloring of G, we obtain that $f(v_i) \neq f(v_j)$. W.l.o.g. $f(v_i) = 1$ and $f(v_j) = 2$. We first consider the clauses given in (C2) containing v_i. Because $\mathcal{M}(v_i) = \mathbb{Z} \backslash \{1\}$, it only remains to consider the world 1. In this world b_1 and b_2 are false. It follows that all clauses containing either $\neg b_1$ or $\neg b_2$ are satisfied in this world. As a reason for this, it only remains to consider clauses of the form $v_i \vee \circledast e^{b_1 b_2}_{v_i v_j} \vee b_1 \vee b_2$. But these are satisfied because $f(v_i) = 1$ implies that $\mathcal{M}(e^{b_1 b_2}_{v_i v_j}) = \mathbb{Z}$.

Consider now the clauses given in (C2) that contain v_j. Using the same argumentation as used above for v_i, we obtain that we only need to consider world 2 and moreover we only need to consider clauses of the form $v_j \vee \neg \circledast e^{\bar{b}_1 b_2}_{v_i v_j} \vee \neg b_1 \vee b_2$. Because $f(v_i) = 1$, we obtain that $\mathcal{M}(e^{\bar{b}_1 b_2}_{v_i v_j}) = \varnothing$, which implies that these clauses are also satisfied.

- The clause $\neg b_1 \vee \neg b_2$ is trivially satisfied, because there is no world in which b_1 and b_2 hold simultaneously.

Now we prove the reverse direction. Therefore we assume that φ is satisfiable and let $\mathcal{M}$ be a temporal interpretation witnessing this. First note that because of the clauses added by C1, it holds that $\mathcal{M}(v_i) \neq \mathbb{Z}$ for every i with $1 \leqslant i \leqslant n$. Let $w\colon V(G) \to \mathbb{Z}$ be defined such that for every i with $1 \leqslant i \leqslant n$, $w(v_i)$ is an arbitrary world in $\mathbb{Z} \backslash \mathcal{M}(v_i)$. We define $f\colon V(G) \to \{1, 2, 3\}$ by setting:

- $f(v_i) = 1$ if $\mathcal{M}, w(v_i) \not\models b_1 \vee b_2$,
- $f(v_i) = 2$ if $\mathcal{M}, w(v_i) \not\models \neg b_1 \vee b_2$, and
- $f(v_i) = 3$ if $\mathcal{M}, w(v_i) \not\models b_1 \vee \neg b_2$.

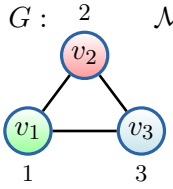
Note that because of the clause added by (C3), f assigns exactly one color to every vertex v_i of G. We claim that f is a 3-coloring of G. To show this it suffices to show that for every $e = \{v_i, v_j\} \in E(G)$, it holds that $f(v_i) \neq f(v_j)$. Assume for a contradiction that this is not the case, i.e., there is an edge $e = \{v_i, v_j\} \in E(G)$ such that $f(v_i) = f(v_j)$. W.l.o.g. $f(v_i) = f(v_j) = 1$. Consider the clause $v_i \vee \circledast e^{b_1 b_2}_{v_i v_j} \vee b_1 \vee b_2$ (which was added by C2). Then, because of the definition of w and f, we obtain that $\mathcal{M}, w(v_i) \not\models v_i \vee b_1 \vee b_2$. It follows that $\mathcal{M}, w(v_i) \models \circledast e^{b_1 b_2}_{v_i v_j}$. Consider now the clause $v_j \vee \neg \circledast e^{b_1 b_2}_{v_i v_j} \vee b_1 \vee b_2$ (which was added by C2). Then, again because of the choice of w and f, we obtain that $\mathcal{M}, w(v_j) \not\models v_j \vee b_1 \vee b_2$. As a consequence, $\mathcal{M}, w(v_j) \models \neg \circledast e^{b_1 b_2}_{v_i v_j}$ contradicting $\mathcal{M}, w(v_i) \models \circledast e^{b_1 b_2}_{v_i v_j}$.
We have shown the membership in **paraNP**, as well as the **paraNP**-hardness by giving an polytime-reduction from the **NP**-hard problem 3Col. Consequently, the proof is complete.

□

Figure 4.10 illustrates a temporal interpretation $\mathcal{M}$ for a simple graph $G = (\{v_1, v_2, v_3\}, \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\})$ with a coloring $f(v_i) = i$ for $1 \leqslant i \leqslant 3$. The temporal interpretation $\mathcal{M}$ is described in the table, with $\mathcal{M} \models \varphi$. Each row of the table corresponds to a world indicated by the first column of the table. Accordingly the first row corresponds to the first green-colored world. Additionally each column represents the assignments of a variable as indicated in the first row. The assignment "–" has a "*do not care*"- meaning, in other words the assignment does not influence whether $\mathcal{M} \models \varphi$.

For the remainder of the section we want to consider the more flexible fragment where we are able to talk about the past as well as about the future. We will show how to encode **NP**-complete problems into the HORN-fragment and obtain a **paraNP** lower bound.

