Parameterized Aspects of Team-Based Formalisms and Logical Inference

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation

von Herrn

M.Sc. Yasir Mahmood

Geboren am 11.05.1992 in Khushab, Pakistan

2022

Referent:	PD. Dr. Arne Meier, Leibniz Universität Hannover
Korreferent:	Dr. Jonni Virtema, University of Sheffield
Vorsitzender:	Prof. Dr. Ziawasch Abedjan, Leibniz Universität Hannover
Tag der Promotion:	29.09.2022

To my lovely wife,
and my parents,
. . . and myself!

ACKNOWLEDGEMENT

It is advised to read this section in a random but fixed order.

I want to thank Juha Kontinen for introducing me to the beautiful realm of team-based logics and the lovely logic group of Hannover. Thank you, Arne! You have been an amazing colleague, a friend, a translator, a Steuerberater, a chess partner, a co-author, a Latex guide, a PC wizard, an awesome supervisor, and now a reviewer. I appreciate that I rarely had to book an appointment to discuss anything with you. All I needed was to show up at your door, and you managed to help me on the spot. A special thanks to Arne and Heribert for their trust and support and for allowing me more time to complete my thesis. I also want to thank my old and current colleagues at THI: Martin Lück, Anselm Haak, Fabian Müller, Timon Barlag, Sabrina Gaube, and Rahel Becker.

The brilliant people with whom I have co-authored also deserve a separate mention. I admire Johannes Schmidt for his knowledge regarding co-clones and his help in enabling me to get some of it. Johannes also helped me a lot in understanding the co-clones during the writing phase of my thesis. I also want to thank Markus Hecher and Johannes Fichte for including me in their list of co-authors. Thanks to Jonni, for the helpful discussions regarding the dependence and inclusion logic. Working with each of you was always fun, and I learned a lot from all of you.

I am grateful to my parents and siblings for their love and support. The fact that I am the first Ph.D. in my whole family would not have been possible without their encouragement. My wife, Tahira, also deserves special acknowledgment for her unbreakable support and patience. She never complained, even if I was working in the evening or at weekends. Thanks for always standing by my side and not getting angry when I lied to you for a whole month that *I will submit my thesis today*. I also want to thank my sister Afia and her husband (my dearest cousin) Mudassar who live in Frankfurt. Their love and support was all I had in the difficult time when my wife was not here.

Yasir Mahmood
Hanover, 2022

Abstract

Parameterized complexity is an interesting subfield of complexity theory that has received a lot of attention in recent years. Such an analysis characterizes the complexity of (classically) intractable problems by pinpointing the computational hardness to some structural aspects of the input. In this thesis, we study the parameterized complexity of various problems from the area of team-based formalisms as well as logical inference.

In the context of team-based formalism, we consider propositional dependence logic ($\mathcal{PDL}$) [114]. The problems of interest are model checking (MC) and satisfiability (SAT). Peter Lohmann studied the classical complexity of these problems as a part of his Ph.D. thesis [75] proving that both MC and SAT are **NP**-complete for $\mathcal{PDL}$. This thesis addresses the parameterized complexity of these problems with respect to a wealth of different parameterizations. Interestingly, SAT for $\mathcal{PDL}$ boils down to the satisfiability of propositional logic as implied by the downwards closure of $\mathcal{PDL}$-formulas. We propose an interesting satisfiability variant (mSAT) asking for a satisfiable team of size m. The problem mSAT restores the 'team semantic' nature of satisfiability for $\mathcal{PDL}$-formulas. We propose another problem (MaxSubTeam) asking for a maximal satisfiable team if a given team does not satisfy the input formula.

From the area of logical inference, we consider (logic-based) abduction and argumentation. The problem of interest in abduction (ABD) is to determine whether there is an explanation for a manifestation in a knowledge base (KB). Following Pfandler et al. [43], we also consider two of its variants by imposing additional restrictions over the size of an explanation ($\text{ABD}_{\leq}$ and $\text{ABD}_{=}$). In argumentation, our focus is on the argument existence (ARG), relevance (ARG-Rel) and verification (ARG-Check) problems. The complexity of these problems have been explored already in the classical setting, and each of them is known to be $\mathbf{\Sigma}_2^P$-complete (except for ARG-Check which is **DP**-complete) for propositional logic. Moreover, the work by Nord and Zanuttini [89] (resp., Creignou et al. [22]) explores the complexity of these problems with respect to various restrictions over allowed KBs for ABD (ARG). In this thesis, we explore a two-dimensional complexity analysis for these problems. The first dimension is the restrictions over KB in Schaefer's framework (the same direction as Nord and Zanuttini [89] and Creignou et al. [22]). What differentiates the work in this thesis from an existing research on these problems is that we add another dimension, the parameterization.

The results obtained in this thesis are interesting for two reasons. First (from a theoretical point of view), ideas used in our reductions can help in developing further reductions and prove (in)tractability results for related problems. Second (from a practical point of view), the obtained tractability results might help an agent designing an instance of a problem come up with the one for which the problem is tractable.

Keywords: Parameterized complexity, propositional dependence logic, Schaefer's framework, non-classical logic, abductive reasoning, argumentation.

Zusammenfassung

Parametrisierte Komplexität ist ein interessantes Teilgebiet der Komplexitätstheorie, das in den letzten Jahren viel Aufmerksamkeit erhalten hat. Eine solche Analyse charakterisiert die Komplexität von praktisch unlösbaren Problemen, indem sie die rechnerische Schwierigkeit auf einige strukturelle Aspekte der Eingabe zurückführt. In dieser Arbeit untersuchen wir die parametrisierte Komplexität verschiedener Probleme aus dem Bereich der teambasierten Formalismen sowie der logischen Inferenz.

Im Kontext des teambasierten Formalismus betrachten wir die Propositional Dependence Logic ($\mathcal{PDL}$) [114]. Die Probleme, die uns interessieren, sind Model Checking (MC) und Erfüllbarkeit (SAT). Peter Lohmann untersuchte die Komplexität dieser Probleme im Rahmen seiner Doktorarbeit [75] und bewies, dass sowohl MC und SAT **NP**-vollständig für $\mathcal{PDL}$ sind. Diese Arbeit befasst sich mit der parametrisierten Komplexität dieser Probleme in Bezug auf verschiedener Parametrisierungen. Interessanterweise läuft SAT für $\mathcal{PDL}$ auf das Erfüllbarkeitsproblem der Aussagenlogik hinaus, wie sie durch die Abwärtsschliessung von $\mathcal{PDL}$-Formeln impliziert wird. Wir schlagen eine interessante Erfüllbarkeitsvariante (mSAT) vor, die nach einem erfüllbaren Team der Grösse m fragt. Das Problem mSAT stellt die "team-semantische" Natur der Erfüllbarkeit von $\mathcal{PDL}$-Formeln wieder her. Wir schlagen ein weiteres Problem (MaxSubTeam) vor, das nach einem maximal erfüllbaren Team fragt, wenn ein gegebenes Team die Eingabeformel nicht erfüllt.

Aus dem Bereich der logischen Inferenz betrachten wir (logikbasierte) Abduktion und Argumentation. Das Problem, das bei der Abduktion (ABD) von Interesse ist, besteht darin festzustellen, ob es eine Erklärung für eine Manifestation in einer Wissensdatenbank (*KB*) gibt. In Anlehnung an Pfandler et al. [43] betrachten wir auch zwei seiner Varianten, indem wir zusätzliche Einschränkungen an die Grösse einer Erklärung auferlegen ($\text{ABD}_{\leq}$ und $\text{ABD}_{=}$). Bei der Argumentation konzentrieren wir uns auf die Probleme der Existenz (ARG), Relevanz (ARG-Rel) und Verifikation (ARG-Check) von Argumenten. Die Komplexität dieser Probleme wurde bereits im klassischen Rahmen erforscht und jedes von ihnen ist $\mathbf{\Sigma}_2^P$-vollständig (außer ARG-Check, das **DP**-vollständig ist) für Aussagenlogik. Darüber hinaus untersucht die Arbeit von Nord und Zanuttini [89] (bzw. Creignou et al. [22]) die Komplexität dieser Probleme in Bezug auf verschiedene Einschränkungen an erlaubte *KB*s für ABD (ARG). In dieser Arbeit untersuchen wir eine zweidimensionale Komplexitätsanalyse für diese Probleme. Die erste Dimension sind die Einschränkungen an *KB* im Rahmen von Schaefer ([89],[22]). Was die Arbeit in dieser Dissertation von der bestehenden Forschung zu diesen Problemen unterscheidet, ist, dass wir eine weitere Dimension, die Parametrisierung, hinzufügen.

Die in dieser Arbeit erzielten Ergebnisse sind aus zwei Gründen interessant. Erstens, können die Ideen, die in unseren Reduktionen verwendet werden, dabei helfen, weitere Reduktionen zu entwickeln und praktische (Un-)lösbarkeit für verwandte Probleme zu beweisen. Zweitens, können die Lösbarkeitsergebnisse einem Agenten, der eine Instanz eines Problems entwirft, dabei helfen, diejenige zu finden, für die das Problem praktisch berechenbar ist.

PREFACE

The title of my thesis is *Parameterized Aspects* of Team-based Formalisms and Logical Inference with an intentional emphasis on "parameterized aspects". In a Dagstuhl seminar on Logics for Dependence and Independence in January 2019, Arne Meier mentioned in his introduction that "I work on parameterized complexity and want to give all of you some taste of this amazing subfield." The person next to Arne in the queue was me, and I started my introduction as: "My name is Yasir Mahmood, and I am one of those who are already affected by Arne's idea." I found parametrized analysis a fascinating subfield of complexity theory. Here, one closely looks at an intractable problem and assigns a degree of intractability to various components of an instance. It answers the question "what role does this component play in the overall intractability" for considered problems. This thesis presents parts of my research since September 2018. In the following, I give some insights into each chapter with an overview of my contributions.

Chapter 3 I initiated my research by analyzing the parameterized complexity of propositional dependence logic ($\mathcal{PDL}$), jointly with Arne Meier. Initially, we listed five parameters, including the treewidth, arity (dep-arity) and size of the formula ($|\phi|$), size of the team ($|T|$), and the number of variables ($|\mathsf{var}(\phi)|$) in the formula. Then we started to explore the complexity of satisfiability (SAT) and model checking (MC) with respect to these parameters. Soon, we realized that the disjunction operator in dependence logic is quite strong in the sense that the model-checking problem for a fixed first-order dependence logic ($\mathcal{D}$) formula with two split-junctions is **NP**-complete [68]. However, the fragment of propositional dependence logic ($\mathcal{PDL}$) without any split-junction renders the problem tractable [39]. This resulted in considering the number of split-junctions (#splits) as an interesting parameter for $\mathcal{PDL}$. In the same vein, the failure of the distributivity of conjunction over disjunction also impacts the normal forms of $\mathcal{PDL}$-formulas. Namely, a $\mathcal{PDL}$-formula can be seen as a tree where leaves are atomic subformulas. This yielded yet another parameter, namely the depth of this syntax tree of the input formula (formula-depth). However, the question of defining the treewidth for a formula was still unresolved so far. Arne suggested an approach proposed by Lück et al. [78] to consider the syntax circuit of an input

formula. Later, we realized that the treewidth for $\mathcal{PDL}$-formulas, which are already *tree-like*, may not be interesting. We concluded that the notion of syntax structures is better suited for $\mathcal{PDL}$-formulas than that of syntax circuits. This way one can also express the team (for MC) in a structure and consider the treewidth for the Gaifman graph of this structure. Moreover, this yielded two different parameters regarding the treewidth, one with the team included in the representation (formula-team-tw) and one without it (formula-tw). The complexity results for MC with respect to these two parameters suggest that this distinction is indeed interesting. I proved that formula-team-tw also bounds the teamsize, which is not the case for formula-tw. My other contributions include a recursive bottom-up algorithm for solving MC when parameterized by the teamsize. Interestingly, I later proved that this algorithm applies to any team-based logic $\mathcal{L}$ such that $\mathcal{L}$-atoms can be evaluated in polynomial time [84]. Additionally, I presented an **FPT**-algorithm for SAT when parameterized by #splits. We also proposed a satisfiability variant (mSAT) for $\mathcal{PDL}$ which asks for a satisfying team of size $m \in \mathbb{N}$. I proved that mSAT is **NP**-complete if m is given in unary and **NEXP**-complete otherwise. After an exchange of emails with Phokion Kolatis (UC Santa Cruz), it turned out that mSAT bears a close resemblance with the notion of Armstrong databases. Given a set of database dependencies, the question is whether or not there is a database that satisfies precisely these dependencies. However, the Armstrong database is stricter in that the resulting database must satisfy the given dependencies precisely (nothing more and nothing less). Whereas it does not impose a size restriction as mSAT does. Nevertheless, such a problem has not been considered in the context of dependence logic, as pointed out by Juha Kontinen (Helsinki). This work was first published in the proceedings of the eleventh International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2020) [79] and later with new results on the complexity of mSAT in the Annals of Mathematics and Artificial Intelligence [80]. Results regarding the two parameters (#conjunctions and #atoms) are new to this thesis and not published before. Moreover, I introduce another problem MaxSubTeam and prove that this is an **NP**-hard problem.

Chapter 4 We started exploring the parameterized complexity of abductive reasoning in Schaefer's framework when Johannes Schmidt (Sweden) first visited Hannover. Our analysis included three flavors of the problem (ABD, $\text{ABD}_{=}$ and $\text{ABD}_{\leq}$), and after some discussion, we agreed on four parameterizations. These include the size of the set of hypotheses (H), variables (V), manifestations (M) and the solution set (E). It took me some time to digest discussions on the Schaefer's framework. In the beginning, my contributions were mainly on the *meta* level. Such as, one can reduce IndependentSet to an $\text{ABD}(\Gamma)$-instance if the constraint language (CL) Γ allows clauses of the form $(\neg x \vee \neg y)$ in the knowledge base (KB). This resulted in proving $\mathbf{W}[1]$-hardness of $\text{ABD}_{=}$ for certain CLs when parameterized by $|E|$. Later, Johannes extended these ($\text{ABD}_{=}$) cases to $\mathbf{W}[1]$-completeness as well as presented $\mathbf{W}[2]$-completeness of some further CLs. I argued that these cases are also $\mathbf{W}[2]$-complete for $\text{ABD}_{\leq}$ when parameterized by $|E|$ but could not find the exact reduction. I also presented a so-called 'monotone argument' for dualHorn languages, which implied that $\text{ABD}_{\leq}$ and $\text{ABD}_{=}$ are equivalent when KB is a

dualHorn formula. Furthermore, I proved that $\text{ABD}_{\leq}$ and $\text{ABD}_{=}$ for dualHorn languages are **FPT** when parameterized by $|M|$. This was proven by finding a reduction to the problem MaxSATs. This reduction, together with other results for the case of $|E|$, resulted in several **FPT** cases for the case of $|M|$. The first version of this work was published in the proceedings of the International Symposium on Logical Foundations of Computer Science (LFCS 2020) [81]. After some time, I finally succeeded in achieving the $\mathbf{W}[2]$-membership of $\text{ABD}_{\leq}$ for aforementioned CLs when parameterized by $|E|$. I achieved this through a reduction from our problem to the halting problem for multi-tape NTMs. Later, I also reduced cases of ABD and $\text{ABD}_{\leq}$ when parameterized by $|M|$ to the halting problem for single-tape NTMs, yielding $\mathbf{W}[1]$-membership. This (together with other additions to the LFCS-version) has been published in the Journal of Logic and Computation [82]. We also proved several implementation results for constraint languages which are independent of the problems we considered here. Finally, the results regarding the constant-depth reduction between $\text{ABD}_{\leq}$ and $\text{ABD}_{=}$, as well as the relationship between the parameters $|V|$ and $|KB|$ are new to this thesis.

Chapter 5 The fifth chapter addresses the parameterized complexity of problems in logic-based argumentation. This was also a joint work together with Arne Meier and Johannes Schmidt. We considered three problems from argumentation, namely the existence problem (ARG), the verification of an argument (ARG-Check), and the relevance problem (ARG-Rel). The meaningful parameters include the size of the knowledge base (Δ), the claim (α), and the support or the solution size (Φ). We argued that each of these parameters has three values associated with them. In other words, one can consider either the number of formulas, the number of variables, or the *size* (the encoding length) for a set χ of formulas. In the conference version of our work, we proved that all three versions of each parameterization are equivalent. However, with some careful analysis, I recently observed that this equivalence is not true for the number of formulas. Indeed, the number of formulas in a set Φ is bounded by the total number of variables, whereas one can not prove the bound in the other direction. For the parameter claim-size ($|\alpha|$), I established the $\mathbf{W}[1]$-membership for essentially negative (EN) and essentially positive (EP) constraint languages by reducing ARG to Clique. Johannes later extended this reduction to languages that are either EN or EP. Additionally, he switched the reduction and reduced it to *his favorite* $\mathbf{W}[1]$-complete problem, the weighted satisfiability for $\Gamma_{1,d}$-formulas. In this thesis, I prove that the same reduction works for the relevance problem with slight modifications, but only if the constraint language is both (EN and EP). If a CL is either EN or EP, but not both, I could only establish the membership in $\mathbf{W}[2]$, since the reduction yields a $\Gamma_{2,d}$ formula. A preliminary version of this work has been published in the proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021) [83]. During our parameterized complexity analysis, we encountered instances of implication (IMP) and tautology problems (TAUT) in Schaefer's framework. It turned out that these problems had not been studied for parameterized complexity in Schaefer's framework before. After an exchange of emails with Nadia Creignou (Aix Marseille), we found that the complexity of TAUT in Schaefer's framework had not been considered before,

even in the classical setting. We proved that TAUT is in fact in **P**. My contributions included formalizing the proof of **P**-membership for TAUT and classifying the parameterized complexity of IMP in Schaefer's framework. This also helped in achieving **FPT**-results for each problem when parameterized by the number of variables in the knowledge base ($|\mathsf{var}(KB)|$) and the support ($|\mathsf{var}(\Phi)|$). The extended version with these additions has been accepted to be published in ACM's journal of Transactions on Computational Logic. Finally, in this thesis I include tractability versus intractability frontier for each problem in argumentation whe parameterized by $|KB|$ and $|\Phi|$. Results regarding these two parameters in argumentation (as well as several corrections to the conference version) are new to this thesis as they did not appear in the journal version.

Other Work Apart from what is included in this thesis, I worked with Arne Meier and Juha Kontinen on the parameterized complexity of MC for first-order dependence logic ($\mathcal{D}$). This work was published in the proceedings of the International Symposium on Logical Foundations of Computer Science (LFCS 2022) [70]. The notable contributions include answering an open question by Virtema [111, P.88] on the expression complexity of the fixed variable fragment ($\mathcal{D}^k$) of dependence logic. Recently, we extended our analysis to independence logic and the resulting work is under review for the Journal of Logic and Computation. Together with Jonni Virtema (Sheffield), I explored the parameterized complexity of propositional independence ($\mathcal{PIND}$) and inclusion logic ($\mathcal{PINC}$) [84]. My major contribution in this work includes answering an open question by Hella and Stumpf [58, P.13] regarding the complexity of satisfiability for the fixed arity fragment of propositional inclusion logic ($\mathcal{PINC}$). I find it worth mentioning that, although the title of my thesis consists of *team-based formalisms*, only propositional dependence logic ($\mathcal{PDL}$) is included in this writing. Including other logics would make this thesis too large to be readable.

During the visit of Markus Hecher and Johannes Fichte (TU Vienna), we discussed how interesting 'treewidth' as a parameter is. The motivation stems from the fact that given a tree decomposition for an instance of a problem P, then a decomposition-guided (DG) reduction from P to the satisfiability problem (SAT or QBF) can help in solving P using SAT- or QBF-solvers. Moreover, one can prove tight lower bounds on the runtime of an algorithm that solves P under the exponential time hypothesis (ETH) via a reduction from SAT/QBF to P. We explored DG-reductions for all three problems in argumentation, and the resulting work was published in the proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI 2021) [44]. We also proved runtime lower bounds under ETH. Later, we discussed with Johannes Schmidt the possibility of finding such reductions for problems in abductive reasoning. We concluded that one could find DG-reductions for ABD as well. This enables one to use QBF-solvers for solving reasoning problems in abduction. However, we could not prove tight lower bounds as in the case of argumentation. This can be explained by the fact that: reducing a QBF-instance to an ABD-instance yields either a non-linear blow up in the treewidth or the resulting theory in an instance of ABD is not of the desired form (CNF/DNF).

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Complexity Theory

Since the beginning of the human race, mankind has strived for *efficiency*. Be this the invention of the wheel to cut down the traveling time, using birds for faster communication, or inventing a computing device to do the math for them. There are many ways to make the term 'efficiency' precise. Generally, it refers to the *best use* of available resources to solve a particular task. The core of complexity theory lies in analyzing various problems and estimating the required resources to solve them. From a computational point of view *space* and *time* are the two most interesting complexity measures of a problem. This is because both space and time are costly: our computer has a finite and limited memory and can do only a certain amount of work in a given time.

Example 1.1. *Consider the scenario where Mr. M wants to search for a flight between two cities. Clearly, the database consisting of the flight schedule is sizeable, while Mr. M does not want his browser to take 5–10 minutes before returning a result.* ◁

The first obvious question arises 'how do we measure space and time?' Do we calculate the time as 'time on a watch', or should we consider the clock ticking of the processor? These absolute measures may yield fascinating results, but they do have a limitation: the obtained results are processor-specific. Nevertheless, we want our complexity analysis to be independent of these details. In other words, we want the complexity of a problem to be independent of whether we solve this on a Mac machine with a 3.1 GHz processor, on a 20th century 1.6 Pentium computer with Windows OS, or maybe on a supercomputer. For this reason, we abstract away all these details and consider the most basic machine model for computation, a Turing machine. A *Turing machine* (TM) is a widely used and, for all practical purposes, an accurate model of our general-purpose computers [106]. We define the time required to solve a problem P as the time taken by a TM $\mathbb{M}$ that solves P. Moreover, we calculate the time with respect to the input

size. This causes another concern as there might be two input instances of the same size, but an algorithm may solve one relatively faster than the other. To overcome this issue, we consider the time taken in the *worst case* to solve a problem of size n. It is then clear that there can be no input instance of size n that takes more than this worst-case time. We defer the detailed description and the precise formulation of TMs until preliminaries.

The goal of the complexity analysis is twofold: (1) to find efficient algorithms and (2) to compare the complexity of considered problems relative to each other. Some problems are intrinsically hard, that is, there is no better way (known) to solve them than to brute-force all possible solutions. Finding an *efficient algorithm* amounts to finding a procedure to solve a problem that runs in polynomial time with respect to the input. If a problem is efficiently solvable, we call it *tractable* and it belongs to the complexity class **P**. Whereas, if there is no efficient (known) way of solving a problem, then it is *intractable*. Intractable problems fall in various complexity classes depending on the running time or space usage of a Turing machine that solves them. One such intractability class is **NP**, which includes the problems solvable by a non-deterministic TM (NTM) running in polynomial time. An NTM is a TM that can also guess during its computation. Equivalently, **NP** consists of problems for which the correctness of a solution to an instance can be verified in polynomial time. Characterizing the relative complexity of two problems requires finding an efficient way of *reducing* one of the problem to the other. This is achieved by the concept of *polynomial-time reduction* between problems. A problem A reduces to a problem B in poly-time if there is a polynomial-time computable function f mapping yes-instances (resp., no-instances) of A to yes-instance (no-instances) of B. Finally, **NP**-complete problems are the hardest problems in **NP**. A problem P in **NP** is **NP**-complete if every problem $\mathrm{A} \in \mathbf{NP}$ reduces to P in polynomial-time.

Example 1.1 presents an instance of a well-known problem known as query evaluation, or the model checking problem. The input consists of a structure (a database) and a query, the task is either to determine whether the query is true in the structure or to return the result of the query. Moreover, in literature, there exist two additional flavors of these problems (cf. [74, Def. 6.1]). The problem when the query and the database are parts of the input is known as the *combined complexity*. Whereas, if the structure (resp., query) is fixed and the input consists of the query (structure), the problem is known as the expression (data) complexity of the model checking.

In practice, it is often the case that an input instance of a problem is exceedingly large. Consider Example 1.1 where the query and the database are parts of the input. Notice that it is unlikely that either the database or the query is fixed in advance. Consequently, one can not study either the data or the expression complexity in this example. Searching even a small-sized query against a large-sized database can be computationally expensive. Let us consider some additional details, such as Mr. M wanting to search the database for flights between Hannover in Germany and Lahore in Pakistan. One might use the fact that the query is small in size and, among other things, it is preferable to find a flight between the two cities using at most one stopover. This way, an algorithm that uses this additional information may perform better. This concept of *additional information* is formalized further in the following discussion.

This work focuses on the parameterized complexity analysis for problems. The motivation stems from the fact that although some problems are intrinsically hard, the principal source of the complexity is often just a little part of the input. This allows one to separate that part of the input (called the *parameter*) and to study the complexity with respect to the parameter and the input separately. Parameterized complexity [32] is a widely studied subfield of complexity theory. The idea (once again) is to characterize the relative complexity of problems but also considering the runtime dependence on the parameter this time. A parameterized problem (PP) Π consists of two parts, the inputs and the parameterization. We say that Π is fixed-parameter tractable (**FPT** for short), if it can be decided by a DTM that runs in polynomial time with respect to the input and the runtime regarding the parameter is some arbitrary computable function. **FPT** is the counterpart of **P** in the parameterized complexity world. However, problems not in **FPT** have a much varying degree of intractability in the parameterized world. The class **paraNP** is the non-deterministic variant of **FPT**. A PP Π is in **paraNP** if it can be decided by an NTM that runs in polynomial time with respect to the input and some arbitrary computable function with respect to the parameter. Moreover, the class **XP** includes problems decidable by a DTM in polynomial time for each fixed value of the parameter. One can define additional complexity classes by restricting the non-determinism of an NTM. The class $\mathbf{W}[\mathbf{P}]$ consists of problems decideable by an NTM that runs in polynomial time with respect to the input, some arbitrary computable function f with respect to the parameter, and the number of non-deterministic steps is bounded by a function h of the parameter. Moreover, in between **FPT** and $\mathbf{W}[\mathbf{P}]$, there exists an infinite hierarchy of classes known as the **W**-hierarchy. The following example highlights this interesting degree of intractability and promotes the parameterized complexity analysis.

Example 1.2. *Let $\mathcal{G} = (V, E)$ be an undirected graph with the set of vertices V, edges $E = \{\{v, w\} \mid v, w \in V\}$ and $k \in \mathbb{N}$ be a natural number. Two vertices v and w are* adjacent *if $\{v, w\} \in E$. Consider the following decision problems.*

VertexCover: *Is there a set S of size k, such that S includes at least one endpoint of every edge $e \in E$?*

IndependentSet: *Is there a set S of size k, such that no two vertices $v, w \in S$ are adjacent?*

DominatingSet: *Is there a set S of size k, such that every vertex $v \in V \backslash S$ is adjacent to some $w \in S$?*

k-Coloring: *Is there a coloring of V into k-colors, such that no two adjacent vertices have same color?* ◁

Concerning the classical complexity, these problems are **NP**-complete (see for example [106]) and enjoy the same complexity up to polynomial-time reductions. However, in parameterized setting, these problems fall in different complexity degrees when parameterized by k. The problem VertexCover is **FPT**, IndependentSet is $\mathbf{W}[1]$-, DominatingSet is $\mathbf{W}[2]$- and k-Coloring is **paraNP**-complete (see [46]).

1.2 Logic and Reasoning

We use logic and reasoning in almost everything we do or say in our daily life. This includes stating observations, defining a concept and formalizing theories, drawing conclusions, and convincing others with our arguments. The story does not end here, as logic is one of the fundamental aspect of working in computers. It is logic that makes our computers as powerful as they are today. At the primary level, logic tells a system what to do when a key is pressed. At more advanced levels, it is used in systems to validate various engineering designs, diagnose failures and return the result of a query against a database.

The rudimentary components of logic are *statements* as well as *expressions* formed using a specific set of rules. Statements are used to depict the relationship between various entities, which are interpreted in a *model*. A model is simply a view of the universe that contains various objects and relationships between them. In this thesis, we work with *propositional logic*. A proposition is a statement that can either be true or false, such as 'it is raining' or 'today is Sunday'. The universe of our model is then $\{0,1\}$ or $\{F,T\}$. The unary operator '$\neg$: negation' switches the truth value of propositions. Other complex statements (or *formulas*) in propositional logic are constructed from propositions or their negations by connecting them with the help of '$\wedge$: conjunction' and '$\vee$: disjunction'. The truth of a formula is evaluated under an *assignment* of propositions to 0 or 1. In the *satisfiability* problem, the task is to search an assignment that evaluates a given formula to true. Propositional logic ($\mathcal{PL}$) is a simple form of logic where formulas are constructed from propositions by using connectives $\{\wedge,\vee,\neg\}$. More powerful formulas can be constructed by *quantifying* variables. That is, there is some valuation of x ($\exists x$) or for every valuation of x ($\forall x$). There are extensions and generalizations of $\mathcal{PL}$ such as first-order logic, temporal logic, and modal logic. In this thesis, we consider an extension (propositional dependence logic) and a generalization (the constraint satisfaction) of propositional logic. Moreover, we focus on the complexity-theoretic aspects of various problems pertaining to these logics and not involve ourselves in the expressiveness issues — a different area of research in logic (see further [74]).

In the context of propositional logic, one often considers a formula in a so-called *conjunctive normal form* (CNF). That is, a formula is a conjunction of *clauses* where each clause is a disjunction of *literals*. A literal is simply a proposition or its negation. The collection of assignments that satisfies a formula can be seen as a *logical relation* over $\{0,1\}$. However, one can go the other way around by first identifying n-ary relations over $\{0,1\}$ and then constructing formulas by taking conjunctions of these relations.

Example 1.3. *Consider the scenario where* 4 *people (p_i for $i \leq 3$) sit in two different offices at the same faculty. The pandemic and its resulting social distancing allow only specific sitting arrangements at any time. An example 'office visiting plan' is given by the logical relation $R = \{1011, 1110, 0011, 0101\}$. The tuple* 1011 *depicts that $\{p_0, p_1, p_3\}$ are allowed to visit office whereas $\{p_1\}$ is not allowed.* ◁

Schaefer's framework [103] provides this very generalization of formulas in CNF. For a k-ary relation R, a constraint is an expression of the form $R(x_1, \ldots, x_k)$ and it generalizes the notion of

a clause. Moreover, for a set Γ of relations, a Γ-formula is simply a conjunction of constraints with relations from Γ. The main idea behind Schaefer's framework is that: given a set Γ of relations, what other relations can be defined from Γ by allowing conjunctions and the existential quantification? This approach provides an infinite class of satisfiability problems that contains classical CNF-satisfiability as a special subcase. Schaefer's dichotomy theorem states that every member of this class of problems is either tractable (in **P**) or **NP**-complete. The results proved by Schaefer [103] emphasize the importance of the framework on its own.

> "[The dichotomy theorem] is potentially very useful in expediting **NP**-completeness proofs, for the reason that [it] gives one a much broader 'target cross-section' for use in reductions."

The motivation behind such a generalized framework is that it allows one to no longer aim for a specific **NP**-complete problem for proving hardness, as now there are a latitude of problems available. A similar (yet different) approach is due to Post [96]. In Post's framework, one considers *functions* and their closure under functional composition. Our focus in this work is on Schaefer's approach and therefore, we omit the detailed discussion on Post's framework.

Dependence and independence among various entities are natural phenomena observed in our everyday life and scientific reasoning. Whether or not it rains today is independent of whether today is Friday, whether or not the author visits Pakistan this year is dependent on whether he completes his thesis within the deadline. In mathematics, an expression of the form $y = f(x)$ depicts the dependence of y on an independent variable x. Moreover, there exists a well-known notion of *functional dependencies* (introduced by Codd in 1970) in the context of databases. For variables x and y, the dependency $x \to y$ states that for every pair of rows (tuples) in the database, whenever they have the same value for x, then they must also have the same value for y.

In order to motivate the necessity of dependence in logic, we consider the following scenario. It was recognized that the usual dependence of variables in a formula of the form $\forall x \exists y \forall u \exists v \phi$ is such that: each quantified variable depends on all others appearing before itself in the sequence. In particular, the valuation of y depends upon x, whereas the valuation of v depends on that of x, y, u. An obvious question arose, what if we want v not to depend on x and y. One possible solution was presented in terms of Henkin's *partially ordered quantifiers* [59]. Henkin introduced the idea of *branching quantifiers* and an expression of the following form.

$$\Phi := \begin{pmatrix} \forall x \exists y \\ \forall u \exists v \end{pmatrix} \phi \tag{1.1}$$

The formula Φ has an interpretation that the valuation of y (resp., v) depends only upon the valuation of x (u). As an alternative approach, Hintikka and Sandu [60] proposed *independence friendly logic* ($\mathcal{IF}$). The idea was to introduce the so-called *slashed*-quantifiers, resulting in expressions of the form $(\forall x)(\exists y)(\forall u)(\exists v/\forall x)\phi$. The intuition once again is to express the (in)dependence of variables on each other. Hintikka and Sandu gave game-theoretic semantics

for their $\mathcal{IF}$ logic (instead of the compositional/Tarski-like semantics). The challenge of devising compositional semantics in opposition to Hintikka's game-theoretic semantics was later tackled by Hodges [62] through the introduction of *trump semantics*. For a better historical background, a reader is advised to consult the introduction chapter of Martin Lück's Ph.D. thesis [77].

Jouko Väänänen introduced a new atomic formula to the classical logic in order to capture the notion of dependence [109]. A *dependence* atom of the form $\mathsf{dep}(\mathbf{x}; y)$ states that the value of y depends on the values of variables in $\mathbf{x}$ and nothing else. Along the same lines as Hodges, Väänänen introduced *team semantics* where the truth of a formula is evaluated over a set of assignments called a *team*. The central idea of a teams is the following fact by Väänänen [109].

> "Dependence manifests itself in the presence of the multitude. A single event cannot manifest dependence, as it may have occurred as a matter of chance."

In this thesis, our focus is on the propositional dependence logic ($\mathcal{PDL}$) and therefore we only consider propositional teams. The usual Tarski semantics for $\mathcal{PL}$-formulas is extended to the team semantics by simply requiring that every assignment in the team satisfies a formula. Enriching the classical propositional logic with team semantics by adding the dependence atoms yields the propositional dependence logic ($\mathcal{PDL}$). The example formula Φ from Equation 1.1 in dependence logic becomes $\forall x \exists y \forall u \exists v(\phi \wedge \mathsf{dep}(u; v))$. Moreover, a dependence atom naturally corresponds to the notion of functional dependency in the database setting.

Example 1.4. *The database considered in Example 1.1 when seen as a team satisfies the dependence atom* $\mathsf{dep}(\{\texttt{Flight}, \texttt{Date}, \texttt{Time}\}; \texttt{Destination})$. *It highlights that the destination is uniquely determined by the flight for a particular date and time.* ◁

It is worth pointing out that the satisfiability and the model checking problems for $\mathcal{PDL}$ enrich these two problems from the classical $\mathcal{PL}$-setting. Furthermore, one can encode the domain D of a fixed first-order ($\mathcal{FO}$) structure in binary and thereby translate an $\mathcal{FO}$-structure into the propositional setting. This implies that propositional dependence logic (although a restricted version of first-order dependence logic) is still relevant in practice.

The introduction of dependence logic has resulted in an active community of researchers spanning different areas including databases, statistics, and social choice to name a few. The literature on team-based formalisms is very rich, as it includes various other dependency notions such as *independence* [50], *inclusion* and *exclusion* [47, 56]. Moreover, the focus also has widened from the first-order dependence logic further to modal logic [110], temporal logics [72, 71], probabilistic logics [36], multi-team semantics [35], and poly-team semantics [54].

Logical Reasoning Reasoning (in general) is the process of reaching conclusions on the basis of a careful consideration of the available information. Abductive reasoning as a type of inference is frequently employed on various occasions in our everyday life. It is one of the most important aspects of common sense reasoning [10]. The term 'abduction' finds its origin in the combination of Latin words ab- ('from, away from') and dūcō ('to lead, to draw'). That is, to draw from or

to lead somewhere away from [something]. This makes abduction the process of leading to a reason or an *explanation* for an *event* from the *given information*. Such an explanation must (1) align with one's belief or the given information, also known as the *knowledge base*, and (2) explain the manifestation of the given event logically. One can further impose a minimality or the size criteria for an acceptable explanation. Pierce [90, (CP 5.171)] proposed that abductive rasoning is the only component of logic that produces any new knowledge.

> "Abduction is the process of forming an explanatory hypothesis. It is the only logical operation that introduces any new idea; for induction does nothing but determine a value, and deduction merely evolves the necessary consequences of pure hypotheses"

The abductive reasoning has two different aspects or flavors, namely the *creative* and the *selective* abduction (cf. [105]). Creative abductions can introduce new concepts or models, whereas the task of selective abductions is to choose the *best candidate* among a given multitude of possible explanations. Moreover, abduction can only offer hypotheses which may be refuted with additional information. This is the reason why abduction is often referred to as an inference to the best 'available' explanation. The following example highlights how abduction is engrained as a fundamental component of common sense reasoning in human.

Example 1.5. *Mr. M informs his family that he plans to visit them in March. However, in March, he could not travel, and his family members are wondering what might be the reason. Either M was very busy at work, there were new Corona regulations, or he ran out of money. Consider some additional information that M would visit his family only when he has completed his thesis, as well as, he received his booster vaccine for Covid19, and people vaccinated three times are allowed to travel. Moreover, M has saved enough money for his tour. With the help of abduction, his brother concludes that M was busy writing his thesis, and he will travel once it is complete.* ◁

Abductive reasoning is a non-monotonic process. This is because adding new information to the knowledge base may result in some explanations to become invalid. The components of abductive reasoning include a *knowledge base* (given information), a *manifestation* (an event to be explained) and *hypotheses* (candidate explanations). In this thesis, our focus is on the logic-based abduction. Thereby, the knowledge base (KB) is modeled as a propositional logic theory, whereas the manifestation and hypotheses are modeled by propositional variables. An instance of the abduction problem consists of these three components. The task is to find a subset of hypotheses that, together with the knowledge base, is logically consistent and entails the manifested event. Abductive reasoning is an important concept in AI as emphasised by Morgan [88] and Pole [94]. Moreover, abduction is also used in the process of medical diagnosis, a physician hypothesizes the disease that best explains a patient's symptoms [64, 91]. Other applications of abductive reasoning include planning [41], database updates [65], natural language understanding and text generation [15, 61], and machine learning [20, 63, 66, 95].

In abductive reasoning, it is often argued that the lack of a particular piece of information does not play any role, and only the 'available information' is relevant in finding an explanation. This is

contrary to the *closed-world assumption* (introduced by Reiter [98]) where the lack of information is also important in reasoning. One assumes that all the relevant information has been stated in the knowledge base and any fact not specified is assumed to be false. Often in abduction one does not assume that the facts not included in an explanation are false. Nevertheless, Pfandler et al. [92] proposed a new definition of explanations based on a closed-world approach. Their motivation lies in the fact that the *open-world assumption* may not always yield minimal explanations.

The study of *argumentation* is also closely connected to that of reasoning. Upon making a claim, someone may request further *support* for the claim in the form of reasons. However, there are differences between abductive reasoning and the process of argumentation. In abductive reasoning, an individual's viewpoint is addressed, and consequently, this view of each individual or the *knowledge base* is assumed to be consistent. However, argumentation manifests itself while exchanging reasons in the presence of more than one views (possibly) contrary to one's own belief or the knowledge base.

Argumentation is a crucial aspect of communication in our daily lives. The importance of argumentation, as put forward by Dung [33] is emphasized by the following fact.

> "The ability to engage in an argument is essential for humans to understand new problems, to perform scientific reasoning, to express, clarify and to defend their opinions in their daily lives."

Clearly, to gain, understand, and process new information, one has to question themself, especially when the new information is in conflict with one's previous knowledge. For humans, argumentation works on the principle of "*the one who has the last word laughs best*". That is, in an exchange of arguments, the winning argument is the one that cannot be further counter-argued. The research area in argumentation is broad, as discussed next. Some important aspects of argumentation include: (1) constructing arguments and studying relationships between them, (2) drawing conclusions based on a set of arguments, and (3) finding consistent (in some precise/pre-given sense) sets of arguments that can be accepted together. Each of these three components is well-studied, as we shortly discuss.

Dung's argumentation framework [33] models the collection of arguments as a directed graph where each node represents an argument. The (directed) edge represents the attack relation between two arguments. The task is to find a coherent set of arguments which can be accepted simultaneously. Assumption-based argumentation (ABA) [34] is a richer formalism spanning all three aspects of the argumentation. In ABA, one considers the whole process starting from constructing arguments in an underlying logic, identifying conflicts between them, and returning conclusions based on the acceptance of specific arguments. Clearly, the considered logic plays an important role in the complexity of reasoning problems in ABA.

The focus of this thesis is *logic-based argumentation* introduced by Besnard and Hunter [7, 8]. One considers propositional logic ($\mathcal{PL}$) together with its *entailment relation* ($\models$) to construct arguments over a knowledge base. In logic-based argumentation, one is given a *knowledge base* as a collection of formulas (can be inconsistent altogether) together with a $\mathcal{PL}$-formula as a *claim*.

The task is then to construct a minimal *support* for the claim. That is, a minimal (with respect to set inclusion) consistent subset of the knowledge base which logically entails the claim. The pair consisting of the support and the claim is then called an *argument*. Below we illustrate an example for better understanding.

Example 1.6. *(A1): One needs money and visa to be able to travel, and a job to fulfil these two requirements. Mr. M argues that he will travel to Pakistan. His argument in favour of travelling is supported by the fact that he adheres to both conditions. Now consider another scenario. (A2): New Corona regulations caused the visa office to shut down. As a result, the argument that M can not have a visa anytime soon, follows.* ◁

The overall information in Example 1.6 is inconsistent because the claim of A2 (M do not have a visa) is contrary to the support of A1 (M has a visa). This highlights the very nature of argumentation that one wants to find the support for a particular claim in the presence of contradictory information.

1.3 Related Work

This thesis addresses the computational problems from three different domains of logic. The literature on the team-based formalisms is very rich, as we shortly discuss next. The three most popular logics in team-semantics are dependence $\mathcal{D}$ [109], independence $\mathcal{FO}(\perp)$ [50] and inclusion logic $\mathcal{FO}(\subseteq)$ [47]. Concerning the computational problems, the satisfiability and the finite satisfiability for all three logics are undecidable (see [69]). Erich Grädel [49] explored the complexity of model checking for $\mathcal{D}$ and $\mathcal{FO}(\perp)$. His analysis includes the expression and the combined complexity, whereas, the results regarding the data complexity were explored by Kontinen [68]. Recently, Kontinen et al. [70] explored the parameteriezed complexity of model checking for $\mathcal{D}$. Fan Yang has explored the propositional logic of dependence in depth [114]. However, model checking and satisfiability for modal dependence logic (and propositional dependence logic) were studied by Peter Lohmann [39, 76]. Hannula et al. [55] (resp., Mahmood and Virtema [84]) explored the classical (the parameterized) complexity of model checking and satisfiability for propositional independence and inclusion logic.

An overview of the complexity results in abstract argumentation or the assumption-based argumentation can be found by the work of Dvorák and Dunne [38]. The complexity of logic-based argumentation in Schaefer's Framework was first attacked by Schmidt et al. [22]. The problems in abductive reasoning have also been explored in depth with various restrictions over the allowed knowledge-base, the manifestation, and the hypotheses. The work by Nord and Zanuttini [89] includes a detailed analysis with respect to these restriction over an input-instance for problem in abduction. Fellows et al. [43] initiated the parameterized complexity analysis of abductive reasoning for the fragments of the CNF-formulas. The work in this thesis is essentially an expansion of their work, as we now consider not only the CNF-fragments but also further restriction over the allowed formulas.

Organization

We begin with preliminaries in Chapter 2. This gives a brief and comprehensive introduction to all the concepts and notions used in this writing. In Chapters 3, we discuss the parameterized complexity of propositional dependence logic. This is followed by the abductive reasoning in Chapter 4 and argumentation in Chapter 5. Each chapter begins with a discussion on the considered parameters. This includes defining each parameterization and proving relationships between them. Definitions used only in a specific proof or a theorem are included in the appropriate subsections to make preliminaries less exhaustive. Chapter 6 concludes the thesis with interesting remarks and possible directions for the future work.

Figure 1.1 presents a pictorial overview of the related work with brown circles depicting the work included in this thesis.

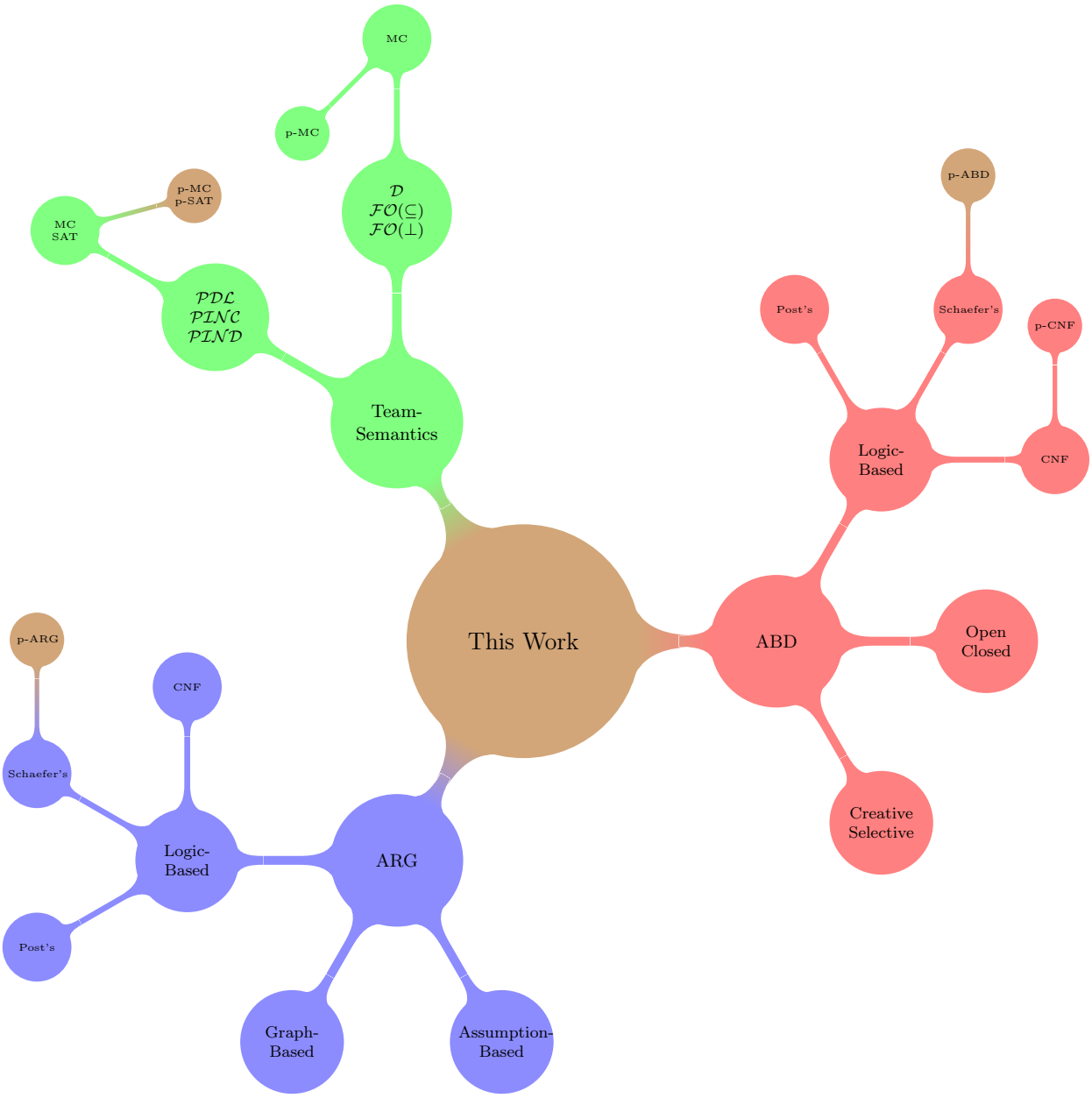


Figure 1.1: An overview of the related work. A p-suffix indicates the parameterized complexity analysis of the problem. Circles in brown indicate the content of this work.

CHAPTER 2

PRELIMENARIES

The reader is assumed to have at least a basic mathematical knowledge. This includes familiarity with objects such as variables, sets, functions, etc. We begin by fixing the notation and defining the terminology used in this thesis.

2.1 Propositional Logic

A *propositional variable* can take one of the two possible values, 0 or 1 (equivalently, False or True). All variables considered in this work are propositional, unless stated otherwise. We denote variables by small English letters such as x, y, p, q etc., possibly with indices and reserve capital letters X, Y, P, Q for sets of variables. A collection of variables $\{x_1, \ldots, x_n\}$ when seen as a tuple is denoted in boldface font $\mathbf{x}$. If X is a set then $|X|$ denotes the number of elements in X. The classical propositional logic constitutes propositional statements which are often denoted by variables. Formulas of propositional logic are obtained from variables using connectives, namely 'conjunction' ($\wedge$), 'disjunction' ($\vee$), and 'negation' ($\neg$). That is, if p and q are propositions then each of $p, q, \neg p, p \wedge q, p \vee q$ are propositional formulas. We find it convenient to write $x \to y$ instead of $\neg x \vee y$ and $x \leftrightarrow y$ for $(x \to y) \wedge (y \to x)$. A literal is a variable x or its negation $\neg x$. There are two special formulas, True ($\top$), which is always true and False ($\bot$), which is always false. We denote propositional formulas by small Greek letters $\phi, \psi, \theta, \gamma$ and reserve capital letters Φ, Ψ, Γ for collection of formulas. The propositional logic is denoted by $\mathcal{PL}$. An observation, or more formally, an *assignment* s with a domain X of variables assigns a truth value of 0 or 1 to each proposition. That is $s\colon X \to \{0,1\}$ is a mapping. For a tuple $\mathbf{x} = (x_1, \ldots, x_n)$ of variables we denote the tuple $(s(x_1), \ldots, s(x_n))$ by $s(\mathbf{x})$. In such an event, we sometimes find it convenient to write s as a bit string $\mathbf{b}$ of length n to indicate that $s(x_i) = b_i$ for $i \leq n$. Moreover, we also write $s \in 2^X$ and view s as a subset of X comprising of $x \in X$ with $s(x) = 1$. We say that s satisfies a proposition $p \in X$ iff $s(p) = 1$ and denote this as $s \models p$. Then s does not satisfy p ($s \not\models p$) or p

is false in s if $s(p) = 0$. The definition of satisfaction (also known as *Tarski semantics*) extends to other $\mathcal{PL}$-formulas as follows.

$$\begin{aligned} s &\models \top \quad \text{and} \quad s \not\models \bot, \text{ always,} \\ s &\models \neg\phi \quad \text{iff} \quad s \not\models \phi, \\ s &\models \phi \wedge \psi \text{ iff} \quad s \models \phi \text{ and } s \models \psi, \\ s &\models \phi \vee \psi \text{ iff} \quad s \models \phi \text{ or } s \models \psi. \end{aligned}$$

Variables in a $\mathcal{PL}$-formula ϕ are denoted as $\mathrm{Vars}(\phi)$. We say that ϕ is *satisfiable* if there is an assignment s over $\mathrm{Vars}(\phi)$ such that $s \models \phi$ and otherwise, ϕ is *unsatisfiable*. The propositional logic satisfies the so-called *law of excluded middle*. That is, for every $\mathcal{PL}$-formula ϕ and an assignment s, either $s \models \phi$ or $s \models \neg\phi$. It is crucial to mention this property here because we will soon introduce an extension of $\mathcal{PL}$ (propositional dependence logic) that fails it.

In the following, we enlist problems that interest us in this thesis. The *model checking* and the *satisfiability* problem asks whether an assignment s satisfies a formula ϕ or whether there is such a satisfying assignment, respectively.

Problem:	$\mathcal{PL}$-MC
Input:	A $\mathcal{PL}$-formula ϕ and an assignment s over $\mathrm{Vars}(\phi)$.
Question:	Does $s \models \phi$?

Problem:	$\mathcal{PL}$-SAT
Input:	A $\mathcal{PL}$-formula ϕ.
Question:	Is there an assignment s over $\mathrm{Vars}(\phi)$ such that $s \models \phi$?

The complement problem to satisfiability is the *unsatisfiability* problem (UNSAT) asking whether the input formula is not satisfiable. A $\mathcal{PL}$-formula ϕ is *valid* or a *tautology* if $s \models \phi$ for every assignment $s\colon \mathrm{Vars}(\phi) \to \{0,1\}$. For two $\mathcal{PL}$-formulas ϕ and ψ, we say that ϕ *entails* (or *implies*) ψ iff for every assignment s over $\mathrm{Vars}(\phi) \cup \mathrm{Vars}(\psi)$ we have: if $s \models \phi$ then $s \models \psi$. This is denoted as $\phi \models \psi$. Finally, ϕ and ψ are *equivalent*, $\phi \equiv \psi$ if and only if $\phi \models \psi$ and $\psi \models \phi$. The *tautology* problem asks whether a given formula is a tautology.

Problem:	$\mathcal{PL}$-TAUT
Input:	A $\mathcal{PL}$-formula ϕ.
Question:	Does $s \models \phi$ for every assignment s over $\mathrm{Vars}(\phi)$?

Finally, the *implication* problem asks for a pair of formulas ϕ and ψ whether $\phi \models \psi$.

Problem:	$\mathcal{PL}$-IMP
Input:	Two $\mathcal{PL}$-formulas ϕ and ψ.
Question:	Does $\phi \models \psi$?

The classical $\mathcal{PL}$-formulas are often considered in a so-called *conjunctive* or *disjunctive normal forms* (CNF/DNF). A *clause* is a disjunction of literals written as $c = (l_1 \vee l_2 \vee \cdots \vee l_r)$. A formula ϕ is in CNF if $\phi = \bigwedge_i c_i$ where c_i for each $i \leq n$ is a clause. A *term* is a conjunction of literals written as $t = (l_1 \wedge l_2 \wedge \cdots \wedge l_r)$. A formula ϕ is in DNF if $\phi = \bigvee_i t_i$ where t_i for each $i \leq n$ is a term. A clause, term or a CNF/DNF is called positive (resp., negative) if it includes only positive (negative) literals. One can impose these additional restrictions on the type of formulas included in a problem P. In such a scenario, for example, P(CNF) denotes that an input to P consists of only formulas in CNF. Finally, when there is no ambiguity, we drop the logic prefix from the problem and write, for example, SAT when the considered logic is obvious from the context.

Sometimes, we come across an assignment s that has a subset $Y \subseteq \mathrm{Vars}(\phi)$ as its domain. In this case, we talk about the *reduct* of ϕ under s. Formally, let ϕ be a formula in CNF and s be an assignment over Y. The reduct $\phi[s]$ of ϕ is defined as the propositional formula obtained from ϕ by deleting all clauses satisfied by s and deleting all literals set to 0 by s from the remaining.

Propositional Dependence Logic

We first extend *Tarski's semantics* from the previous subsection to the *team semantics*. A *team* T over a domain X is a collection of assignments s_i such that $s_i \colon X \to \{0,1\}$. Given a propositional logic formula $\mathcal{PL}$ and a team T over $\mathrm{Vars}(\phi)$, we say that T satisfies ϕ (written as $T \models \phi$) iff $s \models \phi$ for every $s \in T$. Now we enrich the syntax of propositional logic by adding *dependence atoms* of the form $\mathsf{dep}(P;Q)$ where P, Q are sets of variables. The so-obtained logic is called *propositional dependence logic* ($\mathcal{PDL}$). It is worth mentioning that we only allow negation symbols to appear in front of atomic formulas. Allowing arbitrary negation yields the formation of more complex formulas and therefore is out of the scope of this thesis (see further [55]). The semantics for $\mathcal{PDL}$-formulas is stated below.

$$
\begin{array}{lll}
T \models x & \text{iff} & \forall s \in T : s(x) = 1 \\
T \models \neg x & \text{iff} & \forall s \in T : s(x) = 0 \\
T \models \mathsf{dep}(P;Q) & \text{iff} & \forall s, s' \in T : \bigwedge_{p \in P} s(p) = s'(p) \text{ implies } \bigwedge_{q \in Q} s(q) = s'(q) \\
T \models \neg\mathsf{dep}(P;Q) & \text{iff} & T = \emptyset \\
T \models \phi \wedge \psi & \text{iff} & T \models \phi \text{ and } T \models \psi \\
T \models \phi \vee \psi & \text{iff} & \exists T_1 \exists T_2 (T_1 \cup T_2 = T), T_1 \models \phi \text{ and } T_2 \models \psi
\end{array}
$$

When either set P or Q in an atom $\mathsf{dep}(P;Q)$ is a singleton, we simply write it as an element rather than the set. For example, we write $\mathsf{dep}(P;q)$ instead of $\mathsf{dep}(P;\{q\})$. The disjunction operator ($\vee$) in the context of $\mathcal{PDL}$ is often referred to as the *split-junction*. In literature, there are two different semantics for the split-junction (see further [57]). The one defined above is the so-called *lax*-semantics. An alternative is the *strict*-semantics, which forces the split of the team into disjoint subteams. Nevertheless, both semantics coincide for $\mathcal{PDL}$ [109] and we use the lax-semantics.

Flight	Date	Departs	Route	Price($)
EY002	20.03.22	11:20	FRA-AUH-LHR	370
TK592	20.03.22	13:00	FRA-IST-LHR	320
TK102	20.03.22	11:00	HAJ-IST-LHR	350
QR068	21.03.22	17:35	FRA-DOH-LHR	330
EY002	21.03.22	11:20	FRA-AUH-LHR	350
QR068	22.03.22	17:35	FRA-DOH-LHR	310
QR528	22.03.22	10:35	BRE-DOH-LHR	340

Table 2.1: An example flights database.

Example 2.1. *Consider the team T[1] presented in Table 2.1 (seen as a database). Then T satisfies the dependence atom* $\mathsf{dep}(\{\texttt{Flight},\texttt{Date},\texttt{Departs}\};\texttt{Route})$. *Whereas,* $\mathsf{dep}(\{\texttt{Flight},\ \texttt{Route}\};\texttt{Price})$ *fails in T, as depicted by the presence of the pair (EY002, 20.03.2022, 11:20, FRA-AUH-LHR, 370) and (EY002, 21.03.2022, 11:20, FRA-AUH-LHR, 350).* ◁

We now define some well-known properties of $\mathcal{PDL}$-formulas which are also relevant to the results in this thesis. $\mathcal{PDL}$-formulas are *local* in the sense that for a formula ϕ and a team T over variables $X \supseteq \mathrm{Vars}(\phi)$, the satisfaction $T \models \phi$ is only determined by variables in $\mathrm{Vars}(\phi)$. That is, $T \models \phi \iff T\restriction_{\mathrm{Vars}(\phi)} \models \phi$. For this reason, we omit writing the domain of T explicitly since it is clear that T is a team over $\mathrm{Vars}(\phi)$. For propositional literals, the team semantics is defined with respect to individual assignments in the team. We can strengthen this property to cover all $\mathcal{PL}$-formulas. A formula ϕ is *flat* if, given a team T, we have $T \models \phi \iff \{\, s \,\} \models \phi$ for every $s \in T$. The classical $\mathcal{PL}$-formulas are flat. Moreover, the team semantics for such formulas coincide with the Tarski's semantics. As a consequence, the following is true for a $\mathcal{PL}$-formula ϕ.

$$T \models \phi \text{ iff } \quad \forall s \in T\colon \{s\} \models \phi \text{ iff } \quad \forall s \in T\colon s \models \phi.$$

$\mathcal{PDL}$-formulas are *downwards closed* in the sense that for every $\mathcal{PDL}$-formula ϕ and the team T, if $T \models \phi$ then $P \models \phi$ for every $P \subseteq T$. Finally, a formula ϕ is 2-*coherent* if for every team T, we have that $T \models \phi \iff \{\, s_i, s_j \,\} \models \phi$ for every $s_i, s_j \in T$. Every dependence atom, as well as every $\mathcal{PDL}$-formula without a split-junction is 2-coherent [68]. It is worth mentioning some properties that $\mathcal{PDL}$-formulas fail. These are also relevant to results in this work.

- Excluded middle: For a formula ϕ and a team T, either $T \models \phi$ or $T \models \neg\phi$.
- Absorption law: For a formula ϕ, $\phi \vee \phi \equiv \phi \equiv \phi \wedge \phi$.

Remark 2.2. *$\mathcal{PDL}$ does not satisfy the law of excluded middle. This is because the team $\{s_1, s_2\}$ where $s_i(y) = i$ for $i = 0, 1$, neither satisfies y nor $\neg y$. Moreover, $\mathcal{PDL}$ fails the absorption law (w.r.t.*

[1]Notice that T is not a propositional team. Nevertheless, it can be translated to the propositional setting via a binary encoding of the possible entries for the variables. This might cause a logarithmic blow-up (by binary encoding the universe values for each column. The parameter values we consider in this chapter correspond to the propositional setting and therefore there is no need to consider this blow-up separately.

disjunction). The team $\{00, 01, 10, 11\}$ *over* $\{x, y\}$[2] *satisfies* $\mathsf{dep}(x; y) \vee \mathsf{dep}(x; y)$*, whereas, it does not satisfy* $\mathsf{dep}(x; y)$.

The computational problems from $\mathcal{PDL}$ that interest us include SAT and MC. The validity and the implication problems have also been studied [52, 112] but will not be considered in this thesis.

Problem:	$\mathcal{PDL}$-MC
Input:	A $\mathcal{PDL}$-formula ϕ and a team T over $\mathrm{Vars}(\phi)$.
Question:	Does $T \models \phi$?

For $\mathcal{PDL}$-SAT, in principle, the question amounts to finding a satisfying team for an input formula ϕ. However, since $\mathcal{PDL}$-formulas are downwards closed, we have for every $\mathcal{PDL}$-formula ϕ and a team T that: if $T \models \phi$ then for every $s \in T, \{s\} \models \phi$. As a consequence, if a $\mathcal{PDL}$-formula ϕ is satisfiable then there is a singleton team $T = \{s\}$ such that $T \models \phi$. On the other hand, if $T \models \phi$ for some singleton team T, then ϕ is clearly satisfiable. This implies that it is enough to find a singleton team to determine the satisfiability of ϕ. Moreover, $\{s\} \models \mathsf{dep}(P; Q)$ trivially for a dependence atom $\mathsf{dep}(P; Q)$ and any assignment s. Consequently, the team semantics and the usual Tarskian semantics coincide for the case of singleton teams. This yields the following version of SAT for $\mathcal{PDL}$.

Problem:	$\mathcal{PDL}$-SAT
Input:	A $\mathcal{PDL}$-formula ϕ.
Question:	Is there an assignment s over $\mathrm{Vars}(\phi)$ such that $\{s\} \models \phi$?

Schaefer's Framework

We now introduce concepts and terminology required to work in Schaefer's framework. Constraints generalize the notion of clauses, whereas constraint languages generalize the notion of classes of clauses. A relation $R \subseteq \{0,1\}^k$ is called a *logical relation* of arity $k \in \mathbb{N}$. A constraint language (CL) Γ is a finite set of relations $\{R_1^{k_1}, \ldots, R_n^{k_n}\}$, where k_i is the arity of R_i. A *constraint* C over R is a formula $C := R(x_1, \ldots, x_k)$, where R is a k-ary logical relation and $x_1, \ldots, x_k$ are (possibly repeating) variables. An assignment $s\colon \{x_1, \ldots, x_k\} \to \{0,1\}$ *satisfies* C, if $(s(x_1), \ldots, s(x_k)) \in R$. Let Γ be a constraint language, then a Γ-formula ϕ is a finite conjunction of constraints over Γ, that is $\phi = \bigwedge_{i=1}^{n} R_i(x_{i,1}, \ldots, x_{i,k_i})$, where each $R_i \in \Gamma$ has arity k_i for $i \leq n$. Finally, let ϕ be a Γ-formula, an assignment s over $\mathrm{Vars}(\phi)$ *satisfies* ϕ if s simultaneously satisfies every constraint in ϕ. The notion of satisfiability, entailment, and equivalence is employed as for propositional logic. Similarly, every problem $\mathrm{P} \in \{\mathrm{SAT}, \mathrm{IMP}, \mathrm{TAUT}\}$ can be defined with respect to a constraint language Γ, denoted as $\mathrm{P}(\Gamma)$.

[2] An assignment s over $X = \{x_1, \ldots x_n\}$ can be seen as a bit string $s(x_1) \ldots s(x_n)$

Example 2.3. *Recall the relation R stating the 'office visiting plan' from Example 1.3. The* constraints *such as 'who else must be present today to take care of other duties' further restricts the combination of people allowed to visit offices. Let* $\Gamma = \{R, S, T\}$ *be a CL where* $R = \{1011, 1110, 0011, 0101\}$, $S = \{11, 00\}$ *and* $T = \{1\}$ *are relations. Then* $\phi := R(x_1, \ldots, x_4) \wedge S(x_1, x_3) \wedge T(x_4)$ *is a* Γ*-formula. Moreover,* $s \models \phi$ *for the assignment* $s\colon \mathbf{x} \mapsto 0101$. ◁

Let C be a constraint, V be a set of variables, and u a variable, then $C[V/u]$ denotes the constraint obtained from C by replacing each occurrence of variables in V by u. Whenever a Γ-formula ϕ (or a constraint C) is logically equivalent to a clause or a term, we write the corresponding clause or term in place of ϕ. We formalize this in the following. A k-ary relation R is *represented* by a formula ϕ in CNF if ϕ is a formula over k distinct variables $x_1, \ldots, x_k$ and $\phi \equiv R(x_1, \ldots, x_k)$. Moreover, we say that the relation R is

- *Horn* (resp., *dual-Horn*) if ϕ contains at most one positive (negative) literal in each clause.
- *Bijunctive* if ϕ contains at most two literals in each clause.
- *Affine* if ϕ is a conjunction of linear equations of the form $x_1 \oplus \ldots \oplus x_n = a$ where $a \in \{0, 1\}$.
- *Essentially negative* if every clause in ϕ is either negative or unit positive. R is *essentially positive* if every clause in ϕ is either positive or unit negative.
- 1-*valid* (resp., 0-*valid*) if every clause in ϕ contains at least one positive (negative) literal.

Furthermore, we say that R is *Schaefer* if it is Horn, dual-Horn, bijunctive, or affine. If a relation R is 1- or 0-valid or both, then we say that R is ε-*valid*. Finally, for any property P of a relation, we say that a CL Γ has P if every relation in Γ has P. Some proofs in Chapter 4 (regarding the base independence of co-clones) require the following equivalent criteria for relations defined above. The binary operations of conjunction, disjunction, and negation are applied coordinate-wise. A relation R is

- *Horn* if and only if $m_1, m_2 \in R$ implies $m_1 \wedge m_2 \in R$.
- *dualHorn* if and only if $m_1, m_2 \in R$ implies $m_1 \vee m_2 \in R$.
- *essentially negative* if and only if $m_1, m_2, m_3 \in R$ implies $m_1 \wedge (m_2 \vee \neg m_3) \in R$.
- *essentially positive* if and only if $m_1, m_2, m_3 \in R$ implies $m_1 \vee (m_2 \wedge \neg m_3) \in R$.

In general, this characterisations is given in terms of the polymorphisms of functions. However, their details are not necessary to understand proofs in this work, and we omit defining these concepts. An interested reader is referred to the literature (e.g. [29]) for a detailed exposition.

Definition 2.4. *Let* Γ *be a CL.*

1. *The set $\langle\Gamma\rangle$ is the set of all relations that can be expressed as a $\Gamma \cup \{=\}$-formula with existentially quantified variables. That is, the smallest set of relations that contains Γ, the equality constraint, $=$, and which is closed under primitive positive first-order definitions of the following form: if ϕ is a $\Gamma \cup \{=\}$-formula and $R(x_1, \ldots, x_n) \equiv \exists y_1 \ldots \exists y_l \phi(x_1, \ldots, x_n, y_1, \ldots, y_l)$, then $R \in \langle\Gamma\rangle$.*

2. *The set $\langle\Gamma\rangle_{\neq}$ is the set of relations that can be expressed as a Γ-formula with existentially quantified variables (the equality relation is not allowed).*

3. *The set $\langle\Gamma\rangle_{\exists,\neq}$ is the set of relations that can be expressed as a Γ-formula (neither the equality relation nor existential quantification is allowed).*

The set $\langle\Gamma\rangle$ is called a *relational clone* or a *co-clone* and Γ is called the *base* [13]. Notice that for a co-clone C and a CL Γ the statements $\Gamma \subseteq \mathsf{C}$, $\langle\Gamma\rangle_{\neq} \subseteq \mathsf{C}$, $\langle\Gamma\rangle_{\exists,\neq} \subseteq \mathsf{C}$, and $\langle\Gamma\rangle \subseteq \mathsf{C}$ are equivalent.

Example 2.5. *Let $R(x_1, x_2, x_3) := (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Then*

$$(x_1 \vee x_2) \wedge (x_2 \oplus x_3 = 0) \equiv \exists y(R(x_1, x_2, y) \wedge F(y) \wedge (x_2 = x_3)),$$

where $F = \{0\}$. This implies that $(x_1 \vee x_2) \wedge (x_2 \oplus x_3 = 0) \in \langle\{ R, F \}\rangle$. ◁

We follow Schaefer's terminology [103] and refer to different types of Boolean relations and the corresponding co-clones in this work. Table 2.2 presents an overview of co-clones with their relational properties and bases. Finally, notice that $\langle\Gamma\rangle_{\neq} \subseteq \langle\Gamma\rangle$ is true by definition whereas the other direction is not true in general. In order to achieve that $\langle\Gamma\rangle_{\neq} = \langle\Gamma\rangle$, one must prove that $(x = y) \in \langle\Gamma\rangle_{\neq}$. Figure 2.1 depicts a graph representation of the co-clone structure, also known as Post's lattice [97]. Each vertex in the graph corresponds to a co-clone, whereas edges depict the (bottom-up) subset relationship in this lattice. That is, $\mathsf{C} \subseteq \mathsf{D}$ for two co-clones C, D, if C appears below D and there is an edge between the two.

In Section 2.4, we discuss how Post's lattice helps in considering fewer types of CLs when classifying the complexity of a problem in Schaefer's framework.

2.2 Logical Inference

A logical theory for problems in abduction (KB) and argumentation (Δ) consists of formulas that are conjunctions of constraints. An input instance of the abduction problem (ABD) is a tuple (V, H, M, KB) where V is the set of variables, $H, M \subseteq V$ are known as *hypotheses* and *manifestation* respectively, and KB is a Γ-formula known as the *knowledge base*. A subset $E \subseteq H$ is called an *explanation* for M in KB if $E \wedge KB$ is satisfiable and $E \wedge KB \models M$. This version of the abduction problem is known in the literature as *positive*-abduction [89]. The name emphasizes that both H and M are sets of variables. In contrast, $H \subseteq V \cup V^{-}$ can be any (or closed under negation) subset of literals. Similarly, one can introduce various restrictions over M as being a literal, clause, term, or a formula in CNF. We consider M as a positive term denoted as $\bigwedge_i m_i$ for $m_i \in M$.

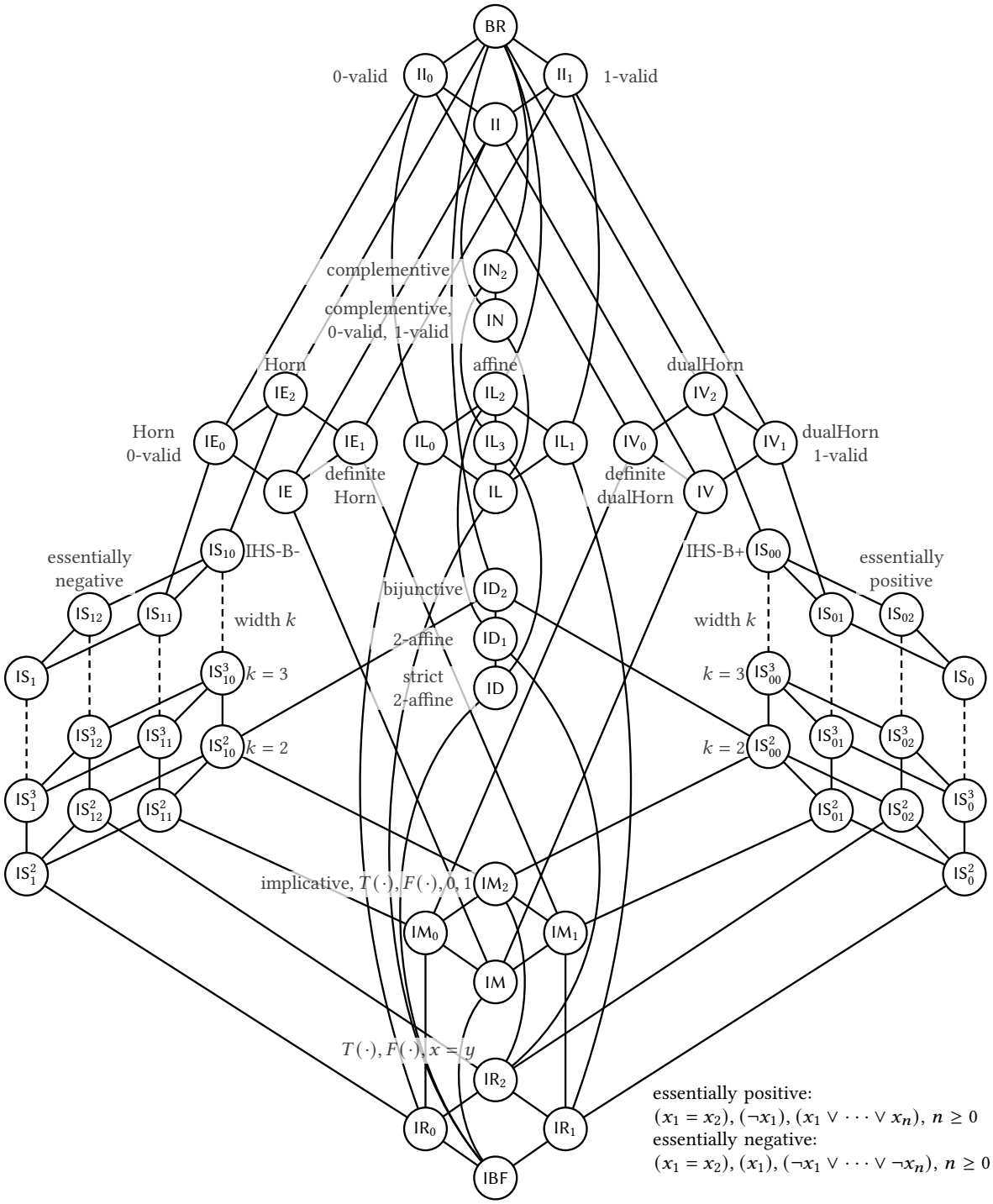


Figure 2.1: Post's lattice of Boolean relational clones

co-clone	base	clause type	name/indication
BR (II_2)	1-IN-3 $= \{001, 010, 100\}$	all clauses	all Boolean relations
II_1	$x \vee (y \oplus z)$	at least one positive literal per clause	1-valid
II_0	DUP, $x \to y$	at least one negative literal per clause	0-valid
II	EVEN^4, $x \to y$	at least one negative and one positive literal per clause	1- and 0-valid
IN_2	NAE $= \{0,1\}^3 \setminus \{000, 111\}$	cf. previous column	complementive
IN	DUP $= \{0,1\}^3 \setminus \{101, 010\}$	cf. previous column	complementive and 1- and 0-valid
IE_2	$x \wedge y \to z, x, \neg x$	clauses with at most one positive literal	Horn
IE_1	$x \wedge y \to z, x$	clauses with exactly one positive literal	definite Horn
IE_0	$x \wedge y \to z, \neg x$	$(x_1 \vee \neg x_2 \vee \cdots \vee \neg x_n), n \geq 2, (\neg x_1 \vee \cdots \vee \neg x_n), n \geq 1$	Horn and 0-valid
IE	$x \wedge y \to z$	$(x_1 \vee \neg x_2 \vee \cdots \vee \neg x_n), n \geq 2$	Horn and 1- and 0-valid
IV_2	$x \vee y \vee \neg z, x, \neg x$	clauses with at most one negative literal	dualHorn
IV_1	$x \vee y \vee \neg z, x$	$(\neg x_1 \vee x_2 \vee \cdots \vee x_n), n \geq 2, (x_1 \vee \cdots \vee x_n), n \geq 1$	dualHorn and 1-valid
IV_0	$x \vee y \vee \neg z, \neg x$	clauses with exactly one negative literal	definite dualHorn
IV	$x \vee y \vee \neg z$	$(\neg x_1 \vee x_2 \vee \cdots \vee x_n), n \geq 2$	dualHorn and 1- and 0-valid
IL_2	EVEN^4, x, $\neg x$	all affine clauses (all linear equations)	affine
IL_1	EVEN^4, x	$(x_1 \oplus \cdots \oplus x_n = a)$, $n \geq 0, a = n \pmod 2$	affine and 1-valid
IL_0	EVEN^4, $\neg x$	$(x_1 \oplus \cdots \oplus x_n = 0)$, $n \geq 0$	affine and 0-valid
IL_3	EVEN^4, $x \oplus y$	$(x_1 \oplus \cdots \oplus x_n = a)$, n even, $a \in \{0,1\}$	-
IL	EVEN^4	$(x_1 \oplus \cdots \oplus x_n = 0)$, n even	affine and 1- and 0-valid
ID_2	$x \oplus y, x \to y$	clauses of size 1 or 2	bijunctive, KROM, 2CNF
ID_1	$x \oplus y, x, \neg x$	affine clauses of size 1 or 2	2-affine
ID	$x \oplus y$	affine clauses of size 2	strict 2-affine
IM_2	$x \to y, x, \neg x$	$(x_1 \to x_2), (x_1), (\neg x_1)$	implicative
IM_1	$x \to y, x$	$(x_1 \to x_2), (x_1)$	implicative and 1-valid
IM_0	$x \to y, \neg x$	$(x_1 \to x_2), (\neg x_1)$	implicative and 0-valid
IM	$x \to y$	$(x_1 \to x_2)$	implicative and 1- and 0-valid
IS_{10}	cf. next column	$(x_1), (x_1 \to x_2), (\neg x_1 \vee \cdots \vee \neg x_n), n \geq 0$	IHS-B-
IS^k_{10}	cf. next column	$(x_1), (x_1 \to x_2), (\neg x_1 \vee \cdots \vee \neg x_n), k \geq n \geq 0$	IHS-B- of width k
IS_{12}	cf. next column	$(x_1), (\neg x_1 \vee \cdots \vee \neg x_n), n \geq 0, (x_1 = x_2)$	essentially negative
IS^k_{12}	cf. next column	$(x_1), (\neg x_1 \vee \cdots \vee \neg x_n), k \geq n \geq 0, (x_1 = x_2)$	essentially negative of width k
IS_{11}	cf. next column	$(x_1 \to x_2), (\neg x_1 \vee \cdots \vee \neg x_n), n \geq 0$	-
IS^k_{11}	cf. next column	$(x_1 \to x_2), (\neg x_1 \vee \cdots \vee \neg x_n), k \geq n \geq 0$	-
IS_1	cf. next column	$(\neg x_1 \vee \cdots \vee \neg x_n), n \geq 0, (x_1 = x_2)$	negative
IS^k_1	cf. next column	$(\neg x_1 \vee \cdots \vee \neg x_n), k \geq n \geq 0, (x_1 = x_2)$	negative of width k
IS_{00}	cf. next column	$(\neg x_1), (x_1 \to x_2), (x_1 \vee \cdots \vee x_n), n \geq 0$	IHS-B+
IS^k_{00}	cf. next column	$(\neg x_1), (x_1 \to x_2), (x_1 \vee \cdots \vee x_n), k \geq n \geq 0$	IHS-B+ of width k
IS_{02}	cf. next column	$(\neg x_1), (x_1 \vee \cdots \vee x_n), n \geq 0, (x_1 = x_2)$	essentially positive
IS^k_{02}	cf. next column	$(\neg x_1), (x_1 \vee \cdots \vee x_n), k \geq n \geq 0, (x_1 = x_2)$	essentially positive of width k
IS_{01}	cf. next column	$(x_1 \to x_2), (x_1 \vee \cdots \vee x_n), n \geq 0$	-
IS^k_{01}	cf. next column	$(x_1 \to x_2), (x_1 \vee \cdots \vee x_n), k \geq n \geq 0$	-
IS_0	cf. next column	$(x_1 \vee \cdots \vee x_n), n \geq 0, (x_1 = x_2)$	positive
IS^k_0	cf. next column	$(x_1 \vee \cdots \vee x_n), k \geq n \geq 0, (x_1 = x_2)$	positive of width k
IR_2	$x_1, \neg x_2$	$(x_1), (\neg x_1), (x_1 = x_2)$	-
IR_1	x_1	$(x_1), (x_1 = x_2)$	-
IR_0	$\neg x_1$	$(\neg x_1), (x_1 = x_2)$	-
IR (IBF)	$\emptyset$	$(x_1 = x_2)$	-

Table 2.2: Overview of bases [13] and clause descriptions [89] for co-clones, where EVEN^4 = $x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus 1$.

Example 2.6. *The scenario from Example 1.5 can be rewritten as below.*[3]

$$\begin{aligned} KB &= \{boosted \rightarrow \neg new\text{-}rules,\ (thesis \vee semester) \rightarrow busy,\\ &\quad\ boosted,\ \neg no\text{-}money,\ (busy \vee no\text{-}money \vee \neg boosted) \rightarrow no\text{-}travel\} \\ M &= \{no\text{-}travel\} \\ H &= \{thesis,\ new\text{-}rules,\ no\text{-}money\} \end{aligned}$$

Clearly, the set $E = \{thesis\}$ *is an explanation for* M*. Moreover,* $E_1 = \{thesis, no\text{-}money\}$ *is an explanation for* M *in* $KB_1 = KB \backslash \{\neg no\text{-}money\}$*. This highlights how abductive reasoning is a non-monotonic process since adding* $\{\neg no\text{-}money\}$ *to* KB_1 *results in invalidating* E_1 *as an explanation.* ◁

The computational problem in abductive reasoning that interests us is the following.

Problem:	$\mathrm{ABD}(\Gamma)$
Input:	(V, H, M, KB), where V is a set of variables, KB is a set of Γ-formulas and $H, M \subseteq V$.
Question:	Is there an explanation E for M in KB?

In addition to the problem defined above, we consider two size variants of $\mathrm{ABD}(\Gamma)$. The problem $\mathrm{ABD}_{\leq}(\Gamma)$ (resp., $\mathrm{ABD}_{=}(\Gamma)$) ask whether there is an explanation E of size atmost (exactly) s for some $s \in \mathbb{N}$. Accordingly, an instance to both these problems is then the tuple (V, H, M, KB, s).

Definition 2.7. *Let* Φ *be a set of formulas and* α *be a formula. The pair* (Φ, α) *is called an* argument *if*

1. Φ *is consistent,*
2. $\Phi \models \alpha$*, and*
3. Φ *is minimal with respect to set inclusion that satisfies (1) and (2).*

Let Δ *be a large repository of information from which one aims to construct arguments for a given claim. If* $\Phi \subseteq \Delta$*, then we say that* (Φ, α) *is an argument for* α *in* Δ*. The set* Φ *is called the* support *for the* claim α*.*

The knowledge base Δ for problems in argumentation is not always expected to be consistent. The underlying intuition is that from Δ one can construct arguments for and against arbitrary claims [9]. This is in contrast to the case of abductive reasoning. An instance of argumentation existence problem (ARG) is the pair (Δ, α), where Δ is a collection of Γ-formulas, and α is a Γ-formula.

[3] Clearly *no-money* is the same as $\neg$*money*. We used the rewording just to depict H and M as *positive* sets.

Example 2.8. *The scenario from Example 1.6 can be described as below.*

$$\begin{aligned}\Delta &= \{\mathit{money}, \mathit{visa}, \mathit{new\text{-}rules}, (\mathit{money} \wedge \mathit{visa}) \rightarrow \mathit{travel},\\ &\qquad \mathit{new\text{-}rules} \rightarrow \neg \mathit{processing}, \neg \mathit{processing} \rightarrow \neg \mathit{visa}\}\\ \alpha &= \{\mathit{travel}\}\end{aligned}$$

Clearly, $\Phi := \{\mathit{money}, \mathit{visa}, (\mathit{money} \wedge \mathit{visa}) \rightarrow \mathit{travel}\}$ *constitutes a support for* α *and therefore the pair* (Φ, α) *is an argument in* Δ. ◁

Argument existence (ARG) is the following decision problem.

Problem:	ARG(Γ)
Input:	(Δ, α) s.t. Δ is a set of Γ-formulas and α is a Γ-formula.
Question:	Is there a set $\Phi \subseteq \Delta$ such that (Φ, α) is an argument in Δ?

Two further interesting problems are argument verification (ARG-Check) and the relevance problem (ARG-Rel).

Problem:	ARG-Check(Γ)
Input:	(Φ, α) s.t. Φ is a set of Γ-formulas and α is a Γ-formula.
Question:	Is (Φ, α) an argument?

Problem:	ARG-Rel(Γ)
Input:	(Δ, α, ψ) s.t. Δ is a set of Γ-formulas, α,ψ are Γ-formulas.
Question:	Is there a set $\Phi \subseteq \Delta$ such that $\psi \in \Phi$ and (Φ, α) is an argument in Δ?

It is worth pointing out that the minimality condition (see Def. 2.7) plays no role in the complexity of ARG. This follows because there exists a minimal support if and only if there exists a support. Nevertheless, minimality indeed needs to be checked for ARG-Check because the support is given for this problem. Moreover, regarding ARG-Rel, one needs to assure that the formula ψ is indeed relevant. In other words, there exists a support Φ for α such that $\psi \in \Phi$ and $\Phi \setminus \psi$ is not a support for α.

The relevance and the verification problems for abduction have also been explored [22]. Furthermore, one can explore the size variants of problems for argumentation, but we do not cover them in this work.

2.3 Complexity Theory

We briefly describe our model of computation, a Turing machine (TM). For a detailed exposition, consult any textbook on the theory of computation (such as Sipser's Introduction to the theory of computation [106]). A single tape Turing machine is a tuple $\mathbb{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r, \sqcup)$ where,

- Q is a state set with three distinguished states $q_0, q_a, q_r \in Q$ an initial, an accepting and a rejecting state respectively,
- Σ is the input alphabet and $\Gamma \supseteq \Sigma$ is the tape alphabet, $\sqcup \in \Gamma \backslash \Sigma$ denotes a blank symbol,
- $\delta \colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.

During the computation of $\mathbb{M}$, the current state, the tape content, and the current head location changes. A particular setting of these three elements is called a *configuration*. That is, for two strings $s, t, \in \Gamma^\star$ and the state $q \in Q$ a configuration C represented as sqt highlights that the current content of the tape is st, the current state is q, and the head is scanning the cell containing the first symbol of t. Moreover, for two configurations C_1, C_2 we say that C_1 *yields* C_2 if $\mathbb{M}$ can go from C_1 to C_2 in a single step using the transition function δ. The starting configuration of $\mathbb{M}$ on an input string x is $q_0 x$ and a machine enters a halting state if it ever encounters a configuration with the state either q_a or q_r. $\mathbb{M}$ *accepts* an input x if there is a sequence of configurations $C_i, i \leq k$ such that $C_1 = q_0 x$, C_i yields C_{i+1} for each $i \leq k$ and C_k is an accepting configuration (a configuration with state q_a). On the other hand, if C_k is a rejecting configuration (a configuration with state q_r) then $\mathbb{M}$ halts by rejecting x. A *multitape Turing machine* is a Turing machine with several tapes such that each of these has its own head for reading and writing. A TM $\mathbb{M}$ is *deterministic* (DTM) if every configuration has at most one successor configuration, otherwise $\mathbb{M}$ is *non-deterministic* (NTM). For a NTM, the relation $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is called the transition relation.

Let $\mathbb{M}$ be a DTM that halts on all inputs and $f \colon \mathbb{N} \to \mathbb{N}$ be a function. We say that $\mathbb{M}$ runs in time $f(n)$ if the maximum number of steps taken by $\mathbb{M}$ on any input of length n is $f(n)$. It is convenient to analyze the approximated runtime by considering the asymptotic bounds on the running time. This gives rise to the so-called $\mathcal{O}$-notation. Let f and g be functions $f, g \colon \mathbb{N} \to \mathbb{R}^+$. Then $f(n) \in \mathcal{O}(g(n))$ if there exists positive integers c and n_0 s.t. $f(n) \leq c \cdot g(n)$ for every $n \geq n_0$.

We assume a reasonable encoding scheme for the inputs that can be computed in polynomial time. Moreover, we do not distinguish an input instance x from its encoding. Finally, for an element x, $|x|$ denotes its encoding length, or in other words the *size* of x. Let Σ be an alphabet and P be a decision problem over Σ, that is, a subset of Σ^*. We say that P is *trivial* if either $\mathrm{P} = \emptyset$ or $\mathrm{P} = \Sigma^*$, otherwise P is *nontrivial*. Moreover, we say that P is decidable in time $f(n)$ on a DTM, if there is a DTM M that halts on each input, decides every $x \in \mathrm{P}$ correctly and runs in time $f(n)$ where $|x| = n$. Two important complexity classes are **P** and **NP**, defined as classes of problems decidable in polynomial time on a DTM and NTM, respectively. In order to compare the complexity of two problems, the notion of reducibility is used.

Definition 2.9. *Let Σ and Δ be two alphabets with $A \subseteq \Sigma^*$ and $B \subseteq \Delta^*$ two problems. We say that A is* polynomial time reducible *to B, written $A \leq_m^{\mathbf{P}} B$, if there exists a polynomial time computable function f such that for every $x \in \Sigma^*$, we have $x \in A \iff f(x) \in B$.*

Two problems A and B are *polynomial time equivalent* if $A \leq_m^{\mathbf{P}} B$ and $B \leq_m^{\mathbf{P}} A$. A problem B is **NP**-*complete* if $B \in \mathbf{NP}$ and $A \leq_m^{\mathbf{P}} B$ for every $A \in \mathbf{NP}$. Another interesting class is the

$$
\begin{array}{lll}
\mathbf{\Delta}_1^P = \mathbf{P} & \mathbf{\Sigma}_1^P = \mathbf{NP} & \mathbf{\Pi}_1^P = \mathbf{CoNP} \\
\mathbf{\Delta}_2^P = \mathbf{P}^{\mathbf{\Sigma}_1^P} & \mathbf{\Sigma}_2^P = \mathbf{NP}^{\mathbf{\Sigma}_1^P} & \mathbf{\Pi}_2^P = \mathbf{CoNP}^{\mathbf{\Sigma}_1^P} \\
\vdots & \vdots & \vdots \\
\mathbf{\Delta}_i^P = \mathbf{P}^{\mathbf{\Sigma}_{i-1}^P} & \mathbf{\Sigma}_i^P = \mathbf{NP}^{\mathbf{\Sigma}_{i-1}^P} & \mathbf{\Pi}_i^P = \mathbf{CoNP}^{\mathbf{\Sigma}_{i-1}^P} \\
\\
\mathbf{PH} = \bigcup_i \mathbf{\Delta}_i^P & = \bigcup_i \mathbf{\Sigma}_i^P & = \bigcup_i \mathbf{\Pi}_i^P
\end{array}
$$

Figure 2.2: Complexity classes in the Polynomial hierarchy (**PH**)

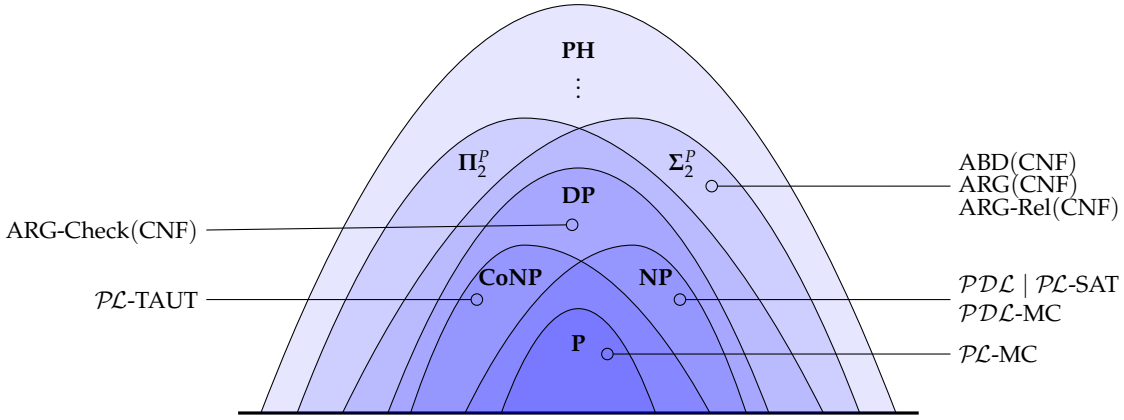


Figure 2.3: Complexity classes in the classical world (cf. Figure 2.4)

complementary class of **NP**, the class **CoNP**. A problem A is in **CoNP** if the complement problem A^c of A is in **NP**, where $A^c := \{x \mid x \notin A\}$. The complexity classes can be generalised using the notion of an oracle. An *oracle machine* is a Turing machine with an oracle which is able to solve certain problems in a single step. Let C be a complexity class, then $\mathbf{P}^C$ (resp., $\mathbf{NP}^C$) is the class of problems decidable by a DTM (NTM) in polynomial time with access to an oracle that can decide problems in C. Figure 2.2 defines the complexity classes in the *polynomial hierarchy* (**PH**). Moreover, Figure 2.3 depicts an overview of these classes with some example problems complete for them. Classes that interest us the most in this thesis are $\mathbf{P}, \mathbf{NP}, \mathbf{CoNP}, \mathbf{\Sigma}_2^P$ and the complexity class **DP**. **DP** is the class of problems C such that $C = A \cap B$ for some $A \in \mathbf{NP}$ and $B \in \mathbf{CoNP}$.

Parameterized Complexity Theory

To define parameterized problems, we borrow the notation from Downey and Fellows [32]. Let Σ be an alphabet. A *parameterized problem (PP)* Π is a subset of $\Sigma^* \times \mathbb{N}$. For $(x, k) \in \Pi$, x is called the input and k is the parameter value. A problem Π is *fixed-parameter tractable*, or **FPT** if there is a DTM deciding Π in time $f(k) \cdot p(x)$ for every instance (x, k) of Π, where f is some computable function and p is a polynomial. Whereas, problems decidable by an NTM with the same time constraints are in the complexity class **paraNP**. For a problem to be **FPT**, the polynomial degree

must be independent of the parameter. One can relax this condition and allow a running time of the form $|x|^{f(k)}$ for an instance (x,k). The problems decidable in the running time of this kind are in the class **XP**. It is known that **FPT** $\subseteq$ **W**[**P**] $\subseteq$ **XP** $\cap$ **paraNP**. The class **W**[**P**] contains problems decidable by an NTM running in at most $f(k) \cdot p(n)$ steps, such that at most $h(k) \cdot \log n$ of them are nondeterministic for some computable function h. In between **FPT** and **W**[**P**], there exists a (presumably) infinite hierarchy of complexity classes called the **W**-hierarchy. The notion of hardness in parameterized complexity is employed by **FPT**-*reductions*.

Definition 2.10. *Let Σ and Δ be two alphabets with $\Pi \subseteq \Sigma^* \times \mathbb{N}$ and $\Theta \subseteq \Delta^* \times \mathbb{N}$ two PPs. We say that Π* **FPT**-reduces *to Θ in symbols $\Pi \leq^{\mathbf{FPT}} \Theta$, if*

- *there is an* **FPT**-*computable function f s.t. for all $(x,k) \in \Sigma^* \times \mathbb{N}$: $(x,k) \in \Pi \Leftrightarrow f(x,k) \in \Theta$,*
- *there is a computable function g such that, for all $(x,k) \in \Sigma^* \times \mathbb{N}$ and $f(x,k) = (y,\ell)$: $\ell \leq g(k)$.*

Similar to the classical setting, we say that two problems Π and Θ are **FPT**-*equivalent* if $\Pi \leq^{\mathbf{FPT}} \Theta$ and $\Theta \leq^{\mathbf{FPT}} \Pi$. One can define the parameterized counter parts for each classical complexity class via the notion of *precomputation on the parameter*. We say that a complexity class $\mathcal{C}$ is *robust* if it satisfies the following two conditions (for all alphabets Σ and Δ).

1. For every problem $P \subseteq \Sigma^*$ with $P \in \mathcal{C}$ and every word $y \in \Delta^*$: $P \times \{y\} \in \mathcal{C}$,
2. For every $P \subseteq \Sigma^* \times \Delta^*$ with $P \in \mathcal{C}$ and every word $y \in \Delta^*$: $P_y = \{x \in \Sigma^* \mid (x,y) \in P\} \in \mathcal{C}$

The notion of robustness was introduced by Flum and Grohe [45] to relate classical complexity classes with their counterparts in parameterized world. The authors also state the fact that the complexity classes in the polynomial hierarchy are all robust.

Definition 2.11. *Let $\mathcal{C}$ be a robust (classical) complexity class. Then* **para**$\mathcal{C}$ *is the class of all PPs $\Pi \subseteq \Sigma^* \times \mathbb{N}$ such that there exists a computable function $\pi\colon \mathbb{N} \to \Delta^*$ and a problem $L \in \mathcal{C}$ with $L \subseteq \Sigma^* \times \Delta^*$ such that for all $(x,k) \in \Sigma^* \times \mathbb{N}$ we have that $(x,k) \in \Pi \Leftrightarrow (x,\pi(k)) \in L$.*

It is easy to observe that **paraP** $=$ **FPT** and the definition of **paraNP** coincides with the one defined before. Additionally, the classes **paraCoNP**, **paraDP** and **para**$\mathbf{\Sigma}_2^P$ interest us the most in this work. We refer the reader to Figure 2.4 for an overview of complexity classes in the parameterized world. A broader picture with other complexity classes can be found in the work of Elberfeld et al. [40]. In particular, it is believed that **XP** $\not\subseteq$ **para**$\mathbf{\Sigma}_i^P$ for any $i \in \mathbb{N}$ [45, Prop. 8].

The Notation We sometimes find it convenient to describe a parameter as a function $\kappa\colon \Sigma^* \to \mathbb{N}$ using the notation from Flum and Grohe [46]. In this setting, a PP is a pair (P, κ) where $\mathrm{P} \subseteq \Sigma^*$ and κ is a parameterization. There should be no ambiguity as to how a parameterized problem is defined. Indeed, an instance (x,k) of a PP $\Pi \subseteq \Sigma^* \times \mathbb{N}$ can be seen as an instance of (P, κ) where $\kappa(x) = k$. Our analysis concerns the parameterized complexity, and the considered problems are parameterized problems. We denote the parameterized version of a problem P by p-P. Moreover,

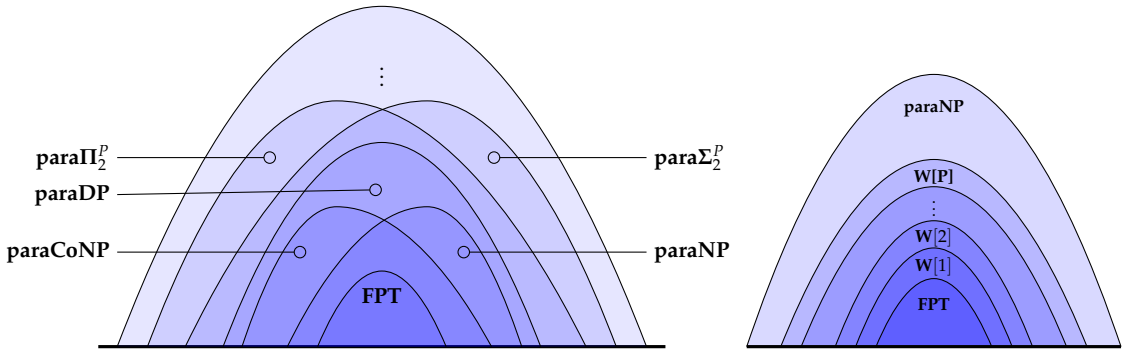


Figure 2.4: Classes in parameterized world (left) and additional classes below **paraNP** (right)

we find it convenient to mention the considered parameter alongside the problem. In other words, we write the parameterization κ inside the parenthesis, along with other restrictions imposed on an input instance. For example, p-ABD(Γ, κ) denotes the abduction problem parameterized by κ, and the knowledge base in the input instance is a Γ-formula.