G : $\mathcal{M}$:



	b_1	b_2	v_1	v_2	v_3	$e^{b_1 b_2}_{v_1 v_2}$	$e^{\bar{b}_1 b_2}_{v_1 v_2}$	$e^{b_1 \bar{b}_2}_{v_1 v_2}$	$e^{b_1 b_2}_{v_1 v_3}$	$e^{\bar{b}_1 b_2}_{v_1 v_3}$	$e^{b_1 \bar{b}_2}_{v_1 v_3}$	$e^{b_1 b_2}_{v_2 v_3}$	$e^{\bar{b}_1 b_2}_{v_2 v_3}$	$e^{b_1 \bar{b}_2}_{v_2 v_3}$
1	0	0	0	1	1	1	0	–	1	–	0	–	1	0
2	1	0	1	0	1	1	0	–	1	–	0	–	1	0
3	0	1	1	1	0	1	0	–	1	–	0	–	1	0

Figure 4.10: Temporal interpretation $\mathcal{M}$ for a given graph G, where $\mathcal{M} \vDash \varphi$

Theorem 4.33.
$\text{EVAL}^{\Box_F,\Box_P}(\text{HORN})$ is **paraNP**-complete.

Proof. As in Theorem 4.32, the membership of $\text{EVAL}^{\Box_F,\Box_P}(\text{HORN})$ in **paraNP** follows from the work of Artale et al., where the authors show the satisfiability problem of $\text{LTL}^{\Box_F,\Box_P}(\text{HORN})$ is in **NP**, [AKRZ13, Table 1].
To prove the **paraNP**-hardness we construct again, as in Theorem 4.32 a polynomial time reduction $3\text{COL} \leqslant^{fpt} \text{EVAL}^{\Box_F,\Box_P}(\text{HORN})$, but this time for a backdoor size four. And again, we construct an $\text{LTL}^{\Box_F,\Box_P}_{\text{CNF}}$-formula $\varphi := \Psi \wedge \circledast\, \Phi$, which has a strong HORN-backdoor set B of size four, such that the graph G has a 3-coloring if and only if φ is satisfiable. We assume $V(G) = \{v_1, \ldots, v_n\}$ and $E(G) = \{e_1, \ldots, e_m\}$, and the formula φ contains the following variables:

(V1) The backdoor set contains the variables $B := \{c_1, c_2, c_3, p'_n\}$

(V2) For the initial world we have a variable s

(V3) For every i with $1 \leqslant i \leqslant n$, three variables v_i^1, v_i^2, v_i^3

(V4) For every i with $1 \leqslant i \leqslant n$ the variable p_i

We set Ψ to be the formula s and the formula Φ contains the following clauses:

(C1) The clauses $c_1 \vee c_2 \vee c_3$, $\neg c_1 \vee \neg c_2 \vee \neg c_3$, $c_1 \vee \neg c_2 \vee \neg c_3$, $\neg c_1 \vee \neg c_2 \vee c_3$, and $\neg c_1 \vee c_2 \vee \neg c_3$. These clauses force that exactly one of the variables c_1, c_2, c_3 is true in every world. Observe that the clause $c_1 \vee c_2 \vee c_3$ is not HORN, note that, all of its variables are contained in the backdoor set B.

(C2) For every i and c with $1 \leqslant i \leqslant n$ and $1 \leqslant c \leqslant 3$, the clauses $v_i^c \rightarrow \Box_F v_i^c = \neg v_i^c \vee \Box_F v_i^c$ and $v_i^c \rightarrow \Box_P v_i^c = \neg v_i^c \vee \Box_P v_i^c$. These clauses force that the variable v_i^c either holds in every world or in no world for every i and c as above. Observe that both of these clauses are HORN.

(C3) Informally, the following set of clauses ensures together that for every i with $1 \leqslant i \leqslant n$, it holds that p_i is true in every world apart from the i-th world (where p_i is false). Here, the first world is assumed to be the starting world.

(C3.1) The clauses $s \to \neg p_1 = \neg s \vee \neg p_1$, $s \to \Box_F p_1 = \neg s \vee \Box_F p_1$, and $s \to \Box_P p_1 = \neg s \vee \Box_P p_1$. Informally, these ensure that p_1 is only false in the starting world (and otherwise true).

(C3.2) The clause $p_i \wedge \Box_F p_i \to \Box_F p_{i+1} = \neg p_i \vee \neg \Box_F p_i \vee \Box_F p_{i+1}$ for every i with $1 \leqslant i < n$. Informally, these clauses (together with the clauses from C3.1) ensure that for every i with $2 \leqslant i \leqslant n$, it holds that p_i is true in every world after the i-th world.

(C3.3) The clause $\neg p_i \to \neg \Box_F p_{i+1} = p_i \vee \neg \Box_F p_{i+1}$ for every i with $1 \leqslant i < n$. Roughly, these clauses (together with the clauses from (C3.1) and (C3.2) ensure that for every i with $2 \leqslant i \leqslant n$, it holds that p_i is false at the i-th world. Observe that the clauses from (C3.1) to (C3.3) already ensure that $\neg p_i \wedge \Box_F p_i$ holds if and only if we are at the i-th world of the model for every i with $1 \leqslant i \leqslant n$.