We next define the levels of the **W**-hierarchy through a weighted version of the satisfiability problem. Let I be a nonempty finite index set and $t, d \in \mathbb{N}$. Consider the following special subclasses of formulas:

$$\begin{aligned}
\Gamma_{0,d} &= \{\, \ell_1 \wedge \ldots \wedge \ell_c \mid \ell_1, \ldots, \ell_c \text{ are literals and } c \leq d \,\}, \\
\Delta_{0,d} &= \{\, \ell_1 \vee \ldots \vee \ell_c \mid \ell_1, \ldots, \ell_c \text{ are literals and } c \leq d \,\}, \\
\Gamma_{t,d} &= \left\{ \bigwedge_{i \in I} \alpha_i \;\middle|\; \alpha_i \in \Delta_{t-1,d} \text{ for } i \in I \right\}, \\
\Delta_{t,d} &= \left\{ \bigvee_{i \in I} \alpha_i \;\middle|\; \alpha_i \in \Gamma_{t-1,d} \text{ for } i \in I \right\}.
\end{aligned}$$

Finally, denote by $\Gamma^+_{t,d}$ (resp. $\Gamma^-_{t,d}$)[4] the class of all positive (negative) formulas in $\Gamma_{t,d}$. The parameterized weighted satisfiability problem (p-WSAT) for $\Gamma_{t,d}$-formulas is defined as follows.

Problem:	p-WSAT$(\Gamma_{t,d}, \kappa)$
Input:	A $\Gamma_{t,d}$-formula ϕ with $t, d \geq 1$ and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Is there a satisfying assignment for ϕ of weight k?

The *weight* of an assignment s is the number of variables mapped to 1 by s. The third line (Parameter) in the problem definition is read as: k is the *parameter value*. We sometimes write $\kappa(x)$ instead of k to relate the function κ with its value k for a given instance x.

The following proposition characterizes the **W**-hierarchy.

[4] We also use Γ to denote a constraint language. There should be no ambiguity in using the letter Γ as a CL and as a $\Gamma_{t,d}$-formula defined here. When Γ is a CL, it is written without indices.

Proposition 2.12 ([46, Theorem 7.1]). *Each of the following problem is* $\mathbf{W}[t]$*-complete, under* $\leq^{\mathbf{FPT}}$*-reductions:*

- p-WSAT$(\Gamma^{+}_{t,1}, \kappa)$ *if* t *is even,*
- p-WSAT$(\Gamma^{-}_{t,1}, \kappa)$ *if* $t > 1$ *and odd,*
- p-WSAT$(\Gamma_{t,d}, \kappa)$ *for every* $t \geq 1$ *and* $d \geq 1$*,*

where the parameterization κ *is the weight of a satisfying assignment.*

Let $c \in \mathbb{N}$ and $\Pi \subseteq \Sigma^* \times \mathbb{N}$ be a PP, then the *c-slice of* Π, written as Π_c is defined as $\Pi_c := \{ (x,k) \in \Sigma^* \times \mathbb{N} \mid k = c \}$. To prove that a PP Π is **para**$\mathcal{C}$-hard for some complexity class $\mathcal{C}$, it is enough to prove that Π_c is $\mathcal{C}$-hard for some $c \in \mathbb{N}$. This is an easy consequence of the following proposition by Flum and Grohe [45].

Proposition 2.13 ([45, Proposition 14]). *Let* $\mathcal{C}$ *be a robust complexity class and let* P *be* $\mathcal{C}$*-complete under polynomial-time reductions. Then there is* $c \in \mathbb{N}$ *such that* $P \times \{c\}$ *is* **para**$\mathcal{C}$*-complete under* **FPT***-reductions.*

Finally, the following folklore result from the parameterized complexity theory helps achieve upper bounds in conjunction with the relationship between parameters.

Proposition 2.14. *Let* Q *be a problem such that* (Q, κ) *is* **FPT** *and let* γ *be another parameterization such that* $\kappa(x) \leq f(\gamma(x))$ *for some computable function* f *and every* x*, then* (Q, γ) *is also* **FPT***.*

We define considered parameterizations at the beginning of each chapter. In the following, we define a very generic parameter, namely the *treewidth*. The notion of treewidth is due to Robertson and Seymour [100] and has proven itself to be an important structural parameter (further see [18, 37, 51, 16, 17]). It is defined on top of a graph structure $G = (V, E)$.

Definition 2.15 (Treewidth). *Let* $\mathcal{G} := (V, E)$ *be an undirected graph. The* tree decomposition *of* $\mathcal{G}$ *is a tree* $T = (B, E_T)$*, where* $B \subseteq \mathcal{P}(V)$ *(called a collection of* bags*) and* E_T *is the edge relation such that the following is true.*

- $\bigcup_{b \in B} b = V$*,*
- *for every* $\{ u, v \} \in E$ *there is a bag* $b \in B$ *with* $u, v \in b$*, and*
- *for all* $v \in V$ *the restriction of* T *to* v *(the subset with all bags containing* v*) is connected.*

The width *of a tree decomposition* $T = (B, E_T)$ *is the size of the largest bag minus one:* $\max_{b \in B} |b| - 1$*. The* treewidth *of a given graph* G *is the minimum over all widths of tree decompositions of* G*.*

Observe that if G is a tree, then the treewidth of G is one. Intuitively, the treewidth measures how *tree-like* a given graph is. The decision problem to determine whether the treewidth of a graph $\mathcal{G}$ is at most k, is **NP**-complete [2]. See Bodlaender's Guide [12] for an overview of

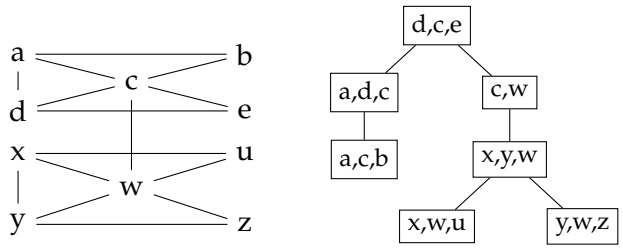


Figure 2.5: A graph (Left) with the a possible tree decomposition (Right).

algorithms that compute tree decompositions. Notice that an instance of a PP is the pair (x, k) where x is the input and k is the parameter value. In other words, the parameter value is given together with an input instance, therefore one does not have to worry about computing it from the given input.

Example 2.16. *Figure 2.5 presents a graph $\mathcal{G}$ on the left and its tree decomposition on the right. Since the largest bag has size three, the treewidth of this decomposition is two.* ◁

The problems discussed in Example 1.2 are also relevant in the reductions presented during the course of this work. For this reason, we present a formal definition for each of these.

Problem:	VertexCover
Input:	$\mathcal{G} = (V, E)$, and a number $k \in \mathbb{N}$.
Question:	Is there a vertex cover for $\mathcal{G}$ of size k?

Problem:	IndependentSet
Input:	$\mathcal{G} = (V, E)$, and a number $k \in \mathbb{N}$.
Question:	Is there an independent set for $\mathcal{G}$ of size k?

Problem:	k-Coloring
Input:	$\mathcal{G} = (V, E)$, and a number $k \in \mathbb{N}$.
Question:	Can $\mathcal{G}$ be *properly* colored using k-colors?

Classically, each of the above three problems is known to be **NP**-complete (consult any textbook on the complexity theory, such as [106]). When parameterized by k, the problem p-VertexCover is **FPT** [46, Cor. 1.19], p-IndependentSet is $\mathbf{W}[1]$-complete [46, Thm.6.1], and k-Coloring is **paraNP**-complete [46, Cor. 1.12].

2.4 Galois Connection

Now we discuss how Post's lattice helps in considering fewer types of CLs when classifying the complexity of a problem in Schaefer's framework. Recall that the knowledge base in the context

of abduction and argumentation consists of Γ-formulas for a CL Γ. The Galois connection in the general setting for a problem P is the following important property. Let Γ and Γ′ be two CLs,

$$\mathrm{P}(\Gamma') \leq_m^{\mathbf{P}} \mathrm{P}(\Gamma), \quad \text{if } \Gamma' \subseteq \langle\Gamma\rangle. \tag{2.1}$$

It states that when studying the complexity of a problem P for finite constraint languages Γ, it is enough to consider only one generating constraint language per each co-clone. In other words, the complexity of P does not change for constraint languages which generate the same co-clone. Property 2.1 might fail for some problems as discussed during the course of this thesis. Lemma 2.17 highlights the Galois connection for the satisfiability problem.

Lemma 2.17 (Galois Connection)**.** *Let* Γ *and* Γ′ *be two finite CLs. If* $\Gamma' \subseteq \langle\Gamma\rangle$ *then* $\mathrm{SAT}(\Gamma') \leq_m^{\mathbf{P}} \mathrm{SAT}(\Gamma)$.

Proof. The idea is to translate the given Γ′-formula ψ to an equivalent Γ-formula ϕ through the following steps.

- Replace every constraint in ψ by its equivalent $\Gamma \cup \{=\}$-formula.
- Remove existential quantifiers.
- Remove equality clauses and replace all variables connected via a chain of these equality clauses with a common fresh variable.

It is easy to observe that $\psi \equiv \phi$. Moreover, the translation can be achieved in polynomial time (in log-space to be precise). □

The following propositions exhibit the known results regarding the Galois connection for problems in abduction and argumentation.

Proposition 2.18 ([89, Lemma 22])**.** *Let* Γ *and* Γ′ *be two finite constraint languages. If* $\Gamma' \subseteq \langle\Gamma\rangle$ *then* $\mathrm{ABD}(\Gamma') \leq_m^{\mathbf{P}} \mathrm{ABD}(\Gamma)$.

Proposition 2.19 ([22, Theorems 5.3 & 6.1])**.** *Let* Γ *and* Γ′ *be two finite CLs. If* $\Gamma' \subseteq \langle\Gamma\rangle$ *then* $\mathrm{ARG}(\Gamma') \leq_m^{\mathbf{P}} \mathrm{ARG}(\Gamma)$ *and* $\text{ARG-Check}(\Gamma') \leq_m^{\mathbf{P}} \text{ARG-Check}(\Gamma)$.

Regarding $\mathrm{ABD}_{\leq}$ or $\mathrm{ABD}_{=}$, there are no known results. Nevertheless, we prove during the course of this thesis that the Galois connection is indeed true for these two problems as well. Regarding ARG-Rel, the Galois connection fails [22, Thm. 7.1]. In other words, there are two CLs Γ and Γ′ such that $\langle\Gamma\rangle = \langle\Gamma'\rangle$ but $\text{ARG-Rel}(\Gamma')$ is in **P** whereas $\text{ARG-Rel}(\Gamma)$ is **NP**-complete. We state and prove the Galois connection (when it holds) for each considered problem in the parameterized setting under **FPT**-reductions.

CHAPTER 3

PROPOSITIONAL LOGIC OF DEPENDENCE

In this chapter we discuss the parameterized complexity of model checking (MC) and satisfiability (SAT) for $\mathcal{PDL}$. We begin by defining each parameterization and various relationships between them. The last section of this chapter is devoted to defining some related problems. We introduce a satisfiability variant (mSAT) and characterize its complexity in classical, as well as the parameterized setting. Moreover, we introduce another problem (MaxSubTeam) that might be relevant from the database repairs' pesepective. Table 3.1 summarises the main results of this chapter.

3.1 A Note on Parameterizations

Formulas in this chapter are $\mathcal{PDL}$-formulas unless stated otherwise. Moreover, we drop the logic prefix ($\mathcal{PDL}$-) and write MC and SAT for model checking and satisfiability for $\mathcal{PDL}$. Given a team T and a formula ϕ, then it is assumed that the domain $dom(T)$ of T is $\mathrm{Vars}(\phi)$ (due to the locality principle). We consider ten different parameters for each of the two problems MC and SAT. Our parameterizations include formula-tw, formula-team-tw, $|T|$, $|\phi|$, $|\mathsf{var}(\phi)|$, formula-depth, #splits, #conjunctions, #atoms and dep-arity. These parameters arise naturally in an instance for each problem under consideration. Moreover, some of these parameters render the problem trivial (such as $|\phi|$ for SAT), and the soul purpose of including them is to complete the picture with respect to parameterized complexity. Let T be a team and ϕ a $\mathcal{PDL}$-formula. An instance of p-MC(κ) (resp., p-SAT(κ)) is the tuple (T, ϕ, k) $((\phi, k))$ where $\kappa(T, \phi) = k$ is the parameter value. Let I be an instance of either p-MC(κ) or p-SAT(κ), then each parameterization (κ) is defined in the following discussion.

1. $\#\mathsf{splits}(I)$ is the number of times a split-junction ($\vee$) appears in ϕ, $\#\mathsf{conjunctions}(I)$ is the number of occurrences of conjunction, and $|\mathsf{var}(\phi)|$ is the number of distinct propositional variables in ϕ.

2. The arity of a dependence atom $\mathsf{dep}(P;Q)$ is the size of P and $\mathsf{dep\text{-}arity}(I)$ is the maximum arity of any dependence atom in ϕ. Moreover, $\#\mathsf{atoms}(I)$ is the number of occurrences of dependence atoms in ϕ.

3. $|T|$ is the team-size (the number of assignments in T) and $|\phi|$ is the size (the encoding length) of ϕ. Both these parameters can be considered as a function f such that $f(I)$ returns the appropriate value for an instance I.

Remark 3.1. *It is worth pointing out that if a parameterization κ is seen as a function, then the expression $\kappa(I)$ only makes sense when the parameter value can be actually computed from the input. For example, an instance I of* MC *is the tuple (T,ϕ) and $|T|$ is the number of assignments in T. Whereas, an instance I of* SAT *is simply a $\mathcal{PDL}$-formula, and one might argue that the parameter $|T|$ does not make sense. One can follow the remark by Flum and Grohe ([46]) and "always make the parameter an explicit part of the input". Thereby, the parameterization is a constant function in such cases. In other words, $|T| = c$ and an input instance is the tuple (ϕ, c). Nevertheless, the reader can choose himself the meaning of $\kappa(I)$ in such cases*[1].

To consider further structural parameters, we associate a graph structure (a representation) with an input instance I. We achieve this goal in the following subsection.

Representation of Inputs as Graphs

The classical $\mathcal{PL}$-formulas are represented via various kinds of graphs (such as, the Gaifman graph or the primal graph) [101]. However, this setting typically considers CNF-formulas, and the graph representation depicts the relationship between literals and clauses of a formula. A generalization of this approach to consider arbitrary formulas was presented by Lück et al. [78], where the authors defined 'syntax circuits' for temporal logic formulas. We adhere to the same idea and define the syntax (or formula) structure for a $\mathcal{PDL}$-formula.

It is also worth pointing out an important observation regarding the graph representation for the $\mathcal{PDL}$-formulas due to Grädel [49]. In the usual setting, for logics with team semantics, we take the syntax tree and not the associated syntax structure and distinguish between different occurrences of the same subformula. The reason for this choice is illustrated in Remark 2.2. That is, $\phi \vee \phi$ is not equivalent to ϕ since different teams are entitled to the two occurrences of ϕ in their evaluation. Consequently, the well-formed formulas of $\mathcal{PDL}$ are seen as binary trees with leaves as atomic subformulas (variables and dependence atoms). This yields another interesting parameterization defined below.

4. $\mathsf{formula\text{-}depth}(I)$ is the depth of the syntax tree of the formula ϕ. That is, the length of the longest path from the root to any leaf in the syntax tree.

If a $\mathcal{PDL}$-formula is seen as a tree, as discussed above, the parameter treewidth (Def. 2.15) is not meaningful anymore. For this reason, we consider the syntax structure rather than the syntax

[1] Clearly, if the parameter value does not correspond to an input instance then it does not tell anything interesting about the tractability. We follow this intuition and only consider a parameterization if it 'makes sense'.

tree as a graph structure to consider treewidth as a parameter. Moreover, in the case of MC, one might include assignments in a graph representation. In the latter case, one considers the Gaifman graph of the structure that models the team and the input formula.

Definition 3.2 (Syntax structure)**.** *Let (T, ϕ) be an instance of* MC*, where ϕ is a $\mathcal{PDL}$-formula with propositional variables $X \subseteq VAR$ and $T = \{ s_1, \dots s_m \}$ is a team of assignments $s_i \colon X \to \{0, 1\}$. The* syntax structure $\mathcal{A}_{T,\phi}$ *over the vocabulary*

$$\tau_{T,\phi} := \{ \mathsf{VAR}^1, \mathsf{SF}^1, \succcurlyeq^2, \mathsf{DEP}^2, \mathsf{inTeam}^1, \mathsf{isTrue}^2, \mathsf{isFalse}^2, r, c_1, \dots, c_m \},$$

where superscripts denote the arity of each relation, then is defined as follows.

The universe of $\mathcal{A}_{T,\phi}$ is $A := SF(\phi) \cup \mathrm{Vars}(\phi) \cup T$, where $SF(\phi)$ and $\mathrm{Vars}(\phi)$ denote the sets of subformulas and variables appearing in ϕ, respectively. Clearly, $SF(\phi) \cap \mathrm{Vars}(\phi) \neq \emptyset$ since some variables can also appear as subformulas of ϕ. However, $\mathrm{Vars}(\phi) \not\subseteq SF(\phi)$ as not every variable in $\mathrm{Vars}(\phi)$ appears as a subformula of ϕ (such as variables occurring only in dependence atoms).

- SF *and* VAR *are unary relations representing 'is a subformula of ϕ' and 'is a variable in ϕ' respectively.*
- $\succcurlyeq$ *is a binary relation such that $\theta \succcurlyeq^{\mathcal{A}} \psi$ iff ψ is an immediate subformula of θ and r is a constant symbol representing ϕ.*
- DEP *is a binary relation which connects each dependence atom with the used variables.*
- *The set $\{ c_1, \dots, c_m \}$ encodes the team T, where $c_i^{\mathcal{A}} := s_i$ for each $i \leq m$. That is, each c_i corresponds to an assignment $s_i \in T$ for $i \leq m$. Moreover, $\mathsf{inTeam}(c)$ is true if and only if $c^{\mathcal{A}} \in T$.*
- isTrue *and* $\mathsf{isFalse}$ *relate variables with the team elements.* $\mathsf{isTrue}(c, x)$ *(resp.,* $\mathsf{isFalse}(c, x)$*) is true if and only if x is mapped* 1 *(resp.,* 0*) by the assignment interpreted by c.*

Analogously, the syntax structure *$\mathcal{A}_\phi$ over a respective vocabulary τ_ϕ is defined. In this case, the team-related relations are not present and the universe does not contain constants $c_i^{\mathcal{A}}$ for $1 \leq i \leq m$.*

Definition 3.3 (Gaifman graph)**.** *Let T be a team and ϕ be a $\mathcal{PDL}$-formula, the* Gaifman graph *$\mathcal{G}_{T,\phi} = (A, E)$ of the $\tau_{T,\phi}$-structure $\mathcal{A}_{T,\phi}$ is defined as*

$$E := \left\{ \{u, v\} \;\middle|\; u, v \in A, \text{ such that there is an } R \in \tau_{T,\phi} \text{ with } (u, v) \in R \right\}.$$

Analogously, we let $\mathcal{G}_\phi$ to be the Gaifman graph *for the τ_ϕ-structure $\mathcal{A}_\phi$.*

Note that $E := \mathsf{DEP} \cup \succcurlyeq$ for $\mathcal{G}_\phi$, and $E := \mathsf{DEP} \cup \succcurlyeq \cup \mathsf{isTrue} \cup \mathsf{isfalse}$ for $\mathcal{G}_{T,\phi}$.

Example 3.4. *Let $\phi := (x_3 \vee \neg x_1) \wedge \big(\mathsf{dep}(x_3; x_4) \vee (x_1 \wedge x_2)\big)$ be a $\mathcal{PDL}$-formula. Figure 3.1 represents the Gaifman graph of the syntax structure $\mathcal{A}_\phi$ (in the middle) with a tree decomposition (on the right). Since the largest bag has a size of three, the treewidth of the given decomposition is two. Figure 3.2 presents the Gaifman graph of the syntax structure $\mathcal{A}_{T,\phi}$, that is, when the team $T = \{s_1, s_2\} = \{0011, 1110\}$ is also part of the input.* ◁

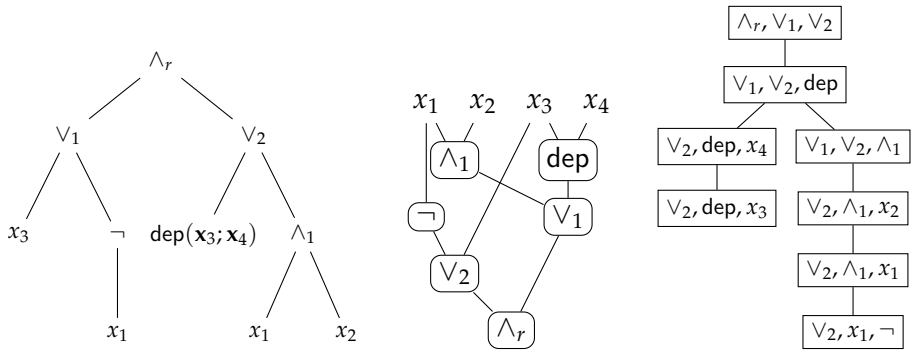


Figure 3.1: The syntax tree (left) with the corresponding Gaifman graph of the syntax structure (middle) and a tree decomposition (right) for $\phi := (x_3 \vee \neg x_1) \wedge \big(\mathsf{dep}(x_3; x_4) \vee (x_1 \wedge x_2)\big)$. For a better presentation, we abbreviated subformulas in the inner vertices of the Gaifman graph.

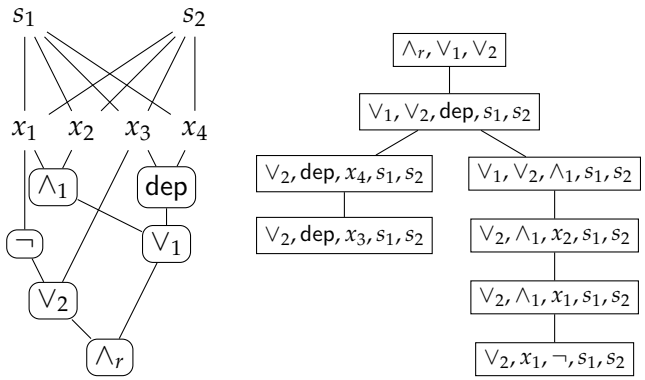


Figure 3.2: The Gaifman graph for (T, ϕ) (from Example 3.4) with a possible tree decomposition.

For MC, including the assignment-variable relation in the graph representation yields two treewidth notions, namely formula-tw and formula-team-tw. The name emphasizes whether the team is also part of the graph representation or not. Turning back to our list of parameters, we append the following two items.

5. formula-tw(I) is the treewidth of $\mathcal{G}_\phi$.

6. formula-team-tw(I) is the treewidth of $\mathcal{G}_{T,\phi}$.

Clearly, formula-team-tw is only relevant for MC. Moreover, we will shortly prove that the treewidth increases when a team is also included in the graph representation (Lemma 3.6).

The following lemma proves relationships between several parameters considered in our analysis. This is further visualized in Figure 3.3. As before, the notation $\kappa(T, \phi)$ stands for the parameter value of the input instance (T, ϕ).

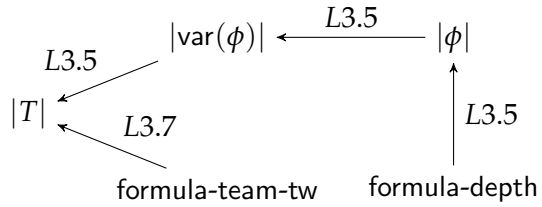


Figure 3.3: The relationship among different parameters. The direction of arrow in $p \leftarrow q$ implies that bounding q results in bounding p. *Li* means the proof can be found in Lemma i.

Lemma 3.5. *Let $I = (T, \phi)$ where T is a team and ϕ is a $\mathcal{PDL}$-formula, then*

1. $|T| \leq 2^{|\mathsf{var}(\phi)|}$,
2. $|T| \leq 2^{f(|\phi|)}$ *for some function* f,
3. $|\phi| \leq 2^{2 \cdot \mathsf{formula\text{-}depth}(I)}$.

Proof. If a $\mathcal{PDL}$-formula ϕ has m variables, then there are 2^m many assignments (due to the locality) and the maximum size for a team is 2^m. As a result, we have $|T| \leq 2^{|\mathsf{var}(\phi)|}$. Furthermore, the number of variables in a $\mathcal{PDL}$-formula ϕ is bounded from above by $|\phi|$. Recall that we assume an encoding scheme which is computable in polynomial time. This implies that $|\mathsf{var}(\phi)|$ is bounded by the encoding length of $\mathrm{Vars}(\phi)$, which in turn is bounded by $|\phi|$. As a result, we have $|\mathsf{var}(\phi)| \leq f(|\phi|)$ for some function f (which depends on the encoding scheme). This implies that $2^{|\mathsf{var}(\phi)|} \leq 2^{f(|\phi|)}$ and proves the second claim.

If $\mathsf{formula\text{-}depth}(I) = d$ for an input instance I, then there are $\leq 2^d$ leaves in the (binary) syntax tree of ϕ and $\leq 2^d$ internal nodes. Then $|\phi| \leq 2^{2d}$ is true. □

Now we prove the following non-trivial lemma stating that treewidth of the structure $\mathcal{A}_{T,\phi}$ bounds either the team size or the number of variables in ϕ. This implies that bounding the treewidth of the structure also bounds one of the two parameters. Recall that for formula-team-tw, we consider the treewidth of the Gaifman graph $\mathcal{G}_{T,\phi}$ underlying the structure $\mathcal{A}_{T,\phi}$ that encodes an instance of MC.

Lemma 3.6. *Let $I = (T, \phi)$ be an instance of* MC. *Then the following relationship between the parameters is true,*

$$\mathsf{formula\text{-}team\text{-}tw}(I) \geq \min\{\, |T|, |\mathsf{var}(\phi)| \,\}$$

Proof. We prove that if there is a tree decomposition of $\mathcal{G}_{T,\phi}$ with the treewidth smaller than the two values, then such a decomposition must have cycles and hence cannot be a valid tree decomposition. The proof uses the fact that in the Gaifman graph $\mathcal{G}_{T,\phi}$, every team element is related to each variable. As a consequence, in any tree decomposition, the assignment-variable relations 'isTrue' and 'isFalse' force some bag to have their size larger than either the team size or the number of variables (based on which of the two values is smaller). Without loss of generality, assume that $|T| \geq 2$, if $|T| < 2$ there is nothing to prove since $\mathsf{formula\text{-}team\text{-}tw}(I) \geq 1$ trivially.

This is due to the reason that the treewidth of a tree is one and a structure has treewidth zero only if there are no relation symbols (see Def. 2.15). We consider individual bags corresponding to an edge in the Gaifman graph due to the relations from $\tau_{T,\phi}$.

Let $\mathrm{Vars}(\phi) = \{ x_1, \ldots, x_n \}$ and consider a minimal tree decomposition $(\mathcal{B}, E_T)$ for $\mathcal{G}_{T,\phi}$. Denote by $B(x_i, c_j)$ a bag[2] that covers the edge between a variable x_i and an assignment-element c_j, that is, either $\mathsf{isTrue}(x_i, c_j)$ or $\mathsf{isFalse}(x_i, c_j)$ is true. Moreover, denote by $B(x_i, \alpha)$ the bag covering the edge between a variable x_i and its immediate $\succcurlyeq$-predecessor α. Recall (Def. 3.2), there is a path from each variable x_i to the formula ϕ due to $\succcurlyeq$. This implies the existence of a path between each pair of variables in the Gaifman graph, which also passes through some subformula ψ of ϕ. Let $B(x, \alpha_1), B(\alpha_1, \alpha_2), \ldots, B(\alpha_q, \psi), B(\psi, \beta_r), \ldots, B(\beta_2, \beta_1), B(\beta_1, y)$ be the sequence of bags that cover $\succcurlyeq^{\mathcal{A}}$-edges between x and y (where $q, r \leq |\mathrm{SF}(\Phi)|$). Without loss of generality, we assume that all these bags are distinct. Now, for any pair x, y of variables, the bags $B(x, c_i)$ and $B(y, c_i)$ contain c_i for each $i \leq m$ and as a consequence, we have either of the following two cases.

Case 1. The two bags are the same, that is $B(x, c_i) = B(y, c_i)$ and as a consequence, we have $|B(x, c_i)| \geq 3$ because $B(x, c_i)$ contains at least x, y and c_i. Moreover, if this is true (otherwise case two applies) for each pair of variables, then there is a single bag, say $B(c_i)$, that contains all variables and the element c_i. This means the maximum bag size must be larger than the total number of variables, a contradiction.

Case 2. Every bag in the E_T-path between $B(x, c_i)$ and $B(y, c_i)$ contains c_i. We know that if a $B(x, \alpha_1)$-$B(\beta_1, y)$-path between x and y due to relations in $\{\succcurlyeq\} \cup \{\mathsf{DEP}\}$ exist, then the bags $B(x, c_i)$ and $B(y, c_i)$ cannot be E_T-adjacent (because this will produce a cycle) unless the whole $B(x, \alpha_1)$-$B(\beta_1, y)$-path collapses to these two bags. Now, since $|T| \geq 2$, consider two different elements c_i, c_j, and the bags $B(y, c_i)$ and $B(y, c_j)$. If these two bags are E_T-adjacent then $B(x, c_i)$ and $B(y, c_i)$ cannot be E_T-adjacent and the path between $B(x, c_i)$ and $B(y, c_i)$ must contain c_i. Notice that both $B(y, c_i), B(y, c_j)$ and $B(x, c_i), B(y, c_i)$ cannot be adjacent since this would create a cycle. Consequently, the two possible cases are (see Figure 3.4 explaining this situation): (1) $B(y, c_i)$ and $B(y, c_j)$ are not E_T-adjacent and every path between these bags contains y, or (2) $B(x, c_i)$ and $B(y, c_i)$ are not E_T-adjacent and every path between these bags contains c_i. Finally, since this is true for all variables and elements c_i with $i \leq m$, this proves that either there is a bag that contains all variables, or there is one that contains all c_i's. The remaining case that there are cycles in the tree decomposition is not applicable.

This proves the claim and completes the proof to the lemma. □

The following corollary is immediate due to the previous lemma.

Corollary 3.7. *Let $I = (T, \phi)$ where T is a team and ϕ is a $\mathcal{PDL}$-formula. Then there is a fuction f such that $|T| \leq f(\mathsf{formula\text{-}team\text{-}tw}(I))$.*

[2] Such a bag always exists by definition. There can be many bags covering an edge, we simply choose one of them.

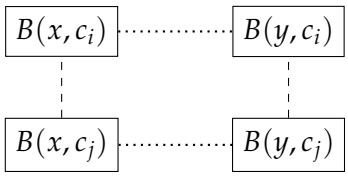


Figure 3.4: The rectangles represent bags corresponding to the variable-assignment relation. If the two c_i-bags do not contain c_j-nodes, then there can be only either dotted (horizontal) or dashed (vertical) edges between the bags to avoid cycles.

Parameter (κ)	p-MC(κ)	p-SAT(κ)
formula-tw	**paraNP**[3.9]	**FPT**[3.20]
dep-arity	**paraNP**[3.10]	**paraNP**[3.19]
#splits	**paraNP**[3.11]	**FPT**[3.21]
#conjunctions	**paraNP**[3.12]	
#atoms	**paraNP**[3.13]	**paraNP**[3.19]
$\lvert T\rvert$	**FPT**[3.14]	$\star$
$\lvert\phi\rvert$	**FPT**[3.15]	**FPT**[3.23]
$\lvert\mathsf{var}(\phi)\rvert$	**FPT**[3.15]	**FPT**[3.23]
formula-depth	**FPT**[3.15]	**FPT**[3.23]
formula-team-tw	**FPT**[3.16]	**FPT**[3.20]

Table 3.1: Complexity classification overview with pointers to the proof details. All **paraNP**-results are completeness results. For $\star$: see Remark 3.1

Proof. It follows from Lemma 3.6 that $\mathsf{formula\text{-}team\text{-}tw}(I) \geq \min\{|T|, |\mathsf{var}(\phi)|\}$. In the first case, $\mathsf{formula\text{-}team\text{-}tw}(I) \geq |T|$ and there is nothing to prove. Otherwise, $|\mathsf{var}(\phi)| \leq \mathsf{formula\text{-}team\text{-}tw}(I)$ and the claim follows since $|T| \leq 2^{|\mathsf{var}(\phi)|} \leq 2^{\mathsf{formula\text{-}team\text{-}tw}(I)}$.

□

3.2 Model Checking

In this section, we classify the parameterized complexity of model checking (p-MC(κ)) under various parameterizations (κ). Table 3.1 contains a complete list of our results.

Problem:	p-MC(κ)
Input:	A $\mathcal{PDL}$-formula ϕ, a team T over $\mathrm{Vars}(\phi)$, and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does $T \models \phi$?

Classically, MC is **NP**-complete [39, Thm. 3.2]. This also implies that p-MC(κ) is in **paraNP** under any parameterization κ.

Proposition 3.8. p-MC(κ) *is in* **paraNP** *for any parameterization* κ*.*

We first present those parameters κ that render the complexity of p-MC(κ) as **paraNP**-complete. The upper bound in each case follows from Proposition 3.8 and we only present **paraNP**-hardness for individual cases. Our first result is regarding the parameter formula-tw.

Theorem 3.9. p-MC(formula-tw) *is* **paraNP**-*complete.*

Proof. For **paraNP**-hardness, we prove that the 1-slice of the problem is **NP**-hard by reducing from 3SAT. The reduction presented by Ebbing and Lohmann ([39, Thm. 1]) uses Kripke semantics (as they aim for results for modal logic). We slightly modify the reduction to fit our presentation and the correctness proof is the same. Let $\psi := C_1 \wedge \ldots \wedge C_m$ be an instance of 3SAT over $\{x_1, \ldots, x_n\}$ where each C_i is a clause, that is, a set of literals. We define an instance (T, ϕ) of $\mathcal{PDL}$-MC such that $\mathrm{Vars}(\phi) = \{p_1, \ldots, p_n, r_1 \ldots, r_n\}$. The team $T = \{s_1, \ldots, s_m\}$ contains m assignments, where each $s_i \colon \mathrm{Vars}(\phi) \to \{0, 1\}$ is defined as follows,

$$\begin{aligned} &s_i(p_j) = s_i(r_j) = 1, \quad \text{if } x_j \in C_i, \\ &s_i(p_j) = 0, s_i(r_j) = 1, \text{ if } \neg x_j \in C_i, \\ &s_i(p_j) = s_i(r_j) = 0, \quad \text{if } x_j, \neg x_j \notin C_i. \end{aligned}$$

In other words, there is an assignment s_i per clause. The value $s_i(r_j)$ encodes whether or not x_j appears in the clause C_i, whereas, $s_i(p_j)$ encodes whether x_j appears positively or negatively in C_i. Finally, let $\phi := \bigvee_{j=1}^{n} \big(r_j \wedge \mathsf{dep}(; p_j)\big)$. The proof of $\psi \in$ SAT iff $T \models \phi$ is similar to the one presented by Ebbing and Lohmann [39, Thm. 1].

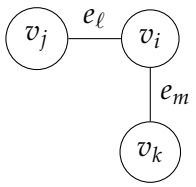
"$\Rightarrow$". Let θ be a satisfying assignment for ψ. We construct T_j for each $j \leq n$ such that,

$$T_j := \begin{cases} \{s_i \mid s_i(p_j) = 1 = s_i(r_j)\} & \text{if } \theta(x_j) = 1, \\ \{s_i \mid s_i(p_j) = 0, s_i(r_j) = 1\} & \text{if } \theta(x_j) = 0. \end{cases}$$

That is, T_j contains an assignment s_i if and only if the clauses C_i is satisfied by $\theta(x_j)$ where $i \leq m$. Clearly, $T_j \models r_j \wedge \mathsf{dep}(; p_j)$. Moreover, since every clause is satisfied, this implies that $\bigcup_{j \leq n} T_j = T$ and consequently $T \models \phi$.

"$\Leftarrow$". Suppose that $T \models \phi$, then there are $T_1, \ldots, T_n$ such that $T = \bigcup_{j \leq n} T_j$ and $T_j \models r_j \wedge \mathsf{dep}(; p_j)$. Clearly the value of p_j is fixed by T_j for each $j \leq n$. We construct a satisfying assignment for ψ by considering each variable separately. For $j \leq n$, let $V_j = \{i \mid s_i \in T_j\}$. Now, $s_i(p_j) = 1$ implies $x_j \in C_i$, and we set $\theta(x_j) = 1$, otherwise set $\theta(x_j) = 0$. This implies that we set $\theta(x_j) = 0$ if $\neg x_j \in C_i$. Since for every $s_i \in T$ there is a $j \leq n$ with $s_i \in T_j$, we have an evaluation that satisfies every clause $C_i \in \psi$ and, as a consequence, $\theta \models \psi$.

We conclude by observing that there is no interleaving of variables in ϕ and the value of the parameter is fixed in advance. This is because G_ϕ is a tree; as a consequence, $\text{formula-tw}(\phi) = 1$. This completes the proof. □



	x_i	x_j	x_k	$y_{\ell,i}$	$y_{\ell,j}$	$y_{\ell,k}$	$y_{m,i}$	$y_{m,j}$	$y_{m,k}$
s_i	0	1	1	1	1	1	1	1	1
s_j	1	0	0	1	1	1	1	0	1
s_k	1	0	0	1	1	0	1	1	1

Figure 3.5: A graph $\mathcal{G} : (\{ v_i, v_j, v_k \}, \{ e_l, e_m \})$ with its corresponding team.

The following corollary follows from the proof of Theorem 3.9 since each dependence atom in ϕ has arity zero.

Corollary 3.10. p-MC(dep-arity) *is* **paraNP**-*complete.*

It turns out that a split-junction is also a major source of the hardness in the model checking problem for $\mathcal{PDL}$. Furthermore, SAT and MC for the fragment of $\mathcal{PDL}$ without splits are in **P** [87]. The following theorem establishes the **paraNP**-completeness of p-MC(#splits).

Theorem 3.11. p-MC(#splits) *is* **paraNP**-*complete.*

Proof. We achieve **paraNP**-hardness by a reduction from the 3-colouring problem (3COL) and applying Proposition 2.13. An instance of 3COL constitutes a graph $\mathcal{G} = (V, E)$ and the problem is to determine whether $\mathcal{G}$ can be colored using three colors such that no two endpoints of an edge are colored same. We map $\mathcal{G}$ to an instance (T, ϕ) where T is a team, and ϕ is a $\mathcal{PDL}$-formula with two split-junctions. The reduction implies that the 2-slice of p-MC(#splits) is **NP**-hard. The idea of the reduction from 3COL is to construct a team (as shown in Figure 3.5) in combination with the formula containing two split-junctions, where each disjunct is ψ for $\psi := \bigwedge_{e_k = \{ v_i, v_j \}} \mathsf{dep}(\mathbf{y}_k; x_i)$. Intuitively, each vertex of the graph corresponds to an assignment in the team, and the subteams for the three disjuncts are then mapped to three colors.

Let $V = \{ v_1, \ldots, v_n \}$ be the vertex set and $E = \{ e_1, \ldots, e_m \}$ be the set of edges of $\mathcal{G}$. We let

$$\mathrm{Vars}(\phi) := \{ x_1, \ldots, x_n \} \cup \{ y_{1,1}, \ldots, y_{1,n}, \ldots, y_{m,1}, \ldots, y_{m,n} \}.$$

That is, we have (1) a variable x_i corresponding to each node v_i and (2) a variable $y_{j,k}$ corresponding to each edge e_j and each node v_k. For convenience, we will sometimes write $\mathbf{y}_j$ instead of $(y_{j,1} \ldots y_{j,n})$ when it is clear that we are talking about the tuple of variables corresponding to the edge e_j. This is because we have an n-tuple of variables $\mathbf{y}_j$ for each edge e_j, where $1 \leq j \leq m$. We construct a team that contains an assignment s_i corresponding to each node v_i. The assignment s_i encodes the neighborhood of v_i and all the edges that contain v_i in the graph. This is achieved by mapping each variable $y_{\ell,j}$ in the tuple y_ℓ to 1 under the assignment s_j if $v_j \in e_\ell$ whereas $s_j(y_{\ell,j}) = 0$ if $v_j \notin e_\ell$ and for every $j \neq i$, $s_j(y_{\ell,i}) = 1$. Figure 3.5 depicts an example graph to get some intuition on this construction.

Formally, the team $T = \{s_1, \ldots, s_n\}$ is defined as follows.

1. If $\mathcal{G}$ has an edge $e_\ell = \{ v_i, v_j \}$ then we set $s_i(x_j) = 1$ and $s_j(x_i) = 1$, and let $s_i(y_{\ell,1}) = \ldots = s_i(y_{\ell,n}) = 1$ as well as $s_j(y_{\ell,1}) = \ldots = s_j(y_{\ell,n}) = 1$.

2. For the case $v_j \notin e_\ell$, we set $s_j(y_{\ell,j}) = 0$ and $s_j(y_{\ell,i}) = 1$ for $i \neq j$.

3. Since, we can assume w.l.o.g. the graph has no loops (self-edges) we always have $s_i(x_i) = 0$ for all $1 \leq i \leq n$.

Consequently, two assignments s_i, s_j agree on a tuple $\mathbf{y}_k$ and we have $s_i(y_{k,q}) = 1 = s_j(y_{k,q})$ for each $q \leq n$, if the corresponding e_k is the edge between v_i and v_j.
Now let ϕ be the following $\mathcal{PDL}$-formula

$$\phi := \psi \vee \psi \vee \psi,$$

where

$$\psi := \bigwedge_{e_k = \{ v_i, v_j \}} \mathsf{dep}(\mathbf{y}_k; x_i).$$

The choice of x_i or x_j to appear in the formula is irrelevant. The idea is that if there is an edge e_k between v_i, v_j and accordingly $s_i(\mathbf{y}_k) = s_j(\mathbf{y}_k)$ then the two assignments cannot be in the same split of the team. This is always true because in that case the assignments s_i, s_j cannot agree on any of x_i or x_j. Since, by (3.), we have $s_i(x_i) = 0$ but there is an edge to v_j and we have $s_j(x_i) = 1$. We prove the correctness of the reduction by the following claim.

Claim 3.1. *$\mathcal{G}$ is 3-colourable iff $\{s_1, \ldots, s_n\} \models \phi$.*

Proof of Claim. *"$\Rightarrow$": Let V_1, V_2, V_3 be a partition of V into three colours. Consequently, for every $v \in V$ we have an $r \leq 3$ such that $v \in V_r$. Moreover, for every $v_i, v_j \in V_r$ there is no ℓ s.t. $e_\ell = \{ v_i, v_j \}$. Let $T_r = \{ s_i \mid v_i \in V_r \}$ for each $r \leq 3$, then we show that $\bigcup_{r<3} T_r = T$ and $T_r \models \psi$. This will prove that $T \models \phi$ because we can split T into three subteams such that each satisfies one disjunct.*

Since for each $v_i, v_j \in V_r$, there is no edge $e_\ell = \{ v_i, v_j \}$ this implies that for $s_i, s_j \in T_r$ and the tuple $\mathbf{y}_\ell$, we have $s_i(\mathbf{y}_\ell) \neq s_j(\mathbf{y}_\ell)$. As a result, $\{s_i, s_j\}$ evaluates each dependence atom in ψ trivially true. Moreover, since every dependence atom is 2-coherent (see page 16), it is enough to check only for pairs s_i, s_j and since the condition holds for every edge, we have $T_r \models \psi$. Since we assume that V can be split into three subsets, we have the split of T into three subteams. This proves that $T \models \phi$.

"$\Leftarrow$": Conversely, assume that T can be split into three subteams, each satisfying ψ. Then we show that V_1, V_2, V_3 is the partition of V into three colors where $V_r = \{ v_i \mid s_i \in T_r \}$. Clearly, $\bigcup_{r \leq 3} V_r = V$ and we prove that for any $v_i, v_j \in V_r$ there is no edge between v_i, v_j. Suppose to the contrary that there is an edge $e_\ell = \{ v_i, v_j \}$ for some $v_i, v_j \in V_r$. Then we must have $s_i, s_j \in T_r$ such that $s_i(\mathbf{y}_\ell) = s_j(\mathbf{y}_\ell)$. That is, $s_i(y_{\ell,q}) = 1 = s_j(y_{\ell,q})$ for each $q \leq n$. Since we have that $s_i(x_i) = 0$ whereas $s_j(x_i) = 1$, this implies that $\{ s_i, s_j \} \not\models \psi$, which is a contradiction since $T_r \models \psi$. ■

Finally, the reduction can be achieved in polynomial time and this concludes the full proof. □

'variable'	'parity'	'clause'	'position'
x	y	u	v
p_1	1	1	0
p_2	0	1	1
p_3	0	1	2

'variable'				'par'	'clause'				'pos'	
x_1	...	x_{r-1}	x_r	y	u_1	...	u_{s-1}	u_s	v_1	v_2
0	...	0	0	1	0	...	0	0	0	0
0	...	0	1	0	0	...	0	0	0	1
0	...	1	1	0	0	...	0	0	1	0

Table 3.2: A first-order team (left) and its propositional translation (right) for $(p_1 \vee \neg p_2 \vee \neg p_3)$.

We considered #splits as a parameter because a split-junction forces a team to split into subteams. One might ponder whether the number of conjunctions also plays a role in the complexity of the model checking problem. We claim that #conjunctions is also an interesting parameter for the following reason. Peter Lohmann posed an open question in his Ph.D. thesis [76] about the precise complexity of MC for the fragment of $\mathcal{PDL}$ which disallows the conjunction operator. However, it is worth pointing out that the author considers only the dependence atoms of the form $\mathsf{dep}(P;q)$, rather than $\mathsf{dep}(P;Q)$. In other words, the second argument in an atom is a variable instead of a set of variables. In contrast, we use atoms of the form $\mathsf{dep}(P;Q)$ in our analysis by observing that $\mathsf{dep}(P;Q) \equiv \bigwedge_{q\in Q} \mathsf{dep}(P;q)$. Clearly, the absence of '$\wedge$' renders this impossible. In the proof of the following theorem, we present a reduction from 3SAT that uses no conjunction operator (or, in the light of the above discussion, only a few conjunction operators). The reduction in the proof of the following theorem was established during a collaboration with Jonni Virtema, and I include this with his kind permission.

Theorem 3.12. p-MC(#conjunctions) *is* **paraNP**-*complete.*

Proof. Jarmo Kontinen ([68]) proved that the model checking problem for a fixed $\mathcal{D}$-formula (first-order dependence logic) of the form $\mathsf{dep}(x;y) \vee \mathsf{dep}(u;v) \vee \mathsf{dep}(u;v)$ is still **NP**-complete. We briefly sketch the reduction from 3SAT presented by the author.

Let $\psi := C_1 \wedge \ldots \wedge C_m$ be an instance of 3SAT over propositions $\{p_1, \ldots, p_n\}$, where each C_i is a clause of length at most three, for $i \leq m$. Consider the structure $\mathcal{A}$ over the empty vocabulary, that is, $\tau = \emptyset$. Let $A = \{p_1, \ldots, p_n\} \cup \{c_1, \ldots, c_m\} \cup \{0, 1, 2\}$. The team T is constructed over variables $\{x, y, u, v\}$ that take values from A. Moreover, T encodes the position and the parity of each variable in a clause. Finally, the desired $\mathcal{D}$-formula is $\varphi := \mathsf{dep}(x;y) \vee \mathsf{dep}(u;v) \vee \mathsf{dep}(u;v)$.

We do not prove the correctness of the reduction here and only encode $(\mathcal{A}, T, \varphi)$ in binary so that the resulting team T_b is a team over propositional variables, and the formula ϕ is a $\mathcal{PDL}$-formula. Let $r = \lceil \log(n) \rceil$ and $s = \lceil \log(m) \rceil$. The idea is to encode the universe A in binary using $r + s + 2$ additional propositions. The variable x is encoded by propositions $\{x_1, \ldots, x_r\}$, u by $\{u_1, \ldots, u_s\}$, v by $\{v_1, v_2\}$, and finally y remains unchanged. As an example, the assignments corresponding to the clause $(p_1 \vee \neg p_2 \vee \neg p_3)$ and their binary encoding is depicted in Table 3.2. Finally, $(\mathcal{A}, T, \varphi)$ (or indeed the instance ψ of 3SAT) is mapped to an instance (T_b, ϕ) of $\mathcal{PDL}$-MC, where T_b is the binary encoding of T over the propositional variables

$\{x_1, \ldots, x_r, y, u_1, \ldots, u_s, v_1, v_2\}$ and ϕ is the following $\mathcal{PDL}$-formula.

$$\phi := \mathsf{dep}(\mathbf{x}; y) \vee \mathsf{dep}(\mathbf{u}; \mathbf{v}) \vee \mathsf{dep}(\mathbf{u}; \mathbf{v}) \tag{3.1}$$

Notice that ϕ does not include any conjunction operator, and this proves that 0-slice of p-MC(#conjunctions) is **NP**-hard, resulting in the desired **paraNP**-hardness. □

In order to make a reference to the (above stated) open question by Peter Lohmann, we rewrite the formula ϕ from Equation 3.1 as follows.

$$\phi' := \mathsf{dep}(\mathbf{x}; y) \vee [\mathsf{dep}(\mathbf{u}; v_1) \wedge \mathsf{dep}(\mathbf{u}; v_2)] \vee [\mathsf{dep}(\mathbf{u}; v_1) \wedge \mathsf{dep}(\mathbf{u}; v_2)]$$

Surprisingly, ϕ' contains only two conjunction operators. Furthermore, notice that ϕ contains only three dependence atoms. This yields the following interesting corollary, where $\#\mathsf{atoms}(\phi)$ denotes the number of dependence atoms in ϕ.

Corollary 3.13. p-MC(#atoms) *is* **paraNP**-*complete.*

Proof. The **paraNP**-hardness follows from the proof of Theorem 3.12 □

It is interesting to observe that the $\mathcal{PDL}$-formula ϕ in the proof of Theorem 3.12 also consists of two splits. As a result, the **paraNP**-hardness of p-MC(#splits) follows as a corollary to Theorem 3.12. Nevertheless, the hardness of p-MC(#splits) was first proven via a reduction presented in Theorem 3.11 and we include the original reduction in the thesis.

Now we analyse the parameter $|T|$. Notice that due to the downwards closure property of $\mathcal{PDL}$, it suffices to consider only the strict splits for teams. As a result, a team of size k has 2^k different candidates for the correct split corresponding to each split-junction, and each can be verified in polynomial time. This implies that an exponential runtime in the input length seems necessary. However, when $|T|$ is a parameter, the problem can be solved in polynomial time with respect to the input-size and exponentially in the parameter.

Theorem 3.14. p-MC($|T|$) *is* **FPT**.

Proof. We present a procedure (Algorithm 1) that solves the model checking problem for $\mathcal{PDL}$ and runs in fpt-time. The correctness follows because the procedure is simply a recursive definition of the truth evaluation of $\mathcal{PDL}$-formulas in a bottom-up fashion.

Recall that the input formula ϕ is a binary tree. The procedure starts by labeling every satisfying subteam $P \subseteq T$ with each atomic (or negated atomic) subformula α. Notice that this step also deals with negated atomic suformulas because we only allow atomic negations. Then recursively, if $P \models \alpha_i$ for $i = 1, 2$ and there is a subformula α such that $\alpha = \alpha_1 \wedge \alpha_2$ then it adds P to the label of α. Moreover, if $P_i \models \alpha_i$ for $i = 1, 2$ and there is a subformula α such that $\alpha = \alpha_1 \vee \alpha_2$ then it adds $P = P_1 \cup P_2$ to the label of α.

The first loop runs in $\mathcal{O}(2^k) \cdot |\phi|$ steps for each leaf node and the number of iterations is also bounded by $|\phi|$. At each inner node, there are at most 2^k candidates for each of P_1 and P_2,

Algorithm 1: Recursive bottom-up algorithm solving p-MC($|T|$).

Input : A $\mathcal{PDL}$-formula ϕ and a team T
Output: true if $T \models \phi$, otherwise false

1 **foreach** *non-root node v in the syntax tree* **do** $L_v = \{\emptyset\}$
2 **foreach** *atomic/negated atomic $\ell \in SF(\phi)$* **do** `// find all subteams for` ℓ
3 $\quad L_\ell = \{\emptyset\}$
4 $\quad$ **foreach** $P \subseteq T$ **do**
5 $\quad\quad$ **if** $\ell = x$ *and* $\forall s \in P : s(x) = 1$ **then** $L_\ell \leftarrow L_\ell \cup \{P\}$
6 $\quad\quad$ **else if** $\ell = \neg x$ *and* $\forall s \in P : s(x) = 0$ **then** $L_\ell \leftarrow L_\ell \cup \{P\}$
7 $\quad\quad$ **else if** $\ell = \neg\mathsf{dep}(\mathbf{P};\mathbf{Q})$ **then** $L_\ell \leftarrow L_\ell$ `// because` $\emptyset \models \neg\mathsf{dep}(\mathbf{P};\mathbf{Q})$
8 $\quad\quad$ **else if** $\ell = \mathsf{dep}(\mathbf{P};\mathbf{Q})$ *and* $\forall s_i \forall s_j \bigwedge_{p\in P} s_i(p) = s_j(p) \Rightarrow \bigwedge_{q\in Q} s_i(q) = s_j(q)$ **then**
9 $\quad\quad\quad$ $L_\ell \leftarrow L_\ell \cup \{P\}$
10 **foreach** α_1, α_2 *with* $\alpha = \alpha_1 \circ \alpha_2$ *and* $L_{\alpha_i} \neq \{\emptyset\}$ *for* $i = 1, 2$ **do**
11 $\quad$ **foreach** $P_1 \in L_{\alpha_1}$, $P_2 \in L_{\alpha_2}$ **do**
12 $\quad\quad$ **if** $\circ = \wedge$ *and* $P_1 = P_2$ **then** $L_\alpha \leftarrow L_\alpha \cup \{P_1\}$
13 $\quad\quad$ **else if** $\circ = \vee$ **then** $L_\alpha \leftarrow L_\alpha \cup \{P_1 \cup P_2\}$
14 **if** $T \in L_\Phi$ **then return** `true` **else return** `false`

and consequently, at most 2^{2k} pairs need to be checked. This implies the loop for each inner node can be implemented in $\mathcal{O}(2^{2k}) \cdot |\phi|$ steps. Furthermore, the loop runs once for each pair of subformulas α_1, α_2 such that $\alpha_1 \circ \alpha_2$ is also a subformula of ϕ for $\circ \in \{\wedge, \vee\}$. Finally, in the last step, a set of size k needs to be checked against a collection containing 2^k such sets. This can be achieved in $\mathcal{O}(k \cdot 2^k)$ steps.

We conclude that the above procedure solves p-MC($|T|$) in $\mathcal{O}(2^{2k}) \cdot p(|\phi|)$ steps for some polynomial p. It does not yield a blow-up in the number of subformulas because the formula tree is binary. The procedure operates on a pair of subformulas in each step, and the size of each label ($|L_\alpha|$) is again bounded by 2^k. □

It is worth pointing out that Algorithm 1 is not optimal because in each label it stores more subteams than required to solve MC for $\mathcal{PDL}$. In other words, for literals (in Line 5–6), one can allow storing only the unique maximal subteam satisfying it, rather than storing each satisfying subteam. Similarly, for dependence atoms (Line 8), store all maximal satisfying subteams. The desired results are achieved due to the downwards closure property of $\mathcal{PDL}$-formulas. The only modification requires taking intersections of subteams from each label for a conjunction node. Nevertheless, in its current form it is easy to observe how Algorithm 1 can solve p-MC($|T|$) for any team based logic $\mathcal{L}$, such that $\mathcal{L}$-atoms can be evaluated in polynomial time.

We conclude this section on the model checking by presenting **FPT** results regarding the remaining parameterizations.

Theorem 3.15. p-MC(κ) *is* **FPT** *for every* $\kappa \in \{|\phi|, |\mathsf{var}(\phi)|, \mathsf{formula\text{-}depth}\}$.

Proof. Recall the relationship we proved between various parameters (Lemma 3.5). The **FPT**-membership for $|\phi|$ and $|\mathsf{var}(\phi)|$ follows due to Proposition 2.14 and the fact the p-MC($|T|$) is

FPT. Furthermore, the **FPT**-membership when parameterized by formula-depth follows due to the relationship between $|\phi|$ and formula-depth. □

Finally, the case for formula-team-tw follows due to Corollary 3.7 in conjunction with the **FPT** result for $|T|$ (Lemma 3.14).

Corollary 3.16. p-MC(formula-team-tw) *is* **FPT**.

3.3 Satisfiability

In this section, we study the parameterized complexity of the satisfiability problem (p-SAT(κ)) under various parameterizations (κ). So the question, is there a team T for a given formula ϕ such that $T \models \phi$? Recall (Sec 2.1) that the question is equivalent to finding a singleton team satisfying the input formula. As a result, team semantics coincides with the usual Tarskian semantics. This facilitates in determining the truth value of (1) disjunctions in the classical way, and (2) dependence atoms to be trivially true. Simplifying the notation slightly, for SAT we now look for an assignment rather than a singleton team that satisfies the formula.

Problem:	p-SAT(κ)
Input:	A $\mathcal{PDL}$-formula ϕ, and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Is there an assignment s over Vars(ϕ) such that $\{s\} \models \phi$?

Notice that an input instance for SAT consists of a $\mathcal{PDL}$-formula ϕ alone. This affects two of our parameterizations $|T|$ and formula-team-tw which include a 'team'. The parameter $|T|$ is not meaningful anymore (see Remark 3.1) whereas formula-team-tw is the same as formula-tw. This is due to the reason that we have only one graph representation, namely G_ϕ for an input instance ϕ.

Corollary 3.17. p-SAT(κ) *for* $\kappa \in \{$formula-team-tw, formula-tw$\}$ *is the same.*

Classically, $\mathcal{PDL}$-SAT is **NP**-complete. The hardness follows from the **NP**-completeness of $\mathcal{PL}$-SAT [21, 73]. The membership holds because it is enough to find a single satisfying assignment. This also implies the trivial **paraNP** upper bound under any parameterization.

Proposition 3.18. p-SAT(κ) *is in* **paraNP** *for any parameterization* κ.

Recall that $\mathcal{PL}$ is a fragment of $\mathcal{PDL}$, with no dependence atoms. The following result is an immediate corollary of the **NP**-completeness of $\mathcal{PL}$-SAT [21, 73].

Corollary 3.19. p-SAT(κ) *is* **paraNP**-*complete for any* $\kappa \in \{$dep-arity, #atoms$\}$.

Proof. The 0-slice regarding dep-arity and #atoms (no dependence atoms at all) is **NP**-hard. □

Interestingly, these are the only parameters that yield intractability results. The remaining cases are all **FPT** as we prove next.

Turning towards treewidth, notice first that $\mathcal{PL}$-SAT(treewidth) is **FPT** due to Samer and Szeider [102, Thm. 1]. However, their result does not immediately translate to our setting because Samer and Szeider study CNF-formulas and we have arbitrary formulas instead. Nevertheless, Lück et al. [78, Cor. 4.7] studying temporal logics under the parameterized approach, classified, as a byproduct, the propositional satisfiability problem with respect to arbitrary formulas to be fixed-parameter tractable. As a result, we have the following corollary.

Corollary 3.20. p-SAT(formula-tw) *is* **FPT**.

Proof. As stated before, we need to find a singleton team. It is important to observe that a split-junction has the same semantics as the classical disjunction, and dependence atoms are trivially satisfied. Consequently, replacing the occurrence of every dependence atom $\mathsf{dep}(P;Q)$ by $\top$ yields a propositional logic formula. This substitution does not increase the treewidth. Finally the result follows due to Lück et al. [78, Cor. 4.7]. □

Now, we turn towards the parameter #splits. We present a procedure that constructs a satisfying assignment s such that $s \models \phi$ if there is one and otherwise it answers no. The idea is that this procedure needs to remember positions where a modification in a satisfying assignment is possible. We show that the number of modifications in such positions is bounded by the parameter #splits.

Consider the syntax tree of ϕ where, as before, multiple occurrences of subformulas are allowed. The procedure starts at the leaf level with satisfying assignment candidates (partial assignments, to be precise). Reaching the root (the formula ϕ), the procedure confirms whether it is possible to have a combined assignment or not. We assume that the leaves of the tree consist of literals, dependence atoms, or negated dependence atoms. Accordingly, the internal nodes of the tree are only conjunction and disjunction nodes. The procedure sets all the dependence atoms to be trivially *true* (an assignment always satisfies them). Moreover, it sets all the negated dependence atoms to be *false* because there can be no satisfying assignment for a negated dependence atom. Additionally, it sets each literal to its respective satisfying assignment. Ascending the tree, it checks the relative condition for conjunction and disjunction by joining the assignments and thereby giving rise to conflicts. A conflict arises (at a conjunction node) when two assignments are joined with contradicting values for some variable. At this point, it sets this variable x to a conflict state c. At disjunction nodes the assignment stores that it has two separate options.

Joining a *true*-value from a dependence atom affects the assignment only at disjunction nodes. This simulates the effect that the formula of the form $\mathsf{dep}(P;Q) \vee \psi$ is true under any assignment. At a conjunction node, when an assignment s joins with a *true*, the procedure returns the assignment s. Since at a split the procedure returns both assignments, for k splits there can be $\leq 2^k$-many different choices. At the root node if at least one assignment is consistent then we have a satisfying assignment. Otherwise, if all the choices contain conflicts over some variables then there is no satisfying assignment for ϕ.

Theorem 3.21. p-SAT(#splits) *is* **FPT**. *Moreover, given a $\mathcal{PDL}$-formula ϕ, there is an algorithm that determines whether ϕ is satisfiable in time $\mathcal{O}(2^{\#\mathsf{splits}(\phi)}) \cdot p(|\phi|)$ for some polynomial p.*

Proof. Consider Algorithm 2 which constructs partial assignments for ϕ of the form $t\colon \mathrm{Vars}(\phi) \to \{0, 1, c\}$. Intuitively, these mappings are used to find a satisfying assignment in the process of the presented algorithm.

If t, t' are two (partial) mappings then $t \mathbin{©} t'$ is the assignment such that

$$(t \mathbin{©} t')(x) := \begin{cases} \text{undefined} & \text{, if both } t(x) \text{ and } t'(x) \text{ are undefined,} \\ c & \text{, if both are defined and } t(x) \neq t'(x), \\ t(x) & \text{, if only } t(x) \text{ is defined,} \\ t'(x) & \text{, if only } t'(x) \text{ is defined.} \end{cases}$$

We prove the following claim.

Claim 3.2. *The formula ϕ is satisfiable if and only if Algorithm 2 returns a consistent (partial) assignment s. Moreover, s can be extended to a satisfying assignment for ϕ over* $\mathrm{Vars}(\phi)$.

Proof of Claim. *We prove the claim using induction on the structure of ϕ.*

Base case. *If $\phi = x$, then ϕ is satisfiable and $s \models \phi$ such that $s(x) = 1$. Moreover, the procedure returns such an assignment as depicted by line 3 of the algorithm. Similarly, the case $\phi = \neg x$ follows from line 4. The case $\phi = \mathsf{dep}(P; Q)$ or $\phi = \top$ is a special case since this is true under any assignment. Line 6 in our procedure returns a partial assignment that can be extended to any consistent assignment. Finally, for $\phi = \bot$ or $\phi = \neg\mathsf{dep}(P; Q)$, the assignment contains a conflict and can not be extended to a consistent assignment and the algorithm returns "ϕ is not satisfiable".*

Induction Step. *Notice first that if either of the two operands is $\top$, then this is a special case and triggers lines 9–11 of the algorithm, thereby giving the satisfying assignment.*

Suppose now that $\phi = \psi_0 \wedge \psi_1$ and that the claim is true for ψ_0 and ψ_1. As a result, both ψ_0 and ψ_1 are satisfiable, if and only if the algorithm returns a satisfying assignment for each. Let S_i for $i = 0, 1$ be such that some consistent $t'_i \in S_i$ can be extended to a satisfying assignment t_i for ψ_i. We claim that S_ϕ returned by the procedure (line 13) is non-empty and contains a consistent assignment for ϕ, if and only if ϕ is satisfiable. First observe that, by construction, S_i contains all the possible partial assignments that satisfy ψ_i for $i = 0, 1$. Consequently, S_ϕ contains all the possible $©$-joins of such assignments that can satisfy ϕ. Let ψ_0 be satisfied by t'_0 and ψ_1 be satisfied by t'_1. Moreover, let $s' \in S_\phi$ be an assignment such that $s' = t'_0 \mathbin{©} t'_1$. If s' is consistent then s' can be extended to a satisfying assignment s for ϕ since $s' \models \psi_i$ for $i = 0, 1$. On the other hand if every $t'_0 \mathbin{©} t'_1$ is conflicting (for $t'_i \in S_i$) then there is no assignment over $\mathrm{Vars}(\phi) = \mathrm{Vars}(\psi_1) \cup \mathrm{Vars}(\psi_2)$ *that satisfies ϕ. As a result, ϕ is not satisfiable.*

The case for split-junction is simpler, as we prove next. Suppose that $\phi = \psi_0 \vee \psi_1$ and that the claim is true for ψ_1 and ψ_2. Then ϕ is satisfiable, if and only if either ψ_0 or ψ_1 is satisfiable. Since the label

Algorithm 2: Bottom-up algorithm solving p-SAT(#splits).

Input : $\mathcal{PDL}$-formula ϕ represented by a syntax tree with atomic/negated atomic subformulas as leaves

Output: An assignment s such that $s \models \phi$ or "ϕ is not satisfiable"

```
1  foreach Leaf ℓ of the syntax tree do                    //atomic or negated-atomic subformula
2      S_ℓ = {∅}
3      if ℓ = x is a variable then S_ℓ ← { { x ↦ 1 } };
4      else if ℓ = ¬x is a negated variable then S_ℓ ← { { x ↦ 0 } };
5      else if ℓ = ⊥ or ℓ = ¬dep(P;Q) then pick x ∈ VAR and S_ℓ ← { { x ↦ c } };
6      else S_ℓ ← { 1 };                                   // case ⊤ or dep(P;Q)
7  foreach Inner node ℓ of the syntax tree in bottom-up order do
8      Let ℓ_0, ℓ_1 be the children of ℓ with S_0, S_1 the resp. sets of partial assignments;
9      if 1 ∈ S_i then
10         if ℓ is a conjunction then S_ℓ ← S_{1−i};
11         else S_ℓ ← { 1 };                               // empty split for a split-junction
12     else if ℓ is a conjunction then
13         foreach s_0 ∈ S_0 and s_1 ∈ S_1 do  S_ℓ ← S_ℓ ∪ { s_0 © s_1 } ;
14     else                                                // ℓ is a split-junction
15         foreach s_0 ∈ S_0 and s_1 ∈ S_1 do  S_ℓ ← S_ℓ ∪ { s_0, s_1 } ;
16 if there exists a non-conflicting assignment s ∈ S_Φ then return s;
17 else return "φ is not satisfiable";
```

S_ϕ for ϕ is the union of the labels for ψ_0 and ψ_1, it is enough to check that either the label of ψ_0 (that is, S_0), or that of ψ_1 (S_1) contains a consistent partial assignment. By the induction hypothesis, this is equivalent to determining whether ψ_0 or ψ_1 is satisfiable. This completes the case for split-junction and the proof of our claim. ■

Finally, notice that the label size adds when a split-junction occurs. In other words, we keep all the assignment candidates separate, and each such candidate is present in the label for the split-junction. In contrast, at conjunction nodes, we 'join' the assignments, and as a result, the label size is the product of the two labels. Notice that we do not get a blow-up in the number of conjunctions. This is because the label size for each node is initially one, and only at a split-junction does the size increase. This implies that the maximum size for any label is bounded by $2^{\#\mathsf{splits}(\phi)}$. As a result, the argument regarding the runtime follows. □

In the following, we present an example as the application of Algorithm 2.

Example 3.22. *Let $\phi = [(x_4 \wedge x_1) \wedge \neg x_2] \wedge [((x_1 \wedge x_2) \vee \mathsf{dep}(x_3; x_4)) \wedge (x_3 \vee \neg x_1)]$. Figure 3.6 depicts the application of the procedure on the syntax tree of ϕ. To simplify the notation, we consider the assignment labels of the form $\{x_i, \neg x_j\}$ rather than $\{x_i \mapsto 1, x_j \mapsto 0\}$.* ◁

The remaining **FPT**-cases for the parameters $|\mathsf{var}(\phi)|$, $|\phi|$, or formula-depth follow easily from the results already proven.

Theorem 3.23. p-SAT(κ) *is* **FPT** *for* $\kappa \in \{|\mathsf{var}(\phi)|, |\phi|, \mathsf{formula\text{-}depth}\}$.

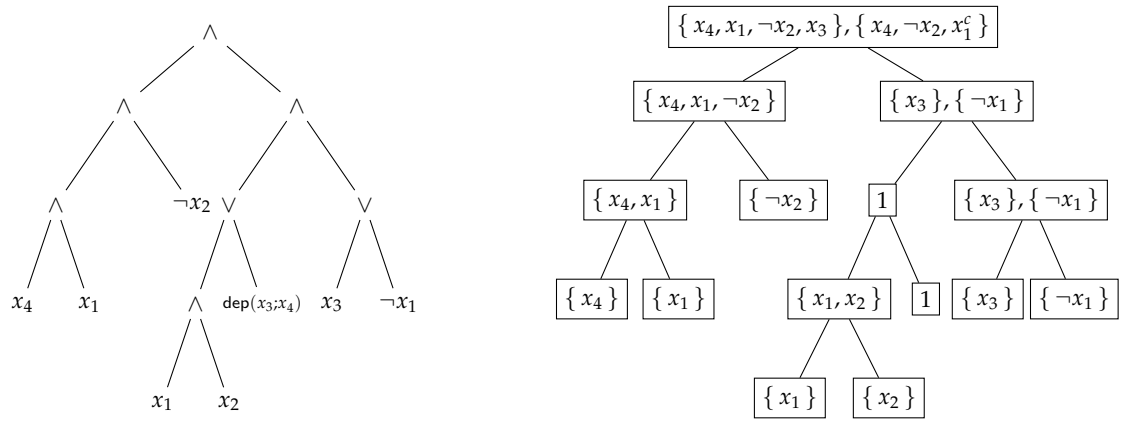


Figure 3.6: (Left) the syntax tree of ϕ, and (right) computation of Algorithm 2. Notation: $x/\neg x/x^c$ means a variable is set to true/false/conflict. Clearly, $\{ x_4, x_1, \neg x_2, x_3 \}$ satisfies the formula.

Proof. If a formula ϕ contains k propositional variables, then there are 2^k different assignments. As a consequence, one can check for each assignment s whether $s \models \phi$, this can be achieved in time $2^k \cdot |\phi|$. Regarding $|\phi|$, note that and any problem Π is **FPT** when parameterized by the input-length. Consequently, p-SAT($|\phi|$) is **FPT**.

If formula-depth$(\phi) = d$ then there are $\leq 2^d$ leaves and $\leq 2^d$ internal nodes. Accordingly we have $|\phi| \leq 2^{2d}$ which shows **FPT** membership, when parameterised by formula-depth. □

The complexity of p-SAT(#conjunctions) is currently unclassified. The author believes that this might yield another **FPT** result, but do not have a proof yet.

3.4 A Digression

Towards the end of our parameterised complexity analysis of propositional dependence logic, we present two additional interesting problems for $\mathcal{PDL}$.

3.4.1 A Satisfiability Variant (*m*SAT)

Our first proposal is a variant of SAT, we call it *m*SAT. An input to *m*SAT includes a $\mathcal{PDL}$-formula ϕ and a unary encoding 1^m of a number $m \geq 2$. The task is to determine whether ϕ has a satisfying team of size at least m.

Problem:	*m*SAT
Input:	A $\mathcal{PDL}$-formula ϕ and $m \in \mathbb{N}$ in unary.
Question:	Is there a satisfying team T for ϕ such that $\lvert T \rvert \geq m$?

Notice that since $\mathcal{PDL}$-formulas are downwards closed, *m*SAT is equivalent to finding a team of size exactly m. The motivation is to study the satisfiability problem for $\mathcal{PDL}$ with its original *team-semantic* nature. The problem of finding a team of a given size has not been studied before

in the classical setting (up to the author's knowledge). Meier and Reinbold [87] considered the enumeration complexity of Poor man's propositional dependence logic, which is a fragment of $\mathcal{PDL}$ with no splits. Another related yet different problem in the context of databases is to find an Armstrong relation [6, 42]. Let Γ be a set of functional dependencies, the question is whether there exists a database D that satisfies a functional dependency $\mathsf{dep}(P;Q)$ if and only if $\mathsf{dep}(P;Q)$ is implied by Γ. In our setting, the problem mSAT considers general $\mathcal{PDL}$ formulas (so not only functional dependencies). Furthermore, mSAT questions whether it is possible to generate a database of size m, and the restriction that it should not satisfy any other functional dependency, is dropped. We explore the classical complexity of mSAT, followed by proving results in the parameterized setting.

We present a scenario to emphasize that mSAT is relevant in practice. Assume that one is given a set of constraints, and the task is to populate the database with a certain number of entries (m in our case). Such a scenario naturally arises, for example, when someone is making a teaching schedule. The weekly lectures should be arranged subject to certain constraints, such as (1) the teacher and the room determine the course, (2) the teacher and the time determine the room and the course, and many more. One would like to have an algorithm with this collection of constraints and the solution size (the number of weekly lectures) as an input, and the algorithm outputs a consistent teaching schedule for them.

Example 3.24. *Let $\phi := \mathsf{dep}(x,y;z) \wedge (x \vee y)$. Then, for the team $T = \{010, 101, 111\}$ over $\{x,y,z\}$, we have that $T \models \phi$. This implies that there is a team of size three for ϕ. Moreover, it is easy to observe that for any other assignment $t \notin T$ over $\{x,y,z\}$, we have that $T \cup \{t\} \not\models \phi$. As a consequence, there is no team of size four that also satisfies ϕ.*

There is a reason for the requirement that m should be in unary, as one might be looking for an arbitrary large solution. Consequently, it makes sense to ask whether it is possible to fill the database (even with an arbitrarily large size) in polynomial time with respect to the input. This corresponds to the intuition that the input to mSAT is an empty database D with allocated memory of size m (that is, with blank rows), and the task is to populate D with the data that satisfies given constraints. If the restriction of m not being in unary is lifted, the problem immediately becomes **NEXP**-complete. The membership follows because given an input ϕ, guessing a team T of (presumably) exponential size and verifying that $T \models \phi$ requires nondeterministic exponential time. The hardness follows due to a reduction from the validity problem for $\mathcal{PDL}$ [112]. In polynomial time, one can count the number of proposition symbols n in a $\mathcal{PDL}$-formula ϕ and ask whether there is a satisfying team for ϕ of size m where $m = 2^n$. It is easy to observe that ϕ is valid if and only if ϕ has a satisfying team of size m.

Remark 3.25. *The problem mSAT is* **NEXP***-complete when m is given in binary.*

Now we prove that restricting m to be unary drops the complexity to **NP**-completeness. The proof uses the idea of a truth function introduced by Yang [113]. For a better understanding and self containment, we include the definition (modified to fit our setting) for a truth function.

Definition 3.26 ([113, Def. 4.17])**.** *Let ϕ be a $\mathcal{PDL}$-formula and T be a team. A function $f\colon SF(\phi) \to 2^T$ is called a truth function for ϕ over T if the following conditions are satisfied.*

- $f(\phi) = T$,
- $f(\psi) = f(\psi_0) = \psi_1$ *if* $\psi = \psi_0 \wedge \psi_1$, *and* $f(\psi) = f(\psi_0) \cup \psi_1$ *if* $\psi = \psi_0 \vee \psi_1$,
- $f(\psi) \models \psi$ *for every* $\psi \in SF(\phi)$.

The truth evaluation of a $\mathcal{PDL}$-formula ϕ under a team T can be determined by the existence of a truth function for ϕ over T.

Proposition 3.27 ([113, Theorem 4.19])**.** *Let ϕ be a $\mathcal{PDL}$-formula and T be a team over* $\mathrm{Vars}(\phi)$*. Then $T \models \phi$ iff there is a truth function $f\colon SF(\phi) \to 2^T$ for ϕ over T.*

Theorem 3.28. *The problem m*SAT *is* **NP***-complete when m is given in unary.*

Proof. Notice first that finding a team of size m is computationally harder than finding a team of size one. Consequently, the hardness follows because SAT for $\mathcal{PDL}$ is **NP**-hard.

We prove **NP**-membership by presenting a non-trivial nondeterministic algorithm running in polynomial time that constructs a team of size m if there exists such. This is achieved by constructing a team T_i for each $i \in \mathrm{SF}(\phi)$, starting with the atomic and negated atomic subformulas. Let ϕ be an input formula, the procedure (Algorithm 3) labels each node of the tree with a satisfying team of size at most m. For each atomic/negated atomic subformula, this team is guessed nondeterministically, for a conjunction (resp., split-junction), the team is the intersection (union) of the two subteams from the successor nodes.

Specifically, for each literal ℓ, the label T_ℓ is a team of size m and $T_\ell = \{\, s_i \mid s_i(\ell) = 1 \,\}$. The question of how are the other variables mapped by each s_i, is answered nondeterministically. For each dependence atom $\alpha = \mathsf{dep}(P;Q)$, the label T_α is a team such that $|T_\alpha| = \min\{m, 2^{|P|}\}$. If $|T_\alpha| = m$, the assignments over $\mathrm{Vars}(\phi)$ are selected nondeterministically in such a way that $T_\alpha \models \alpha$. However, if $|T_\alpha| = 2^{|P|}$, then each assignment over P is selected and these assignments are extended to variables in $\mathrm{Vars}(\phi)\setminus P$ nondeterministically. At conjunctions, the label is the intersection of the two labels from the successor nodes, and at split junctions, it is the union. At the root level, if T_ϕ has a size of at least m, then Algorithm 3 accepts, otherwise, it rejects. The result follows from the downward closure property of $\mathcal{PDL}$-formulas. Moreover, the maximum label size for any node can be at most $|\phi| \cdot m$ because the size only increases (potentially doubles) at a split junction. This is unproblematic since m is in unary. Now we prove the correctness of the procedure through the following claim.

Claim 3.3. *ϕ has a satisfying team of size m if and only if Algorithm 3 outputs such a team when given the input ϕ. Moreover, Algorithm 3 runs in nondeterministic polynomial time.*

Proof of Claim. *The result is justified by the fact that Algorithm 3 constructs the truth function for ϕ in a bottom-up fashion.*

Algorithm 3: Non-deterministic algorithm solving mSAT.

Input : $\mathcal{PDL}$-formula ϕ represented by a syntax tree with atomic/negated atomic subformulas as leaves and m in unary

Output: A team T of size m s.t. $T \models \phi$ or "ϕ does not have a satisfying team of size m"

1 **foreach** *Leaf ℓ of the syntax tree* **do**
2 nondeterministically guess a team T_ℓ such that $|T_\ell| \leq m$ and $T_\ell \models \ell$
3 **foreach** *Inner node ℓ of the syntax tree in bottom-up order* **do**
4 Let ℓ_0, ℓ_1 be the children of ℓ with T_0, T_1 the resp. team labels;
5 **if** *ℓ is a conjunction* **then** $T_\ell = T_0 \cap T_1$;
6 **else** $T_\ell = T_0 \cup T_1$;
7 **if** $|T_\phi| \geq m$ **then return** T_ϕ;
8 **else return** "ϕ does not have a satisfying team of size m";

"⇒": Let T be a satisfying team for ϕ of size m. Consider the syntax tree of ϕ. Due to Proposition 3.27, each node ψ in the tree can be labeled with a satisfying team $f(\psi)$ for ψ. Moreover, $f(\psi) \subseteq T$ for each $\psi \in SF(\phi)$ and consequently $|f(\psi)| \leq m$. This implies that Algorithm 3 can guess teams T_ℓ in such a way that $T_\ell = f(\ell)$, where $\ell \in SF(\phi)$ is a leaf node of the syntax tree of ϕ. These teams correctly 'add-up' to T at the root node and $T \models \phi$ due to our assumption.

"⇐": Suppose that Algorithm 3 outputs a team T of size at least m. Let $T_1, \ldots T_n$ be the teams labeled by the algorithm at the leaves of ϕ, where n is the number of atomic or negated atomic subformulas (leaves in the syntax tree) of ϕ. First note that $T_i \models \ell_i$ where ℓ_i a leaf of ϕ and $i \leq n$. Moreover, for each split-junction, $\psi = \psi_0 \vee \psi_1$ the team label T_ψ for the node ψ is the union $T_{\psi_0} \cup T_{\psi_1}$, where T_{ψ_s} is the team label for ψ_s and $s \in \{0, 1\}$. Similarly, at the conjunction nodes, the team label for the parent node is the intersection of the teams from each conjunct. Finally, at the root level, the label has size m. Now, working in the backward direction, one can obtain the truth function $f\colon SF(\phi) \to 2^T$ for ϕ over T (guaranteed by the downwards closure property of $\mathcal{PDL}$). This proves that $T \models \phi$ (Prop. 3.27), where T is a team of size at least m. ■

It is easy to observe that Algorithm 3 runs in nondeterministic polynomial time with respect to ϕ and m. This completes the proof to the theorem. □

Now, we move on to the parameterised complexity of mSAT. Clearly those parameterizations κ that yield **paraNP**-hardness for p-SAT(κ) also render p-mSAT(κ) intractable. The following result follows from Corollary 3.19. The result regarding $|T|$ follows since SAT is **NP**-hard ($|T| = 1 = m$).

Corollary 3.29. p-mSAT(κ) *is* **paraNP**-*complete for any* $\kappa \in \{|T|, \mathsf{dep\text{-}arity}, \#\mathsf{atoms}\}$.

The following theorem presents **FPT**-results for mSAT under various parameterizations.

Theorem 3.30. p-mSAT(κ) *is* **FPT** *for any* $\kappa \in \{|\mathsf{var}(\phi)|, |\phi|, \mathsf{formula\text{-}depth}\}$, *with time even linear in the input length.*

Proof. Notice that a satisfying team T has a maximal size of 2^k where $k = |\mathsf{var}(\phi)|$. Moreover, there are a total of 2^{2^k}-many teams. Consequently, we can find all the satisfying teams of size m (if any) in **FPT**-time with respect to the parameter $|\mathsf{var}(\phi)|$.

For $|\phi|$, notice first that $|\mathsf{var}(\phi)| \leq |\phi|$, that is, bounding $|\phi|$ also bounds $|\mathsf{var}(\phi)|$ for any formula ϕ (Lem. 3.5). As a result, we have that mSAT parameterized by $|\phi|$ is **FPT**. For formula-depth notice that $|\phi| \leq 2^{2 \cdot \text{formula-depth}}$ and thereby the problem is **FPT** when parameterized by formula-depth.

Finally, consider the brute-force (bottom-up) algorithm in each case. This algorithm evaluates each subformula against the candidate subteam for this subformula. This implies that the running time with respect to the input is linear in the number of subformulas. That is, the algorithm runs in time $f(k) \cdot |\phi|$ for some computable function f. □

3.4.2 A Model Checking Variant (MaxSubTeam)

We propose another interesting problem which might have applications in the context of the database repairing. Let MaxSubTeam be the following problem. Given a $\mathcal{PDL}$-formula ϕ and a team T such that $T \not\models \phi$. The task is to find a maximal subteam $T' \subseteq T$ such that $T' \models \phi$. We also define the decision version (mSubTeam) of MaxSubTeam in the following. Given a team T and a $\mathcal{PDL}$-formula ϕ such that $T \not\models \phi$, mSubTeam asks whether there is subteam T' such that $T' \models \phi$ and $|T'| \geq m$ for some $m \in \mathbb{N}$?

Problem:	mSubTeam
Input:	A $\mathcal{PDL}$-formula ϕ, a team T s.t. $T \not\models \phi$, and $m \in \mathbb{N}$.
Question:	Is there a subteam $T' \subseteq T$, s.t. $\lvert T' \rvert \geq m$ and $T' \models \phi$?

MaxSubTeam (or its decisional version) appears to be an interesting variant of MC for $\mathcal{PDL}$ as we discuss next. The motivation lies in the fact that if $T \not\models \phi$ for a formula ϕ and a team T, then one might be interested in finding a maximal subteam of T that satisfies ϕ. Recall that $\emptyset \models \phi$ is always true (the empty team property), one can restrict the attention to non-empty subteams.

We discuss the applications of such a task from the database perspective. A database D (first-order relational structure[3]) is said to be inconsistent with respect to a collection Ψ of constraints, if $D \not\models \Psi$. Clearly, it is desirable to repair such a *broken* database in order to retrieve any useful information. One popular mean of repairing an inconsistent database is deleting the minimum number of tuples to achieve consistency. Alternatively, one can keep the inconsistent database but select answers only from a consistent subset of it (consistent query answering). In the context of dependence logic, Ψ is a collection of dependence atoms (functional dependencies) and the database D is a team. This is where the question of finding a (maximal) subteam D' of D, such that $D' \models \Psi$, becomes relevant. Hella and Hannula [53] studied the maximal subteam membership MSM(ϕ) for a fixed (first-order) inclusion logic formula ϕ. An input instance of

[3]We only intend to discuss the motivation without digging into details. Therefore it is not necessary to give semantics.

MSM(ϕ) is the tuple $(\mathcal{A}, T, s)$ and the problem is to determine whether s is in the maximal subteam T' of T such that $T' \models \phi$. For a detailed discussion and the usefulness of this problem, we refer the reader to the introduction section of their work [53]. The complexity of the problems related to database repairs and consistent query answering for various classes of dependencies has been extensively studied in the literature (see [1, 3, 19, 107] and references therein).

Example 3.31. *Let $\phi := \mathsf{dep}(x, y; z) \wedge (x \vee y)$ and the team $T = \{000, 010, 101, 111, 110\}$ over $\{x, y, z\}$. Then clearly, $T \not\models \phi$. However, $T_1 = \{010, 101, 111\}$ and $T_2 = \{010, 101, 110\}$ are both the maximal satisfying subteams for ϕ. As a consequence, there is a maximal satisfying subteam of T for ϕ with size three. Moreover, it is easy to observe that there is no satisfying subteam for ϕ with size four.*

Surprisingly, *m*SubTeam is already **NP**-hard for $\mathcal{PDL}$. The hardness is established in the same way as the reduction in the proof of Theorem 3.11.

Theorem 3.32. *The problem* *m*SubTeam *is* **NP***-hard.*

Proof. We reduce IndependentSet to *m*SubTeam by a similar construction as in the proof of Theorem 3.11 and using only one conjunct (no split-junction). Consider the reduction (presented in the proof of Theorem 3.11) from an instance $\mathcal{G} = (V, E)$ of 3COL to the instance (T, ϕ) of MC. Let T be the same team as constructed in the proof of Theorem 3.11, and let $\phi := \bigwedge_{e_k = \{v_i, v_j\}} \mathsf{dep}(\mathbf{y}_k; x_i)$. There is a one-one correspondence between the independent sets S of $\mathcal{G}$ and the subteams T' of T such that $T' \models \phi$. This is because each assignment $s \in T$ corresponds to a node $v \in V$ and vice versa. As a result, for each e_k, an independent set S contains no two end points of e_k if and only if the subteam containing assignments corresponding to nodes in S satisfies $\mathsf{dep}(\mathbf{y}_k; x_i)$. Conversely, $T' \models \phi$ for a subteam $T' \subseteq T$ if and only if the set S containing nodes corresponding to the assignments in T' is an independent set. Consequently, the **NP**-hardness of *m*SubTeam follows since IndependentSet is **NP**-complete ([67]). □

Regarding membership, we can only prove an upper bound of $\mathbf{\Sigma}_2^p$ because guessing a subteam requires **NP**-time and verification can be achieved via an **NP**-oracle. Nevertheless, if ϕ is a Poor man's $\mathcal{PDL}$-formula (a fragment of $\mathcal{PDL}$ without split-junctions) then we can prove **NP**-membership. In other words, we have the following corollary where $\mathcal{PDL}^p$ denotes the Poor man's fragment of $\mathcal{PDL}$.

Corollary 3.33. *m*SubTeam *for $\mathcal{PDL}^p$ is* **NP***-complete.*

Proof. For membership, notice that one only needs to guess the subteam $T' \subseteq T$, and the question whether $T' \models \phi$ can be answered in polynomial time. The hardness follows since the formula in the proof of Theorem 3.32 does not contain any split-junction. □

CHAPTER 4

LOGICAL INFERENCE: ABDUCTION

In this chapter, we explore the parameterized complexity of abductive reasoning in Schaefer's framework. The problem of interest is to find an explanation for a manifestation (ABD). Moreover, we consider two of its size variants ($\text{ABD}_{\leq}$ and $\text{ABD}_{=}$) which impose size restriction for an explanation. We begin by proving several technical expressibility results that are helpful in the proof for this chapter as well as in Chapter 5. This is followed by the parameterized complexity analysis of abduction. At first, we prove general results in the classical and the parameterized setting. This is followed by the parameter-specific results in each separate subsection. We conclude the chapter by a discussion on the base independence (Property 2.1) in the parameterized setting for each considered problem. Figure 4.1 presents a complexity overview for problems in ABD. Throughout this chapter, the knowledge base (KB) is a conjunction of constraints over a fixed language, usually denoted by Γ.

4.1 Technical Implementation Results

We begin by presenting a wealth of technical expressivity results that allow us to prove the crucial property stated in Equation 2.1. The general results regarding the constraint languages are relevant for any problem in Schaefer's framework. These results are independent of the computational problems discussed in this work as they consider the expressibility of equality constraints. Specific results regarding the base independence for each problem are proven inside appropriate subsections.

Implementing Equality Constraints

In the following, we denote the classes of essentially positive and essentially negative languages by EP and EN, respectively. Recall that a relation R is EP (resp., EN) if R can be implemented by a

formula ϕ such that every clause in ϕ is either positive or unit negative (negative or unit positive). Furthermore, we say that a Boolean relation R is *strict essentially positive* denoted as EP^{s} (resp., *strict essentially negative*, EN^{s}) if it can be defined by a conjunction of literals and positive clauses (resp., negative clauses) alone. In similar manner, we say that a Boolean relation R is *strict positive* denoted as P^{s} (resp., *strict negative*, N^{s}) if it can be defined by a conjunction positive clauses (resp., negative clauses) only. Note that the only difference between a strict and its non-strict counterpart is the absence of the equality constraints (see Table 2.2).

The following proposition is a pre-requisite for the proof of Lemma 4.2. Its proof can be found in the work of Creignou et al. [22, 23].

Proposition 4.1. *([22]) Let Γ be a constraint language, then the following is true.*

1. *If Γ is complementive, but neither 1-valid, nor 0-valid, then $(x \neq y) \in \langle\Gamma\rangle_{\neq}$ [22, Lem. 4.6.1].*
2. *If Γ is neither complementive, nor 1-valid, nor 0-valid, then $(x \wedge \neg y) \in \langle\Gamma\rangle_{\neq}$ [22, Lem. 4.6.3].*
3. *If Γ is 1-valid and 0-valid but not trivial, then $(x = y) \in \langle\Gamma\rangle_{\neq}$ [22, Lem. 4.7].*
4. *If Γ is 1-valid, but neither 0-valid, nor essentially positive, then $(x = y) \in \langle\Gamma\rangle_{\neq}$ [22, Lem. 4.8.1].*
5. *If Γ is 0-valid, but neither 1-valid, nor essentially negative, then $(x = y) \in \langle\Gamma\rangle_{\neq}$ [22, Lem. 4.8.2].*

The underlying idea of the following lemma is to express equality by some other constraints.

Lemma 4.2. *Let Γ be a constraint language such that Γ is neither EP nor EN. Then, $(x = y) \in \langle\Gamma\rangle_{\neq}$ and $\langle\Gamma\rangle = \langle\Gamma\rangle_{\neq}$.*

Proof. For constraint languages that are Horn (IE_2) or dualHorn (IV_2) we use the following characterisations by polymorphisms (see, e.g. [29]). The binary operations of conjunction, disjunction and negation are applied coordinate-wise.

1. R is Horn if and only if $m_1, m_2 \in R$ implies $m_1 \wedge m_2 \in R$.
2. R is dualHorn if and only if $m_1, m_2 \in R$ implies $m_1 \vee m_2 \in R$.
3. R is essentially negative if and only if $m_1, m_2, m_3 \in R$ implies $m_1 \wedge (m_2 \vee \neg m_3) \in R$.
4. R is essentially positive if and only if $m_1, m_2, m_3 \in R$ implies $m_1 \vee (m_2 \wedge \neg m_3) \in R$.

To complete the proof of the lemma, we make a case distinction according to whether Γ is 1- or 0-valid.

1-valid and 0-valid. This case follows immediately from Prop. 4.1, 3rd item.

1-valid but not 0-valid. Follows immediately from Prop. 4.1, 4th item.

0-valid but not 1-valid. Follows immediately from Prop. 4.1, 5th item.

Neither 0-valid, nor 1-valid. We make another case distinction according to whether Γ is Horn and/or dualHorn.

Neither Horn, nor dualHorn. It suffices to show that we can express inequality $(x \neq y)$, since $(x = y) \equiv \exists z(x \neq z) \wedge (z \neq y)$. If Γ is complementive, we obtain by the 1st item in Prop. 4.1 that: $(x \neq y) \in \langle\Gamma\rangle_{\neq}$. Now suppose that Γ is not complementive. Since Γ is neither Horn, nor dualHorn, there are relations R and S such that R is not Horn and S is not dualHorn.

Since R is not Horn, there are $m_1, m_2 \in R$ such that $m_1 \wedge m_2 \notin R$. For $i, j \in \{0,1\}$, let $V_{i,j} = \{x \mid x \in V,\ m_1(x) = i,\ m_2(x) = j\}$. Observe that the sets $V_{0,1}$ and $V_{1,0}$ are both nonempty (otherwise $m_1 = m_1 \wedge m_2$ or $m_2 = m_1 \wedge m_2$, a contradiction). Let C denote the $\{R\}$-constraint $C = R(x_1, \ldots, x_k)$. Then, we let

$$M_1(u, x, y, v) := C[V_{0,0}/u, V_{0,1}/x, V_{1,0}/y, V_{1,1}/v].$$

The relation M_1 contains the tuples $\{0011, 0101\}$ (since $m_1, m_2 \in R$) but it does not contain 0001 (since $m_1 \wedge m_2 \notin R$).

Since S is not dualHorn, there are $m_3, m_4 \in S$ such that $m_3 \vee m_4 \notin S$. For $i, j \in \{0,1\}$, let $V'_{i,j} = \{x \mid x \in V,\ m_3(x) = i,\ m_4(x) = j\}$. Once again, the sets $V'_{0,1}$ and $V'_{1,0}$ are nonempty (otherwise $m_3 = m_3 \vee m_4$ or $m_4 = m_3 \vee m_4$, a contradiction). Let D denote the $\{S\}$-constraint $D = S(x_1, \ldots, x_k)$. Then, we let

$$M_2(u, x, y, v) := D[V'_{0,0}/u, V'_{0,1}/x, V'_{1,0}/y, V'_{1,1}/v].$$

The relation M_2 contains the tuples $\{0011, 0101\}$ (since $m_3, m_4 \in R$) but it does not contain 0111 (since $m_3 \vee m_4 \notin R$). Now, consider the following $\{R, S, (t \wedge \neg f)\}$-formula

$$M(f, x, y, t) := M_1(f, x, y, t) \wedge M_2(f, x, y, t) \wedge (t \wedge \neg f).$$

It is easy to verify that $M(f, x, y, t)$ is equivalent to $(x \neq y) \wedge (t \wedge \neg f)$. Due to Prop. 4.1, 2nd item, $(t \wedge \neg f)$ is expressible as a Γ-formula, and therefore so is $M(f, x, y, t)$. Finally, $\exists t \exists f M(f, x, y, t)$ is equivalent to $(x \neq y)$, and we obtain $(x \neq y) \in \langle\Gamma\rangle_{\neq}$.

Horn. Let R be a relation in Γ such that R is Horn but not EN. Then, there are $m_1, m_2, m_3 \in R$ such that $m_4 := m_1 \wedge (m_2 \vee \neg m_3) \notin R$. Moreover, since R is Horn, $m_5 := m_1 \wedge m_2 \in R$. For $i, j, k \in \{0,1\}$, let $V_{i,j,k} = \{x \mid x \in V,\ m_1(x) = i,\ m_2(x) = j,\ m_3(x) = k\}$. Observe that the sets $V_{1,0,0}$ and $V_{1,0,1}$ are nonempty (otherwise $m_5 = m_4$ or $m_1 = m_4$, a contradiction). Let C denote the $\{R\}$-constraint $C = R(x_1, \ldots, x_k)$. Then, we let

$$M(f, x, y, t) := C[V_{0,0,0}/f, V_{0,0,1}/f, V_{0,1,0}/f, V_{0,1,1}/f, V_{1,0,0}/x, V_{1,0,1}/y, V_{1,1,0}/t, V_{1,1,1}/t].$$

The relation M contains $\{00001111, 00000011\}$ (since $m_1, m_5 \in R$) but it does not contain

00001011 (since $m_4 \notin R$). Finally consider the $\{R, (t \wedge \neg f)$-formula

$$M'(f,x,y,t) := M(f,x,y,t) \wedge M(f,y,x,t) \wedge (t \wedge \neg f)$$

It is easy to verify that the relation M' contains $\{0111, 0001\}$ but neither 0101, nor 0011. Therefore $M'(f,x,y,t)$ is equivalent to $(x = y) \wedge (t \wedge \neg f)$. Due to Prop. 4.1, 2nd item, $(t \wedge \neg f)$ is expressible as a Γ-formula, and therefore so is $M'(f,x,y,t)$. Finally, $\exists t \exists f M'(f,x,y,t)$ is equivalent to $(x = y)$, and we obtain $(x = y) \in \langle\Gamma\rangle_{\neq}$.

dualHorn. This case is analogous to the Horn case. We use a relation R that is dualHorn but not EP.

This concludes all the cases and completes the proof of the lemma. □

The following implementation result strengthens Lemma 4.2 to certain essentially positive and essentially negative constraint languages.

Lemma 4.3. *Let Γ be a CL that is not ε-valid. If Γ is either* EP *but not* EP^{s}*, or* EN *but not* EN^{s}*, then we have that $(x = y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\not\equiv,\neq}$, as well as, $(x = y) \in \langle\Gamma\rangle_{\neq}$ and $\langle\Gamma\rangle = \langle\Gamma\rangle_{\neq}$.*

Proof. We prove the statement for a CL Γ that is EP but not EP^{s}. The other case can be treated analogously. Let R be the relation in Γ such that R is EP but not EP^{s}. Furthermore, R is neither 1-valid nor 0-valid. Let k be the arity of R and $V = \{x_1, \ldots, x_k\}$ be a set of k distinct variables. By the definition of EP (cf. IS_{02} in Table 2.2), R can be written as a conjunction of negative literals, positive clauses and equality constraints. If R can be written without any equality, then R is EP^{s}, a contradiction. As a result, any representation of R requires at least one equality constraint. Suppose, w.l.o.g., that $R(x_1, \ldots, x_k) \models (x_1 = x_2)$, while $R(x_1, \ldots, x_k) \not\models x_1$ and $R(x_1, \ldots, x_k) \not\models \neg x_1$. Define the following three subsets of V: $W = \{\, x_i \mid R(x_1, \ldots, x_k) \models (x_1 = x_i) \,\}$, $N = \{\, x_i \mid R(x_1, \ldots, x_k) \models \neg x_i \,\}$, and $P = V \setminus (W \cup N)$.