(C3.4) The clauses $\neg p_n \wedge \Box_F p_n \to p'_n = p_n \vee \neg \Box_F p_n \vee p'_n$ and $\neg p_n \wedge \Box_F p_n \leftarrow p'_n = \neg p_n \wedge \Box_F p_n \vee \neg p'_n = (\neg p_n \vee \neg p'_n) \wedge (\Box_F p_n \vee \neg p'_n)$. Informally, these clauses (together with the clauses from (C3.1) to (C3.3) ensure that p'_n only holds in the n-th world of the model. Observe that all these clauses are HORN after removing the backdoor set variable p'_n.

(C3.5) The clause $p'_n \to \Box_P p_n = \neg p'_n \vee \Box_P p_n$. Informally, this clause (together with the clauses from (C3.1) to (C3.4) ensures that p_n is only false in the n-th world of the model.

(C3.6) The clause $p_i \wedge \Box_P p_i \to \Box_P p_{i-1} = \neg p_i \vee \neg \Box_P p_i \vee \Box_P p_{i-1}$ for every i with $2 \leqslant i \leqslant n$. Informally, these clauses (together with the clauses from (C3.1) to (C3.5) ensure that p_i is true before the i-th world for every i with $2 \leqslant i < n$.

Note that all of the described clauses above are HORN or become HORN after removing all variables from backdoor set B. Note furthermore that all the above clauses ensure that $\Box_P p_i \wedge \Box_F p_i$ holds if and only if we are at the i-th world of the model for every i with $1 \leqslant i \leqslant n$.

(C4) For every i and j with $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant 3$ the clauses $\Box_F p_i \wedge \Box_P p_i \wedge v_i^j \to c_j = \neg \Box_F p_i \vee \neg \Box_P p_i \vee \neg v_i^j \vee c_j$ and $\Box_F p_i \wedge \Box_P p_i \wedge c_j \to v_i^j = \neg \Box_F p_i \vee \neg \Box_P p_i \vee \neg c_j \vee v_i^j$. Informally, these clauses ensure that in the i-th world for every $1 \leqslant i \leqslant n$, the variables c_1, c_2, c_3 are a copy of the variables v_i^1, v_i^2, v_i^3. Observe that all of these clauses are also HORN.

(C5) For every edge $e = \{v_i, v_j\} \in E(G)$ and every c with $1 \leqslant c \leqslant 3$, the clause $\neg v_i^c \vee \neg v_j^c$. Informally, these clauses ensure that the 3-partition (of the vertices of G) given by the (global) values of the variables v_1^1, v_1^2, v_1^3, $\ldots$, v_n^1, v_n^2, v_n^3 is a valid 3-coloring for G. Observe that all of these clauses are HORN.

From the definition of the formula φ follows, $\varphi[\vartheta] \in \mathrm{LTL}^{\Box_F,\Box_P}(\mathsf{HORN})$ for every assignment ϑ of variables in backdoor set B. Therefore holds that B is a strong HORN backdoor set B of the formula φ. Observe that the formula φ is constructible in polynomial time, consequently it remains to show that φ is satisfiable if and only if the graph G has a 3-coloring.
First we prove that φ is satisfiable if the graph G has a 3-coloring. We define the coloring function f as $f : V(G) \to \{1, 2, 3\}$. To show the satisfiability of φ via a construction of a temporal interpretation $\mathcal{M}$, such that $\mathcal{M} \vDash \varphi$.
The constructed temporal interpretation $\mathcal{M}$ has the following properties:

- We set $\mathcal{M}(c_j) = \{i \mid f(v_i) = j\}$, for every j with $1 \leqslant j \leqslant 3$
- We set $\mathcal{M}(p'_n) = \{n\}$
- For every i and c with $1 \leqslant i \leqslant n$ and $1 \leqslant c \leqslant 3$, we set $\mathcal{M}(v_i^c) = \mathbb{Z}$ if $c = f(v_i)$ and otherwise we set $\mathcal{M}(v_i^c) = \varnothing$.
- For every i with $1 \leqslant i \leqslant n$, we set $\mathcal{M}(p_i) = \mathbb{Z}\backslash\{i\}$.

Now we assume φ is satisfiable in a valid temporal interpretation $\mathcal{M}$. We prove the reverse direction by showing the following claims for $\mathcal{M}$:

(M1) For every $a \in \mathbb{Z}$ exactly one of $\mathcal{M}, a \vDash c_1$, $\mathcal{M}, a \vDash c_2$, and $\mathcal{M}, a \vDash c_3$ holds.

(M2) For every i, c, a, and a' with $1 \leqslant i \leqslant n$, $1 \leqslant c \leqslant 3$, and $a, a' \in \mathbb{Z}$, it holds that $\mathcal{M}, a \vDash v_i^c$ if and only if $\mathcal{M}, a' \vDash v_i^c$.

(M3) For every i with $1 \leqslant i \leqslant n$ and every $a \in \mathbb{Z}$, it holds that $\mathcal{M}, a \vDash p_i$ if and only if $a \neq i$.

(M4) For every i and j with $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant 3$, it holds that $\mathcal{M}, i \vDash c_j$ if and only if $\mathcal{M}, i \vDash v_i^j$.