By construction, the three subsets provide a partition of V. Now, W is nonempty by our construction, N is nonempty since R is not 1-valid and P is nonempty since R is not 0-valid. Let C denote the $\{R\}$-constraint $C = R(x_1, \ldots, x_k)$ and let $M(x_1, x_2, t, f) := C[W/x_2, P/t, N/f]$. It is easy to verify that $M(x_1, x_2, t, f) \equiv (x_1 = x_2) \wedge t \wedge \neg f$.

Finally, observe that $(x = y) \equiv \exists t \exists f\, (x = y) \wedge t \wedge \neg f$. We conclude that $(x = y) \in \langle\Gamma\rangle_{\neq}$ and consequently $\langle\Gamma\rangle = \langle\Gamma\rangle_{\neq}$. □

We want to remark that the expressivity results proven in this subsection are independent of the problems we study. In other words, these results are not only applicable to problems in abduction and argumentation, but they are useful in the general setting of Scheafer's framework.

4.2 Abductive Reasoning

In this section, we consider the parameterized complexity of problems for abductive reasoning in Schaefer's framework. From this point on, (V, H, M, KB) denotes an instance of ABD(Γ), where

V is a set of variables, KB is a set of Γ-formulas and $H, M \subseteq V$. As before, we find it convenient to write an instance of the parameterized problem as (V, H, M, KB, k) where $\kappa(V, H, M, KB) = k$.

Problem:	p-ABD(Γ, κ)
Input:	(V, H, M, KB, k).
Parameter:	k.
Question:	Is there an explanation E for M in KB?

Additionally, we consider two size variants of p-ABD(Γ, κ). The problem p-ABD$_{\leq}(\Gamma, \kappa)$ (resp., p-ABD$_{=}(\Gamma, \kappa)$) ask whether there is an explanation E of size at most (exactly) s for some $s \in \mathbb{N}$. An instance to both these problems is then the tuple (V, H, M, KB, s, k) where k is the parameter value. Furthermore, for solution size $|E|$ as a parameter, we omit s (since $s = k$) and only write (V, H, M, KB, k) as an instance. Finally, we denote by ABD$_*$ any abduction problem under consideration, that is, ABD$_* \in \{\text{ABD}, \text{ABD}_{\leq}, \text{ABD}_{=}\}$.

A Note on Parameterizations

The knowledge-base (KB) in an instance of the abduction problem ABD$_*(\Gamma)$ is considered as a single Γ-formula. The meaningful parameters arising from an ABD-instance are $\kappa \in \{|V|, |H|, |M|, |E|\}$, that is, the cardinality of each set. Two further parameters, the treewidth and the backdoor-size[1] have also been considered by Gottlob et al. [48, Theorem 3.10] and Pfandler et al. [93], respectively.

One can also consider the *size* of the knowledge base KB (the encoding length) as another parameter. Nevertheless, we argue that it is enough to consider either the number of variables $|V|$, or the size of KB, since the two parameters are equivalent. Clearly, bounding the encoding length implies having limited space for encoding variables. Furthermore, if one bounds the number of variables, then one also has limited possibilities for defining *different* formulas. This is due to the reason that the constraint language Γ for our problems is fixed in advance. In the following, $|\Phi|$ denotes the number of distinct (up to logical equivalence) formulas, and enc(Φ) denotes the encoding of the formulas in Φ.

Lemma 4.4. *Let Γ be a fixed CL. Then for any set Φ of Γ-formulas over variables V, we have that*

1. $|\Phi| \leq 2^{p \cdot |V|^q}$ *for some constants* $p, q \in \mathbb{N}$,
2. $|V| \leq f(|\text{enc}(\Phi)|)$, *and* $|\text{enc}(\Phi)| \leq g(|V|)$ *for some computable functions f and g.*

Proof. (1). Let $\Gamma = \{R_1^{k_1}, \ldots, R_p^{k_p}\}$ be the constraint language such that each relation R_i has arity k_i for $i \leq p$. Moreover, let $V = \{x_1, \ldots, x_v\}$ be a collection of v distinct variables. For each relation R_i of arity k_i, the number of distinct R_i-constraints are at most $\binom{v}{k_i} = v^{k_i}$. This implies that the

[1] Size of the smallest Horn backdoor. We omit the detailed discussion on this parameter since this is not covered in this work. An interested reader is suggested to consult the cited paper.

total number of distinct Γ-constraints are atmost $p \cdot v^q$ where q is the maximum arity of any relation in Γ. A Γ-formula is simply a conjunction of constraint, and therefore, can be seen as a subset of Γ-constraints. This implies that there are at most $2^{p \cdot v^q}$-many different Γ-formulas. The result follows since the number and the arity of each relation in Γ is constant.

(2). We represent a variable x_i by its binary encoding. This implies that any reasonable encoding of variables requires $\log_2(|V|) + c$ bits for some constant c. In other words, $|\text{enc}(V)| = \log_2(|V|) + c$, or $|V| = 2^{d \cdot |\text{enc}(V)|}$ for some constant d. Moreover, $|\text{enc}(V)| \leq |\text{enc}(\Phi)|$ and therefore $|V| \leq 2^{d \cdot |\text{enc}(\Phi)|}$ is true.

We also know that $|\text{enc}(\Phi)| \leq |\Phi| \cdot |\text{enc}(\psi)|$ where ψ is such that $|\text{enc}(\phi)| \leq |\text{enc}(\psi)|$ for each $\phi \in \Phi$. Now, observe that $|\text{enc}(\psi)|$ is bounded from above by the size of the truth table for ψ, and therefore $|\text{enc}(\psi)| \leq 2^{c \cdot |\text{Vars}(\psi)|}$. This implies that $|\text{enc}(\Phi)| \leq |\Phi| \cdot 2^{c \cdot |\text{Vars}(\psi)|}$. By using (1) from above, we conclude that $|\text{enc}(\Phi)| \leq 2^{p \cdot |V|^q} \cdot 2^{c \cdot |V|}$, since $|\text{Vars}(\psi)| \leq |V|$ □

Notice that Lemma 4.4 is independent of the problems in this chapter. Therefore, we also use this when defining meaningful parameterizations in Chapter 5 for argumentation.

Base Independence for ABD

As stated before, the base independence yields generalized upper as well as lower bounds for certain CLs. The idea is to apply Lemma 4.2 and prove Property 2.1 for problems in abductive reasoning with respect to each parameterization. We first prove these results for each problem in the classical setting (Lem. 4.5, 4.6) and later strengthen them in the parametrized setting (Cor. 4.7).

Lemma 4.5. *Let* Γ *and* Γ′ *be two constraint languages such that* Γ′ *is neither essentially positive nor essentially negative. Let* $\text{ABD}_* \in \{\text{ABD}, \text{ABD}_=, \text{ABD}_\leq\}$. *If* $\Gamma \subseteq \langle \Gamma' \rangle$, *then* $\text{ABD}_*(\Gamma) \leq_m^{\mathbf{P}} \text{ABD}_*(\Gamma')$.

Proof. Let *KB* be a Γ-formula. We transform *KB* into an equivalent Γ′-formula by replacing every Γ-constraint with the corresponding Γ′-formula. This can be achieved by constructing a look-up table, which maps every $R \in \Gamma$ to an equivalent Γ′-formula, as illustrated next. Since Γ′ is neither essentially positive nor essentially negative, we have $\langle \Gamma' \rangle = \langle \Gamma' \rangle_{\neq}$ by Lemma 4.2 and $R \in \Gamma \subseteq \langle \Gamma' \rangle$ implies that $R \in \langle \Gamma' \rangle_{\neq}$. By definition, there is a Γ′-formula ψ_R such that $R(x_1, \ldots, x_n) \equiv \exists y_1 \ldots \exists y_m \psi_R(x_1, \ldots, x_n, y_1, \ldots, y_m)$, where x_i's and y_i's are $n + m$ distinct variables. Finally, we remove the existential quantifiers and map R to the formula ψ_R. Note that the look-up table can be constructed in the constant time, since Γ is a fixed and finite CL which is independent of the input instance. We are now ready to transform *KB* into an appropriate Γ′-formula by applying the following replacement procedure as long as applicable.

- Let $C_R := R(x_1, \ldots, x_n)$ be a Γ-constraint in *KB* (where x_i's are not necessarily n distinct variables). Replace C_R by its corresponding Γ′-formula $\psi_R(x_1, \ldots, x_n, y_1, \ldots, y_m)$, where $y_1, \ldots, y_m$ are fresh variables and unique to C_R (they are not used for any other constraint).

This transformation introduces additional variables. We show that the total number of such variables is bounded polynomially in the input. Let m_R be the number of y_i's added while

replacing C_R (denoted m in the above procedure). It is easy to observes that the total number of additional variables is bounded by the number of constraints in KB times the maximum of all m_R. Since m_R is only dependent on R, it is constant. Moreover, Γ is finite, and therefore, the maximum over all m_R's is also constant. We conclude that this transformation can be achieved in polynomial time.

Furthermore, observe that an $\mathrm{ABD}_*(\Gamma')$-instance after this transformation has exact same solutions as the original $\mathrm{ABD}_*(\Gamma)$-instance. □

In the following lemma, we prove the base independence for $\mathrm{ABD}_*(\Gamma)$ when Γ is essentially positive. The proof idea is to remove the equality constraints while maintaining the size count and preserving the solution of instances.

Lemma 4.6. *Let Γ, Γ' be two constraint languages such that Γ' is essentially positive. Let $\mathrm{ABD}_* \in \{\mathrm{ABD}, \mathrm{ABD}_=, \mathrm{ABD}_\leq\}$. If $\Gamma \subseteq \langle\Gamma'\rangle$, then $\mathrm{ABD}_*(\Gamma) \leq_m^{\mathbf{P}} \mathrm{ABD}_*(\Gamma')$.*

Proof. The case for ABD is due to Nordh and Zanuttini [89, Lemma 22]. We first prove the case of $\mathrm{ABD}_\leq$ in the following. Let (V, H, M, KB, s) be an $\mathrm{ABD}_\leq(\Gamma)$-instance and (V', H', M', KB', s) be an instance where KB' is obtained from KB by replacing every constraint by its equivalent Γ'-formula and removing equality constraints (as well as duplicate variables) similar to the proof of Lemma 2.17. Notice that removing the equality constraints and deleting the duplicate occurrences of variables can only decrease the size of H and M. This implies that $|M'| \leq |M|$ and $|H'| \leq H$. Finally, the result follows by observing that (V, H, M, KB, s) has a solution E of size at most s if and only if E is also a solution for (V', H', M', KB', s).

Now we proceed with proving the case for $\mathrm{ABD}_=$. We show that for any $\mathrm{ABD}_=(\Gamma \cup \{=\})$ instance (V, H, M, KB, s), there is an $\mathrm{ABD}_=(\Gamma)$-instance (V_1, H_1, M_1, KB_1, s) such that the former has an explanation if and only if the latter has one. The proof uses the fact that the negative clauses in KB are the unary clauses alone (of length one). Notice that the existence of a solution is invariant under the equality clauses (the case of ABD). As a result, we only assure that the size of a solution is also preserved after the transformation. To obtain (V_1, H_1, M_1, KB_1, s), we do the following for each clause $(x_i = x_j) \in KB$.

Case 1. If at most one of x_i and x_j appears in H, then remove $(x_i = x_j)$ from KB, replace x_j by x_i everywhere in $KB \cup H \cup V \cup M$ (and delete x_j).

Case 2. If both x_i, x_j are from H. Then,

1. if $\neg x_i$ (resp., $\neg x_j$) appears in KB, we add $\neg x_j$ (resp., $\neg x_i$) to KB and remove the clause $x_i = x_j$ from KB.
2. otherwise $\neg x_i, \neg x_j \notin KB$ and we simply remove $(x_i = x_j)$ from KB and do not remove any variable.

The problem caused by the equality clauses is the following. If we remove a variable x such that $x \in H$, then removing x from H, owing to some equality constraint, may not preserve the size

of the solutions. Furthermore, this problem occurs only when an equality clause contains both variables from H (Case 2.) since the size of H is not changed otherwise (Case 1.). We prove the following correspondence between the solutions of the two instances.

Claim 4.1. *A subset $E \subseteq H$ is an explanation for (V, H, M, KB, s) if and only if E is an explanation for (V_1, H_1, M_1, KB_1, s).*

Proof of Claim. *"$\Rightarrow$": Let E be an explanation for (V, H, M, KB, s). Since $E \wedge KB$ is consistent, we prove that $E \wedge KB_1$ is also consistent. Note that $KB_1 \subseteq KB$, except if $\neg x_j \in KB_1$ for some x_j. This implies that $x_j \notin E$ due to the reason that $x_i = x_j$ and $\neg x_i \in KB$. Finally, $M_1 \subseteq M$ and E is an explanation for M implies that E is also an explanation for M_1.*

"$\Leftarrow$": Suppose that E is an explanation for (V_1, H_1, M_1, KB_1, s). Since $E \wedge KB_1$ is consistent, let θ be a satisfying assignment. We consider each equality constraint separately and prove that $E \wedge KB$ is consistent. If Case 1 applies for $(x_i = x_j)$, then at most one (say x_i) appears in H. Furthermore, if $x_i \in E \subseteq H$ then $\neg x_j \notin KB_1$ since this would imply that $\neg x_i \in KB_1$ and $x_i \notin E$. Consequently, $E \wedge KB_1 \wedge (x_i = x_j)$ is consistent (extending θ to $\theta(x_i) = 1 = \theta(x_j)$ satisfies it). On the other hand, if $\neg x_j \in KB_1$ then $\neg x_i \in KB_1$ and $x_i \notin E$. As a result, θ extended by $\theta(x_i) = 0 = \theta(x_j)$ is a satisfying assignment. If Case 2 applies for $(x_i = x_j)$, that is, both $x_i, x_j \in H$ then we again have two sub-cases based on whether $\neg x_i \in KB_1$ or not. If $\neg x_i \in KB_1$, then due to the sub-case 1, we have that $\neg x_j \in KB_1$ and this implies that $x_i \notin E$. As a consequence, $E \wedge KB_1 \wedge (x_i = x_j)$ is consistent (extending θ to $\theta(x_i) = 0 = \theta(x_j)$ satisfies it). In the sub-case 2, when both $x_i, x_j \in H$ and $\neg x_i, \neg x_j \notin KB_1$ then mapping $\theta(x_i) = 1 = \theta(x_j)$ satisfies $E \wedge KB_1 \wedge (x_i = x_j)$. This is because all the non-unit clauses in KB_1 are positive. Finally, since this is true for all the equality clauses, $E \wedge KB$ is consistent.

For entailment, suppose that for some $m_i, m_j \in M$, we have that $(m_i = m_j) \in KB$ and $m_j \notin M_1$, that is $M_1 \subsetneq M$. Since $E \wedge KB_1$ is consistent and entails M_1, we have $E \wedge KB_1 \wedge (m_i = m_j)$ is also consistent (due to arguments for consistency) and entails $M_1 \cup \{m_j\}$. Moreover, this is true for every pair $m_i, m_j \in M$ such that $(m_i = m_j) \in KB$. This completes the proof in this direction and settles the claim. ■

Finally, the presented reduction can be computed in polynomial time because both steps in the transformation are applied once for each equality clause, and each step takes polynomial time. This shows the desired reduction between $\text{ABD}_{=}(\Gamma \cup \{=\})$ and $\text{ABD}_{=}(\Gamma)$. □

Notice that Lemmas 4.5 and 4.6 are stated with respect to the classical (unparameterized) decision problems. However, these reductions can be generalized to $\leq^{\mathbf{FPT}}$-reductions whenever the parameters are bounded as required by Definition 2.10. That is, in our case, for any parameterization $\kappa \in \{\, |H|, |E|, |M| \,\}$ the reductions are valid. Even more, the values of the parameters remain same as the sizes of H, E, and M remain unchanged in the reduction.

Corollary 4.7. *Let Γ, Γ' be any two constraint languages except for essentially negative ones. Let $\text{ABD}_* \in \{\text{ABD}, \text{ABD}_{=}, \text{ABD}_{\leq}\}$. If $\Gamma \subseteq \langle \Gamma' \rangle$, then $\text{p-ABD}_*(\Gamma, \kappa) \leq^{\mathbf{FPT}} \text{p-ABD}_*(\Gamma', \kappa)$ for any $\kappa \in \{|H|, |E|, |M|\}$.*

It is cumbersome to mention the base independence in almost every proof. As a result, we omit this reference and show the results only for concrete bases implicitly using Corollary 4.7. When we deal with essentially negative constraint languages, we do not have a general result about the base independence. In these cases we have direct constructions showing membership and hardness for all bases (e.g., Lemmas 4.15 and 4.25).

4.2.1 General Complexity Results

In this subsection, we prove general observations and reductions between defined problems. The following two results regarding the classical problems relate the complexity of our problems $\mathrm{ABD}_* \in \{\mathrm{ABD}, \mathrm{ABD}_{\leq}, \mathrm{ABD}_{=}\}$ to each other.

Lemma 4.8. *For every constraint language* Γ*, we have that* $\mathrm{ABD}(\Gamma) \leq_m^{\mathbf{P}} \mathrm{ABD}_{\leq}(\Gamma)$.

Proof. Clearly, an instance (V, H, M, KB) of $\mathrm{ABD}(\Gamma)$ has an explanation if and only if the instance (V, H, M, KB, s) of $\mathrm{ABD}_{\leq}(\Gamma)$ has one, where $s = |H|$. In other words, an ABD-instance has an explanation if and only if it has one with size at most that of the hypotheses set. □

We can relate the complexity of $\mathrm{ABD}_{\leq}$ with that of $\mathrm{ABD}_{=}$ via a Turing reduction (using the concept of oracles) between the two problems. Clearly, an instance (V, H, M, KB, s) of $\mathrm{ABD}_{\leq}(\Gamma)$ has a solution of size at most s iff there is some $i \leq s$ such that the instance (V, H, M, KB, i) of $\mathrm{ABD}_{=}(\Gamma)$ has a solution of size exactly i. Such a reduction is indeed interesting but it does not help in transferring the hardness results from $\mathrm{ABD}_{\leq}$ to $\mathrm{ABD}_{=}$.

In the following, we relate the complexity of ABD with that of $\mathrm{ABD}_{=}$. The idea (presented by Fellows et al. [43, Cor., 16]) is to add additional hypotheses which then help to assure the size restriction for an explanation.

Lemma 4.9. *Let* Γ *be a constraint language such that* $\langle\Gamma\rangle \supseteq \mathsf{IE}$*, then we have that* $\mathrm{ABD}(\Gamma) \leq_m^{\mathbf{P}} \mathrm{ABD}_{\leq}(\Gamma)$.

Proof. Let (V, H, M, KB) be an ABD-instance where $H = \{h_1, \ldots, h_p\}$. Moreover, let $H' = \{h'_1, \ldots, h'_p\}$, $M' = \{m'_1, \ldots, m'_p\}$ be collections of fresh variables, $V' = H' \cup M'$, and $KB' = \bigwedge_{i \leq p} (\neg h_i \vee \neg h'_i) \wedge (\neg h_i \vee m_i) \wedge (\neg h'_i \vee m'_i)$. Then we construct the $\mathrm{ABD}_{=}$-instance $(V \cup V', H \cup H', M \cup M', KB \wedge KB', s)$ where $s = |H| = p$. Clearly, (V, H, M, KB) admits an explanation $E \subseteq H$ iff $E \cup \{h'_i \mid h'_i \in H' \text{ and } h_i \notin E\}$ is an explanation for $(V \cup V', H \cup H', M \cup M', KB \wedge KB', s)$. □

Notice that the reduction in Lemma 4.9 is only useful for certain CLs (when Horn clauses can be represented). We wish to point that we use Lemma 4.9 for only those CLs Γ, such that $\langle\Gamma\rangle = \mathsf{BR}$ (see Theorems 4.11 and 4.28). As a result, the restriction that 'Γ must implement a Horn clause' is irrelevant. Moreover, when proving results for $|H|$ (Theorem 4.11) the set H has size already one and therefore the reduction does not increase the parameter value. However, H is not expected to have fixed size when proving results for $|M|$ (Theorem 4.28). Nevertheless, we use the fact that the set M' can be simulated by a single fresh variable m'. This is achieved by

adding the formula $(m' \leftrightarrow \bigwedge_{i \leq |M'|} m'_i)$ to KB. Consequently, the parameter value $|M|$ increases by one due to the reduction between ABD and $\text{ABD}_=$ (in Theorem 4.28).

Now, we proceed by proving the equivalence of $\text{ABD}_\leq$ and $\text{ABD}_=$ for dualHorn languages.

Lemma 4.10. *For every constraint language* Γ *such that* $\mathsf{IBF} \subseteq \langle\Gamma\rangle \subseteq \mathsf{IV}_2$*, we have that* $\text{ABD}_\leq(\Gamma) \equiv_m^{\mathbf{P}} \text{ABD}_=(\Gamma)$.

Proof. "$\leq_m^{\mathbf{P}}$": We claim that one can check in polynomial time whether a positive instance (V, H, M, KB, s) of $\text{ABD}_\leq(\Gamma)$ has a solution E of size exactly s. Given a solution E such that $|E| \leq s$, then a solution of size exactly s can be constructed (in polynomial time w.r.t. $|H|$) by adding one element h at a time from H to E and checking that $\neg h \notin KB$. Moreover, if there are no such elements in H then clearly there can be no solution of size exactly s.

"$\geq_m^{\mathbf{P}}$": Every solution of size exactly s is a solution of size at most s. □

Intractable cases

Interestingly, for 0-valid, 1-valid, and complementive languages, all three problems remain intractable under any parametrization except for the case of $|V|$.

Theorem 4.11. *The problems* $\text{p-ABD}(\Gamma, \kappa)$, $\text{p-ABD}_\leq(\Gamma, \kappa)$ *and* $\text{p-ABD}_=(\Gamma, \kappa)$ *are*

1. **paraCoNP**-*hard if* $\mathsf{IN} \subseteq \langle\Gamma\rangle \subseteq \mathsf{II}_1$ *and* $\kappa \in \{\, |H|, |E|, |M| \,\}$,
2. **paraDP**-*hard if* $\mathsf{C} \subseteq \langle\Gamma\rangle \subseteq \mathsf{BR}$ *and* $\mathsf{C} \in \{\mathsf{IN}_2, \mathsf{II}_0\}$ *and* $\kappa \in \{|H|, |E|\}$,
3. **para**$\mathbf{\Sigma}_2^P$-*hard if* $\mathsf{C} \subseteq \langle\Gamma\rangle \subseteq \mathsf{BR}$ *and* $\kappa = |M|$ *for* $\mathsf{C} \in \{\mathsf{IN}_2, \mathsf{II}_0\}$.

Proof. (1). We prove **paraCoNP**-hardness of $\text{p-ABD}(\Gamma, \kappa)$ when $\mathsf{IN} \subseteq \langle\Gamma\rangle$ regarding all three parameters simultaneously. Notice that $\text{IMP}(\Gamma)$ is **CoNP**-hard when $\langle\Gamma\rangle \subseteq \mathsf{II}_1$ [89, Thm. 34], even if the right side contains only a single variable. We describe in the following a modified proof from [89, Prop. 48]. Since $\langle\mathsf{IN} \cup \{\, T \,\}\rangle = \mathsf{II}_1$ (where $T = \{1\}$), we have that $\text{IMP}(\mathsf{IN} \cup \{\, T \,\})$ is **CoNP**-hard, even if the right side contains only a single variable. We reduce $\text{IMP}(\mathsf{IN} \cup \{\, T \,\})$ to our abduction problems with $|H| = 1$, $|M| = 1$, and $|E| = 1$. Let (Φ_T, q) be an instance of $\text{IMP}(\mathsf{IN} \cup \{\, T \,\})$, where $\Phi_T = \Phi \wedge \bigwedge_{x \in V_T} T(x)$ and Φ is an IN-formula. We map (Φ_T, q) to an instance $(V, \{h\}, \{q\}, KB)$ of $\text{ABD}(\Gamma)$, where $V = \text{Vars}(\Phi) \cup \{h\}$, h is a fresh variable, and KB is obtained from Φ by replacing every variable from V_T by h. Note that $\Phi_T \equiv KB \wedge h$. Since Φ and KB are 1-valid $KB \wedge h$ is always satisfiable and there exists an explanation iff $KB \wedge h \models q$, iff $\Phi_T \models q$. Furthermore, observe that $\Phi_T \models q$ if and only if $(V, \{h\}, \{q\}, KB, |H|) \in \text{p-ABD}(\Gamma, |H|)$ if and only if $(V, \{h\}, \{q\}, KB, 1, |H|) \in \text{p-ABD}_\leq(\Gamma, |H|)$ if and only if $(V, \{h\}, \{q\}, KB, 1, |H|) \in \text{p-ABD}_=(\Gamma, |H|)$, where $\langle\Gamma\rangle = \mathsf{IN}$. Finally, the equivalence is also true when $|H|$ is replaced by $|E|$ or $|M|$ since $|E| \leq |H|$ and $M = q$. This proves the claimed **paraCoNP**-hardnesses by noting that $|E| = |H| = |M| = 1$, in each case.

(2). We prove **paraDP**-hardness for IN_2 and II_0 regarding both parameters simeltenously. From Fellows et al. [43, Prop. 4] we know that $\text{ABD}_*(\Gamma)$ for $\langle\Gamma\rangle \subseteq \mathsf{BR}$ is **DP**-complete, even if $|H| = 0$

and $|M| = 1$. We argue that the hardness can be extended to a CL Γ such that $\langle\Gamma\rangle = \mathsf{IN}_2$. Note that $\langle \mathsf{IN}_2 \cup \{F\}\rangle = \mathsf{BR}$ where $F = \{0\}$. Creignou & Zanuttini [30, Lem. 21] proved that $\mathrm{ABD}(\Gamma \cup \{ F \}) \leq_m^{\mathbf{P}} \mathrm{ABD}(\Gamma \cup \{ \mathtt{SymOR}_{2,1} \})$ where $\mathtt{SymOR}_{2,1}(x,y,z) = ((x \to y) \wedge T(z)) \vee ((y \to x) \wedge F(z))$. Moreover, they also prove that $\mathtt{SymOR}_{2,1} \in \langle\Gamma\rangle$ if $\mathsf{IN}_2 \subseteq \langle\Gamma\rangle$ [30, Lem. 27]. Finally, having $|M| = 1$ allows us to use their proof ([30, Lem. 21]) and, as a consequence, $\mathrm{ABD}(\Gamma \cup \{ F \}) \leq_m^{\mathbf{P}} \mathrm{ABD}(\Gamma)$ such that $\langle\Gamma\rangle = \mathsf{IN}_2$. Regarding II_0, the proof follows by a similar argument using the observations that $\langle \mathsf{II}_0 \cup \{ T \}\rangle = \mathsf{BR}$ and $\mathtt{OR}_{2,1} \in \langle\Gamma\rangle$ such that $\mathsf{II}_0 \subseteq \langle\Gamma\rangle$ where $\mathtt{OR}_{2,1}(x,y) = x \to y$ [30, Lem. 19/27]. The desired **paraDP**-hardness follows by noting that $|E| = |H| = 0$ in each case.

(3). Nordh and Zanuttini [89, Prop. 46/47] proved that the problem $\mathrm{ABD}(\Gamma)$ with positive literal manifestations is $\mathbf{\Sigma}_2^P$-hard if $\langle\Gamma\rangle \subseteq \mathsf{C}$ and $\mathsf{C} \in \{ \mathsf{IN}_2, II_0 \}$. This implies that the 1-slice of p-$\mathrm{ABD}(\Gamma, |M|)$ is $\mathbf{\Sigma}_2^P$-hard in each case, which gives the desired result. For $\mathrm{ABD}_{\leq}(\Gamma, |M|)$ and $\mathrm{ABD}_{=}(\Gamma, |M|)$, the results follow from Lemmas 4.8 and 4.9. □

Notice that the **para$\mathbf{\Sigma}_2^P$**-hardness from Theorem 4.11 can be strengthened to completeness. This is because, guessing an explanation for an instance of each $\mathrm{ABD}_* \in \{\mathrm{ABD}, \mathrm{ABD}_{\leq}, \mathrm{ABD}_{=}\}$ takes nondeterministic time and the verification can be done by an **NP**-oracle.

Fixed-parameter tractable cases

The following corollary is immediate because ABD corresponding to these cases is in **P** due to Nordh and Zanuttini [89].

Corollary 4.12. *The problem* p-$\mathrm{ABD}(\Gamma, \kappa)$ *is* **FPT** *for any parameterization* $\kappa \in \{|H|, |M|\}$ *and* $\langle\Gamma\rangle \subseteq \mathsf{C}$ *with* $\mathsf{C} \in \{ \mathsf{IV}_2, \mathsf{ID}_1, \mathsf{IE}_1, \mathsf{IS}_{12} \}$.

The result for each case when parameterized by $|V|$ is already due to Fellows et al. [43, Prop. 13].

Corollary 4.13. *The problems* p-$\mathrm{ABD}(\Gamma, |V|)$, p-$\mathrm{ABD}_{\leq}(\Gamma, |V|)$, p-$\mathrm{ABD}_{=}(\Gamma, |V|)$ *are all* **FPT** *for all Boolean constraint languages* Γ.

Now, we prove **P**-membership of $\mathrm{ABD}_s(\Gamma)$ for $s \in \{ \leq, = \}$ and start with the case when Γ is essentially positive. The idea is to first apply unit propagation. The positive clauses (after unit propagation) do not explain anything, and one only has to check for each $m \in M$ whether $m \in KB \cup H$. Then, we need to adjust the size accordingly.

Lemma 4.14. *The classical problems* $\mathrm{ABD}_{=}(\Gamma)$ *and* $\mathrm{ABD}_{\leq}(\Gamma)$ *are in* **P** *if* $\langle\Gamma\rangle \subseteq \mathsf{IS}_{02}$.

Proof. We only prove the claim for $\mathrm{ABD}_{=}(\Gamma)$, whereas, the result for $\mathrm{ABD}_{\leq}(\Gamma)$ follows from Lemma 4.10. Let (V, H, M, KB, s) be an instance of $\mathrm{ABD}_{=}(\Gamma)$ where $\Gamma \subseteq \mathsf{IS}_{02}$. Let H', M', KB' denote the result of applying the unit propagation on each literal y such that $y \in \mathrm{Lit}(KB) \setminus (H^+ \cup M^-)$. Recall that for a set Y of literals, Y^+ (resp., Y^-) denotes the set of positive (negative) literals formed upon Y. In unit propagation, for each literal u, any clause containing u is deleted and, $\sim u$ from any clause is deleted, where $\sim u = x$ if $u = \neg x$ is a negative literal and $\sim u = \neg x$ if $u = x$ is positive. Note that each literal $y \in H^+ \cup M^-$ (that is, $y \in H$ or $y = \neg m$ for $m \in M$) is

excluded from this rule. The reason for this choice is as follows. If $\neg m \in KB$ for some $m \in M$ then removing m from $KB \cup M$ transforms a 'no solution'- to a 'yes solution'-instance. Similarly, removing an $h \in H$ from $KB \cup H$ may result in decreasing the solution size of an instance. Finally, the positive literal $m \in M$ may or may not be processed. However, it is important to consider $h \in H^-$ since this helps in invalidating the clauses of length ≥ 2.

Let P and N denote the positive, respectively negative unit clauses of KB'. Note that if $N \neq \emptyset$ then there can be no explanation for M. This is due to the reason that only negative unprocessed literals are over M, implying that $KB \wedge M$ is inconsistent. Consequently, we have $N = \emptyset$. Moreover, positive clauses of length ≥ 2 in KB' do not explain anything as a variable cannot be forced to 0 by an explanation E. Therefore, a positive literal x cannot explain anything more than x itself. This implies that: there is an explanation for M if and only if $M' \subseteq H' \cup P$. The set $M' \setminus P$ consists of those $m \in M$ which are not already explained by KB and must be explained by H'. As a consequence, there exists an explanation for $\mathrm{ABD}_{\leq}(\Gamma)$ if and only if $M' \setminus P \subseteq H'$ and $|M' \setminus P| \leq s$. The consistency is assured already by the fact that $N = \emptyset$. Finally, to determine whether there is an explanation $E \subseteq H$ of size s, it suffices to check additionally whether $|H'| \geq s$. This argument assures whether we can artificially increase the solution size since, in that case, any $E \subseteq H'$ with the above conditions constitutes an explanation for $\mathrm{ABD}_{=}(\Gamma)$. Moreover, if $|H'| < s$, then no explanation of size s exists. The unit propagation and the size comparisons can be achieved in polynomial time. This proves the claim. □

The following lemma proves that $\mathrm{ABD}_{\leq}(\Gamma)$ also remains tractable when Γ is EN.

Lemma 4.15. *The classical problem* $\mathrm{ABD}_{\leq}(\Gamma)$ *is in* **P** *if* $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}$.

Proof. First, we prove the result with respect to $\langle\Gamma\rangle_{\neq} \subseteq \mathsf{IS}_{12}$ (that is, without base independence). Let P denote the set of positive unit clauses in KB and let $E_{MP} = M \setminus P$. Now, we have the following two observations.

Observation 1 There exists an explanation iff $E_{MP} \subseteq H$ and $KB \wedge M$ is consistent. In other words, what is not yet explained by P must be explainable directly by H because negative clauses can not contribute to explaining anything, they can only contribute to 'rule out' certain subsets of H as possible explanations.

Observation 2 If E is an explanation for M in KB, then $E \supseteq E_{MP}$.

As a result, E_{MP} represents a cardinality-minimal and a subset-minimal explanation. We conclude that there exists an explanation E with $|E| \leq s$ iff E_{MP} constitutes an explanation and $|E_{MP}| \leq s$. Now, we proceed with base independence for this case.

Claim 4.2. $\mathrm{ABD}_{\leq}(\Gamma \cup \{=\}) \leq_m^{\mathbf{P}} \mathrm{ABD}_{\leq}(\Gamma)$ *for* $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}$.

Proof of Claim. *We remove equality clauses and delete the duplicating occurrences of variables. This only decreases the size of H and possibly the size of an explanation E as well. It is important to notice that a clause $(x = y) \in KB$ does not enforce both x and y into E.* ■

This completes the proof of the lemma. □

Finally, $\mathrm{ABD}_{=}(\Gamma)$ and $\mathrm{ABD}_{\leq}(\Gamma)$ are also tractable when Γ is 2-affine, as we prove in the following lemma. The proof idea is, similar to Creignou et al. [26, Prop. 1], to change the representation of the knowledge base.

Lemma 4.16. *The classical problems* $\mathrm{ABD}_{=}(\Gamma)$ *and* $\mathrm{ABD}_{\leq}(\Gamma)$ *are in* **P** *if* $\langle\Gamma\rangle \subseteq \mathsf{ID}_1$.

Proof. Analogously to Creignou et al. [26, Prop. 1], we change the representation of the *KB*. Without loss of generality, suppose *KB* is satisfiable and contains no unit clauses since unit clauses can be dealt with by unit propagation. Each clause in Γ expresses either equality or inequality between two variables. With the transitivity of the equality relation and the fact that (in the Boolean case) $a \neq b \neq c$ implies $a = c$, we can identify equivalence classes of variables such that every two classes are either independent or they must have contrary truth values. We call a pair of dependent equivalence classes (X, Y) a *cluster* (X and Y must take contrary truth values). Denote by $X_1, \ldots, X_p$ the equivalence classes that contain variables from M such that $X_i \cap M \neq \emptyset$. Denote by $Y_1, \ldots, Y_p$ the equivalence classes such that for each i the pair (X_i, Y_i) represents a cluster. We make the following stepwise observations.

1. There is an explanation iff $\forall i : H \cap X_i \neq \emptyset$.
2. The size of a minimal explanation (E_{min}) is p. E_{min} is constructed by taking exactly one representative from each X_i.
3. There exists an explanation of size $\leq s$ iff $p \leq s$.
4. An explanation of maximal size (E_{max}) can be constructed as follows:
   (a) Begin by setting $E := \emptyset$,
   (b) for each i add to E all variables from $X_i \cap H$,
   (c) for each cluster $(X, Y) \notin \{(X_i, Y_i) \mid 1 \leq i \leq p\}$:
      i. if $|X \cap H| \geq |Y \cap H|$: add to E the set $X \cap H$,
      ii. else: add to E the set $Y \cap H$.
5. Any explanation of size between $|E_{min}|$ and $|E_{max}|$ can be constructed.
6. There is an explanation of size $= s$ iff $|E_{min}| \leq s \leq |E_{max}|$.

This completes the proof. □

Lemmas 4.14–4.16 yield the following corollary.

Corollary 4.17. *The following problems are* **FPT** *for any* $\kappa \in \{\, |H|, |E|, |M| \,\}$.

1. $\text{p-}\mathrm{ABD}_{=}(\Gamma, \kappa)$ *if* $\langle\Gamma\rangle \subseteq C$ *for* $\mathsf{C} \in \{\, \mathsf{IS}_{02}, \mathsf{ID}_1 \,\}$,
2. $\text{p-}\mathrm{ABD}_{\leq}(\Gamma, \kappa)$ *if* $\langle\Gamma\rangle \subseteq C$ *for* $\mathsf{C} \in \{\, \mathsf{IS}_{02}, \mathsf{ID}_1, \mathsf{IS}_{12} \,\}$.

Now we prove complexity results for the considered problems under each parameterization.

4.2.2 Parameter 'number of hypotheses' $|H|$

When parameterized by $|H|$, the only intractable cases are those pointed out in Theorem 4.11.

Theorem 4.18. p-ABD$(\Gamma, |H|)$, p-ABD$_{\leq}(\Gamma, |H|)$ *and* p-ABD$_{=}(\Gamma, |H|)$ *are*

1. **paraDP**-*hard if* $\mathsf{C} \subseteq \langle\Gamma\rangle \subseteq \mathsf{BR}$, *where* $\mathsf{C} \in \{\mathsf{IN}_2, \mathsf{II}_0\}$,
2. **paraCoNP**-*hard if* $\mathsf{IN} \subseteq \langle\Gamma\rangle \subseteq \mathsf{II}_1$,
3. **FPT** *if* $\langle\Gamma\rangle \subseteq \mathsf{C} \in \{\, \mathsf{IE}_2, \mathsf{IV}_2, \mathsf{ID}_2, \mathsf{IL}_2 \,\}$.

Proof. (1+2). Follows from Theorem 4.11.

(3). Recall that SAT(Γ) and IMP(Γ) are both in **P** for every Γ in the question (cf. [103, 104]). Note that $|H| \geq |E|$, and $\binom{|H|}{|E|} = |H|^{|E|} \in \mathcal{O}(k^k)$, where $k = |H|$. Consequently, one can brute-force the candidates for E and verify them in polynomial time. This yields **FPT** membership. □

What addition can make ABD **tractable along with** $|H|$ **as a parameter?** It is worth stopping here and asking ourselves: is there some parameterization κ, such that ABD$(\Gamma, \kappa + |H|)$ is tractable for every CL Γ? This question is answered by Pfandler et al. [93], where authors prove that the size of the smallest (Horn) backdoor is one such parameter. One of the main results by the authors is that ABD(CNF) is **FPT** when parameterized by $|H|$ and backdoor-size.

4.2.3 Parameter 'number of explanations' $|E|$

In this subsection, we consider the solution size as a parameter. Notice first that the parameter $|E|$ for problem p-ABD is not meaningful anymore since an input instance does not include the solution size. As a result, we only consider the two size-variants ABD$_{=}$ and ABD$_{\leq}$. The following theorem classifies both problems into six different complexity degrees.

Theorem 4.19. *The problems* p-ABD$_{\leq}(\Gamma, |E|)$ *and* p-ABD$_{=}(\Gamma, |E|)$ *are*

1. **paraDP**-*hard if* $\mathsf{C} \subseteq \langle\Gamma\rangle \subseteq \mathsf{BR}$, *where* $\mathsf{C} \in \{\mathsf{IN}_2, \mathsf{II}_0\}$,
2. **paraCoNP**-*hard if* $\mathsf{IN} \subseteq \langle\Gamma\rangle \subseteq \mathsf{II}_1$,
3. $\mathbf{W}[\mathbf{P}]$-*complete if* $\mathsf{IE} \subseteq \langle\Gamma\rangle \subseteq \mathsf{IE}_2$,
4. $\mathbf{W}[2]$-*complete if* $\mathsf{IM} \subseteq \langle\Gamma\rangle \subseteq \mathsf{C}$ *for* $\mathsf{C} \in \{\mathsf{ID}_2, \mathsf{IS}^{\ell}_{10}, \mathsf{IV}_2\}$,
5. **FPT** *if* $\langle\Gamma\rangle \subseteq \mathsf{ID}_1$ *or* $\langle\Gamma\rangle \subseteq \mathsf{IS}_{02}$,

Moreover, if $\mathsf{IS}^2_1 \subseteq \langle\Gamma\rangle \subseteq \mathsf{IS}_{12}$, *then* p-ABD$_{\leq}(\Gamma, |E|)$ *is* **FPT** *and* p-ABD$_{=}(\Gamma, |E|)$ *is* $\mathbf{W}[1]$-*complete.*

Proof. (1+2). Follows from Theorem 4.11.

(3). The $\mathbf{W}[\mathbf{P}]$-membership follows from the fact that SAT(Γ) and IMP(Γ) are both in **P** when $\langle\Gamma\rangle \subseteq \mathsf{IE}_2$ (cf. [103, 104]). Guessing E takes $k \cdot \log n$ non-deterministic steps, and the verification

takes polynomial time. For the lower bound, we argue that the proof by Fellows et al. [43, Cor. 9] for definite Horn theories (IE_1) can be adapted. The only types of clauses used in the presented reduction are $(x \wedge y \rightarrow z)$ and $(x \rightarrow y)$, which are both expressible by Γ if $\mathsf{IE} \subseteq \langle\Gamma\rangle$. Finally, the membership and the hardness arguments are valid for $\text{ABD}_{\leq}(\Gamma, |E|)$ as well since the problem used for the reduction in [43, Cor. 9] is the Monotone Circuit SAT (which is *monotone*).

(4). The $\mathbf{W}[2]$-completeness for $\text{p-ABD}_{=}(\Gamma, |E|)$ such that $\langle\Gamma\rangle = \mathsf{IM}$ is proven in Lemma 4.20. Then Lemma 4.21 strengthens this completeness result by showing $\mathbf{W}[2]$-membership of the problem $\text{p-ABD}_{=}(\Gamma, |E|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{IV}_2$. The question $\text{p-ABD}_{\leq}(\Gamma, |E|)$ for the above two cases follows from the monotone argument of Lemma 4.10. Furthermore, the result for $\text{p-ABD}_{=}(\Gamma, |E|)$ and $\text{p-ABD}_{\leq}(\Gamma, |E|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{ID}_2$ follows from the proof by Fellows et al. [43, Thm. 21]. The membership for $\text{p-ABD}_{=}(\Gamma, |E|)$ and $\text{p-ABD}_{\leq}(\Gamma, |E|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$ is proven in Lemma 4.22, 4.23 respectively. The hardness for both cases follows from Lemma 4.20 and the Galois connection of ABD_{*} (Corollary 4.7).

(5). Follows from Corollary 4.17.

The **FPT**-membership for $\text{p-ABD}_{\leq}(\Gamma, |E|)$ when $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}$ is established in Corollary 4.17. Finally, the $\mathbf{W}[1]$-hardness for $\text{p-ABD}_{=}(\Gamma, |E|)$ when $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}$ is proven in Lemma 4.24. We achieve the mentioned result by proving $\mathbf{W}[1]$-hardness for a language Γ, such that $\neg x \vee \neg y \in \langle\Gamma\rangle_{\neq}$. The $\mathbf{W}[1]$ membership for $\text{p-ABD}_{=}(\Gamma, |E|)$ if $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}$ (including the base independence for this case) is established in Lemma 4.25

This concludes all the cases in Theorem 4.19. □

Intermediate Lemmas

Observe that when solving an instance of $\text{ABD}_{=}(\Gamma)$ and $\text{ABD}_{\leq}(\Gamma)$ if $\mathsf{IM} \subseteq \langle\Gamma\rangle$, a computationally expensive step is the case when a solution of size larger than k is found. This solution must be reduced to the one of size $= k$ (resp. $\leq k$). First, we prove the $\mathbf{W}[2]$-completeness of $\text{p-ABD}_{=}(\Gamma, |E|)$ such that $\langle\Gamma\rangle = \mathsf{IM}$. Later, we extend the membership to the cases when $\langle\Gamma\rangle = \mathsf{IV}_2$ and $\langle\Gamma\rangle = \mathsf{IS}^{l}_{10}$.

Lemma 4.20. $\text{p-ABD}_{=}(\Gamma, |E|)$ *is* $\mathbf{W}[2]$*-complete if* $\langle\Gamma\rangle = \mathsf{IM}$.

Proof. For membership we prove that $\text{p-ABD}_{=}(\Gamma, |E|) \leq^{\mathbf{FPT}} \text{p-WSAT}(\Gamma_{2,1}, \kappa)$ if $\langle\Gamma\rangle = \mathsf{IM}$. The problem $\text{p-WSAT}(\Gamma_{2,1}, \kappa)$ is known to be $\mathbf{W}[2]$-complete when κ is the weight of a satisfying assignment (Proposition 2.12). Let (V, H, M, KB, k) be an instance of $\text{ABD}_{=}(\mathsf{IM}, |E|)$, where the solution size is the parameter value (that is, $s = k$). Specifically, let $KB = \bigwedge_{i \leq r} (x_i \rightarrow y_i)$ and $M = m_1 \wedge \ldots \wedge m_n$, where $n = |M|$. Note that, in order to explain an $m_i \in M$, a single $h \in H$ suffices. As a result, for each $m_i \in M$, we associate a set $H_i \subseteq H$ of hypotheses that explains m_i. This implies that every element (singleton subset) of H_i explains m_i. Now, it is enough to determine whether at least one such $h \in H_i$ can be selected for each m_i. For this we map (V, H, M, KB, k) to (ϕ, k) where $\phi = \bigwedge_{i \leq n} \bigvee_{x \in H_i} x$. Then our claim is that (V, H, M, KB, k) has an

explanation E if and only if ϕ has a satisfying assignment of size k. Clearly, there is a 1-1-correspondence between solutions E of (V, H, M, KB, k) and satisfying assignments θ for ϕ of weight k. That is, $\theta(x) = 1 \iff x \in E$. We conclude by observing that H_i can be computed in polynomial time for each m_i and $|H_i| \leq H$.

For the hardness, we reduce from p-WSAT$(\Gamma^{+}_{2,1}, \kappa)$ which is also $\mathbf{W}[2]$-complete when κ is the weight of a satisfying assignment (Proposition 2.12). Let (ϕ, k) be an instance of p-WSAT$(\Gamma^{+}_{2,1}, \kappa)$ where ϕ is given as: $\bigwedge_{i \leq q} \bigvee_{j \leq r} (x_{i,j}) = \bigwedge_{i \leq q} (x_{i,1} \vee \ldots \vee x_{i,r})$ and $\text{Vars}(\phi) = \{\, x_{i,j} \mid i \leq q, j \leq r \,\}$. For $i \leq q$, let m_i be a fresh variable, we use these variables to simulate each of the q clauses. Moreover, we slightly abuse the notation and write $x \in m_i$ to depict that the variable x appears in ith clause. Finally, we let $KB = \bigwedge_{i \leq q} \bigwedge_{x \in m_i} (x \to m_i)$, $H = \text{Vars}(\phi)$, $M = \bigwedge_{i \leq q} m_i$ and $V = H \cup M$. It is easy to observe that a subset $E \subseteq H$ is an explanation for $(V, H, M, KB, k) \iff \theta \models \phi$ where $\theta(x) = 1 \iff x \in E$. This completes the desired reduction for showing $\mathbf{W}[2]$-hardness, thereby proving the lemma. □

The reduction p-ABD$_{=}(\Gamma, |E|) \leq^{\mathbf{FPT}}$ p-WSAT$(\Gamma_{2,1}, \kappa)$ in the proof of Lemma 4.20 is essential in achieving several other reductions in this sequel. To establish the membership in these cases, we only state required modifications on top of the same reduction. We next prove $\mathbf{W}[2]$-membership of p-ABD$_{=}(\Gamma, |E|)$ for languages in IV_2.

Lemma 4.21. p-ABD$_{=}(\Gamma, |E|)$ *is in* $\mathbf{W}[2]$ *if* $\langle \Gamma \rangle \subseteq \mathsf{IV}_2$.

Proof. Observe that in the proof of Lemma 4.20 (when $\langle \Gamma \rangle = \mathsf{IM}$), we dealt only with clauses of the type $(x \to y)$ for variables x and y. We refer to these 'implication' clauses as clauses of type-0. When $\langle \Gamma \rangle = \mathsf{IV}_2$ we have additional clauses of the following types:

type-1. Unit clauses: positive and negative literals x, $\neg x$.

type-2. Positive clauses of size two or larger: $(x_1 \vee \cdots \vee x_n)$, $n \geq 2$.

type-3. Clauses with exactly one negative literal of size 3 or larger: $(\neg x_0 \vee x_1 \vee \cdots \vee x_n)$, $n \geq 2$.

We can eliminate type-1 clauses by unit propagation and obtain an equivalent formula (regarding satisfiability). Observe that unit propagation gets rid of type-1 clauses, though it might generate additional clauses of type-0, type-2, or type-3. Nevertheless, we end up only with clauses of either type-0, type-2, or type-3 and, particularly, no type-1 clauses anymore. Notice that this transformation does not preserve all the satisfying assignments, but one can maintain the equivalence by adding fixed values of the eliminated variables in an assignment.

Now, we argue that after applying *resolution* on variables in $KB \setminus H$, we can ignore type-2 and type-3 clauses. Let x be a variable and $C \cup \{x\}$, $D \cup \{\neg x\}$ be two clauses, then a single step of resolution on the variable x yields a clause $C \vee D$, whereas it removes both C and D. Observe that we do not apply resolution to the variables in H. This is due to the reason that a satisfying assignment s for the clause $(x \to y_i) \in KB$ such that $s(x) = 1$ also forces $s(y_i) = 1$. Furthermore,

it also forces $s(z_{i,j}) = 1$ for each $z_{i,j}$ such that $(y_i \rightarrow z_{i,j}) \in KB$, and so on. This precisely captures the intuition that x (as a hypothesis) explains each y_i and $z_{i,j}$. Consequently, removing such a variable x from H (in resolution) in the case when x explains some $m \in M$ would be problematic.