Next we prove the claims (M1) to (M4) for $\mathcal{M}$.
(M1) follows since the clauses which enforces that exactly one of the variables c_1, c_2, c_3 is true in every world (C1).
(M2) follows due to (C2). Let $i, c, a,$ and a' be given as in the statement of (M2). Now assume for contradiction $\mathcal{M}, a \vDash v_i^c$ but $\mathcal{M}, a' \nvDash v_i^c$. Consequently, $a \neq a'$. If $a < a'$, then we obtain a contradiction because of the clause $v_i^c \to \Box_F v_i^c$ and if on the other hand $a' < a$, we obtain a contradiction to the clause $v_i^c \to \Box_P v_i^c$.
To prove (M3), we divide it into several auxiliary claims:

(M3.1) For every $a \in \mathbb{Z}$ it holds that $\mathcal{M}, a \vDash p_1$ if and only if $a \neq 1$ (here we assume that 1 is the initial world)

(M3.2) For every i and a with $1 \leqslant i \leqslant n$, $a \in \mathbb{Z}$, and $a > i$, it holds that $\mathcal{M}, a \vDash p_i$

(M3.3) For every i with $1 \leqslant i \leqslant n$, it holds that $\mathcal{M}, i \nvDash p_i$

(M3.4) For every $a \in \mathbb{Z}$, it holds that $\mathcal{M}, a \vDash p'_n$ if and only if $a = n$

(M3.5) For every $a \in \mathbb{Z}$, it holds that $\mathcal{M}, a \not\models p_n$ if and only if $a = n$

(M3.1) holds for the following reasons:
We know by definition that the initial world s are in Ψ and by the clause of (C3.1), namely $s \rightarrow \neg p_1$ follows $\mathcal{M}, 1 \not\models p_1$. In addition, due to the clauses $s \rightarrow \Box_F p_1$ and $s \rightarrow \Box_P p_1$, follows for every $\mathcal{M}, a \models p_1$ for every $a \neq 1$.
Next we prove (M3.2) by induction on i. The induction basis $i = 1$ holds because of (C3.1). For the inductive step we assume that the claim holds for p_{i-1}. The induction hypothesis yields $\mathcal{M}, i \models p_{i-1} \wedge \Box_F p_{i-1}$. Now due to the clause $p_{i-1} \wedge \Box_F p_{i-1} \rightarrow \Box_F p_i$, given in (C3.2), we obtain that $\mathcal{M}, i \models \Box_F p_i$.
Also (M3.3) we show via induction on i. The induction basis $i = 1$ holds because of (C3.1). For the inductive step we assume that the claim holds for p_{i-1}. Due to the induction hypothesis, follows $\mathcal{M}, (i-1) \not\models p_{i-1}$. In addition we know from (M3.2) that $\mathcal{M}, i \models \Box_F p_i$ and by (C3.3) φ contains the clause $\neg p_{i-1} \rightarrow \neg \Box_F p_i$. We obtain due to $\mathcal{M}, i \models \Box_F p_i$, that $\mathcal{M}, (i-1) \models \neg \Box_F p_i$ can only holds if $\mathcal{M}, i \not\models p_i$.
To show (M3.4) observe that due to (M3.2) and (M3.3) we have $\mathcal{M}, a \models \neg p_n \wedge \Box_F p_n$ if and only if $a = n$. Finally the (C3.4) clause $\neg p_n \wedge \Box_F p_n \leftrightarrow p'_n$ provides the desired statement of (M3.4).
To prove the last claim (M3.5) observe that (M3.5) holds for every $a \in \mathbb{Z}$ with $n \leqslant a$, since (M3.2) and (M3.3) hold. Additionally, due to (M3.4) we have $\mathcal{M}, n \models p'_i$. In combination with the clause (C3.5) $p'_n \rightarrow \Box_p p_n$, it holds $\mathcal{M}, a \models p_n$ for every $a < n$. Finally we obtain $\mathcal{M}, a \not\models p_n$ if and only if $a = n$.
Next we combine the auxiliary claims to prove the statement of (M3) as follows:
$\mathcal{M}, a \models p_i$ holds due to (M3.2) and (M3.3) for every $i \leqslant a$. From (M3.5) we obtain that $\mathcal{M}, a \models p_i$ holds for $i = n$. In the final step we prove (M3) via induction on i starting from $i = n$. From the induction hypothesis we obtain $M, i+1 \models p_{i+1} \wedge \Box_P pi + 1$. Then in combination with the clause (C3.6) $p_{i+1} \wedge \Box_P p_{i+1} \rightarrow \Box_P p_n$, we obtain $\mathcal{M}, i+1 \models \Box_P p_i$ and therefore $\mathcal{M}, a \models p_i$ if and only if $a \neq i$.
It remains to prove the last claim (M4). Observe $\mathcal{M}, i \models \Box_F p_i \wedge \Box_P p_i$ holds due to (M3). Now suppose for contradiction that there are i and j, such that either $\mathcal{M}, i \models c_j$ but $\mathcal{M}, i \not\models v_i^j$ or $\mathcal{M}, i \not\models c_j$ but $\mathcal{M}, i \models v_i^j$. In the case of $\mathcal{M}, i \models c_j$ but $\mathcal{M}, i \not\models v_i^j$ consider the clause (C4) $\Box_F p_i \wedge \Box_P p_i \wedge c_j \rightarrow v_i^j$. As $\mathcal{M}, i \models \Box_F p_i \wedge \Box_P p_i$, we obtain that $\mathcal{M}, i \models v_i^j$ which contradicts the assumption. Now for the $\mathcal{M}, i \not\models c_j$ but $\mathcal{M}, i \models v_i^j$ case, consider the clause $\Box_F p_i \wedge \Box_P p_i \wedge v_i^j \rightarrow c_j$, which was added by (C4). Since $\mathcal{M}, i \models \Box_F p_i \wedge \Box_P p_i$, we obtain that $\mathcal{M}, i \models c_j$; again a contradiction.
$\mathcal{M}, a \models v_i^c$ holds for every $1 \leqslant i \leqslant n$ and $a \in \mathbb{Z}$ with exactly one of the three color $1 \leqslant c \leqslant 3$, because of the claims (M1) and (M4). Additionally the choice of the color c is independent of a, due to (M2). Accordingly, the coloring f that assigns the unique color c to every vertex v_i such that $\mathcal{M}, a \models v_i^c$ forms a partition of the vertex set of G. We claim that f is also a valid 3-coloring of G. Assume for the sake of contradiction that, there is an edge $\{v_i, v_j\} \in E(G)$ such that $c = f(v_i) = f(v_j)$. Observe the clause (C5) $\neg v_i^c \vee \neg v_j^c$. Due to the definition of f, we obtain that $\mathcal{M}, a \not\models \neg v_i^c \vee \neg v_j^c$ for every $a \in \mathbb{Z}$, a contradiction to our assumption that $\mathcal{M} \models \varphi$. This concludes the proof.