Finally, we claim that we can ignore the type-2 and type-3 clauses after the resolution. Type-2 clauses are irrelevant since such clauses do not explain anything. In other words, the satisfaction of a type-2 clause does not force any particular variable to 1. In a type-3 clause of the form $C = (\neg x_0 \vee x_1 \vee \ldots \vee x_r)$ an assignment s such that $s(x_0) = 1$ forces a whole clause $(x_1 \vee \ldots \vee x_r)$ to be true. That is, $s(x_i) = 1$ for at least one x_i. This implies that type-3 clauses cannot be ignored right-away because there might exist further clauses of the form $\neg x_j \vee m$ for each $2 \leq j \leq r$ with $m \in M$. This is due to the reason that the candidate explanation x_1 for m would be lost. However, after applying resolution, we know that a type-3 clause only forces one of the many variables to 1 instead of a single one. As a result, this allows us to ignore type-3 clauses as well. Consequently, we only need to consider type-0 clauses. This completes the proof in conjunction with the reduction in Lemma 4.20. □

The complexity results for p-ABD$_{\leq}(\Gamma, |E|)$ in the above two cases follow from the *monotone* argument of Lemma 4.10. Moving forward, the hardness of p-ABD$_{=}(\Gamma, |E|)$ and p-ABD$_{\leq}(\Gamma, |E|)$ when $\langle\Gamma\rangle = \mathsf{IS}^{\ell}_{10}$ is a consequence of the $\mathbf{W}[2]$-hardness of these problems when $\langle\Gamma\rangle \supseteq \mathsf{IM}$. However, we strengthen these results to $\mathbf{W}[2]$-completeness by showing the membership in $\mathbf{W}[2]$ in the following two lemmas.

Lemma 4.22. *Let $\ell \geq 2$, then* p-ABD$_{=}(\Gamma, |E|)$ *is in* $\mathbf{W}[2]$ *if* $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$.

Proof. Consider the reduction from Lemma 4.20, where we reduce an instance (V, H, M, KB, k) of p-ABD$_{=}(\Gamma, |E|)$ to the instance (ϕ, k) of p-WSAT$(\Gamma_{2,1}, \kappa)$ with $\phi = \bigwedge_{i \leq n} \bigvee_{x \in H_i} x$. The only difference from the aforementioned reduction is that now there are additional constraints of the form $(\neg x_1 \vee \ldots \vee \neg x_r)$ in Γ, where $r \leq \ell$. Now we have two cases to consider.

Case 1. If all the additional (negative) constraints contain exclusively variables from H, then we simply add them to ϕ and obtain a new formula ψ. Since any satisfying assignment for ψ satisfies ϕ and these constraints, this yields an explanation E as required. Conversely, any such explanation E yields a satisfying assignment for ψ since $E \wedge KB$ is consistent.

Case 2. If some constraints contain variables not from H, we transform these constraints into their equivalents which only contain variables from H. This is achieved by repeating the following procedure as long as applicable:

Let C_u be a constraint and u be a variable such that $u \notin H$. We first compute the set of hypotheses $H_u \subseteq H$ that explain u (analogously to Lemma 4.20). Let $H_u = \{h_1, \ldots, h_r\}$, now we replace C_u by r copies of itself and in each C^i_u, substitute h_i for u. Note that this does not change the width of any clause. We repeat this procedure for every u such that $u \notin H$. Finally, we add these new clauses to ϕ and obtain a new formula ψ.

Now we prove the correctness of this transformation via the following claim.

Claim 4.3. *The construction in Case* 2 *preserves the correspondence between the solutions of an instance of* $\mathrm{ABD}_{=}(\Gamma, |E|)$ *and the satisfying assignments for* ϕ *of weight* k*. Moreover, this transformation can be achieved in time polynomial in the instance* (V, H, M, KB, k)*.*

Proof of Claim. *Note that the difference between Lemma* 4.20 *and this case lies in the fact that a solution to* $\mathrm{ABD}_{=}(\Gamma, |E|)$ *must satisfy additional constraints as specified above. The problematic part is when* $(\neg x_{i,1} \vee \ldots \vee \neg x_{i,j}) \in KB$ *for some variables* $\{x_{i,1}, \ldots x_{i,j}\} \subseteq H$*. The formula* ψ *assures that a satisfying assignment* s *can not have* $s(x_{i,p}) = 1$ *for each* $p \leq j$ *as restricted by the constraint* $(\neg x_{i,1} \vee \ldots \vee \neg x_{i,j})$*. This implies that an explanation* E *for* (V, H, M, KB, k) *can not include the set* $\{x_{i,1}, \ldots x_{i,j}\}$*. This proves the correctness claim in conjunction with the arguments in Lemma 4.20 for* ϕ*.*

Now we prove that this transformation can be achieved in polynomial time. The worst case is when there is a clause C *which contains no variable from* H*. Furthermore, assume that* C *has the maximum arity. That is,* $C = (\neg x_1 \vee \ldots \vee \neg x_q)$ *where* $q \leq \ell$ *is the maximum arity of the constraint language* Γ*. Then we know that the set* H_x *for each* $x \in C$ *can have a size at most* m*, where* $|H| = m$ *is bounded by the input size. As a result,* C *yields at most* m^q *new constraints at the completion of the above procedure. Since* q *is constant (it only depends on the constraint language and not on the input instance), the factor* m^q *is polynomial. Finally, there are polynomial many constraints to check for this procedure and we conclude that the transformation takes only polynomial time.* ■

Eventually, similar arguments as in Lemma 4.20 for ψ complete the proof. □

We wish to point out that the reduction in Lemma 4.22 does not immediately settle the complexity for p-$\mathrm{ABD}_{\leq}(\Gamma, |E|)$ when $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$. Nevertheless, we achieve $\mathbf{W}[2]$-membership by reducing p-$\mathrm{ABD}_{\leq}(\Gamma, |E|)$ to Short-NTM-Halt(κ) (the halting problem for non-deterministic multi-tape Turing machines, parameterized by the number of steps).

Problem:	p-Short-NTM-Halt(κ)
Input:	A nondeterministic multi-tape TM $\mathbb{M}$ and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does $\mathbb{M}$ accept the empty string in at most k steps?

The problem Short-NTM-Halt(κ) is $\mathbf{W}[2]$-complete [46, Thm. 7.28]. The following reduction provides the $\mathbf{W}[2]$-membership for p-$\mathrm{ABD}_{\leq}(\Gamma, |E|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$.

Lemma 4.23. *Let* $\ell \geq 2$*, then* p-$\mathrm{ABD}_{\leq}(\Gamma, |E|)$ *is in* $\mathbf{W}[2]$ *if* $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$*.*

Proof. The proof is an extension of Lemma 4.22. Proceeding as before, we map (V, H, M, KB, k) to (ψ, k), where ψ is a collection of positive and negative clauses. Let U_i for $i \leq P$ (resp., $V_j, j \leq N$) denote the collection of positive (negative) clauses and $L = P + N$. We give a reduction that provides a multi-tape NTM $\mathbb{M}$ such that ψ has a satisfying assignment of size at most k if and only if $\mathbb{M}$ accepts the empty string in at most k' steps and $k' \leq g(k)$ for some function g. $\mathbb{M}$

consists of $L+1$ tapes with one tape per each clause. The initial P tapes are dedicated to positive clauses and the following N tapes to the negative ones. For convenience, we denote the ith tape (corresponding to U_i) as u_i for $i \leq P$, and $(P+j)$th tape (corresponding to V_j) as v_j $j \leq N$. The last tape is referred to as the tape $L+1$. The computation of $\mathbb{M}$ has the following four phases.

1. For each $j \leq N$, mark the length of V_j on the tape v_j. At the same time, non-determinstically write k elements $x_1, \ldots, x_k$ from V on the tape $L+1$.

2. Remove duplicates from the tape $L+1$.

3. Read the tape $L+1$. At the same time, for each tape $w_r \in \{u_i, v_j\}$, mark a cell if the elements being read appears in the respective clause $W_r \in \{U_i, V_j\}$ where $r \leq L$.

4. If at least one cell of each tape u_i ($i \leq P$) is marked and at least one cell of the tape v_j ($j \leq N$) is unmarked, then accept.

We first claim that negative clauses of length greater than k can be ignored. The reason is that, for any assignment s with weight k, a negative clause of length more than k still contains $\neg z$ for some variable z such that $s(z) = 0$. As a result, each negative clause of length greater than k is trivially satisfied. This implies that the length of each tape corresponding to a negative clause is bounded by k. For positive clauses, the length does not matter. This is because $\mathbb{M}$ only needs to determine if at least one variable appearing in each positive clause is selected by the guessed assignment. Consequently, $\mathbb{M}$ only scans at most k cells on each of its tapes.

Our construction requires that the length of each V_j is hardcoded on the tape v_j for $j \leq N$. This ensures that $\mathbb{M}$ runs in parallel and does not need a state set of exponential size to ensure the correct computation. For each negative clause V_j of length l_j, the tape v_j has length $l_j + 1$, where $j \leq N$. This is achieved through having a collection of $r+1$ states, where $r = \max\{l_j \mid j \leq N\}$ and $r \leq k$. Moreover, even though $\mathbb{M}$ can guess duplicate elements, it must work with the distinct collection of variables in the subsequent steps. In other words, multiple occurrences of any variable must be removed from the guessed assignment. The alphabet of $\mathbb{M}$ constitutes $V \cup \{yes, *, \#\}$ where V is the collection of variables in ψ. We now present a detailed but high-level description of $\mathbb{M}$ consisting of the following five phases in its computation.

1. In the first k steps, the head of the tape v_j writes the symbol '$*$' for l_j-many cells and the symbol '#' in the cell $l_j + 1$, where l_j is the length of V_j for $j \leq N$. In the same steps, the head of the tape $(L+1)$ non-deterministically writes k elements $x_1, \ldots, x_k$ from V, into the first k cells. After k steps, all heads go back to the first cell.

2. In the next (at most) k^2 steps, the head on the tape $L+1$ removes any duplicates.

3. In the following h steps (where $h \leq k$), the head of the tape $(L+1)$ reads the guessed distinct elements $x_1, \ldots, x_h$. At the same time, in rth of these steps (for $r \leq h$), the head of each tape u_i determines whether U_i contains x_r, for $i \leq P$, while the head of each tape v_j determines if $\neg x_r$ appears in V_j for $j \leq N$. In the first case, $\mathbb{M}$ marks the cell with '*yes*',

and it does not mark anything new in the latter case ($*$ remains there). After marking the cell, each head moves right. Finally, if the variable x_r does not appear in the corresponding clause, the heads neither move nor mark anything.

4. After the previous $h \leq k$ steps, the head on tapes u_i for $i \leq P$ moves one cell to the left while, for other tapes, it stays in the same cell.

5. If each head reads '*yes*' in the u_i-tapes ($i \leq P$) and a '$*$' in v_j-tapes ($j \leq N$), then $\mathbb{M}$ accepts.

We claim that the above translation between (ψ, k) and $(\mathbb{M}, k')$ for some value of k', is indeed an **FPT**-reduction. Moreover, the reduction works as desired.

Claim 4.4. $\mathbb{M}$ *can be constructed from* ψ *in* **FPT***-time. Moreover,* $\mathbb{M}$ *accepts the empty input in at most* k' *steps where* $k' = k^2 + 3k + 2$*, if and only if* ψ *has a satisfying assignment of size at most* k*.*

Proof of Claim. *Recall that* $\mathbb{M}$ *has* $L + 1$ *tapes, where* L *is the number of clauses in* ψ*. Moreover, the alphabet of* $\mathbb{M}$ *constitutes* $V \cup \{yes, *, \#\}$ *where* V *is the collection of variables in* ψ*. Finally,* $\mathbb{M}$ *has* $k + \mathcal{O}(1)$ *states, where* k *of these states are required to ensure the first phase of the computation.*

In the phase 1, the head on the tape $L + 1$ *moves right, writing* $x \in V$ *(non-deterministically). Moreover, the head on* v_j *(*$j \leq N$*) writes the symbol '*$*$*' for* l_j *many cells and the symbol '#' in the last cell, where* $l_j \leq k$ *is the length of* V_j*. In each case, head stays in the last cell. To bring the head back to the first cell on the tape* $L + 1$*, it can read any element* $x \in V$*. However, the head on each* v_j*-tape (*$j \leq N$*) reads the symbol '#' exactly once and the symbol '*$*$*' in the remaining cells.* $\mathbb{M}$*'s transitions force every head on the tape* v_j *to move one step to the left by reading '#', after that, it can only read the symbol '*$*$*'. This implies that the transition relation of* $\mathbb{M}$ *has the size* $O(k \cdot |\psi|^2)$ *for the first phase. Finally, the transition in the following phases for removing duplicates, variables comparison, and the final check each has size* $\mathcal{O}(k \cdot |\psi|)$*. As a result,* $\mathbb{M}$ *can be constructed from* ψ *in* **FPT***-time. This proves the first part of the claim.*

For the second part of the claim, notice that $\mathbb{M}$ *runs for* $2k + k^2 + k + 2$ *many steps. The first* $2k$ *steps account for marking the length of each negative clause on the corresponding tapes and for guessing* k *elements on the tape* $L + 1$*. In both cases, the head of each tape must move back to scan the first cell (the reason for the* $2k$ *steps). The following* k^2 *steps are required to determine and remove duplicates from the guessed list of variables. Lastly, at most* k *steps are required to compare variables against each clause, and the final two steps determine the acceptance criteria for each tape.*

We prove the correctness through the following equivalence, where an assignment s *is seen as a set of variables* x *such that* $s(x) = 1$*.*

- $\mathbb{M}$ *guesses* k *elements in such a way that the head of the tape* u_i *(*$i \leq P$*) reads 'yes', no head of the tape* v_j *(*$j \leq N$*) reads '#' and the machine halts in the accepting state.*

- *The assignment* s *of weight at most* k *guessed by* $\mathbb{M}$ *is such that* s *contains at least one variable per each positive clause and for each negative clause,* s *does not contain all of its variables.*

- ψ *has a satisfying assignment* s *of weight at most* k*,* $s(x) = 1$ *for at least one* $x \in U_i$ *and every* $i \leq P$*, and* $s(y) \neq 1$ *for each* $y \in V_j$ *and* $j \leq N$*.*

Consequently, if ψ has a satisfying assignment s of weight k, then $\mathbb{M}$ simply guesses this assignment and halts in the accepting state. Conversely, if $\mathbb{M}$ accepts, then the guessed elements constitute a satisfying assignment for ψ. ■

This completes the proof to Lemma 4.23 by noting that the pair (ψ, k) in Lemma 4.22 can be constructed from an instance (V, H, M, KB, k) of $\mathrm{ABD}_{\leq}(\Gamma, |E|)$ in **FPT**-time. □

Regarding the parameter $|E|$, the only cases where $\text{p-ABD}_{\leq}(\Gamma, |E|)$ and $\text{p-ABD}_{=}(\Gamma, |E|)$ have different complexity is when $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}$. The problem $\text{p-ABD}_{\leq}(\Gamma, |E|)$ is **FPT** (Corollary 4.17). In the following two lemmas, we prove the **W**[1]-completeness of $\mathrm{ABD}_{=}(\Gamma, |E|)$. The **W**[1]-hardness for $\mathrm{ABD}_{=}(\Gamma, |E|)$ is proven for the languages Γ such that $\neg x \vee \neg y \in \langle\Gamma\rangle_{\neq}$.

Lemma 4.24. $\mathrm{ABD}_{=}(\Gamma, |E|)$ *is* **W**[1]*-hard for any constraint language* Γ *such that* $\neg x \vee \neg y \in \langle\Gamma\rangle_{\neq}$.

Proof. We reduce $\text{p-}\textsc{IndependentSet}(\kappa)$ to $\mathrm{ABD}_{=}(\Gamma, |E|)$ where Γ is a CL such that $\neg x \vee \neg y \in \langle\Gamma\rangle_{\neq}$. An instance $(\mathcal{G}, k)$ of $\text{p-}\textsc{IndependentSet}(\kappa)$ constitutes a graph $\mathcal{G} = (\tilde{V}, \tilde{E})$[2] and a number $\kappa(\mathcal{G}) = k$, the question is whether there is an independent set of size k in $\mathcal{G}$. The problem $\text{p-}\textsc{IndependentSet}(\kappa)$ is **W**[1]-complete when κ is the size of the independent set [31]. Let $(\mathcal{G}, k)$ be an instance of $\text{p-}\textsc{IndependentSet}(k)$ where $\mathcal{G} = (\tilde{V}, \tilde{E})$ is a graph and k is the solution size. We map it to $(V, H, M, KB, k+1)$, where $V = H := \tilde{V} \cup \{z\}$, $M := z$, and

$$KB := \bigwedge_{(x,y)\in\tilde{E}} (\neg x \vee \neg y).$$

Let S be an independent set of size k for $\mathcal{G}$, then $S \wedge KB$ is consistent since no two elements of S share an edge. As a consequence, $S \cup \{z\}$ is an explanation for $(V, H, M, KB, k+1)$. Conversely, an explanation E for $(V, H, M, KB, k+1)$ of size $k+1$ must include z as well as k other variables from H. Now, $E \wedge KB$ is consistent, and this implies that no two variables in E share an edge in $\tilde{E}$, consequently giving an independent set of size k. This implies that $\mathcal{G}$ admits an independent set of size k if and only if (V, H, M, KB) admits an explanation of size $k+1$. □

Now we prove **W**[1] membership for $\mathrm{ABD}_{=}(\Gamma, |E|)$ with $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}^{\ell}$ (for any arbitrary base) in the lemma below.

Lemma 4.25. *Let* $\ell \geq 2$, *then* $\text{p-ABD}_{=}(\Gamma, |E|)$ *is in* **W**[1] *if* $\langle\Gamma\rangle \subseteq \mathsf{IS}_{12}^{\ell}$.

Proof. We reduce $\text{p-ABD}_{=}(\Gamma, |E|)$ to $\text{p-WSAT}(\Gamma_{1,\ell}, \kappa)$, which is **W**[1]-complete when κ is the weight of a satisfying assignment (Prop. 2.12). It is worth pointing out that the proof is correct even when equality constraints are present. As a result, the base independence is not implied by the previous lemmas but follows directly from the proof below.

According to Lemma 4.15, we can solve $\mathrm{ABD}_{\leq}(\Gamma)$ when $\Gamma \subseteq \mathsf{IS}_{12}^{\ell}$ in polynomial time. In other words, in polynomial time, one can determine whether there exists a solution of size $\leq s$

[2]We deote $\mathcal{G} = (\tilde{V}, \tilde{E})$ to differentiate nodes and edges of $\mathcal{G}$ from V and E in the context of an abduction instance and an explanation, respectively.

of an instance. Let (V, H, M, KB, k) be an instance of p-ABD$_=(\Gamma, |E|)$ with $KB = \bigwedge_{i \leq r} C_i \wedge N \wedge P \wedge Q$, where $C_i = (\neg x_1^i \vee \cdots \vee \neg x_\ell^i)$. Moreover, P, N denote the positive and negative unit clauses, respectively, and Q are the equality clauses. Without loss of generality, assume that (V, H, M, KB, k) admits a solution of size $\leq k$ (otherwise, there is no solution of size k). It follows from the proof of Lemma 4.15 that $E_{MP} \subseteq E \subseteq H$ is true for any explanation E of the give instance. This implies that $k \geq |E_{MP}|$. We also know from Lemma 4.15 that E_{MP} is an explanation for M and that both E_{MP} and M are consistent with all clauses in KB.

The question now reduces to whether we can extend E_{MP} to a solution of size k by adding $k - |E_{MP}|$ variables from $H \setminus E_{MP}$? To answer this, we map an abduction instance (V, H, M, KB, k) to a satisfiability instance $(\phi, k - |E_{MP}|)$, where ϕ is obtained from KB by the application of the following consecutive steps.

1. For each $(x_i = x_j) \in KB$, such that $x_i, x_j \in H$, add clauses $(\neg x_i \vee x_j)$ and $(x_i \vee \neg x_j)$ to ϕ. This ensures that corresponding to each clause of the form $(x_i = x_j)$, either both x_i, x_j are in the solution, or none is.

2. Remove all clauses C_i containing only variables from $V \setminus H$.

3. Remove all negative unit clauses $(\neg x) \in N$ such that $x \notin H$. Note that after this step all remaining negative unit clauses are built upon variables from $H \setminus E_{MP}$ only.

4. For each clause C_i, denote by X_H^i (resp., $X_{\overline{H}}^i$) the variables from H (resp., not from H). Execute the following steps:

   (a) Remove C_i.

   (b) If $X_{\overline{H}}^i \subseteq P$: add the clause $(\neg x_1 \vee \cdots \vee \neg x_p)$ to ϕ, where $\{x_1, \ldots, x_p\} = X_H^i \setminus E_{MP}$. Otherwise, nothing needs to be done as $X_{\overline{H}}^i \nsubseteq P$ is true. That is, we have $x \notin P$ for some variable $x \in X_{\overline{H}}^i$ and $\neg x \vee \bigvee_{x_j \in X_H^i} \neg x_j$ is satisfiable via setting x to 0 if all x_j are mapped to 1.

   Note that, after this step, all remaining clauses C_i are built upon variables from H alone.

5. Remove all positive unit clauses $(x) \in P$ such that $x \notin H \setminus E_{MP}$. After this step, it holds that $\text{Vars}(\phi) = H$ and remaining positive unit clauses are built upon variables in $H \setminus E_{MP}$ only.

6. For each clause C_i: remove from C_i all literals built upon variables in E_{MP}. Note that in the so-obtained C_i' at least one literal remains, otherwise $E_{MP} \wedge C_i$ would be inconsistent.

After the above implementation, it holds that $\text{Vars}(\phi) = H \setminus E_{MP}$. As a consequence, we have the following equivalence.

- (V, H, M, KB, k) admits a solution of size exactly k.
- E_{MP} extends to a solution E of size k by adding $k - |E_{MP}|$ variables from H.

- ϕ has a satisfying assignment of size exactly $k - |E_{MP}|$.

Finally, notice that ϕ can be constructed from (V, H, M, KB, k) in polynomial time and that $k - |E_{MP}|$ is bounded by a function of k alone (because $|E_{MP}| \leq k$) This implies that the reduction is indeed an **FPT**-reduction and completes the proof of the lemma. □

4.2.4 Parameter 'number of manifestations' $|M|$

The complexity landscape regarding the parameter $|M|$ is more diverse than the parameter $|E|$. The classification differs for each of the investigated problem variant. Consequently, we treat each case separately and start with the general abduction problem, which provides a pentachotomy.

Theorem 4.26. *The problem* p-ABD$(\Gamma, |M|)$ *is*

1. **paraΣ_2^P**-*complete if* $\mathsf{C} \subseteq \langle\Gamma\rangle \subseteq \mathsf{BR}$, *where* $\mathsf{C} \in \{\mathsf{IN}_2, \mathsf{II}_0\}$,
2. **paraCoNP**-*complete if* $\mathsf{IN} \subseteq \langle\Gamma\rangle \subseteq \mathsf{II}_1$,
3. **paraNP**-*complete if* $\mathsf{IE}_0 \subseteq \langle\Gamma\rangle \subseteq \mathsf{IE}_2$,
4. $\mathbf{W}[1]$-*complete if* $\mathsf{IS}_{11}^2 \subseteq \langle\Gamma\rangle \subseteq \mathsf{C}$, *where* $\mathsf{C} \in \{\mathsf{ID}_2, \mathsf{IS}_{10}^\ell\}$,
5. **FPT** *if* $\langle\Gamma\rangle \subseteq \mathsf{C} \in \{\mathsf{ID}_1, \mathsf{IS}_{12}, \mathsf{IE}_1, \mathsf{IV}_2\}$.

Proof. (1+2). The membership is trivial since ABD(Γ) is in $\mathbf{\Sigma}_2^P$ (resp. **CoNP**) in this case. Moreover, we proved in Lemma 4.11 using the fact that 1-slice of each problem is (classically) hard for respective classes.

(3). The membership is trivial since ABD(Γ) is **NP**-complete in this case. For hardness, we prove that 1-slice of p-ABD$(\Gamma, |M|)$ is **NP**-hard.

Notice that due to Nordh and Zanuttini [89, Lemma 29] any abduction instance can be reduced to an equivalent one with only one manifestation, provided that one can express certain clauses in the *KB*. The idea is to encode the set M of manifestations by a single new manifestation y while adding the clause $y \vee \bigvee_{m \in M} \neg m$ to the *KB*. Recall that M is a (positive) set of propositions, implying that the clause $y \vee \bigvee_{m \in M} \neg m$ is a Horn clause. Consequently, the aforementioned reduction is valid if *KB* is a Γ-formula such that $\mathsf{IE}_0 \subseteq \langle\Gamma\rangle$. This reduction in conjunction with the result for abduction problem with single literal manifestation ([89, Prop. 53]) implies that 1-slice of p-ABD$(\Gamma, |M|)$ is **NP**-hard if $\mathsf{IE}_0 \subseteq \langle\Gamma\rangle$. As a consequence, the problem is **paraNP**-complete.

(4). The $\mathbf{W}[1]$-membership for ABD$(\Gamma, |M|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{ID}_2$ follows from a result by Fellows et al. [43, Thm. 25]. Notice that the authors (in [43]) prove the completeness for the languages in ID_2 alone, but using the fact that the formulas (or clauses) in their reduction are Γ-formulas where $\langle\Gamma\rangle = \mathsf{IS}_{11}^2$, we derive the hardness for p-ABD$(\Gamma, |M|)$ such that $\langle\Gamma\rangle = \mathsf{IS}_{11}^2$. The $\mathbf{W}[1]$-membership for p-ABD$(\Gamma, |M|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{IS}_{10}^\ell$ is established in Lemma 4.29. As a consequence, we have the desired completeness result.

(5). Follows from Corollary 4.12. □

If Γ is definite Horn, that is, $\langle\Gamma\rangle = \mathsf{IE}_1$, then p-ABD$_{\leq}(\Gamma, |M|)$ surprisingly behaves different and has a much higher complexity as compared to p-ABD$(\Gamma, |M|)$.

Theorem 4.27. *The problem* p-ABD$_{\leq}(\Gamma, |M|)$ *is*

1. **paraΣ$_2^P$**-*complete if* $\mathsf{C} \subseteq \langle\Gamma\rangle \subseteq \mathsf{BR}$, *where* $\mathsf{C} \in \{\mathsf{IN}_2, \mathsf{II}_0\}$,
2. **paraCoNP**-*hard if* $\mathsf{IN} \subseteq \langle\Gamma\rangle \subseteq \mathsf{II}_1$,
3. **paraNP**-*complete if* $\mathsf{IE} \subseteq \langle\Gamma\rangle \subseteq \mathsf{IE}_2$,
4. **W**[1]-*complete if* $\mathsf{IS}_{11}^2 \subseteq \langle\Gamma\rangle \subseteq \mathsf{C}$, *where* $\mathsf{C} \in \{\mathsf{ID}_2, \mathsf{IS}_{10}^{\ell}\}$,
5. **FPT** *if* $\langle\Gamma\rangle \subseteq \mathsf{C} \in \{\mathsf{ID}_1, \mathsf{IS}_{12}, \mathsf{IV}_2\}$.

Proof. (1+2). Follows from Theorem 4.26 in conjunction with Lemma 4.8.

(3). The membership is trivial since ABD$_{\leq}(\Gamma)$ is in **NP** for this case. For hardness, we prove that 1-slice of p-ABD$_{\leq}(\Gamma, |M|)$ is **NP**-hard by reducing from the classical vertex cover problem (VertexCover). We argue that the reduction preseted by Fellows et al. [43, Thm. 5] is applicable to our case as well. An instance $(\mathcal{G}, s)$ of VertexCover, where $\mathcal{G} = (\tilde{V}, \tilde{E})$ is translated into an abduction instance (V, H, M, KB, s), where $V := \tilde{V} \cup \tilde{E} \cup \{m\}$, $H := \tilde{V}$, $M := \{m\}$, and $KB := (m \bigvee_{e \in \tilde{E}} \neg e) \wedge \bigwedge_{e=\{x,y\}} ((x \rightarrow e) \wedge (y \rightarrow e))$. It is important to observe that KB is a Γ-formula for $\langle\Gamma\rangle = \mathsf{IE}$, consequently giving the desired hardness result.

(4). The membership for ABD$_{\leq}(\Gamma, |M|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{ID}_2$ follows from Fellows et al.[43, Thm. 25]. Notice that the authors in [43] prove the completeness for the languages in ID_2 alone, but using the fact that the formulas (or clauses) in their reduction are Γ-formulas where $\langle\Gamma\rangle = \mathsf{IS}_{11}^2$, we derive the hardness for p-ABD$_{\leq}(\Gamma, |M|)$ such that $\langle\Gamma\rangle = \mathsf{IS}_{11}^2$. The **W**[1]-membership for the languages Γ such that $\langle\Gamma\rangle \subseteq \mathsf{IS}_{10}^{\ell}$ is established in Lemma 4.30.

(5). **FPT**-membership for p-ABD$_{\leq}(\Gamma, |M|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{IM}$ is proven in Lemma 4.31. Lemma 4.32 strengthens this result to p-ABD$_{\leq}(\Gamma, |M|)$ when $\langle\Gamma\rangle \subseteq \mathsf{IV}_2$. The remaining cases follow from Corollary 4.17. □

We conclude this subsection by presenting the theorem for ABD$_{=}(\Gamma, |M|)$. Surprisingly, the majority of cases are already **paraNP**-complete. Even the case of the essentially negative co-clones, which are **FPT** for ABD$_{\leq}$ yield **paraNP**-completeness in this situation. Merely the 2-affine and dualHorn cases are **FPT**.

Theorem 4.28. *The problem* p-ABD$_{=}(\Gamma, |M|)$ *is*

1. **paraΣ$_2^P$**-*complete if* $\mathsf{C} \subseteq \langle\Gamma\rangle \subseteq \mathsf{BR}$, *where* $\mathsf{C} \in \{\mathsf{IN}_2, \mathsf{II}_0\}$,
2. **paraCoNP**-*hard if* $\mathsf{IN} \subseteq \langle\Gamma\rangle \subseteq \mathsf{II}_1$,
3. **paraNP**-*complete if* $\mathsf{C}_1 \subseteq \langle\Gamma\rangle$, *where* $\mathsf{C}_1 \in \{\mathsf{IS}_1^2, \mathsf{IE}\}$ *and* $\langle\Gamma\rangle \subseteq \mathsf{C}_2 \in \{\mathsf{IE}_2, \mathsf{ID}_2\}$,
4. **FPT** *if* $\langle\Gamma\rangle \subseteq \mathsf{C} \in \{\mathsf{ID}_1, \mathsf{IV}_2\}$.

Proof. (1+2). Follow from Theorem 4.26 in conjunction with Lemma 4.9.

(3). The membership is trivial since $\mathrm{ABD}_{=}(\Gamma)$ is in **NP** for each of these case. The hardness for $\mathrm{ABD}_{=}(\Gamma)$ when $\mathsf{IE} \subseteq \langle\Gamma\rangle$ follows from the argument used in the proof of Theorem 4.27 for the IE case. We prove the hardness for the remaining cases in Lemma 4.33 where we show that $\text{p-ABD}_{=}(\Gamma, |M|)$ is **paraNP**-hard as long as $(\neg x \vee \neg y) \in \langle\Gamma\rangle_{\neq}$. The case for $\text{p-ABD}_{=}(\Gamma, |M|)$, such that $\langle\Gamma\rangle \supseteq \mathsf{IS}_1^2$ (also for arbitrary bases) then follows as a corollary.

(4). The proof for $\text{p-ABD}_{=}(\Gamma, |M|)$ such that $\langle\Gamma\rangle \subseteq \mathsf{IV}_2$ is due to the monotone argument of Lemma 4.10 and Theorem 4.27. The result for $\langle\Gamma\rangle \subseteq \mathsf{ID}_1$ is due to Corollary 4.17. □

Intermediate Lemmas

Our first result is the $\mathbf{W}[1]$-membership of $\text{p-ABD}(\Gamma, |M|)$ when $\langle\Gamma\rangle \subseteq \mathsf{IS}_{10}^{\ell}$. The membership is achieved by reducing $\text{p-ABD}(\Gamma, |M|)$ to Short-NSTM-Halt(κ) (the halting problem for non-deterministic *single-tape* Turing machines, parameterized by the number of steps).

Problem:	p-Short-NSTM-Halt(κ)
Input:	A non-deterministic single-tape TM $\mathbb{M}$ and $k \in \mathbb{N}$.
Parameter:	k.
Question:	Does $\mathbb{M}$ accept the empty string in at most k steps.?

The problem Short-NSTM-Halt(κ) is $\mathbf{W}[1]$-complete [46, Thm. 6.17]. The proof of the following lemma uses the reduction from Lemma 4.23 and the fact that a multi-tape TM $\mathbb{M}$ can be simulated by a single-tape machine $\mathbb{M}'$. Moreover, this can be achieved only with a quadratic blow-up in the running time of $\mathbb{M}'$ [106, Theorem 7.8].

Lemma 4.29. *Let $\ell \geq 2$ then the problem* $\text{p-ABD}(\Gamma, |M|)$ *is in* $\mathbf{W}[1]$ *if* $\langle\Gamma\rangle \subseteq \mathsf{IS}_{10}^{\ell}$.

Proof. Notice first that the problem $\text{p-ABD}(\Gamma, |M|)$ does not impose any restriction on the size of a solution. As a consequence, an instance (V, H, M, KB, k) of $\text{p-ABD}(\Gamma, |M|)$ has a solution iff the formula ψ constructed for the reduction in Lemma 4.22 is satisfiable. Recall that the parameter value k in the present case is $|M|$, which corresponds to the number of positive clauses in ψ. We claim that it is enough to determine if ψ has a satisfying assignment with weight at most k.

Claim 4.5. *Let ϕ be any $\Gamma_{2,1}$-formula with k positive and n negative clauses. Then ϕ is satisfiable if and only if ϕ has a satisfying assignment of weight at most k.*

Proof of Claim. *Let U_i (resp. V_j) for $i \leq k$ ($j \leq n$) denote the collection of positive (negative) clauses in ϕ. If ϕ has a satisfying assignment with weight at most k, then ϕ is clearly satisfiable. For the other direction, note that if $s \models \phi$ for some assignment s then, $s \cap U_i \neq \emptyset$ for any $i \leq k$ and $V_j \not\subseteq s$ for $j \leq N$. Where we consider s as the collection of variables that are mapped to 1. Let s' be the assignment obtained from s such that, for each positive clause U_i, s' selects exactly one variable from U_i (with repetition allowed for different clauses). Then $s' \models \phi$ and $|s'| \leq k$. This is because s' selects exactly one variable from each positive clause and $s' \subseteq s$.* ■

Now we modify the reduction from Lemma 4.23 and argue that the resulting multi-tape TM $\mathbb{M}$ can be simulated by a single-tape machine $\mathbb{M}'$. This completes the desired reduction from p-ABD$(\Gamma, |M|)$ to p-Short-NSTM-Halt(κ), when $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$. We further claim that the blow-up in the size of $\mathbb{M}'$ is only in terms of $|M| = k$, and $\mathbb{M}'$ runs for $f(k)$ steps for some function f. We ignore negative clauses in ψ of length greater than k, as before. This implies that there are at most 2^k negative clauses. Moreover, ψ contains k positive clauses. Consequently, the number of tapes in $\mathbb{M}$ (see Lemma 4.23) is bounded by $k + 2^k + 1$. We argue that the size of $\mathbb{M}'$ is $\mathcal{O}(2^k \cdot p(|\psi|))$ where p is some polynomial. This is because there are $2^k + k + 1$ tapes in the worst case and therefore, the size of each transition (of $\mathbb{M}'$) is bounded by $\mathcal{O}(2^k)$. As a result, the size of $\mathbb{M}'$ is $\mathcal{O}(2^k \cdot |\mathbb{M}|)$ where $|\mathbb{M}| = \mathcal{O}(k \cdot |\psi|^2)$. Moreover, $\mathbb{M}'$ runs for $2^{2k} \cdot f(k)^2$ steps where $f(k)$ is the number of steps taken by $\mathbb{M}$ (for details of the simulation, see the textbook of Sipser [106, Theorem 7.8]). The correctness follows due to Claim 4.5 and Lemma 4.23. This completes the proof of the lemma by observing that the given translation is indeed an **FPT**-reduction. □

The translation presented in the proof of Lemma 4.29 poses an interesting question: can one always reduce an instance $(\mathbb{M}, k)$ of p-Short-NTM-Halt(κ) to an instance $(\mathbb{M}', k')$ of p-Short-NSTM-Halt(κ')? The answer to this question is clearly 'no'. Although one can achieve this translation in **FPT**-time, this is not always an **FPT**-reduction (see Def. 2.10) unless the number of tapes in $\mathbb{M}$ is also bounded by some function in k. In other words, the presence of a function g such that $k' \leq g(k)$ implies that the reduction between $\mathbb{M}$ and $\mathbb{M}'$ is an **FPT**-reduction. Indeed this is impossible in the general case unless the first two levels of the **W**-hierarchy collapse.

Now we argue that with some slight modifications, the translation from Lemma 4.29 also answers p-ABD$_{\leq}(\Gamma, |M|)$ if $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$.

Lemma 4.30. *Let $\ell \geq 2$ then the problem* p-ABD$_{\leq}(\Gamma, |M|)$ *is in* **W**[1] *if* $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$.

Proof. Let (V, H, M, KB, s, k) be an instance of p-ABD$_{\leq}(\Gamma, |M|)$, such that the task is to find an explanation of size at most s where $|M| = k$ is the parameter value. We argue that the reduction in Lemma 4.29 can be extended to solve the problem in this case. In Lemma 4.29 we proved that there is an explanation for p-ABD$(\Gamma, |M|)$ if and only if there is an explanation of size $|M|$ at most, if $\langle\Gamma\rangle \subseteq \mathsf{IS}^{\ell}_{10}$. Now we have the following two cases.

Case 1. If $k \leq s$, then the result holds already due to Lemma 4.29 (in particular Claim 4.5). This is because a solution of size at most k is also a solution of size at most s and there can be no solution of size in between k and s if there is no solution of size at most k.

Case 2. If $s < k$, the solution size is still bounded by the value of the parameter. Our reduction in the proof of Lemma 4.29 takes care of this change by producing two parameter values k_1 and k_2 such that $k_1 = k = |M|$ and $k_2 = s$. This refined reduction is still an **FPT**-reduction because the parameter value $k_1 + k_2$ is bounded by a function in k, that is, $k_1 + k_2 < 2k$. The rest of the reduction remains the same. The only difference now is that the machine (say $\mathbb{M}''$) guesses k_2 elements and not k_1 as $\mathbb{M}'$ does in Lemma 4.29.

This completes the proof to our lemma. □

Now we prove **FPT**-membership for p-ABD$_{\leq}(\Gamma, |M|)$, where $\langle\Gamma\rangle \subseteq \mathsf{IM}$, by reducing our problem to the parameterized MaxSATs problem which asks, given an instance I as a collection of m clauses, is it possible to set at most s variables to true so that at least $\kappa(I)$ clauses are satisfied. The problem p-MaxSATs(κ) (formally defined below) is **FPT** when κ is the minimum number of clauses to be satisfied [14, Prop. 4.3]. We alter the notation slightly to fit our setting.

Problem:	p-MaxSATs(κ)
Input:	A collection C of clauses, and $s, k \in \mathbb{N}$.
Parameter:	k (the number of clauses to be satisfied).
Question:	Does setting at most s variables true satisfies at least k clauses?

Lemma 4.31. *The problem* p-ABD$_{\leq}(\Gamma, |M|)$ *is* **FPT** *if* $\langle\Gamma\rangle \subseteq \mathsf{IM}$.

Proof. Given an instance (V, H, M, KB, s, k) with $KB = \bigwedge_{i \leq r} (x_i \rightarrow y_i)$ and $M = m_1 \wedge \ldots \wedge m_k$. Recall that each $m_i \in M$ can be explained by a single $h_i \in H$. If $k \leq s$, then there is nothing to prove. That is, there are fewer than s many sets of the form H_i each explaining an $m_i \in M$ (see the proof of Lemma 4.20). As a consequence, we need only select one $h_{i,j}$ from each H_i as the part of an explanation to yield a solution of size $\leq s$. Accordingly, assume that $k > s$. Proceed as in the proof of Lemma 4.20 and associate a set $H_i \subseteq H$ of hypotheses with each m_i, such that H_i explains m_i for $i \leq k$. It is enough to check whether selecting at most s many elements $h_i \in H$ can explain all the manifestations $m_i \in M$.

Let H' be the collection of all H_i's. For each i let C_i be the clause $\bigvee_{h_{i,j} \in H_i} h_{i,j}$. Furthermore, let C be the collection of all such clauses. Then C is built over variables in $V' = \bigcup_i H_i$. Our reduction maps an instance (V, H, M, KB, s, k) of p-ABD$_{\leq}(\Gamma, |M|)$ to an instance (C, s, k) of p-MaxSATs(κ). Note that we only have $|M| = k$ many clauses in C and, as a result, the question reduces to whether it is possible to set at most s variables from V' to satisfy every clause in C? Since each H_i can be computed in polynomial time, the whole computation takes polynomial time. Finally, the parameter value k of the two instances (C, s, k) and (V, H, M, KB, s, k) is the same, and consequently, the reduction is indeed an $\leq^{\mathbf{FPT}}$-reduction. This completes the proof. □

It is easy to observe that the **FPT**-membership from Lemma 4.31 can be extended to languages Γ such that $\langle\Gamma\rangle \subseteq \mathsf{IV}_2$. This is achieved using the same argument as in the proof of Lemma 4.21 regarding the type of clauses in the knowledge base KB.

Corollary 4.32. *The problem* p-ABD$_{\leq}(\Gamma, |M|)$ *is* **FPT** *if* $\langle\Gamma\rangle \subseteq \mathsf{IV}_2$.

Proof. After applying unit propagation and resolution, we can ignore positive clauses of length ≥ 2 and clauses with one negative literal of length ≥ 3. □

Finally, we present the intractability of p-ABD$_{=}(\Gamma, |M|)$ for the majority of the cases.

Lemma 4.33. *The problem* p-ABD$_=(\Gamma, |M|)$ *is* **paraNP***-hard, for any CL* Γ *such that* $\neg x \vee \neg y \in \langle\Gamma\rangle_{\neq}$.

Proof. We prove that the 1-slice of the problem is **NP**-hard by reducing from IndependentSet (which is **NP**-complete [67]) to p-ABD$_=(\Gamma, |M|)$ with $|M| = 1$. The reduction is essentially the classical counterpart of the one presented in Lemma 4.24. Let $\mathcal{G} := (\tilde{V}, \tilde{E})$ be an instance of IndependentSet, we map it to (V, H, M, KB, s), where $V = H := \tilde{V} \cup \{z\}$, $M := z$,

$$KB := \bigwedge_{(x,y)\in\tilde{E}} (\neg x \vee \neg y),$$

and $s := k + 1$. Then $\mathcal{G}$ admits an independent set of size k if and only if (V, H, M, KB, s) admits an explanation of size s. □

Notice that we did not mention the base independence for essentially negative languages in the previous proof. This is because, the **paraNP**-membership as well as the base independence already holds for p-ABD$_=(\Gamma, |M|)$, when $\langle\Gamma\rangle \subseteq \mathsf{C} \in \{\mathsf{IE}_2, \mathsf{ID}_2\}$. This gives the desired results for essentially negative languages as well.

We conclude this chapter by pointing out that the Galois connection (Prop. 2.1) is true for p-ABD$_*(\Gamma, \kappa)$ under **FPT**-reductions for each constraint language Γ and parameterization κ. Figure 4.1 presents an overview of our results for ABD$_*(\Gamma, \kappa)$ for each CL Γ and κ.

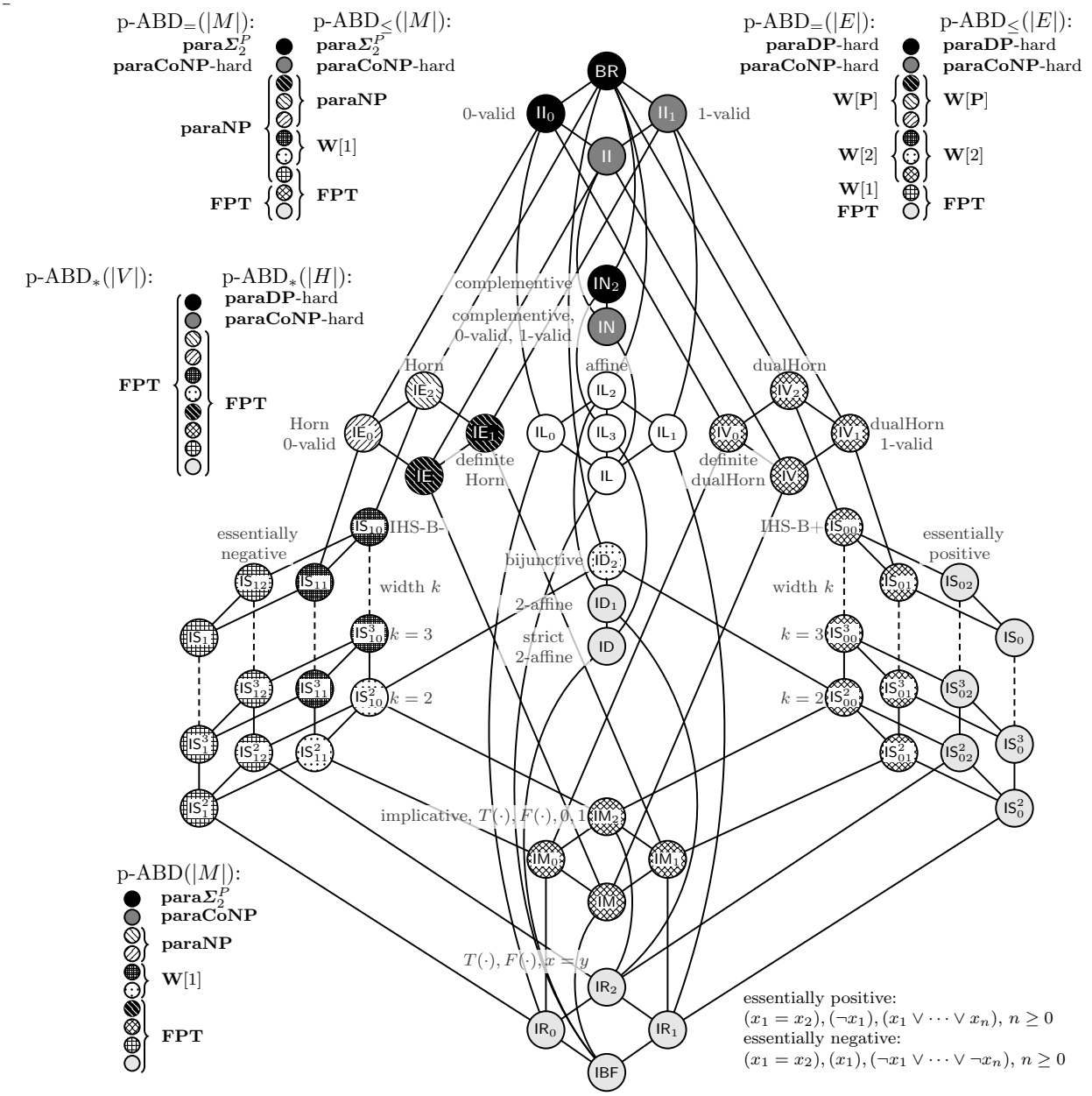


Figure 4.1: Complexity landscape of abductive reasoning with respect to each parameter $\kappa \in \{|M|, |H|, |V|, |E|\}$. White colouring means unclassified. ABD$_\star$ means same result for all three variants. Each result depicts completeness for the mentioned class except for **FPT**-cases, or specified otherwise.

CHAPTER 5

LOGICAL INFERENCE: ARGUMENTATION

In this chapter, we explore the parameterized complexity of problems in logic-based argumentation concerning Schaefer's framework. The problems of interest are (1) to find a support Φ for a claim α (ARG), (2) to determine whether the pair (Φ, α) is an argument (ARG-Check), and (3) to determine whether a formula ψ is *relevant* for the support of an argument (ARG-Rel). The proofs in this chapter also use the technical expressibility results from Chapter 4. Moreover, we state and prove a few more expressibility results (regarding inequality) that we need in this chapter. In the second section, we prove that the tautology problem for any constraint language Γ can be decided in polynomial time. This is followed by the parameterized complexity analysis for the implication problem. For problems in argumentation, we consider each parameter in a separate subsection. Notice that, an input instance for the problems considered in this chapter (ARG, IMP, etc.) often consists of more than one *types of* formulas. For example, an instance of IMP(Γ) is the pair (Φ, α). This yields two variants of the parameterization 'number of variables', namely $|\mathsf{var}(\Phi)|$ and $|\mathsf{var}(\alpha)|$. We conclude the chapter by a discussion on the base independence (Property 2.1) in the parameterized setting for each considered problem. Figure 5.1 present a complexity overview for problems in ARG.

5.1 Technical Implementation Results

We begin by proving some further expressivity results that are essential in achieving certain reductions for problems in argumentation.

Lemma 5.1. *Let Γ be a CL that is neither ε-valid, nor* EP, *nor* EN. *Then, if Γ is*

1. *neither Horn, nor dualHorn, nor complementive, then $(x \neq y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\exists,\neq}$,*

2. *complementive, but neither Horn, nor dualHorn, then* $(x \neq y) \in \langle\Gamma\rangle_{\exists,\neq}$, *and*

3. *Horn or dualHorn, then* $(x = y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\exists,\neq}$.

Proof. This follows immediately from the proof of Lemma 4.2. The proof makes a case distinction according to whether Γ is 0- and/or 1-valid. In the case of non ε-valid Γ, a further case distinction is made according to Γ being Horn and/or dualHorn. □

Let $\mathrm{T} = \{(1)\}$ and $\mathrm{F} = \{(0)\}$ denote the unary relations. The relation T (resp., F) implements true (false). Then we have the following implementation results.