□

In [AKRZ13, Theorem 5] the authors show the satisfiability of $\mathrm{LTL}^{O}_{\mathsf{KROM}}$ for $O \in \{\Box_P, \Box_F\}$ is **NP**-hard. From this we conclude the **NP**-completeness already holds for backdoor sets of size zero.

Corollary 4.34.
Let $O \in \{\Box_P, \Box_F\}$. Then $\textsc{Eval}^{O}(\mathsf{KROM})$ is **paraNP**-complete.

CHAPTER 5

CONCLUSION

In this thesis we examine the parameterized complexity of default logic as well as temporal logics. First we introduced a dynamic programming algorithm, that decides whether a given default theory has a stable extension, and runs in linear time. It can be also used to enumerate all stable default sets with a pre-computation that is linear and followed by linear delay to output the solutions. The algorithm operates on tree-decompositions with bounded width of the semi-primal graph of a given default theory. Despite already known linear time results for the stable extension existence problem, we are able to improve the runtime that is triple exponential instead of at least quintuply exponential in the treewidth. We think that our algorithm can be modified to work on the tree-decomposition of the incidence graph. Therefore we require to handle the cases where the prerequisite, justification and conclusion do not appear in one bag. Consequently, we need some auxiliary states, as in concept of answer set programming [FHMW17]. An interesting task for further research is whether and how we can improve the runtime of the algorithm.

We also introduced the concept of strong backdoors for default logic and examine the problems of backdoor detection and backdoor evaluation, both parametrized by the size of the backdoor set. We have shown, starting from the initial CNF-class, the backdoor evaluation problem is fixed-parameter tractable for all considering target classes HORN, KROM, MONOTONE and POSITIVE-UNIT. Contrary, the evaluation problem is more challenging. It turned out that it is in **para-NP** for the target class KROM and **para-NP**-complete for the target class HORN. Further we have a upper bound **para-**Δ_2^{p} for the target class MONOTONE while it is fixed-parameter tractable for POSITIVE-UNIT. A consequent step will be the investigation of the remaining lower bounds. It would be also interesting to examine the remaining Schaefer classes, like dual-HORN, 1-and 0-valid [Sch78] as well as renameable-HORN and QHORN [BCH90, BHS94].

In the second part of the thesis we established an almost complete classification of all possible operator fragments of the temporal satisfiability problem parameterized by temporal depth and different notions of treewidth or path-width respectively. Additionally, we give an almost complete classification with respect to the Boolean fragments in Post's lattice. We adapted the concept of [Pra13] and show that the **FPT**-result holds for all three considered temporal logics ($\mathcal{CTL}^\star, \mathcal{CTL}^\star$ and $\mathcal{LTL}$), when we restrict it to the temporal operator *next*. By adding only one additional operator the parameterized satisfiability problem become **W**[1]-hard. We present two new formula representations as relational structures, the syntax tree and syntax circuit. It turns out that the most parts of the results do not distinguish between the representation forms. Only the {AF} case remains open for the syntax trees. If we forgo the parameter temporal depth, the cases {AX} or {A, X} remains open. It is not clear whether these fragments are fixed-parameter tractable with the support of syntax tree parameterized by path-width. We have prove the **W**[1]-hardness for those fragments for both, syntax tree and syntax circuit, parameterized by treewidth. Surprisingly, the satisfiability problem in $\mathcal{LTL}$ restricted to only X operator on syntax trees parameterized by treewidth is fixed-parameter tractable, but on syntax circuits parameterized by path-width it is **W**[1]-hard.
Further we present a generalisation of Courcelle's Theorem to work on infinite signatures for parameterized problems expressed by a uniform family of MSO-formulae. Further interesting research is to investigate other parameters beyond the common measures of path-width or treewidth and temporal depth.
In the last part of this thesis we bring together the concepts of strong backdoors and the causal fragment of linear temporal logic. We have proved the fixed parameter tractability for the backdoor detection problem for any considered operators $\circledast, \Box_P, \Box_F$ for both fragments HORN and KROM. In contrast to the evaluation of backdoors is only for the target class HORN restricted to the $\circledast$-operator fixed-parameter tractable. The backdoor evaluation problem becomes **para-NP**-complete if we allow both operators $\Box_P$ and $\Box_F$. For the target class KROM the evaluation problem is **para-NP**-complete independently of the chosen operators. Merely the case for the target class HORN of the backdoor evaluation problem restricted to either $\Box_P$ or $\Box_F$ remains open and poses a possible research task.