Proposition 5.2 (Creignou et al. [23])**.** *If* Γ *is a CL that is*

1. *complementive but not* ε*-valid, then* $(x \neq y) \in \langle\Gamma\rangle_{\exists,\neq}$,

2. *neither complementive, nor* ε*-valid, then* $(t \wedge \neg f) \in \langle\Gamma\rangle_{\exists,\neq}$,

3. 1*-valid but not* 0*-valid, then* $\mathrm{T} \in \langle\Gamma\rangle_{\exists,\neq}$,

4. 0*-valid but not* 1*-valid, then* $\mathrm{F} \in \langle\Gamma\rangle_{\exists,\neq}$, *and*

5. 0*-valid and* 1*-valid, then* $(x = y) \in \langle\Gamma\rangle_{\exists,\neq}$.

5.2 Implication and Tautology

In this section, we consider the parameterized complexity of the tautology (TAUT) and the implication problem (IMP). These two problems play a central role in proving results for argumentation. The problem TAUT is defined as follows.

Problem:	TAUT(Γ) — Γ is a constraint language
Input:	A Γ-formula ϕ.
Question:	Is ϕ a tautology?

Since TAUT has not been considered before in Schaefer's framework, we first discuss its classical complexity. The following notation is essential in formalizing the proofs in this section. We say that a relation R is *full with respect to a constraint* $R(x_1, \ldots, x_k)$ if each assignment s over $\{x_1, \ldots, x_k\}$ yields a tuple in R. Furthermore, a k-ary relation R is said to be *full* if $R = \{0,1\}^k$. It is worth mentioning that the variables $x_1, \ldots, x_k$ in the constraint $R(x_1, \ldots, x_k)$, are not necessarily distinct. Finally, a constraint $R(x_1, \ldots, x_k)$ is a tautology if the relation R is full with respect to $R(x_1, \ldots, x_k)$. We elaborate these concepts with the help of an example.

Example 5.3. *Let* $\phi(x) = R(x, x)$ *be a constraint where* $R = \{00, 11\}$. *Then* R *is full with respect to* $R(x, x)$, *however,* R *is not full with respect to* $R(x, y)$. *As a consequence,* $\phi(x)$ *is a tautology, whereas,* $\psi(x, y) = R(x, y)$ *is not.* ◁

As Example 5.3 demonstrates, a constraint can be a tautology, even though the underlying relation is not full. We now prove that the tautology problem (for ay CL) can be solved in polynomial time.

Lemma 5.4. *Let* Γ *be any constraint language. Then,* $\mathrm{TAUT}(\Gamma) \in \mathbf{P}$.

Proof. Let a ϕ be a Γ-formula given as a conjunction of Γ-constraints. It suffices to check whether each constraint $R(x_1, \ldots, x_k)$ in ϕ is a tautology. Determining if $R(x_1, \ldots, x_k)$ is a tautology, amounts to verifying whether the relation R is full with respect to $R(x_1, \ldots, x_k)$. It is possible to check in constant time whether R is full with respect to $R(x_1, \ldots, x_k)$ because of the following observation. The maximum arity of the underlying relations in Γ is fixed, this implies that the arity of R is constant. As a result, the number of distinct variables in $R(x_1, \ldots, x_k)$ is also constant. Consequently, one can determine whether each constraint in ϕ is a tautology in time $\mathcal{O}(|\phi|)$. □

Let ϕ be a Γ-formula and s be an assignment. Then, $\phi[s]$ denotes the *reduct of* ϕ *under* s. That is, the formula obtained from ϕ by instantiating each variable $x \in \mathrm{dom}(s)$ by $s(x)$. Notice that $\phi[s]$ is a Γ-formula with constants from $\{0, 1\}$. Moreover, the notion of a relation R being full with respect to a constraint $R(\alpha_1, \ldots, \alpha_k)$ can be easily lifted to the case where some of the $\alpha_i \in \{0, 1\}$ are constants and other α_j's are variables. Formally, a relation R is full with respect to $R(\alpha_1, \ldots, \alpha_k)$, if $R \supseteq S$, where

$$S = \left\{ (a_1, \ldots, a_k) \in \{0,1\}^k \;\middle|\; a_i = \alpha_i \text{ if } \alpha_i \in \{0,1\}, i \leq k \right\}.$$

In other words, each tuple in R must agree with constants for specified positions. We give an example below for better understanding of the notation.

Example 5.5. *Let* $\phi = R(x, y, z)$ *be a constraint where* $R = \{000, 100, 101, 110, 111\}$ *and let* s *be an assignment such that* $s(x) = 0$*. Clearly,* $\phi[s] = R(0, y, z)$ *and* R *is not full with respect to* $R(0, y, z)$ *since* 000 *is the only allowed tuple from* R *(that agree with* $s(x) = 0$*). However, considering* $s(x) = 1$ *implies that* R *is full with respect to* $R(1, y, z)$*.* ◁

We strengthen Lemma 5.4 to cases when constraints can also take constant values. In other words, we prove that determining whether $\phi[s]$ is a tautology for a formula ϕ and some assignment s can be solved in polynomial time. Keep in mind that $\phi[s]$ is a tautology, if and only if $\theta \models \phi[s]$ for every assignment θ over $\mathrm{Vars}(\phi) \backslash \mathrm{dom}(s)$.

Lemma 5.6. *Let* Γ *be any CL,* ϕ *be a* Γ*-formula, and* s *be an assignment. Then determining whether* $\phi[s]$ *is a tautology can be solved in polynomial time.*

Proof. The task is to determine whether each constraint $R(\alpha_1, \ldots, \alpha_k)$ in $\phi[s]$ is a tautology. Moreover, this is true if each relation R is full with respect to the constraint $R(\alpha_1, \ldots, \alpha_k)$. As a result, in the worst case, one needs to verify that all 2^k-many tuples are present in R. A similar argument as in the proof of Lemma 5.4 implies that this can be achieved in constant time (as k is bounded by a constant). This gives a running time linear in $|\phi|$ and membership in $\mathbf{P}$. □

Now we explore the parameterized complexity of implication (IMP) in Schaefer's framework. Let Φ denote a set of Γ-formulas α be a Γ-formula.

Problem:	p-IMP(Γ, κ)
Input:	(Φ, α, k).
Parameter:	k.
Question:	Is $\Phi \models \alpha$ true?

For p-IMP(Γ, κ), we consider three parameters $\kappa \in \{|\Phi|, |\mathsf{var}(\Phi)|, |\mathsf{var}(\alpha)|\}$ (see Lemma 4.4). Where, $|\Phi|$ denotes the number of formulas in Φ and $|\mathsf{var}(\chi)|$ is the number of variables in $\chi \in \{\Phi, \alpha\}$. The following corollary is due to Schnoor and Schnoor [104, Theorem 6.5].

Corollary 5.7. *Let Γ be a CL.* IMP(Γ) *is in* **P** *when Γ is Schaefer and* **CoNP***-complete otherwise.*

Observe that **P**-membership (resp., **CoNP**) of IMP(Γ) from Corollary 5.7 can be adapted to the case when Φ is a set of formulas. This is due to the reason that $\Phi \models \alpha$ iff $\Psi \models \alpha$ where $\Psi := \bigwedge_{\phi \in \Phi} \phi$. Consequently, the parameterized problem p-IMP(Γ, κ) is **FPT** when Γ is Schaefer for each $\kappa \in \{|\Phi|, |\mathsf{var}(\Phi)|, |\mathsf{var}(\alpha)|\}$. Now we consider the cases when Γ is non-Schaefer. Clearly, p-IMP(Γ, κ) is in **paraCoNP** for each $\kappa \in \{|\Phi|, |\mathsf{var}(\Phi)|, |\mathsf{var}(\alpha)|\}$. In the following, we differentiate the restrictions on Φ from the ones on α. In other words, we introduce a technical variant, IMP(Γ', Γ) of the (classical) implication problem. An instance of IMP(Γ', Γ) is a tuple (Φ, α), where Φ is a set of Γ'-formulas and α is a Γ-formula. The following corollary also follows from the work of Schnoor and Schnoor [104, Theorem 6.5]. This is essential in achieving the base independence (Property 2.1) for the implication problem.

Corollary 5.8. *Let Γ and Γ' be non-Schaefer CLs. If $\Gamma' \subseteq \langle \Gamma \rangle$ then* IMP$(\Gamma', \Gamma) \leq_m^{\mathbf{P}}$ IMP(Γ).

Regarding non-Schaefer CLs, it turns out that the parameter $\kappa \in \{|\Phi|, |\mathsf{var}(\alpha)|\}$ does not help in achieving tractability. The hardness of p-IMP$(\Gamma, |\Phi|)$ when Γ is non-Schaefer, follows from Corollary 5.7 because instances of the implication problem considered by Schnoor and Schnoor are pairs (ϕ, α) (that is, $|\Phi| = 1$).

Corollary 5.9. p-IMP$(\Gamma, |\Phi|)$ *is* **paraCoNP***-complete when Γ is a non-Schaefer CL.*

The hardness of p-IMP$(\Gamma, |\mathsf{var}(\alpha)|)$ is established in the following lemma. The underlying idea of the reduction is that the formulas in Φ do not necessarily share a set of variables with α.

Lemma 5.10. p-IMP$(\Gamma, |\mathsf{var}(\alpha)|)$ *is* **paraCoNP***-complete if Γ is a non-Schaefer CL.*

Proof. To achieve the lower bounds, we reduce from the unsatisfiability problem (UNSAT). UNSAT(Γ) asks whether an input Γ-formula ϕ is unsatisfiable. Moreover, checking unsatisfiability of a Γ-formula is **CoNP**-complete, if Γ is non-Schaefer (follows by Schaefer's [103] SAT classification).

We inherently use Corollary 5.8 and make a case distinction as whether (Φ, α) is 1-valid, 0-valid or complementive (according to sub-cases of Proposition 5.2). Moreover, in each case, we prove that for some well chosen languages Γ' and Γ, the problem p-IMP$(\Gamma', \Gamma, |\mathsf{var}(\alpha)|)$ is **paraCoNP**-hard. The correctness in each case follows by observing that the variables of α do not appear in ϕ. As a result, if there is an assignment s such that $s \models \phi$, then the assignment s' that extends s in such a way that $s' \not\models \alpha$ in each case, yields a contradiction.

Γ **is complementive but not** ε**-valid.** According to the first item in Proposition 5.2, we have $(x \neq y) \in \langle \Gamma' \rangle_{\exists,\neq}$. The desired reduction follows as we take ϕ to be a Γ'-formula and $\alpha := (x \neq x)$ for a fresh variable x.

Γ **is neither complementive, nor** ϵ**-valid.** According to item (2). in Proposition 5.2, we have $(x \wedge \neg y) \in \langle \Gamma' \rangle_{\exists,\neq}$. As before, we take ϕ to be a Γ'-formula and $\alpha := (x \wedge \neg y)$ for fresh variables x and y.

Γ **is** 1**-valid but not** 0**-valid.** According to item (3). in Proposition 5.2, we have $\mathrm{T} \in \langle \Gamma' \rangle_{\exists,\neq}$. Let ϕ be a Γ'-formula and $\alpha := \mathrm{T}(x)$ for a fresh variable x.

Γ **is** 0**-valid and not** 1**-valid.** According to item (4). in Proposition 5.2, we have $\mathrm{F} \in \langle \Gamma' \rangle_{\exists,\neq}$. Similar to the previous case, we let ϕ be a Γ'-formula and $\alpha := \mathrm{F}(x)$ for a fresh variable x.

Γ **is** 0**- and** 1**-valid.** According to item (5). in Proposition 5.2, we have $(x = y) \in \langle \Gamma' \rangle_{\exists,\neq}$. We let ϕ to be a Γ'-formula and $\alpha := (x = y)$ for fresh variables x and y.

Clearly, $\phi \models \alpha$ if and only if ϕ is unsatisfiable in each subcase. The **paraCoNP**-hardness follows by noticing that α in each reduction has a constant size. This completes the proof. □

Observe that $(x = y) \in \langle \Gamma \rangle_{\neq}$ if Γ is not Schaefer (Lem. 4.2). Nevertheless, we can not simply let $\alpha := (x = y)$ in the proof of Lemma 5.10, and rather use different constraints for each subcase. This is due to the reason that we can not bound the size of the constraints in $\langle \Gamma \rangle_{\neq}$ that implement equality, whereas, in the proof we need that $|\mathsf{var}(\alpha)|$ is fixed. Finally, observe that $\Phi = \{\phi\}$ (and $|\Phi| = 1$) is true in each reduction of Lemma 5.10. This yields the following corollary.

Corollary 5.11. p-IMP(Γ, κ) *is* **paraCoNP**-*complete when* $\kappa = (|\mathsf{var}(\alpha)| + |\Phi|)$, *if* Γ *is non-Schaefer.*

The following theorem settles the parameterized complexity of the implication problem under the parameterization $|\mathsf{var}(\Phi)|$.

Theorem 5.12. p-IMP$(\Gamma, |\mathsf{var}(\Phi)|) \in$ **FPT** *for any constraint language* Γ.

Proof. We construct an **FPT**-algorithm deciding p-IMP$(\Gamma, |\mathsf{var}(\Phi)|)$. Let (Φ, α, k) be an instance of p-IMP$(\Gamma, |\mathsf{var}(\Phi)|)$, where Φ is a Γ-formula and $|\mathsf{var}(\Phi)| = k$ is the parameter value. Clearly, $\Phi \models \alpha$ if and only if $\alpha[s]$ is a tautology for each $s \in \{\, s \mid s \in 2^{\mathrm{Vars}(\Phi)} \text{ and } s \models \Phi \,\}$. Consequently, the entailment problem can be reduced to 2^k-many questions asking whether the resulting formula is a tautology, which can be solved in polynomial time (Lemma 5.6). This proves the desired **FPT**-membership. □

We conclude this section by noting that the Galois connection is also true for the problem p-IMP(Γ, κ), for any CL Γ and parameterization $\kappa \in \{|\Phi|, |\mathsf{var}(\Phi)|, |\mathsf{var}(\alpha)|\}$.

5.3 Logic-Based Argumentation

In this section, we explore the parameterized complexity of various problems in logic-based argumentation concerning the Schaefer's framework. From this point on, Δ and Φ denote a collection of Γ-formulas, and α denotes a Γ-formula.

Problem:	p-ARG(Γ, κ)
Input:	(Δ, α, k).
Parameter:	k.
Question:	Is there a set $\Phi \subseteq \Delta$ s.t. (Φ, α) is an argument in Δ?

Two further problems of interest are argument verification (ARG-Check) and the relevance problem (ARG-Rel).

Problem:	p-ARG-Check(Γ, κ)
Input:	(Φ, α, k).
Parameter:	k.
Question:	Is (Φ, α) an argument?

Problem:	p-ARG-Rel(Γ, κ)
Input:	$(\Delta, \alpha, \psi, k)$ where $\psi \in \Delta$.
Parameter:	k.
Question:	Is there $\Phi \subseteq \Delta$ s.t. $\psi \in \Phi$ & (Φ, α) is an argument in Δ?

A Note on Parameterizations

It is worth pointing out that the set of variables V is not included in an instance of any problem in argumentation, contrary to the case of problems in abduction. Nevertheless, recall from Lemma 4.4 that the problems ARG, ARG-Check, ARG-Rel parameterized by $|\mathsf{enc}(\chi)|$ and $|\mathsf{var}(\chi)|$ are **FPT**-equivalent for any $\chi \in \{\Delta, \Phi, \alpha\}$. As a result, we choose to write $|\mathsf{var}(\chi)|$ for $\chi \in \{\Delta, \Phi\}$, but $|\alpha|$ rather than $|\mathsf{var}(\alpha)|$ for proving results.[1]

An instance of the argumentation problem ARG(Γ) is the pair (Δ, α) where Δ is a collection of Γ-formulas, and α is a Γ-formula. An instance of ARG-Rel(Γ) constitutes a tuple (Δ, α, ψ) and uses an additional formula ψ in the input. Meaningful parameters arising from both these instances are $\kappa \in \{|\Delta|, |\mathsf{var}(\Delta)|, |\alpha|\}$. Moreover, an instance of ARG-Check(Γ) is a tuple (Φ, α) This also

[1] Recall from Chapter 2: $|\chi|$ denotes the number of elements in χ for a set χ, and the encoding length if χ is not a set.

yields three parameters $\kappa \in \{|\Phi|, |\mathsf{var}(\Phi)|, |\alpha|\}$ for ARG-Check. Notice that the parameter $|\Delta|$ for ARG-Check and $|\Phi|$ for ARG are not meaningful since an input instance does not include them. The parameter treewidth was also considered by Mahmood et al. [44] for all three problems in argumentation.

5.3.1 Parameter 'size of the claim '$|\alpha|$

In this section, we discuss the complexity results regarding the parameter α. This includes the number of variables and the encoding size of α. It turns out that the computational complexity of the argumentation problems is hidden in the structure of the underlying CL. That is, in many cases, considering the claim-size as a parameter does not lower the complexity. This is proved by noting that certain slices of the parameterized problems already yield hardness results.

Theorem 5.13. *Let* Γ *be a CL. Then,* p-ARG$(\Gamma, |\alpha|)$ *is*

1. **paraΣ_2^P**-*complete if* Γ *is neither Schaefer, nor* ε*-valid,* [2]
2. **paraCoNP**-*complete if* Γ *is not Schaefer but* ε*-valid,*
3. **paraNP**-*complete if* Γ *is Schaefer but neither* ε*-valid, nor* EP^s*, nor* EN^s*,*
4. $\mathbf{W}[1]$*-complete if* Γ *is either* EP^s *or* EN^s *but not* ε*-valid, and*
5. **FPT** *if* Γ *is Schaefer and* ε*-valid.*

Proof. For (1). (resp., (2)) The membership follows because the classical problem is in $\mathbf{\Sigma}_2^P$ (resp., **CoNP**) [22, Thm 5.3]. For hardness of p-ARG$(\Gamma, |\alpha|)$ when Γ is ε-valid, notice that, since Δ is ε-valid, an instance (Δ, α, k) of p-ARG$(\Gamma, |\alpha|)$ admits an argument if and only if $\Delta \models \alpha$. The result follows from Lemma 5.10 because the implication problem p-IMP$(\Gamma, |\alpha|)$ is still **paraCoNP**-hard. Finally, when Γ is not Schaefer and not ε-valid, in the proofs of Creignou et al. [22, Prop. 5.2] the constructed reductions yields the claim α that contains either two or three variables (depending on subcases). Accordingly, either the 2-slice or the 3-slice of p-ARG$(\Gamma, |\alpha|)$ is $\mathbf{\Sigma}_2^P$-hard. This gives the desired hardness result.

(3). The upper bound follows because the unparameterized problem ARG(Γ) is in **NP** [22, Prop 5.1]. The lower bounds are established in Lemmas 5.16, 5.17 and 5.18 by a case distinction.

(4). The membership is proven in Lemma 5.19 and the hardness in Lemma 5.20.

(5). The classical problem ARG(Γ) is already in **P** for these cases [22, Thm 5.3]. □

Intermediate Lemmas

For technical reasons, we introduce the following variant of the (classical) argumentation existence problem (ARG). This helps in achieving the base independence for ARG.

[2] For theorems related to problems in abduction we specified different Schaefer languages, such as IE and IV, since the complexity differed for each of them.

Problem:	ARG(Γ, R)
Input:	A set Δ of Γ-formulas and an R-formula α.
Question:	Is there a $\Phi \subseteq \Delta$ s.t. (Φ, α) is an argument in Δ?

Lemma 5.14. *Let Γ, Γ' be two CLs and R be a Boolean relation. If $\Gamma' \subseteq \langle\Gamma\rangle_{\neq}$ and $R \in \langle\Gamma\rangle_{\not\exists,\neq}$, then* ARG$(\Gamma', R) \leq_m^{\mathbf{P}}$ ARG(Γ).

Proof. Let (Δ, α) be an instance of ARG(Γ', R), where $\Delta = \{\, \delta_i \mid i \in I \,\}$ is a collection of Γ'-formulas and $\alpha = R(x_1, \ldots, x_k)$. We map this instance to an instance (Δ', α') of ARG(Γ), where $\Delta' = \{\delta_i' \mid \delta_i \in \Delta\}$ and α' is a Γ-formula equivalent to $R(x_1, \ldots, x_k)$ (which exists because $R \in \langle\Gamma\rangle_{\not\exists,\neq}$). For each i, we obtain δ_i' by replacing δ_i by an equivalent Γ-formula (such a representation exists since $\Gamma' \subseteq \langle\Gamma\rangle_{\neq}$) and deleting all existential quantifiers. □

The previous result can be easily extended in the parameterized setting, as depicted in the following corollary.

Corollary 5.15. *Let Γ, Γ' be two CLs, and R be a Boolean relation. If $\Gamma' \subseteq \langle\Gamma\rangle_{\neq}$ and $R \in \langle\Gamma\rangle_{\not\exists,\neq}$, then* p-ARG$(\Gamma', R, \kappa) \leq^{\mathbf{FPT}}$ p-ARG(Γ, κ) *for any $\kappa \in \{|\Delta|, |\alpha|\}$.*

Proof. Notice that the translation in Lemma 5.14 respects the parameter values as indicated in Definition 2.10. This is because Δ' and Δ have the same number of formulas. Moreover, α' has the number of variables bounded by those in α since no new variable is introduced. Finally, the result holds due to Lemma 4.4. □

The proof of Lemmas 5.16 and 5.17 is achieved via a reduction from the classical problem Pos-1-In-3-Sat. We additionally use the fact that the parameter value $|\alpha|$ in the reduced instance (Δ, α) is constant in each case. This gives the **paraNP**-hardness using Proposition 2.13 and the fact that Pos-1-In-3-Sat is **NP**-complete [103]. An instance of Pos-1-In-3-Sat is a 3CNF-formula with only positive literals, the question is to determine whether there is a satisfying assignment which maps exactly one variable in each clause to true.

Problem:	Pos-1-In-3-Sat
Input:	A 3CNF $\mathcal{PL}$-formula ϕ with only positive literals.
Question:	Is there an assignment s such that $s \models \phi$ and $s(x) = 1$ for exactly one x in each clause?

Lemma 5.16. *Let Γ be a CL such that Γ is Schaefer. If Γ is neither affine, nor ε-valid, nor* EP, *nor* EN, *then* p-ARG$(\Gamma, |\alpha|)$ *is* **paraNP***-hard.*

Proof. We prove that the c-slice of p-ARG$(\Gamma, |\alpha|)$ is **NP**-hard for a constant $c \in \mathbb{N}$. To achieve this, we give a reduction from Pos-1-In-3-Sat to ARG(Γ) such that $|\alpha|$ is constant. Furthermore, we make a case distinction according to the case (1) and (3) in Lemma 5.1. Case (2) is not needed because if a Schaefer CL Γ is neither affine, nor horn, nor dualHorn, then Γ can not be

complementive. We first treat case (3), that is, we have that $(x = y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\not\exists,\neq}$. We then show that the other two cases can be treated with minor modifications of the procedure.

Let ϕ be an instance of Pos-1-In-3-Sat. We first reduce ϕ to an instance (Δ, α) of ARG$(\{\mathrm{T}, \mathrm{F}, =\}, (x = y) \wedge t \wedge \neg f)$, and then conclude with Lemma 5.1 and Corollary 5.15. Let $\phi = \bigwedge_{i=1}^{n}(x_i \vee y_i \vee z_i)$ be an instance of Pos-1-In-3-Sat and let $t, f, c_1, \ldots, c_{n+1}$ be fresh variables. We define Δ and α as follows:

$$\begin{aligned}\Delta = &\bigcup_{i=1}^{n}\{x_i \wedge \neg y_i \wedge \neg z_i \wedge (c_i = c_{i+1}) \wedge t \wedge \neg f\}\\ &\cup \bigcup_{i=1}^{n}\{\neg x_i \wedge y_i \wedge \neg z_i \wedge (c_i = c_{i+1}) \wedge t \wedge \neg f\}\\ &\cup \bigcup_{i=1}^{n}\{\neg x_i \wedge \neg y_i \wedge z_i \wedge (c_i = c_{i+1}) \wedge t \wedge \neg f\},\\ \alpha = &(c_1 = c_{n+1}) \wedge t \wedge \neg f.\end{aligned}$$

Clearly, there is a one-to-one correspondence between the satisfying assignments for ϕ that map exactly one variable in each clause to true, and subsets $\Phi \subseteq \Delta$ such that Φ is support for α in Δ. That is, for each $i \leq n$ a satisfying assignment s sets a variable $v_i \in \{x_i, y_i, z_i\}$ to true if and only if Φ contains the formula in which v_i appears positively.

Note that any formula in Δ is expressible as a Γ-formula since $\{\mathrm{T}, \mathrm{F}, =\} \subseteq \mathsf{IM}_2 \subseteq \langle\Gamma\rangle$ (cf. Table 2.2). Moreover, $\langle\Gamma\rangle_{\neq} = \langle\Gamma\rangle$ (Prop. 4.2) and by construction $(x = y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\not\exists,\neq}$, we have, the desired reduction to p-ARG$(\Gamma, |\alpha|)$ (Cor. 5.15) with $|\mathsf{var}(\alpha)| = 4$ and the claim follows.

For case (1) of Lemma 5.1 we have that $(x \neq y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\not\exists,\neq}$. To cope with this change in the reduction we introduce one additional variable d and replace α by $(c_1 \neq d) \wedge (d \neq c_{n+1}) \wedge t \wedge \neg f$. This completes the proof to our lemma. □

Lemma 5.17. *Let Γ be a CL such that Γ is Schaefer. If Γ is affine but neither ε-valid, nor* EP*, nor* EN*, then* p-ARG$(\Gamma, |\alpha|)$ *is* **paraNP***-hard.*

Proof. We proceed analogously to the proof of Lemma 5.16 and give a reduction from Pos-1-In-3-Sat such that $|\alpha|$ is constant. This time, we make a case distinction according to cases (1) and (2) in Lemma 5.1 (case 3 can not occur for Γ is affine and not EP). First, we treat the second case, that is, we have that $(x \neq y) \in \langle\Gamma\rangle_{\not\exists,\neq}$, and show that the first case can be treated with minor modifications in the reduction.

We reduce an instance ϕ of Pos-1-In-3-Sat to an instance (Δ, α) of ARG$(\{=, \neq\}, \{\neq\})$, and then conclude with Lemma 5.1 and Corollary 5.15. Let $\phi = \bigwedge_{i=1}^{k}(x_i \vee y_i \vee z_i)$ be an instance of Pos-1-In-3-Sat and let $t, d, c_1, \ldots, c_{k+1}$ be fresh variables. Define Δ and α as follows:

$$\begin{aligned}\Delta = &\bigcup_{i=1}^{k}\{(x_i = t) \wedge (y_i \neq t) \wedge (z_i \neq t) \wedge (c_i = c_{i+1})\}\\ &\cup \bigcup_{i=1}^{k}\{(x_i \neq t) \wedge (y_i = t) \wedge (z_i \neq t) \wedge (c_i = c_{i+1})\}\\ &\cup \bigcup_{i=1}^{k}\{(x_i \neq t) \wedge (y_i \neq t) \wedge (z_i = t) \wedge (c_i = c_{i+1})\},\\ \alpha = &(c_1 \neq d) \wedge (d \neq c_{k+1}).\end{aligned}$$

The correctness follows due to the similar argument as in the proof of Lemma 5.16. Note that any formula in Δ is expressible as Γ-formula since $\{=, \neq\} \subseteq \mathsf{ID} \subseteq \langle\Gamma\rangle$ (cf. Table 2.2). Since by

Proposition 4.2 $\langle\Gamma\rangle_{\neq} = \langle\Gamma\rangle$ and by construction $(x \neq y) \in \langle\Gamma\rangle_{\not\exists,\neq}$, we have by Corollary 5.15 the desired reduction to p-ARG$(\Gamma, |\alpha|)$ where $|\alpha| = 4$.

For case (1) of Lemma 5.1 we have that $(x \neq y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\not\exists,\neq}$. To cope with this change in the reduction, we introduce one additional variable f and add the constraints $t \wedge \neg f$ to α as well as to every formula in Δ. This proves the claim by noting that the parameter size $|\alpha|$ is still constant. □

Lemma 5.18. *Let Γ be a CL such that Γ is Schaefer but not ε-valid. If Γ is* EP *and not* EP^s *or* EN *and not* EN^s*, then* p-ARG$(\Gamma, |\alpha|)$ *is* **paraNP***-hard.*

Proof. We can use the same reduction as in Lemma 5.16, except we do not require a case distinction. Note that, by Proposition 5.2, we have that $(t \wedge \neg f) \in \langle\Gamma\rangle_{\not\exists,\neq}$. Since $\exists f\,(t \wedge \neg f) \equiv \mathrm{T}(t)$ and $\exists t\,(t \wedge \neg f) \equiv \mathrm{F}(f)$, we conclude that $\mathrm{T}, \mathrm{F} \in \langle\Gamma\rangle_{\neq}$. Further, by Lemma 4.3, we have that $(x = y) \in \langle\Gamma\rangle_{\neq}$. Together we have $\{\mathrm{T}, \mathrm{F}, =\} \subseteq \langle\Gamma\rangle_{\neq}$, and thus any formula in Δ is expressible as Γ-formula with existential quantifiers but without equality. By Lemma 4.3, it follows that $(x = y) \wedge t \wedge \neg f \in \langle\Gamma\rangle_{\not\exists,\neq}$. As a result, we conclude by applying Corollary 5.15. □

Now we prove $\mathbf{W}[1]$-completeness of p-ARG$(\Gamma, |\alpha|)$ if Γ is either EP^s or EN^s but not ϵ-valid.

Lemma 5.19. *Let Γ be a CL that is either* EP^s *or* EN^s *but not ε-valid, then* p-ARG$(\Gamma, |\alpha|) \in \mathbf{W}[1]$.

Proof. We reduce p-ARG$(\Gamma, |\alpha|)$ to p-WSAT$(\Gamma_{1,d}, \kappa)$ for some fixed d. That is, given a $\Gamma_{1,d}$-formula ϕ, determine if there is a satisfying assignment for ϕ of weight $\kappa(\phi) = k$. This problem is $\mathbf{W}[1]$-complete as depicted in Proposition 2.12. Moreover, we only prove the statement for Γ that is EP^s. The other case is proven analogously. Since we consider only finite constraint languages we have that $\Gamma \subseteq \mathsf{IS}^r_{02}$ for some $r \geq 2$. Therefore, any Γ-formula can be written as a conjunction of positive or negative literals and positive clauses of size at most r.

Let (Δ, α, k) be an instance of p-ARG$(\Gamma, |\alpha|)$ with $\Delta = \{\delta_1, \ldots, \delta_n\}$ and $\alpha = \bigwedge_{i \leq k} \alpha_i$, where each α_i is either a literal, or a positive clause of size $\leq r$. First, we prove the following two claims.

Claim 5.1. *For any $i \leq k$, if α_i has a support at all, then it has a support containing at most r formulas from Δ. Consequently, there is support for α iff there is a one of size at most $r \cdot k$.*

Proof of Claim. *If α_i is a negative literal, then at most 1 formula from Δ is sufficient (one that contains α_i). If α_i is a positive literal, then at most r formulas from Δ are sufficient. This is due to the reason that in the worst case: one $\delta_i \in \Delta$ containing a (positive) clause which contains α_i, then we need at most $r - 1$ more δ_j's to force all other variables in the clause to 0. If α_i is a positive clause, then it suffices to explain one variable from that clause, as a result, at most r formulas from Δ are sufficient as in the previous case. In other words, there is no α_i in α such that more than r formulas from Δ are necessary to explain α_i.* ■

Claim 5.2. *Δ is consistent if and only if every subset $\Delta' \subseteq \Delta$ of size $r + 1$ is consistent .*

Proof of Claim. *This is similar to Claim 5.1. The worst case to create an inconsistency is to take a formula $\delta \in \Delta$ containing a positive clause C of size r and then formulas $\delta_j \in \Delta$ for $j \leq r$, forcing together all variables in C to 0.* ■

In accordance with Claim 5.1, let $A_i^1, \ldots, A_i^{m_i}$ denote all subsets of Δ of size at most r that explain α_i for each i. Note that each $m_i \leq r \cdot |\Delta|^r$. For each α_i, we introduce fresh variables $a_i^1, \ldots, a_i^{m_i}$. The idea is that a_i^ℓ represents the set A_i^ℓ for each $\ell \leq m_i$. Note that, if $i \neq j$, it can be the case that there are ℓ and s such that $A_i^\ell = A_j^s$. In such a case a_i^ℓ and a_j^s are still different variables representing the same subset of Δ. Define $V := \bigcup_{i=1}^{k} \{a_i^\ell \mid 1 \leq \ell \leq m_i\}$. For any $u \in V$, let $S(u)$ denotes the subset of Δ represented by u. For $U \subseteq V$ define $S(U) = \bigcup_{u \in U} S(u)$. Finally, we define the desired $\Gamma_{1,r+1}$ formula as $\phi = \phi_1 \wedge \phi_2$, where

$$\phi_1 := \bigwedge_{i=1}^{k} \bigwedge_{\ell \neq j} (\neg a_i^\ell \vee \neg a_i^j),$$

$$\phi_2 := \bigwedge_{\substack{U \subseteq V \text{s.t.} \\ |U| \leq r+1, \\ S(U) \models \emptyset}} \Big(\bigvee_{u \in U} (\neg u) \Big).$$

Clearly, ϕ is a $\Gamma_{1,r+1}$-formula. The role of ϕ_1 is to assure that at most one subset of Δ is chosen for each α_i. Moreover, since we are looking for a satisfying assignment for ϕ of weight k, ϕ_1 further ensures that for each α_i, exactly one subset of Δ is chosen. The role of ϕ_2 is to make sure that inconsistent explanations are forbidden.

Claim 5.3. *(Δ, α, k) admits a support iff ϕ has a satisfying assignment of weight k.*

Proof of Claim. *Let $\Phi \subseteq \Delta$ be a support for α. Since Φ explains each α_i, by Claim 5.1 there is a set $\Theta_i \subseteq \Phi$ of size at most r such that $\Theta_i \models \alpha_i$. By construction each Θ_i is identical to a certain A_i^ℓ for some ℓ. We can thus choose for each α_i a corresponding a_i^ℓ and obtain an assignment s of weight k. Now, it is easy to verify that $s \models \phi_1$ (because we chose at most one a_i^ℓ for each α_i) and that $s \models \phi_2$ (because Φ is consistent).*

For the other direction, let $W \subseteq V$ denote a satisfying assignment for ϕ of weight k. By construction of ϕ, W contains exactly one a_i^ℓ for each α_i (because ϕ_1 is satisfied) and $S(W)$ is consistent (because ϕ_2 is satisfied). Therefore, $S(W)$ constitutes a support for α. ■

We conclude by observing that (ϕ, k) can be computed in **FPT**-time from (Δ, α, k) and the two instances have the same value for the parameter. □

The following lemma proves the $\mathbf{W}[1]$-hardness for languages considered in Lemma 5.19. The hardness is established by a reduction from the $\mathbf{W}[1]$-complete problem p-Clique(κ) ([32, Lem.2.3]) where $\kappa(\mathcal{G})$ is the size of the clique in $\mathcal{G}$. Given a graph $\mathcal{G} = (V, E)$ and $k \in \mathbb{N}$, the problem Clique asks whether there is a clique (a complete subgraph of $\mathcal{G}$) of size k in $\mathcal{G}$.

Lemma 5.20. *Let Γ be a CL that is EP^s and EN^s but not ε-valid, then* p-ARG$(\Gamma, |\alpha|)$*is* $\mathbf{W}[1]$*-hard.*

Proof. We give a reduction from the $\mathbf{W}[1]$-complete problem p-Clique(κ) ([32, Lem.2.3]) where $\kappa(\mathcal{G})$ is the size of the clique in $\mathcal{G}$. We first reduce p-Clique(κ) to p-ARG$(\{\mathrm{T}, \mathrm{F}\}, |\alpha|)$. Observe that a set of terms is consistent if and only if the terms are pairwise consistent. Let $(\mathcal{G}, k)$ be an

instance of p-CLIQUE(κ), where $\mathcal{G} = (V, E)$ is a graph with $V = \{v_1, \ldots, v_n\}$ and $E \subseteq V \times V$ and $k \in \mathbb{N}$. We construct an instance (Δ, α, k) of p-ARG($\{\mathrm{T}, \mathrm{F}\}, |\alpha|$) in two steps.

Step 1 We represent the graph G by terms such that each node v_i corresponds to a term t_i and two terms t_i, t_j are pairwise consistent if and only if there is an edge $(v_i, v_j) \in E$. This is achieved by the following iterative procedure. Let $u_1, \ldots, u_n$ be a collection of fresh variables. Initialize each $t_i := u_i$. Then for each pair t_i, t_j such that $(v_i, v_j) \notin E$, take a fresh variable, say $x_{i,j}$, and set $t_i := t_i \wedge x_{i,j}$ and $t_j := t_j \wedge \neg x_{i,j}$. After completion of this procedure it holds $(v_i, v_j) \in E$ iff $t_i \wedge t_j$ is consistent. Consequently, G has a clique of size k if and only if we can select k terms from $\{t_1, \ldots, t_n\}$ which are together consistent.

Step 2 We define $\alpha := z_1 \wedge \cdots \wedge z_k$, where $z_1, \ldots, z_k$ are fresh variables and extend the terms $t_1, \ldots, t_n$ from the first step as follows. For each t_r, we introduce k additional copies (t_r^i for $i \leq k$) such that:

1. each t_r^i explains exactly one z_i for $i \leq k$, and
2. $t_r^i \wedge t_r^j$ is inconsistent for each $i, j \leq k$.

Property (1) assures that each t_r can be used to explain any of the z_i (by selecting t_r^i) and property (2) assures that with each t_r we can explain at most one of $z_1, \ldots, z_k$. In order to extend the terms $t_1, \ldots, t_n$ such that properties (1) and (2) hold we use the following procedure. Property (1) is achieved by setting initially each $t_r^i := t_r \wedge z_i$. Property (2) is achieved by the same trick as in the first step: each pair t_r^i, t_r^j is made inconsistent by introducing a fresh variable $y_{r,i,j}$ and setting $t_r^i := t_r^i \wedge y_{r,i,j}$ and $t_r^j := t_r^j \wedge \neg y_{r,i,j}$.

We finally set $\Delta = \{t_r^i \mid 1 \leq r \leq n, 1 \leq i \leq k\}$ and observe that $\mathcal{G}$ has a clique of size k if and only if Δ contains a support (of size k) for α.

It remains to show that Corollary 5.15 can be applied, yielding the desired reduction to p-ARG($\Gamma, |\alpha|$). We can achieve this only after some technical modifications of the above α and Δ. Due to Lemma 5.2, item (2), we have $(t \wedge \neg f) \in \langle\Gamma\rangle_{\not\exists,\neq}$. Consequently, we obtain $\{\mathrm{T}, \mathrm{F}\} \subseteq \langle\Gamma\rangle_{\neq}$ since $\exists f(t \wedge \neg f) \equiv \mathrm{T}(t)$ and $\exists t(t \wedge \neg f) \equiv \mathrm{F}(f)$. Furthermore, observe that α (a positive term) can be written by using only constraints of the form $(t \wedge \neg f)$ if we allow the introduction of one additional variable (f) that occurs as a negative literal in α. In order to explain α, any explanation needs to entail $\neg f$. This is achieved by adding $\mathrm{F}(f)$ to Δ, which will be part of any explanation.

This completes the desired reduction to p-ARG($\{\mathrm{T}, \mathrm{F}\}, |\alpha|$) by noting that α contains only one additional variable f and therefore $|\mathsf{var}(\alpha)| = k + 1$. □

Now we turn towards the problem of verifying an argument under the parameterization claim-size. The complexity picture here exhibits a dichotomy between **FPT**- and **paraDP**-cases.

Theorem 5.21. *Let Γ be a CL. Then* p-ARG-Check($\Gamma, |\alpha|$) *is (1)* **FPT** *if Γ is Schaefer, and (2)* **paraDP***-complete otherwise.*

Proof. (1). This follows from Creignou et al. [22, Theorem 6.1] as classically ARG-Check$(\Gamma) \in \mathbf{P}$ if Γ is Schaefer.

(2). The membership follows as classically ARG-Check$(\Gamma) \in \mathbf{DP}$. Furthermore, the reduction in the proof presented by Creignou et al. [22, Propositions 6.3 and 6.4] always uses the claim (α) of the fixed size. As a consequence, certain slices of p-ARG-Check$(\Gamma, |\alpha|)$ are **DP**-hard, giving the desired results. □

Next, we follow up with the relevance problem for argumentation parameterized by the claim-size and show a tetrachotomy. The precise complexity of ARG-Rel$(\Gamma, |\alpha|)$ is still open if Γ is either EN^s but not N^s, or EP^s but not P^s. We only achieve $\mathbf{W}[2]$-membership with $\mathbf{W}[1]$-hardness.

Theorem 5.22. *Let Γ be a CL. Then,* p-ARG-Rel$(\Gamma, |\alpha|)$ *is*

1. **paraΣ_2^P**-*complete if Γ is not Schaefer,*
2. **paraNP**-*complete if Γ is Schaefer but neither EN^s, nor EP^s,*
3. $\mathbf{W}[1]$-*hard, with membership in* $\mathbf{W}[2]$ *if Γ is either EN^s but not N^s, or EP^s but not P^s,*
4. $\mathbf{W}[1]$-*complete if Γ is EN^s and EP^s but neither N^s, nor P^s,*
5. **FPT** *if Γ is either P^s or N^s.*

Proof. (1). The membership is true because the classical problem ARG-Rel(Γ) is in $\mathbf{\Sigma}_2^P$. The hardness also follows from the results by Creignou et al. [22, Prop. 7.7]. Notice that while proving the hardness for subcases, the claim α has a fixed size in each reduction. This implies that certain slices in each case are $\mathbf{\Sigma}_2^P$-hard. As a consequence, the desired hardness results follows.

(2). The membership follows because the classical problem ARG-Rel(Γ) is in **NP** when Γ is Schaefer. For hardness, we make a case distinction as to whether Γ is ε-valid or not.

Case 1. Let Γ be Schaefer and ε-valid, but neither EN^s, nor EP^s. The hardness follows because the 2-slice of the problem is already **NP**-hard [22, Proposition 7.6].

Case 2. Let Γ be Schaefer but neither ε-valid, nor EN^s, nor EP^s. The hardness follows from Theorem 5.13. This is due to the reason that ARG-Rel(Γ) is always harder than ARG via the reduction $(\Delta, \alpha) \mapsto (\Delta \cup \{\psi\}, \alpha \wedge \psi, \psi)$. Moreover, in this case we can take ψ to be a copy of α with fresh variables. This only doubles the size of α.

(3+4). The hardness in both cases follow because of the $\mathbf{W}[1]$-hardness of p-ARG$(\Gamma, |\alpha|)$ for these cases (Lem. 5.20) and the fact that ARG(Γ) reduces to ARG-Rel(Γ) (via $(\Delta, \alpha) \mapsto (\Delta \cup \{\psi\}, \alpha \wedge \psi, \psi)$ where ψ is a new formula over fresh variables). For membership, we extend the proof of Lemma 5.19 in such a way that an assignment for ϕ of weight k forces the given formula ψ to be included in a support for α. The $\mathbf{W}[1]$-membership for (4) is established in Lemma 5.23 using the fact that the required changes in the reduction still yield a $\Gamma_{1,r+1}$ formula. In contrast, for (3), we only prove $\mathbf{W}[2]$-membership in Lemma 5.24, since the modifications yield a $\Gamma_{2,r+1}$ formula.

(5). This follows as classically ARG-Rel$(\Gamma) \in \mathbf{P}$ [22, Prop. 7.3]. □

Intermediate Lemmas

Lemma 5.23. *Let* Γ *be a CL such that* Γ *is* EN^{s} *and* EP^{s}*, then* p-ARG-Rel$(\Gamma, |\alpha|) \in \mathbf{W}[1]$.

Proof. We reduce p-ARG-Rel$(\Gamma, |\alpha|)$ to p-WSAT$(\Gamma_{1,2}, \kappa)$ by slightly modifying the reduction from Lemma 5.19. Let $(\Delta, \alpha, \psi, k)$ be an instance of p-ARG-Rel$(\Gamma, |\alpha|)$. Note that (by definition), any Γ-formula can be written as a $\{\mathrm{T}, \mathrm{F}\}$-formula. In other words, Δ is a set of terms and α, ψ both are also terms. Let $\Delta = \{t_0, \ldots, t_n\}$ and $\alpha = \alpha_1 \wedge \cdots \wedge \alpha_k$. Moreover, assume that $\psi = t_0$. Then we have the following two observations.

Observation 1: For each α_i a single $t_j \in \Delta$ is sufficient to support α_i. In other words, for each $i \leq k$, if α_i has a support then it has a support of size one at most. This implies that there is a support for α iff there is a support of cardinality at most k.

Observation 2: Let $\Phi \subseteq \Delta$ be a set of terms. Then Φ is consistent iff Φ is pairwise consistent.

For each $\alpha_i \in \alpha$, denote $L_i^+ := \{\, t \in \Delta \mid \alpha_i \in t \,\}$ and $L_i^- := \{\, t \in \Delta \mid \neg\alpha_i \in t \,\}$. Clearly, each $L_i^+, L_i^- \subseteq \Delta$. Moreover, every $t \in L_i^+$ is a candidate support for α_i, whereas, no $t \in L_j^-$ can be in the support, for every $i, j \leq k$. Let $N = \bigcup_{i \leq k} L_i^-$ and denote $L_i = L_i^+ \setminus N$. It is important to notice that there is a support for α only if $L_i \neq \emptyset$ for each $i \leq k$. Otherwise, for some i, the support Φ can not contain a term t for α_i such that Φ is consistent. It remains to determine whether there is a consistent set Φ that includes one term from each L_i. The construction so-far is sufficient to solve the argument existence problem. However, for ARG-Rel, we need to assure that ψ is also contained in Φ, and that $\Phi \setminus \{\psi\}$ is not a support for α. To achieve this, notice first that if $\alpha_i \notin \psi$ for every $i \leq k$ then clearly ψ is not relevant for α. Moreover, a candidate support Φ has to be consistent with ψ. Observe that ψ can be considered as a candidate explanation for each α_i such that $\alpha_i \in \psi$. As a result, we do the following.

1. For each $i \leq k$ such that $\alpha_i \in \psi$, we let $L_i := \{\psi\}$. That is, if $\psi \in L_i$, then we remove all other terms from L_i.

2. For each $i, j \leq k$ such that $L_i = L_j$, then remove L_j. This removes the duplicates from the collection $L_1, \ldots L_k$.

Assume that the above procedure results in a collection $L_1, \ldots, L_{k'}$, where each L_i is a set of terms and $k' \leq k$. Proceed as in the proof of Lemma 5.19 and let $V_i := \{a_i^j \mid t_j \in L_i\}$. That is, V_i includes one variable corresponding to each term t_j and the set L_i. Finally, we let $V := \bigcup_{i \leq k'} V_i$. Let $u \in V_i$, then $T(u)$ denotes the term $t_j \in L_i$ represented by the variable $a_i^j \in V$. Then we let $\phi := \phi_1 \wedge \phi_2$, where

$$\phi_1 := \bigwedge_{i=1}^{k'} \bigwedge_{\ell \neq j} (\neg a_i^\ell \vee \neg a_i^j),$$

$$\phi_2 := \bigwedge_{\substack{u \in V_i, v \in V_{i'} \text{s.t.} \\ T(u) \wedge T(v) \models \emptyset}} \Big(\neg u \vee \neg v\Big).$$

The formula ϕ_1 assures that at most one term is chosen for each α_i, and ϕ_2 assures that these terms (from different L_i's) are pairwise consistent. The correctness follows from Claim 5.3. Clearly, ϕ has a satisfying assignment of weight k' iff there is a support Φ for α with $\psi \in \Phi$.

We conclude by observing that the reduction can be performed in **FPT**-time. The sets L_i^+ and L_i^- can be computed in polynomial time. Moreover, the formula ϕ can be constructed in polynomial time since (for ϕ_2) one needs to check the pairwise consistency for each of the remaining k' sets of terms. This proves that the reduction can be preformed in **FPT**-time and $k' \leq k$ where $|\alpha| = k$ is the parameter value of the input instance. □

Lemma 5.24. *Let Γ be a CL such that Γ is either* EN^{s} *or* EP^{s}*, then* p-ARG-Rel$(\Gamma, |\alpha|) \in \mathbf{W}[2]$.

Proof. We reduce p-ARG-Rel$(\Gamma, |\alpha|)$ to p-WSAT$(\Gamma_{2,d}, \kappa)$ for some fixed d. This problem is $\mathbf{W}[2]$-complete as depicted in Proposition 2.12. Moreover, we only present the changes which are required on top of the reduction in Lemma 5.19.

Let $(\Delta, \psi, \alpha, k)$ be an instance of p-ARG-Rel$(\Gamma, |\alpha|)$ with $\Delta = \{\delta_0, \delta_1, \ldots, \delta_n\}$ with $\psi = \delta_0$ and $\alpha = \alpha_1 \wedge \cdots \wedge \alpha_k$, where each α_i is either a positive or negative literal, or a positive clause of size $\leq r$. Notice that the two claims (Claim 5.1 and 5.2) are still true. Accordingly, let $A_i^1, \ldots, A_i^{m_i}$ denote all subsets of Δ of size at most r that form a support for α_i where $i \leq k$. Furthermore, consider the variables $a_i^1, \ldots, a_i^{m_i}$ for each α_i, and let $V := \bigcup_{i=1}^{k} \{a_i^\ell \mid 1 \leq \ell \leq m_i\}$. As before, $S(u)$ denotes the subset of Δ represented by u for each $u \in V$.