BIBLIOGRAPHY

[Abu10] F. N. Abu-Khzam, *A kernelization algorithm for d-hitting set*, Journal of Computer and System Sciences **76** (2010), no. 7, 524–531.

[AC81] E. A. Emerson and E. M. Clarke, *Design and synthesis of synchronisation skeletons using branching time temporal logic*, Logic of Programs, Lecture Notes in Computer Science, vol. 131, Springer Verlag, 1981, pp. 52–71.

[AH85] E. A. Emerson and J. Y. Halpern, *Decision procedures and expressiveness in the temporal logic of branching time*, Journal of Computer and System Sciences **30** (1985), no. 1, 1–24.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases: The logical level*, 1st ed., Boston, MA, USA, 1995.

[AKRZ13] A. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev, *The complexity of clausal fragments of LTL*, arXiv:1306.5088 (2013).

[AN08] Hajirasouliha I. Hormozdiari F. Sahinalp SC. Alon N., Dao P, *Biomolecular network motif counting and discovery by color coding*, Bioinformatics, 2008.

[BBC+10] M. Bauland, E. Böhler, N. Creignou, S. Reith, H. Schnoor, and H. Vollmer, *The complexity of problems for quantified constraints*, Theory Computing Systems **47** (2010), 454–490.

[BCG+12] E. Böhler, N. Creignou, M. Galota, S. Reith, H. Schnoor, and H. Vollmer, *Complexity classifications for different equivalence and audit problems for Boolean circuits*, Logical Methods in Computer Science **8** (2012), no. 3:27, 1–25.

[BCH90] E. Boros, Y. Crama, and P.L. Hammer, *Polynomial-time inference of all valid implications for horn and related formulae*, no. 1-4, 21–32.

[BCHW16] Bernhard Bliem, Günther Charwat, Markus Hecher, and Stefan Woltran, *D-FLAT^2: Subset minimization in dynamic programming on tree decompositions made easy*, 27–34.

[BdV01] P. Blackburn, M. de Rijke, and Y. Venema, *Modal logic*, Cambridge University Press, New York, NY, USA, 2001.

[BHS94] E. Boros, P. L. Hammer, and X. Sun, *Recognition of q-Horn formulae in linear time*, no. 1, 1–13.

[BK08] H. Bodlaender and A. M. C. A. Koster, *Combinatorial optimization on graphs of bounded treewidth*, no. 3, 255–269.

[BMM+11] O. Beyersdorff, A. Meier, M. Mundhenk, T. Schneider, M. Thomas, and H. Vollmer, *Model Checking CTL is almost always inherently sequential*, Logical Methods in Computer Science **7** (2011), no. 2.

[BMTV09] O. Beyersdorff, A. Meier, M. Thomas, and H. Vollmer, *The Complexity of Propositional Implication*, no. 18, 1071–1077.

[BMTV10] O. Beyersdorff, A. Meier, M. Thomas, and H. Vollmer, *The Complexity of Reasoning for Fragments of Default Logic*, Journal of Logic and Computation (2010).

[BTV12] A. Beyersdorff, O.and Meier, M. Thomas, and H. Vollmer, *The complexity of reasoning for fragments of default logic*, Journal of Logic and Computation **22** (2012), no. 3, 587–604.

[CE12] B. Courcelle and J. Engelfriet, *Graph structure and monadic second-order logic, a language theoretic approach*, Cambridge University Press, 2012.

[CKX10a] J. Chen, I. A. Kanj, and G. Xia, *Improved upper bounds for vertex cover*, no. 40–42, 3736–3756.

[CKX10b] J. Chen, I. A. Kanj, and G. Xia, *Improved upper bounds for vertex cover*, Theoretical Computer Science **411** (2010), no. 40–42, 3736–3756.

[CMVT12] N. Creignou, A. Meier, H. Vollmer, and M. Thomas, *The Complexity of Reasoning for Fragments of Autoepistemic Logic*, ACM Transactions on Computational Logic **13** (2012), no. 2, 1–22.

[Cou90] Bruno Courcelle, *The monadic second-order logic of graphs. i. recognizable sets of finite graphs*, Inf. Comput. **85** (1990), no. 1, 12–75.

[CR14] M. Carrillo and D. A. Rosenblueth, {*CTL*} *update of kripke models through protections*, Artificial Intelligence **211** (2014), 51 – 74.

[DF99] R. G. Downey and M. R. Fellows, *Parameterized complexity*, New York, NY, USA, 1999.

[DF13] R. Downey and M. Fellows, *Fundamentals of parameterized complexity*, 2013.

[DS02] S. Demri and P. Schnoebelen, *The Complexity of Propositional Linear Temporal Logics in Simple Cases*, Information and Computation **174** (2002), no. 1, 84–103 (en).

[EG97] T. Eiter and G. Gottlob, *The complexity class Θ_2^{P}: Recent results and applications in AI and modal logic*, Proceedings of the 11th International Symposium on Fundamentals of Computation Theory (FCT'97) (Kraków, Poland) (Bogdan S. Chlebus and Ludwik Czaja, eds.), vol. 1279, September 1997, pp. 1–18.

[EJT10] M. Elberfeld, A. Jakoby, and T. Tantau, *Logspace versions of the theorems of bodlaender and courcelle*, Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 2010.

[Eme90] E. Allen Emerson, *Temporal and modal logic*, Handbook of theoretical computer science (vol. B): formal models and semantics (Jan van Leeuwen, ed.), MIT Press, Cambridge, MA, USA, 1990, pp. 995–1072.

[FDP01] M. Fisher, C. Dixon, and M. Peim, *Clausal temporal resolution*, ACM Transactions on Computational Logic **2** (2001), no. 1, 12–56.

[Fer10] Henning Fernau, *A top-down approach to search-trees: Improved algorithmics for 3-hitting set*, no. 1, 97–118.

[FG06] J. Flum and M. Grohe, *Parameterized complexity theory*, vol. XIV, Berlin, 2006.

[FHMW17] J. K. Fichte, M. Hecher, M. Morak, and S. Woltran, *Answer set solving with bounded treewidth revisited*, Proceedings of the 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'17) (Espoo, Finland) (Marcello Balduccini and Tomi Janhunen, eds.), vol. 10377, July 2017.

[FHS17] J. K. Fichte, M. Hecher, and I. Schindler, *Default Logic and Bounded Treewidth*, CoRR **arXiv:1706.09393** (2017).

[FL79a] M. J. Fischer and R. E. Ladner, *Propositional dynamic logic of regular programs*, Journal of Computer and System Sciences **18** (1979), no. 2, 194 – 211.

[FL79b] M. J. Fischer and R. E. Ladner, *Propositional modal logic of programs*, Journal of Computer and System Sciences **18** (1979), 194–211.

[FMR08] E. Fischer, J. A. Makowsky, and E. V. Ravve, *Counting truth assignments of formulas of bounded tree-width or clique-width*, no. 4.

[FMS16] J. K. Fichte, A. Meier, and I. Schindler, *Strong backdoors for default logic*, Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT'16), 2016.

[GHR94] D. M. Gabbay, I. Hodkinsion, and M. Reynolds, *Temporal logic: mathematical foundations and computational aspects*, vol. 1, Oxford University Press, Inc. New York, USA, 1994.

[Got92] Georg Gottlob, *Complexity results for nonmonotonic logics*, no. 3, 397–425.

[GS12] S. Gaspers and S. Szeider, *Backdoors to satisfaction*, The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday, LNCS, vol. 7370, Springer, 2012, pp. 287–317.

[Hal76] Rudolf Halin, *S-functions for graphs*, Journal of Geometry **8** (1976), no. 1, 171–186.

[Hal95] Joseph Y. Halpern, *The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logics*, Artif. Intell. **75** (1995), no. 2, 361–372.

[HDE95] J. Flum H.-D. Ebbinghaus, *Finite model theory*, Springer Monographs in Mathematics, vol. 2, Springer Berlin Heidelberg, 1995.

[HSS10] E. Hemaspaandra, H. Schnorr, and I. Schnoor, *Generalized modal satisfiability*, Journal of Computer and System Sciences **76** (2010), 561–578.

[Imm99] Neil Immerman, *Descriptive complexity*, Springer, 1999.

[Kon01] Igor Kononenko, *Machine learning for medical diagnosis: history, state of the art and perspective*, Artificial Intelligence in Medicine **23** (2001), no. 1, 89 – 109.

[KP05] Y. Kesten and A. Pnueli, *A compositional approach to ctl* verification*, Theoretical Computer Science **331** (2005), no. 2, 397 – 428, Formal Methods for Components and Objects.

[Kri63] S. Kripke, *Semantical considerations on modal logic*, Acta Philosophica Fennica, vol. 16, 1963, pp. 84–94.

[KS93] E. Korach and N. Solel, *Tree-width, path-widt, and cutwidth*, Discrete Applied Mathematics **43** (1993), no. 1, 97–101.

[Lew79] H. Lewis, *Satisfiability problems for propositional calculi*, Mathematical Systems Theory **13** (1979), 45–53.

[LM15] M. Lück and A. Meier, *LTL fragments are hard for standard parameterisations*, CoRR **abs/1504.06187** (2015).

[LMS17] M. Lück, A. Meier, and I. Schindler, *Parametrised complexity of satisfiability in temporal logic*, ACM Trans. Comput. Log. **18** (2017), no. 1, 1:1–1:32.

[MMS+10] A. Meier, M. Mundhenk, T. Schneider, M. Thomas, V. Weber, and F. Weiss, *The Complexity of Satisfiability for Fragments of Hybrid Logic – Part I*, Journal of Applied Logic **8** (2010), no. 4, 409–421.

[MMTV09] A. Meier, M. Mundhenk, M. Thomas, and H. Vollmer, *The complexity of satisfiability for fragments of CTL and CTL**, International Journal of Foundations of Computer Science **20** (2009), no. 5, 901–918.

[MOSS16] A. Meier, S. Ordyniak, R. Sridharan, and I. Schindler, *Backdoors for linear temporal logic*, 11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark, LIPIcs, vol. 63, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 23:1–23:17.

[MS13] A. Meier and T. Schneider, *Generalized satisfiability for the description logic ALC*, Theoretical Computer Science **505** (2013), no. 0, 55 – 73, Theory and Applications of Models of Computation 2011.

[Pnu77] A. Pnueli, *The temporal logic of programs*, Proc. 18th Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1977, pp. 46–57.

[Pos41] E. Post, *The two-valued iterative systems of mathematical logic*, Annals of Mathematical Studies **5** (1941), 1–122.

[Pra93] Henry Prakken, *A logical framework for modelling legal argument*, Proceedings of the 4th International Conference on Artificial Intelligence and Law (New York, NY, USA), ICAIL '93, ACM, 1993, pp. 1–9.

[Pra13] M. Praveen, *Does treewidth help in modal satisfiability?*, ACM Transactions on Computational Logic **14** (2013), no. 3, 18:1–18:32.

[Pri57] A. N. Prior, *Time and modality*, Clarendon Press, Oxford, 1957.

[Rei80] R. Reiter, *A logic for default reasoning*, Artificial Intelligence **13** (1980), 81–132.

[Rei87] Raymond Reiter, *A theory of diagnosis from first principles*, Artificial Intelligence **32** (1987), no. 1, 57 – 95.

[RN84] Seymour P. D. Robertson N., *Graph minors. iii. planar tree-width*, Journal of Combinatorial Theory, Series B **36** (1984), no. 1, 49–64.

[Ros99] Riccardo Rosati, *Model checking for nonmonotonic logics: Algorithms and complexity*, Proceedings of the 16th International Joint Conference on Artificial Intelligence (ICJAI'99) (Stockholm, Sweden) (Thomas Dean, ed.), July 1999.

[Sch78] Thomas J. Schaefer, *The complexity of satisfiability problems*, Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78) (San Diego, CA, USA) (Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, eds.), 1978, pp. 216–226.

[Sch81] C. P. Schnorr, *On self-transformable combinatorial problems*, Mathematical Programming at Oberwolfach (H. König, B. Korte, and K. Ritter, eds.), vol. 14, 1981, pp. 225–243.

[Sch94] Petra Scheffler, *A practical linear time algorithm for disjoint paths in graphs with bounded tree width*, Fachbereich Mathematik - Report **396** (1994).

[SS09] M. Samer and S. Szeider, *Fixed-parameter tractability*, Handbook of Satisfiability (Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, eds.), 2009, pp. 425–454.

[Sti90a] Jonathan P. Stillman, *It's not my default: The complexity of membership problems in restricted propositional default logics*, Proceedings of the 8th National conference on Artificial Intelligence (AAAI'90) (Boston, MA, USA) (Thomas Dietterich and William Swartout, eds.), vol. 1, July 1990, pp. 571–578.

[Sti90b] ———, *The Complexity of Horn Theories with Normal Unary Defaults*, Proceedings of the 8th Canadian Artificial Intelligence Conference (AI'90), 1990.

[Vol99] Heribert Vollmer, *Introduction to circuit complexity*, Springer Verlag, 1999.

[VS85] M. Y. Vardi and L. Stockmeyer, *Improved upper and lower bounds for modal logics of programs*, STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing (New York, NY, USA), ACM, 1985, pp. 240–251.

[WGS03] R. Williams, C. Gomes, and B. Selman, *Backdoors to typical case complexity*, Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03) (Acapulco, Mexico) (Georg Gottlob and Toby Walsh, eds.), August 2003, pp. 1173–1178.

LIST OF FIGURES

INDEX

LEBENSLAUF

Persönliche Daten

Name	Irena Schindler
Geburtsdaten	05.07.1984 in Alma-Ata
Familienstand	Verheiratet, zwei Kinder

Schulbildung

1995 – 1996	Grundschule Vinhorst
1996 – 1998	Orientierungsstufe Ahlem
1998 – 2005	Bismarckschule Hannover

Hochschulbildung

2005 – 2006	Studium der Wirtschaftswissenschaften, Leibniz Universität Hannover
2006 – 2013	Studium der Mathematik mit Studienrichtung Informatik, Leibniz Universität Hannover
10.2013 – 12.2017	Wissenschaftliche Mitarbeiterin am Institut für Theoretische Informatik an der Leibniz Universität Hannover