Now we distinguish those subsets of Δ (among A_i^j) that contain ψ. In other words, if $\psi \in A_i^j$ for some $i \leq k$ and $j \leq m_i$, then we know that there is a support for α_i that contains ψ. Whereas, if no subset A_i^j contains ψ then clearly ψ is not relevant for α. Now we only need to make sure that a subset of Δ containing ψ is considered in the support for α_i for some $i \leq k$. That is, there are $i \leq k$ and $j \leq m_i$ such that $\psi \in A_i^j$ and $s(a_i^j) = 1$ for any assignment s such that $s \models \phi$. Let $A_0^1, \ldots, A_0^{m_0}$ denote all subsets from the collection A_i^j such that $\psi \in A_i^j$ for $i \leq k$ and $j \leq m_i$. Moreover, let $a_0^1, \ldots a_0^{m_0}$ denote their corresponding variables. Notice that we do not add any new variable but only consider those that correspond to the sets containing ψ. Finally, we define the desired $\Gamma_{2,r+1}$ formula as $\phi := \phi_1 \wedge \phi_2 \wedge \phi_3$, where

$$\begin{aligned}
\phi_1 &:= \bigwedge_{i=1}^{k} \bigwedge_{\ell \neq j} (\neg a_i^\ell \vee \neg a_i^j), \\
\phi_2 &:= \bigwedge_{\substack{U \subseteq V \text{ s.t.} \\ |U| \leq r+1, \\ S(U) \models \emptyset}} \Big(\bigvee_{u \in U} (\neg u) \Big), \\
\phi_3 &:= \bigvee_{q \leq m_0} (a_0^q).
\end{aligned}$$

The formulas ϕ_1 and ϕ_2 are unchanged. The formula ϕ_3 assures that an explanation is acceptable only if it contains ψ.

Claim 5.4. *(Δ, α) admits a support Φ such that $\psi \in \Phi$ iff ϕ has a satisfying assignment of weight k.*

Proof of Claim. *The equivalence between the existence of a support and the existence of a satisfying assignment for ϕ of weight k follows from Claim 5.3. We only argue that the formula ψ is relevant for the support Φ iff ϕ has a satisfying assignment of weight k.*

If there is a support Φ for α and $\psi \in \Phi$, then clearly there is some $i \leq k$ such that α_i is explained by some $A_i^j \subseteq \Delta$ and $\psi \in A_i^j$ for $j \leq m_i$. This implies that $\phi_1 \wedge \phi_2$ is satisfiable by an assignment s and $s(a_0^q) = 1$ for some $q \leq m_0$. As a result, $s \models \phi_3$ and therefore ϕ is satisfiable.

Conversely, if there is an assignment s such that $s \models \phi$, then there is a support Φ for α since $s \models \phi_1 \wedge \phi_2$. Moreover, since s has a weight k and $s \models \phi_3$, there is some $q \leq m_0$ and the variable a_0^q corresponding to the subset $A_0^q \subseteq \Delta$ such that $s(a_0^q) = 1$ and $\psi \in A_0^q$. This implies that there is a support Φ (guaranteed by the assignment s) with $\psi \in \Phi$. Finally, $\Phi \setminus \{\psi\}$ can not be a support, since this removes all those subsets that contain ψ, and thereby making ϕ_3 unsatisfiable.

This completes the correctness proof and settles the claim. ■

We conclude by observing that ϕ is a $\Gamma_{2,r+1}$-formula. Indeed the big disjunction in ϕ_3 can be rewritten as $\bigvee_{q \leq m_0} \bigwedge(a_0^q)$. □

5.3.2 Parameters 'the knowledge base' (Δ) and 'the support' (Φ)

Regarding Δ and Φ, recall that they both yield two versions for the parameterization. For each $\chi \in \{\Delta, \Phi\}$ we consider (1) the number of variables, or equivalently, the encoding size of χ (see Lemma 4.4), and (2) the number of formulas in χ.

We first prove **FPT**-membership of each problem for any constraint language Γ, if the number of variables are considered.

Theorem 5.25. *Let Γ be any CL. Then,* p-ARG$(\Gamma, |\mathsf{var}(\Delta)|)$ *and* p-ARG-Rel$(\Gamma, |\mathsf{var}(\Delta)|)$ *are* **FPT**.

Proof. Notice that the number of subsets of Δ is bounded by $|\Delta|$, which is in turn bounded by the parameter $|\mathsf{var}(\Delta)|$ (see Lemma 4.4). Consequently, one simply checks each subset of Δ as a possible support Φ for α. Moreover, the size of each support Φ is also bounded by the parameter, as a result, one can determine the satisfiability and the entailment in **FPT**-time. This is because the satisfiability and entailment for Schaefer languages are in **P**. For non-Schaefer languages, the result follows from Theorem 5.12 since p-IMP$(\Gamma, |\mathsf{var}(\Phi)|) \in$ **FPT**. For p-ARG-Rel$(\Gamma, |\mathsf{var}(\Delta)|)$, only consider those subsets of Δ as a candidate support that contain ψ. □

If a parameter related to the support $\{|\Phi|, |\mathsf{var}(\Phi)|\}$ is considered, the problems ARG and ARG-Rel become irrelevant. Consequently, we only consider the problem ARG-Check.

Theorem 5.26. *Let Γ be a CL. Then,* ARG-Check$(\Gamma, |\mathsf{var}(\Phi)|)$ *is* **FPT**.

Proof. We need to determine whether (1) $\Phi \models \alpha$, and (2) for each $\psi \in \Phi$, $\Phi \setminus \{\psi\} \not\models \alpha$. The first question can be answered in a similar way to the implication problem (Theorem 5.12). That is, the problem can be reduced to checking whether the formula $\alpha[s]$ is a tautology for each $s \in 2^{|\mathsf{var}(\Phi)|}$ such that $s \models \Phi$. Finally observe that it can be checked in polynomial time whether $\alpha[s]$ is a

tautology. For the second question, notice that the number of formulas $\psi \in \Phi$ is also bounded from above by $|\mathsf{var}(\Phi)|$ due to Lemma 4.4. This implies that the second question is once again asked **FPT**-many times, and each time it can be answered in **FPT**-time, again. This completes the proof of the theorem. □

Interestingly, the number of formulas as a parameter yields intractability for non-Schaefer languages, as we prove next.

Theorem 5.27. *Let* Γ *be a CL. Then,* p-ARG$(\Gamma, |\Delta|)$ *is*

1. **paraDP**-*hard if* Γ *is neither Schaefer, nor* ϵ*-valid,*
2. **paraCoNP**-*complete if* Γ *is* ϵ*-valid, but not Schaefer,*
3. **FPT** *if* Γ *is Schaefer.*

Proof. (1). We prove that the 1-slice of the problem is **DP**-hard. To achieve this, we reduce from the abduction problem ABD(Γ) such that $H = \{h\}$ and $M = \{m\}$. Observe that ABD(Γ) is **DP**-hard when $|H| = 1$ and $|M| = 1$ (Thm. 4.18). Following Creignou et al. [22, Propositions 5.2], we distinguish between two cases depending on whether Γ is complementive or not. If Γ is complementive, we reduce ABD(Γ) to ARG$(\Gamma \cup \{x \neq y\})$ and then conclude by using Lemma 5.2 and Lemma 5.14. Recall that for ABD, the knowledge base KB can be written as a single formula $\Psi := \bigwedge_{i \leq |KB|} \delta_i$. We map an instance (V, H, M, KB) of ABD to (Δ, α) such that $\Delta := \{\Psi\} \cup \{h \neq f\}$ and $\alpha := (m \neq f)$ for a fresh variable f. Since KB is complementive, it suffices to prove the correctness for $\Delta[s]$ and $\alpha[s]$ where $s(f) = 0$. Observe that (V, H, M, KB) admits an explanation iff $KB \wedge h$ is consistent and $KB \wedge h \models m$ iff Δ is consistent and $\Delta \models \alpha$. This completes the correctness of our reduction.

In the other case, if Γ is not complementive, we reduce (V, H, M, KB) to an instance (Δ, α) of ARG$(\Gamma \cup \{x \wedge \neg y\})$ where $\Delta := \{\Psi\} \cup \{h \wedge \neg f\}$ and $\alpha := (m \wedge \neg f)$. The final reduction is obtained by implementing the constraint $(x \wedge \neg y)$ using Lemma 5.2 and 5.14. We conclude the proof by observing that $|\Delta| = 2$ in each case.

(2). The membership is trivial because ARG$(\Gamma) \in$ **CoNP**. For hardness, notice that since Γ is ϵ-valid, Δ is trivially satisfiable. This implies that there is a support for α iff Δ is the support. As a result, one only needs to determine whether $\Delta \models \alpha$. The result follows due to Lemma 5.9 because p-IMP$(\Gamma, |\Delta|)$ is still **paraCoNP**-hard.

(3). Let $|\Delta| = k$, then there are 2^k candidates for a support Φ and each candidate can be verified in polynomial time for a support. This is due to the reason that both problems, SAT(Γ) and IMP(Γ), are in **P** if Γ is Schaefer.

This completes the proof to the theorem. □

Now, we classify the complexity of the relevance problem.

Theorem 5.28. *Let* Γ *be a CL. Then,* p-ARG-Rel$(\Gamma, |\Delta|)$ *is*

1. **paraDP**-*hard if* Γ *is neither Schaefer, nor* ϵ*-valid,*
2. **paraCoNP**-*hard if* Γ *is* ϵ*-valid, but not Schaefer,*
3. **FPT** *if* Γ *is Schaefer.*

Proof. (1+2). The hardness in each case follows because of the corresponding hardness for p-ARG$(\Gamma, |\Delta|)$ (Theorem 5.27). (3). As in Theorem 5.27, if $|\Delta| = k$, then there are 2^k candidates for a support Φ and each candidate can be checked in polynomial time for a support. The only difference now is that the support should also contain the relevant formula, which is still achievable in **FPT**-time. □

The following theorem characterizes the complexity of the verification problem when parameterized by $|\Phi|$.

Theorem 5.29. *Let* Γ *be a CL. Then* p-ARG-Check$(\Gamma, |\Phi|)$ *is*

1. **paraDP**-*complete if* Γ *is neither Schaefer, nor* ϵ*-valid.*
2. **paraCoNP**-*hard if* Γ *is* ϵ*-valid, but not Schaefer,*
3. **FPT** *if* Γ *is Schaefer.*

Proof. (1). The membership is clear because classically ARG-Check$(\Gamma) \in$ **DP** for this case. For hardness, we argue that the reduction for proving the **paraDP**-hardness in Theorem 5.27 works here as well. This is because taking $\Phi := \Delta$ implies that Φ is a support for α iff Φ is a minimal support. This follows from the fact that Φ only contains two formulas, and removing any of these two violates $\Phi \models \alpha$. This completes the correctness of our reduction.

(2). Observe that when $\Phi = \{\phi\}$, then the problem reduces to the entailment problem which is **paraCoNP**-hard for this case (Cor. 5.9).

(3). Follows from a result by Creignou et al. [22, Theorem 6.1], since ARG-Check$(\Gamma) \in$ **P** if Γ is Schaefer. □

Galois connection

We conclude this chapter on the parameterized complexity of argumentation by a discussion on the Galois connection (Property 2.1). Recall from Proposition 2.19 that Galois connection fails for ARG-Rel. Surprisingly, we have a diverse picture regarding the parameterized problems. On the one hand, the Property 2.1 fails for p-ARG$(|\alpha|)$ and p-ARG-Rel$(|\alpha|)$. On the other hand, it is true for p-ARG-Rel$(|\Delta|)$. In particular, there are CLs Γ and Γ' such that $\langle\Gamma\rangle = \langle\Gamma'\rangle$ but p-ARG$(\Gamma, |\alpha|)$ is **W**[1]-complete whereas p-ARG$(\Gamma', |\alpha|)$ is **paraNP**-complete. Moreover, regarding p-ARG-Rel$(\Gamma, |\alpha|)$, there is a violation of Property 2.1 in two places.

W[1] **vs paraNP.** There are CLs Γ and Γ' such that $\langle\Gamma\rangle = \langle\Gamma'\rangle$ but p-ARG-Rel$(\Gamma, |\alpha|)$ is **W**[1]-complete whereas p-ARG-Rel$(\Gamma', |\alpha|)$ is **paraNP**-complete.

FPT vs paraNP. There are CLs Γ and Γ' such that $\langle\Gamma\rangle = \langle\Gamma'\rangle$ but p-ARG-Rel$(\Gamma, |\alpha|)$ is **FPT** whereas p-ARG-Rel$(\Gamma', |\alpha|)$ is **paraNP**-complete.

Clearly, the Galois connection is violated for p-ARG$(\Gamma, |\alpha|)$ unless $\mathbf{W}[1] = \mathbf{paraNP}$. Moreover, it is violated for p-ARG-Rel$(\Gamma, |\alpha|)$ unless $\mathbf{FPT} = \mathbf{W}[1] = \mathbf{paraNP}$.

We present an example below that highlights how the Galois connection fails for certain CLs.

Example 5.30. *Let* $\Gamma_1 = \{\mathrm{T}, \mathrm{F}\}$ *and* $\Gamma_2 = \{\mathrm{T}, \mathrm{F}, =\}$ *be two constraint languages. Then we have that* $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$ *but* p-ARG$(\Gamma_1, |\alpha|)$ *is* $\mathbf{W}[1]$*-complete, while* p-ARG$(\Gamma_2, |\alpha|)$ *is* **paraNP***-complete.*

Interestingly, the Galois connection is only violated when $|\alpha|$ is considered as a parameter. The property 2.1 is true for each problem when parameterized by $|\chi|$ and $|\mathsf{var}(\chi)|$, for $\chi \in \{\Delta, \Phi\}$.

This concludes our parameterized complexity analysis for each problem in argumentation. Figure 5.1 presents an overview of the complexity results for ARG(Γ, κ), ARG-Check(Γ, κ) and ARG-Rel(Γ, κ) for each CL Γ and considered parameterization κ.

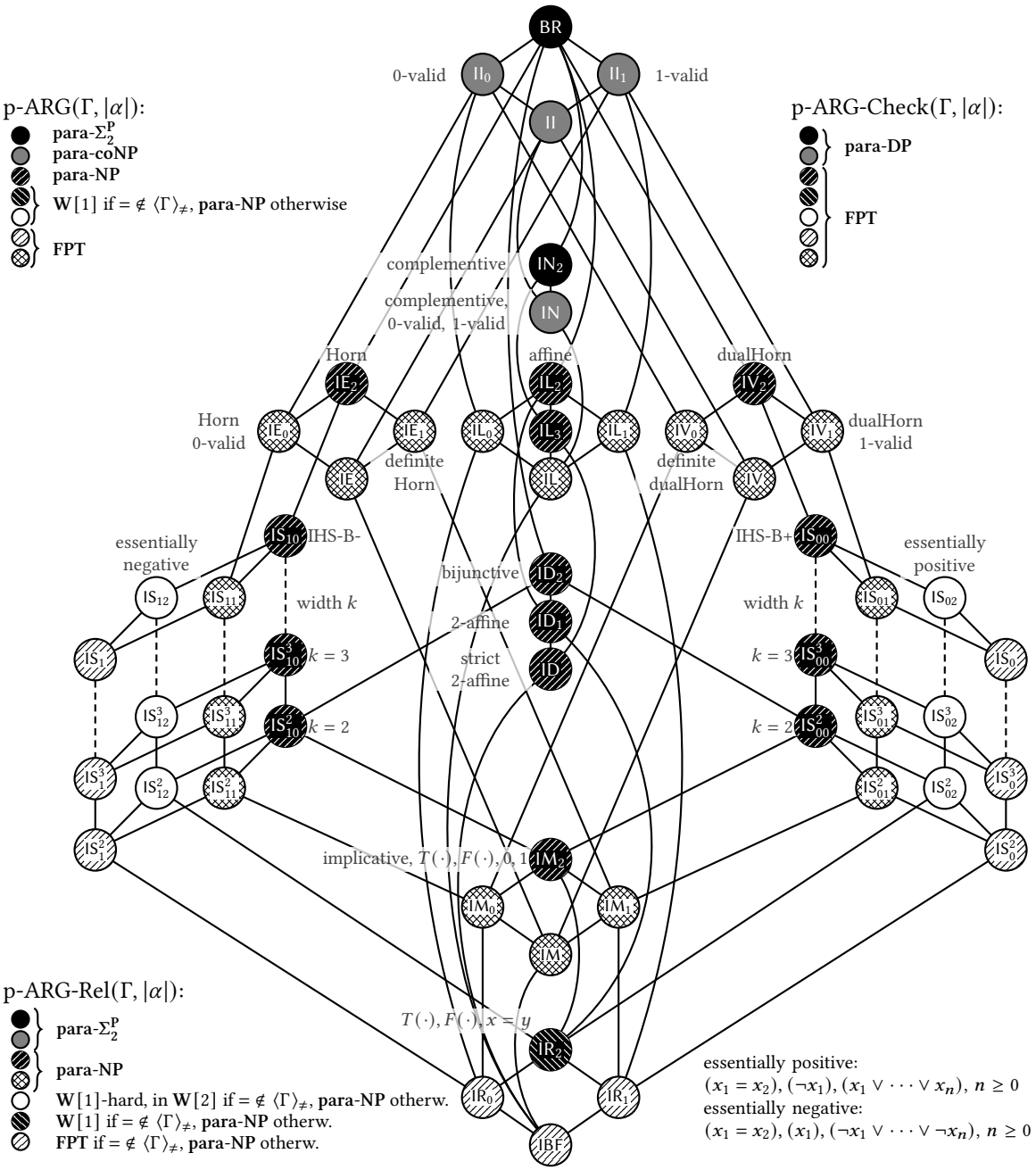


Figure 5.1: Complexity landscape of each problem in argumentation with respect to the parameter $|\alpha|$. The results for parameters $\kappa \in \{|\mathsf{var}(\Delta)|, |\mathsf{var}(\Phi)|, |\Delta|, |\Phi|\}$ have been omitted for better presentation. Each result depicts completeness for the mentioned class except for **FPT**-cases, or when specified otherwise.

CHAPTER 6

CONCLUSION

In this work, we systematically explored the parameterized complexity of various problems in propositional dependence logic, as well as logic-based abduction and argumentation. For an overview of the complexity analysis consult Table 3.1 (for $\mathcal{PDL}$) and Figures 4.1, 5.1 (for ABD and ARG, respectively).

6.1 Outlook for $\mathcal{PDL}$

Regarding $\mathcal{PDL}$, we focused on the model checking (MC) and the satisfiability (SAT) problem. For both problems, we exhibited a complexity dichotomy: **FPT** vs. **paraNP**-completeness (see Table 3.1). This is surprising, considering the fact that there is a well-known infinite **W**-hierarchy in between **FPT** and **paraNP**. In other words, each considered parameterization either yields tractability or renders the problem **NP**-hard for a fixed value of the parameter. Towards the end, we introduced two problems, a variant of the satisfiability question (mSAT) and a variant of model checking (mSubTeam). The problem mSAT asks for a satisfying team of a given size, whereas mSubTeam asks for a satisfying subteam of a given size for the given team. To the best of the author's knowledge, these two problems have not been explored before in the setting of team-semantics (except the maximal subteam membership problem for inclusion logic [53]). We prove that mSAT is **NP**-complete classically. The parameterized complexity of mSAT behaves similarly to SAT, though some parameterizations are open for further research. Moreover, we prove that mSubTeam is also **NP**-hard, and **NP**-complete for Poor man's $\mathcal{PDL}$.

It is also interesting to explore various combinations of the studied parameters. Table 6.1 presents a starting point with an overview of results (already proven) in this direction. Surprisingly, several combinations of our parameterizations already yield intractability. The combination (dep-arity + #atoms) is particularly interesting, as it completely bounds the number and the arity of functions arising in terms of dependence atoms. It is also interesting to observe

Parameter (κ)	p-MC(κ)	p-SAT(κ)
formula-tw + dep-arity	**paraNP**	**FPT**
#conjunctions + #splits	**paraNP**	**FPT**
#conjunctions + #atoms	**paraNP**	-
#splits + #atoms	**paraNP**	**FPT**

Table 6.1: Complexity classification overview with respect to the combination of parameters.

that in our reductions for proving the hardness of p-MC under these two parametrizations, if dep-arity is fixed, then #atoms is unbounded and vice versa. The author believes that the combination (#atoms + dep-arity) might yield membership in **FPT** for MC, although this is just an intuition.

6.2 Outlook for ABD

Regarding logic-based abduction, we presented a two-dimensional classification of three central abductive reasoning problems (ABD, $\text{ABD}_{\leq}$ and $\text{ABD}_{=}$). The first dimension was regarding different parameterizations $|H|, |M|, |V|, |E|$, and the second regarding various constraint languages defined by corresponding co-clones.

The parameter $|V|$ always allows for **FPT** algorithms independent of the co-clone. The parameter $|H|$ is particularly interesting, recognizing that it bounds the solution space for an abduction instance. Nevertheless, for non-Schaefer languages, the problem is still intractable (either **paraCoNP**- or **paraDP**-hard). Even stronger, the intractability of these cases persists even if $|M| = 1$. This compels one to ponder what other parameterization (together with $|H|$) can help achieving tractability. We answered this question by using a result from Fellows et al. [93] that the size of the smallest Horn backdoor is one such parameter. Regarding $|E|$, we argued that only the two size-restricted variants are meaningful. Interesting to notice, if Γ is an essentially negative CL then $\text{ABD}_{\leq}(\Gamma, |E|)$ is **FPT** and $\text{ABD}_{=}(\Gamma, |E|)$ is $\mathbf{W}[1]$-complete. A similar easy/hard difference manifests itself for the parameterization $|M|$. However, here, we distinguish between **paraNP**-completeness of $\text{ABD}_{=}(\Gamma, |M|)$ and **FPT** for $\text{ABD}_{\leq}(\Gamma, |M|)$, if Γ is essentially negative.

Finally, we also proved that the Galois connection (Property 2.1) is true for each problem $\text{ABD}_{*} \in \{\text{ABD}, \text{ABD}_{\leq}, \text{ABD}_{=}\}$ in the classical as well as parameterized setting. Moreover, we also developed several general results regarding the expressivity of equality constraints. Our results imply that one can express equality in $\langle\Gamma\rangle_{\neq}$ for any CL Γ which is neither strict essentially positive, nor strict essentially negative. Furthermore, this can be achieved with only two existentially quantified variables (see Lemma 4.3). As a result, replacing an equality constraint with its equivalent interpretation does not blow up the size of the knowledge base. Regarding the remaining cases (where we did not prove a general expressivity of equality), we still proved by other means that the Galois connection holds. Nevertheless, these techniques might not be useful in other settings, and we already encountered an example where Property 2.1 fails (namely, the

problem p-ARG$(\Gamma, |\alpha|)$).

As a future work, one might explore the parameterized complexity of other versions of the abductive reasoning, such as when M is not a term but an arbitrary Γ-formula. One can also consider the parameterized enumeration complexity [25, 24, 85] of the aforementioned problems. Last but not least, one could attack the open cases for affine co-clones.

A sneak peak into the Affine CLs. Unfortunately, we could not classify the parameterized complexity of abductive reasoning for affine constraint languages. Affine co-clones (and clones) are often considered *notorious* as the problems regarding them often resist a complete complexity classification [4, 28, 27, 108, 5, 86, 99].

The tractable cases for affine languages are often proved using the notion of *projection* of affine formulas. Consider the case when $M := q$, for a variable q. Then the problem (V, H, q, KB) can be answered in polynomial time if KB is a Γ-formula for an affine language Γ. The proof idea ([30, Prop. 9]) is to compute the projection of $KB \wedge (q = 0)$ on to H. That is, an affine formula ψ over H whose satisfying assignments are exactly the satisfying assignments of $KB \wedge (q = 0)$ restricted to H. An abduction instance (V, H, M, KB) has an explanation if and only if the formula $KB \wedge \neg\psi$ is satisfiable. The author believes that the same idea ([30, Prop. 9]) can be generalized at least for the parameter $|M|$. Given an instance (V, H, M, KB) where $M = \bigwedge_{i \leq k} m_i$, then one computes the projections $KB \wedge (\bigwedge_{i \leq k} m_i = 0)$ over H. This yields exponentially (in k) many projections ψ_j. The only open question is to determine whether the satisfiability of $KB \wedge \neg\psi'_j$ for some/every j relates to the existence of an explanation for M. The parameter $|H|$ already yields **FPT**-membership for each problem for affine CLs. This follows as the solution space is bounded by the parameter and that both (SAT and IMP) can be solved in polynomial time. Finally, the complexity concerning the parameter $|E|$ is again difficult to analyze since even the notion of projection does not yield any further direction to explore.

6.3 Outlook for ARG

Regarding the logic-based argumentation, we performed a similar two-dimensional complexity classification of three reasoning problems (ARG, ARG-Check and ARG-Rel). The conclusion regarding argumentation is that the size of α as a parameter does not help to reach tractable fragments of p-ARG, except for very restricted cases (see Theorem 5.13). The parameters $|\mathsf{var}(\Delta)|$ and $|\mathsf{var}(\Phi)|$, on the other hand, yield **FPT**-results for each constraint language. Moreover, the parameters $|\Delta|$ and $|\Phi|$ also yield **FPT**-results for Schaefer languages.

We also explored the (parameterized) complexity of the tautology and the implication problem in Schaefer's framework. We were able to achieve membership in **P** for TAUT. For $\chi \in \{|\Phi|, |\mathsf{var}(\alpha)|\}$, the problem p-IMP$(\Gamma, \chi)$ exhibits a dichotomy: **FPT** if Γ is Schaefer and **paraCoNP**-complete otherwise. In contrast, the problem p-IMP$(\Gamma, |\mathsf{var}(\Phi)|)$ is **FPT** for any constraint language Γ.

In the classical setting, the Galois connection (Property 2.1) failed for ARG-Rel [22, Thm. 7.1]. We proved that this failure is also transferred to the parameterized setting but only when $|\alpha|$ is the parameter. When parameterized by $|\Delta|$ and $|\mathsf{var}(\Delta)|$, the Galois connection is preserved for ARG-Rel somewhat unsurprisingly, since ARG-Rel$(\Gamma, |\mathsf{var}(\Delta)|)$ is **FPT** for any CL Γ, and ARG-Rel$(\Gamma, |\Delta|)$ is **FPT** if Γ is Schaefer. However, the surprise is retained as now we have cases (Example 5.30) when Property 2.1 also fails for ARG when parameterized by α. This implies that the **NP**-complete cases of ARG can be divided into two categories: (1) those cases for which p-ARG remains **paraNP**-complete and (2) those for which the complexity drops to $\mathbf{W}[1]$-completeness, when parameterized by α (unless the **W**-hierarchy collapses at the first level and $\mathbf{W}[1] = \mathbf{paraNP}$).

6.4 Concluding Remarks

Towards the end, we try to compare the problems from each chapter and conclude the overall writing.

It might be interesting to compare ABD and ARG concerning their complexity. Both of these problems are $\mathbf{\Sigma}_2^P$-complete for CNF-fragments, and therefore, they are both equivalent with respect to polynomial-time reductions. Moreover, both problems ask for an *explanation* of a *manifestation* (resp., support for a claim). Nevertheless, their complexity classification is much widespread, with ARG being mostly harder than ABD. One possible explanation for this interesting behavior is that the claim α in an ARG(Γ)-instance can be any Γ-formula. In contrast to that, the manifestation M in an ABD(Γ)-instance (our setting) must be a term. It might be interesting to consider the version of ABD where M is also an arbitrary Γ-formula. Then, a comparison of the parameterized complexity between the two formalisms would be interesting.

Dependence logic manifests its applications on various occasions in the database setting ([1, 11, 53]). One might ask whether it is interesting to consider abduction (or argumentation) for $\mathcal{PDL}$? Notice that the entailment or the implication problem (IMP) is a basic component of both problems we explored. However, IMP for $\mathcal{PDL}$ is $\mathbf{CoNEXP}^{\mathbf{NP}}$-complete [52, Thm., 6.1], which is way more complex than $\mathcal{PL}$-IMP. Nevertheless, it might be interesting to consider the syntactic fragments of $\mathcal{PDL}$ such as Poor man's $\mathcal{PDL}$ ([39, 87]) and study the abductive reasoning for them. Moreover, one can also consider abduction with functional dependencies. Let T be an inconsistent database[1] (a team) with respect to a set Ψ of functional dependencies (dependence atoms), and ϕ be a query. The task of abductive reasoning is then to find an *explanation* for ϕ in T. It seems natural to impose a size-restriction for the subteam to avoid the cases when the dependencies are trivially true (a singleton team). Then one can also explore whether this problem has any connections to consistent query answering (CQA). Intuitively, this problem corresponds to asking whether there exists *significant evidence* for believing that the query is true even if the database is inconsistent. The author believes that this is in contrast to CQA, where

[1] Otherwise the question is simply the *query evaluation*

a query has to be true in every repair of the database. Moreover, it might also be interesting to study this problem in relation to the one we defined earlier (*m*SubTeam). The author believes that the problem *m*SubTeam might have applications in repairing an inconsistent database when a set of functional dependencies are violated. Moreover, the abduction problem for databases might be related to the question of whether a query is true in some repair of the database. Nevertheless, the goal of establishing the importance of these new problems (if any) is left open for further discussion.

BIBLIOGRAPHY

[1] Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 31–41, 2009. doi:10.1145/1514894.1514899.

[2] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. arXiv:https://doi.org/10.1137/0608024, doi:10.1137/0608024.

[3] Pablo Barceló and Marco Calautti, editors. *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, volume 127 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. URL: http://www.dagstuhl.de/dagpub/978-3-95977-101-6.

[4] Michael Bauland, Martin Mundhenk, Thomas Schneider, Henning Schnoor, Ilka Schnoor, and Heribert Vollmer. The tractability of model checking for LTL: the good, the bad, and the ugly fragments. *ACM Transactions on Computational Logic (TOCL)*, 12(2):13:1–13:28, January 2011.

[5] Michael Bauland, Thomas Schneider, Henning Schnoor, Ilka Schnoor, and Heribert Vollmer. The complexity of generalized satisfiability for linear temporal logic. *Logical Methods in Computer Science*, 5(1), 2009. URL: http://arxiv.org/abs/0812.4848.

[6] Catriel Beeri, Martin Dowd, Ronald Fagin, and Richard Statman. On the structure of armstrong relations for functional dependencies. *J. ACM*, 31(1):30–46, January 1984. doi:10.1145/2422.322414.

[7] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, 2001.

[8] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[9] Philippe Besnard and Anthony Hunter. Argumentation based on classical logic. In *Argumentation in Artificial Intelligence*, pages 133–152. 2009. doi:10.1007/978-0-387-98197-0_7.

[10] Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. Abductive commonsense reasoning. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=Byg1v1HKDB.

[11] Thomas Bläsius, Tobias Friedrich, and Martin Schirneck. The Parameterized Complexity of Dependency Detection in Relational Databases. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: http://drops.dagstuhl.de/opus/volltexte/2017/6920, doi:10.4230/LIPIcs.IPEC.2016.6.

[12] Hans L. Bodlaender. A tourist guide through treewidth. *Acta Cybern.*, 11(1-2):1–21, 1993. URL: https://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3417.

[13] Elmar Böhler, Steffen Reith, Henning Schnoor, and Heribert Vollmer. Bases for boolean co-clones. *Inf. Process. Lett.*, 96(2):59–66, 2005. doi:10.1016/j.ipl.2005.06.003.

[14] Édouard Bonnet, Vangelis Th. Paschos, and Florian Sikora. Parameterized exact and approximation algorithms for maximum *k*-set cover and related satisfiability problems. *RAIRO - Theor. Inf. and Applic.*, 50(3):227–240, 2016. doi:10.1051/ita/2016022.

[15] Eugene Charniak. Motivation analysis, abductive unification, and nonmonotonic equality. *Artif. Intell.*, 34(3):275–295, 1988. doi:10.1016/0004-3702(88)90063-X.

[16] Hubie Chen and Stefan Mengel. Counting answers to existential positive queries: A complexity classification. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 315–326. ACM, 2016. doi:10.1145/2902251.2902279.

[17] Hubie Chen and Stefan Mengel. The logic of counting query answers. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005085.

[18] Hubie Chen and Moritz Müller. The fine classification of conjunctive queries and parameterized logarithmic space. *TOCT*, 7(2):7:1–7:27, 2015. doi:10.1145/2751316.

[19] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005. doi:10.1016/j.ic.2004.04.007.

[20] Luca Console, Daniele Theseider Dupré, and Pietro Torasso. On the relationship between abduction and deduction. *J. Log. Comput.*, 1(5):661–690, 1991. doi:10.1093/logcom/1.5.661.

[21] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047.

[22] Nadia Creignou, Uwe Egly, and Johannes Schmidt. Complexity classifications for logic-based argumentation. *ACM Trans. Comput. Log.*, 15(3):19:1–19:20, 2014. doi:10.1145/2629421.

[23] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems*, volume 7 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2001.

[24] Nadia Creignou, Raïda Ktari, Julian-Steffen Müller, Frédéric Olive, and Heribert Vollmer. Parameterised enumeration for modification problems. *Algorithms*, 12(9), 2019. doi:10.3390/a12090189.

[25] Nadia Creignou, Arne Meier, Julian-Steffen Müller, Johannes Schmidt, and Heribert Vollmer. Paradigms for parameterized enumeration. *Theory Comput. Syst.*, 60(4):737–758, 2017. doi:10.1007/s00224-016-9702-4.

[26] Nadia Creignou, Frédéric Olive, and Johannes Schmidt. Enumerating all solutions of a boolean CSP by non-decreasing weight. In *Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*, pages 120–133, 2011. doi:10.1007/978-3-642-21581-0_11.

[27] Nadia Creignou, Johannes Schmidt, and Michael Thomas. Complexity of propositional abduction for restricted sets of boolean functions. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. AAAI Press, 2010. URL: http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1201.

[28] Nadia Creignou, Johannes Schmidt, Michael Thomas, and Stefan Woltran. Sets of boolean connectives that make argumentation easier. In *Proc. 12th European Conference on Logics in Artificial Intelligence*, volume 6341 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2010.

[29] Nadia Creignou and Heribert Vollmer. Boolean constraint satisfaction problems: When does post's lattice help? In Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors, *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar].*, volume 5250 of *Lecture Notes in Computer Science*, pages 3–37. Springer, 2008. doi:10.1007/978-3-540-92800-3_2.

[30] Nadia Creignou and Bruno Zanuttini. A complete classification of the complexity of propositional abduction. *SIAM J. Comput.*, 36(1):207–229, 2006. doi:10.1137/S0097539704446311.

[31] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

[32] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

[33] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995. doi:10.1016/0004-3702(94)00041-X.

[34] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artif. Intell.*, 170(2):114–159, 2006. doi:10.1016/j.artint.2005.07.002.

[35] Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. Approximation and dependence via multiteam semantics. *Ann. Math. Artif. Intell.*, 83(3-4):297–320, 2018. doi:10.1007/s10472-017-9568-4.

[36] Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. Probabilistic team semantics. In Flavio Ferrarotti and Stefan Woltran, editors, *Foundations of Information and Knowledge Systems - 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14-18, 2018, Proceedings*, volume 10833 of *Lecture Notes in Computer Science*, pages 186–206. Springer, 2018. doi:10.1007/978-3-319-90050-6_11.

[37] Arnaud Durand and Stefan Mengel. Structural tractability of counting of solutions to conjunctive queries. *Theory Comput. Syst.*, 57(4):1202–1249, 2015. doi:10.1007/s00224-014-9543-y.

[38] Wolfgang Dvorák and Paul E. Dunne. Computational problems in formal argumentation and their complexity. *FLAP*, 4(8), 2017. URL: http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf.

[39] Johannes Ebbing and Peter Lohmann. Complexity of model checking for modal dependence logic. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings*, volume 7147 of *Lecture Notes in Computer Science*, pages 226–237. Springer, 2012. doi:10.1007/978-3-642-27660-6_19.

[40] Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015.

[41] Kave Eshghi. Abductive planning with event calculus. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15-19, 1988 (2 Volumes)*, pages 562–579, 1988.

[42] Ronald Fagin and Moshe Y. Vardi. Armstrong databases for functional and inclusion dependencies. *Inf. Process. Lett.*, 16(1):13–19, 1983. doi:10.1016/0020-0190(83)90005-4.

[43] Michael R. Fellows, Andreas Pfandler, Frances A. Rosamond, and Stefan Rümmele. The parameterized complexity of abduction. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* AAAI Press, 2012. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5048.

[44] Johannes Klaus Fichte, Markus Hecher, Yasir Mahmood, and Arne Meier. Decomposition-guided reductions for argumentation and treewidth. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1880–1886, 2021. doi:10.24963/ijcai.2021/259.

[45] Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Inf. Comput.*, 187(2):291–319, 2003. doi:10.1016/S0890-5401(03)00161-5.

[46] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

[47] Pietro Galliani. Inclusion and exclusion dependencies in team semantics - on some logics of imperfect information. *Ann. Pure Appl. Logic*, 163(1):68–84, 2012. doi:10.1016/j.apal.2011.08.005.

[48] Georg Gottlob, Reinhard Pichler, and Fang Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Artif. Intell.*, 174(1):105–132, 2010. doi:10.1016/j.artint.2009.10.003.

[49] Erich Grädel. Model-checking games for logics of imperfect information. *Theor. Comput. Sci.*, 493:2–14, 2013. doi:10.1016/j.tcs.2012.10.033.

[50] Erich Grädel and Jouko A. Väänänen. Dependence and independence. *Stud Logica*, 101(2):399–410, 2013. doi:10.1007/s11225-013-9479-2.

[51] Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 657–666. ACM, 2001. doi:10.1145/380752.380867.

[52] Miika Hannula. Validity and entailment in modal and propositional dependence logics. *Log. Methods Comput. Sci.*, 15(2), 2019. doi:10.23638/LMCS-15(2:4)2019.

[53] Miika Hannula and Lauri Hella. Complexity thresholds in inclusion logic. In *Logic, Language, Information, and Computation - 26th International Workshop, WoLLIC 2019, Utrecht, The Netherlands, July 2-5, 2019, Proceedings*, pages 301–322, 2019. doi:10.1007/978-3-662-59533-6_19.

[54] Miika Hannula, Juha Kontinen, and Jonni Virtema. Polyteam semantics. In Sergei N. Artëmov and Anil Nerode, editors, *Logical Foundations of Computer Science - International Symposium, LFCS 2018, Deerfield Beach, FL, USA, January 8-11, 2018, Proceedings*, volume 10703 of *Lecture Notes in Computer Science*, pages 190–210. Springer, 2018. doi:10.1007/978-3-319-72056-2_12.

[55] Miika Hannula, Juha Kontinen, Jonni Virtema, and Heribert Vollmer. Complexity of propositional logics in team semantic. *ACM Trans. Comput. Log.*, 19(1):2:1–2:14, 2018. doi:10.1145/3157054.

[56] Lauri Hella, Antti Kuusisto, Arne Meier, and Jonni Virtema. Model checking and validity in propositional and modal inclusion logics. *J. Log. Comput.*, 2019. doi:10.1093/logcom/exz008.

[57] Lauri Hella, Antti Kuusisto, Arne Meier, and Jonni Virtema. Model checking and validity in propositional and modal inclusion logics. *J. Log. Comput.*, 29(5):605–630, 2019. doi:10.1093/logcom/exz008.

[58] Lauri Hella and Johanna Stumpf. The expressive power of modal logic with inclusion atoms. In *Proceedings of the 6th GandALF*, pages 129–143, 2015.

[59] L. Henkin. Some remarks on infinitely long formulas. In *Journal of Symbolic Logic*, pages 167–183. Pergamon Press, 1961.

[60] Jaakko Hintikka and Gabriel Sandu. Informational independence as a semantical phenomenon. In Jens Erik Fenstad, Ivan T. Frolov, and Risto Hilpinen, editors, *Logic, Methodology and Philosophy of Science VIII*, volume 126 of *Studies in Logic and the Foundations of Mathematics*, pages 571–589. Elsevier, 1989. URL: https://www.sciencedirect.com/science/article/pii/S0049237X08700661, doi:https://doi.org/10.1016/S0049-237X(08)70066-1.

[61] Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and Paul A. Martin. Interpretation as abduction. *Artif. Intell.*, 63(1-2):69–142, 1993. doi:10.1016/0004-3702(93)90015-4.

[62] Wilfrid Hodges. Compositional semantics for a language of imperfect information. *Log. J. IGPL*, 5(4):539–563, 1997. doi:10.1093/jigpal/5.4.539.

[63] Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. Abduction-based explanations for machine learning models. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019,*

Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 1511–1519, 2019. doi:10.1609/aaai.v33i01.33011511.

[64] John R. Josephson, B. Chandrasekaran, Jack W. Smith, and Michael C. Tanner. A mechanism for forming composite explanatory hypotheses. *IEEE Trans. Systems, Man, and Cybernetics*, 17(3):445–454, 1987. doi:10.1109/TSMC.1987.4309060.

[65] Antonis C. Kakas and Paolo Mancarella. Database updates through abduction. In *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings*, pages 650–661, 1990. URL: http://www.vldb.org/conf/1990/P650.PDF.

[66] Antonis C. Kakas and Loizos Michael. Abduction and argumentation for explainable machine learning: A position survey. *CoRR*, abs/2010.12896, 2020. URL: https://arxiv.org/abs/2010.12896, arXiv:2010.12896.

[67] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

[68] Jarmo Kontinen. Coherence and computational complexity of quantifier-free dependence logic formulas. *Studia Logica*, 101(2):267–291, 2013. doi:10.1007/s11225-013-9481-8.

[69] Juha Kontinen, Antti Kuusisto, Peter Lohmann, and Jonni Virtema. Complexity of two-variable dependence logic and if-logic. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 289–298, 2011. doi:10.1109/LICS.2011.14.

[70] Juha Kontinen, Arne Meier, and Yasir Mahmood. A parameterized view on the complexity of dependence logic. In *Logical Foundations of Computer Science - International Symposium, LFCS 2022, Deerfield Beach, FL, USA, January 10-13, 2022, Proceedings*, pages 125–142, 2022. doi:10.1007/978-3-030-93100-1_9.

[71] Andreas Krebs, Arne Meier, and Jonni Virtema. A team based variant of CTL. In Fabio Grandi, Martin Lange, and Alessio Lomuscio, editors, *22nd International Symposium on Temporal Representation and Reasoning, TIME 2015, Kassel, Germany, September 23-25, 2015*, pages 140–149. IEEE Computer Society, 2015. doi:10.1109/TIME.2015.11.

[72] Andreas Krebs, Arne Meier, Jonni Virtema, and Martin Zimmermann. Team semantics for the specification and verification of hyperproperties. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.10.

[73] Leonid Levin. Universal search problems. *Problems of Information Transmission*, 9(3):115–116, 1973.

[74] Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. URL: http://www.cs.toronto.edu/%7Elibkin/fmt, doi:10.1007/978-3-662-07003-1.

[75] Peter Lohmann. *Computational Aspects of Dependence Logic*. PhD thesis, 2012. URL: http://arxiv.org/abs/1206.4564, arXiv:1206.4564.

[76] Peter Lohmann and Heribert Vollmer. Complexity results for modal dependence logic. *Studia Logica*, 101(2):343–366, 2013. doi:10.1007/s11225-013-9483-6.

[77] Martin Lück. *Team logic: axioms, expressiveness, complexity*. PhD thesis, University of Hanover, Hannover, Germany, 2020. URL: https://www.repo.uni-hannover.de/handle/123456789/9430.

[78] Martin Lück, Arne Meier, and Irena Schindler. Parameterised complexity of satisfiability in temporal logic. *ACM Trans. Comput. Log.*, 18(1):1:1–1:32, 2017. doi:10.1145/3001835.

[79] Yasir Mahmood and Arne Meier. Parameterised complexity of model checking and satisfiability in propositional dependence logic. In *Foundations of Information and Knowledge Systems - 11th International Symposium, FoIKS 2020, Dortmund, Germany, February 17-21, 2020, Proceedings*, pages 157–174, 2020. doi:10.1007/978-3-030-39951-1_10.

[80] Yasir Mahmood and Arne Meier. Parameterised complexity of model checking and satisfiability in propositional dependence logic. *Ann. Math. Artif. Intell.*, 90(2):271–296, 2022. doi:10.1007/s10472-021-09730-w.

[81] Yasir Mahmood, Arne Meier, and Johannes Schmidt. Parameterised complexity of abduction in schaefer's framework. In *Logical Foundations of Computer Science - International Symposium, LFCS 2020, Deerfield Beach, FL, USA, January 4-7, 2020, Proceedings*, pages 195–213, 2020. doi:10.1007/978-3-030-36755-8_13.

[82] Yasir Mahmood, Arne Meier, and Johannes Schmidt. Parameterized complexity of abduction in schaefer's framework. *J. Log. Comput.*, 31(1):266–296, 2021. doi:10.1093/logcom/exaa079.

[83] Yasir Mahmood, Arne Meier, and Johannes Schmidt. Parameterized complexity of logic-based argumentation in schaefer's framework. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6426–6434, 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16797.

[84] Yasir Mahmood and Jonni Virtema. Parameterised complexity of propositional logic in team semantics. *CoRR*, abs/2105.14887, 2021. URL: https://arxiv.org/abs/2105.14887, arXiv:2105.14887.

[85] Arne Meier. *Parametrised enumeration*. Habilitation thesis, Leibniz Universität Hannover, 2020. doi:10.15488/9427.

[86] Arne Meier, Martin Mundhenk, Thomas Schneider, Michael Thomas, Volker Weber, and Felix Weiss. The complexity of satisfiability for fragments of hybrid logic — Part I. In *Proc. MFCS*, volume 5734 of *LNCS*, pages 587–599, 2009.

[87] Arne Meier and Christian Reinbold. Enumeration complexity of poor man's propositional dependence logic. In Flavio Ferrarotti and Stefan Woltran, editors, *Foundations of Information and Knowledge Systems - 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14-18, 2018, Proceedings*, volume 10833 of *Lecture Notes in Computer Science*, pages 303–321. Springer, 2018. doi:10.1007/978-3-319-90050-6_17.

[88] Charles G. Morgan. Hypothesis generation by machine. *Artif. Intell.*, 2(2):179–187, 1971. doi:10.1016/0004-3702(71)90009-9.

[89] Gustav Nordh and Bruno Zanuttini. What makes propositional abduction tractable. *Artif. Intell.*, 172(10):1245–1284, 2008. doi:10.1016/j.artint.2008.02.001.

[90] Charles S. Peirce. Collected papers of charles sanders peirce. In Charles Hartshorne and Paul Weiss, editors, *Collected papers of Charles Sanders Peirce*. Cambridge, Massachusetts: The Belknap Press of Harvard University Press, 1931-1958. URL: https://www.hup.harvard.edu/catalog.php?isbn=9780674138001.

[91] Yun Peng and James A. Reggia. *Abductive Inference Models for Diagnostic Problem-Solving*. Artificial Intelligence. Springer-Verlag New York, 1990. doi:10.1007/978-1-4419-8682-5.

[92] Andreas Pfandler, Reinhard Pichler, and Stefan Woltran. The complexity of handling minimal solutions in logic-based abduction. *J. Log. Comput.*, 25(3):805–825, 2015. doi:10.1093/logcom/exu053.

[93] Andreas Pfandler, Stefan Rümmele, and Stefan Szeider. Backdoors to abduction. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 1046–1052, 2013. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6912.

[94] David Poole. Normality and faults in logic-based diagnosis. In N. S. Sridharan, editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 1304–1310. Morgan Kaufmann, 1989. URL: http://ijcai.org/Proceedings/89-2/Papers/073.pdf.

[95] Harry E. Pople. On the mechanization of abductive logic. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, CA, USA, August 20-23, 1973*, pages 147–152, 1973. URL: http://ijcai.org/Proceedings/73/Papers/017.pdf.

[96] Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic. (AM-5)*. Princeton University Press, 2016. URL: https://doi.org/10.1515/9781400882366, doi:doi:10.1515/9781400882366.

[97] Emil Leon Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[98] Raymond Reiter. On closed world data bases. In *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977*, pages 55–76, 1977. doi:10.1007/978-1-4684-3384-5_3.

[99] Steffen Reith. *Generalized satisfiability problems*. PhD thesis, Julius Maximilians University Würzburg, Germany, 2001. URL: http://opus.bibliothek.uni-wuerzburg.de/opus/volltexte/2002/7/index.html.

[100] Neil Robertson and Paul D. Seymour. Graph minors. v. excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.

[101] Marko Samer and Stefan Szeider. Fixed-parameter tractability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 425–454. IOS Press, 2009. doi:10.3233/978-1-58603-929-5-425.

[102] Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.

[103] Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

[104] Henning Schnoor and Ilka Schnoor. Partial polymorphisms and constraint satisfaction problems. In *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, pages 229–254, 2008. doi:10.1007/978-3-540-92800-3_9.

[105] Gerhard Schurz. Patterns of abduction. *Synth.*, 164(2):201–234, 2008. doi:10.1007/s11229-007-9223-4.

[106] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[107] Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. *Theory Comput. Syst.*, 57(4):843–891, 2015. doi:10.1007/s00224-014-9586-0.

[108] Michael Thomas. The complexity of circumscriptive inference in post's lattice. In *Proc. 10th International Conference on Logic Programming and Nonmonotonic Reasoning*, volume 5753 of *Lecture Notes in Computer Science*, pages 209–302. Springer, 2009.

[109] Jouko A. Väänänen. *Dependence Logic - A New Approach to Independence Friendly Logic*, volume 70 of *London Mathematical Society student texts*. Cambridge University Press, 2007. URL: http://www.cambridge.org/de/knowledge/isbn/item1164246/?site_locale=de_DE.

[110] Jouko A. Väänänen. Modal dependence logic. In Krzystof Apt and Robert van Rooij, editors, *New Perspectives on Games and Interaction*. Amsterdam University Press, 2008.

[111] Jonni Virtema. *Approaches to Finite Variable Dependence: Expressiveness and Computational Complexity*. PhD thesis, School of Information Sciences of the University of Tampere, 2014. Available online at https://trepo.tuni.fi/handle/10024/95328.

[112] Jonni Virtema. Complexity of validity for propositional dependence logics. *Inf. Comput.*, 253:224–236, 2017. doi:10.1016/j.ic.2016.07.008.

[113] Fan Yang. Uniform definability in propositional dependence logic. *Rev. Symb. Log.*, 10(1):65–79, 2017. doi:10.1017/S1755020316000459.

[114] Fan Yang and Jouko Väänänen. Propositional logics of dependence. *Ann. Pure Appl. Logic*, 167(7):557–589, 2016. doi:10.1016/j.apal.2016.03.003.

CURRICULUM VITÆ

Personal Information

Name:	Yasir Mahmood
Date of Birth:	May 11, 1992 in Khushab (Pakistan)
Martial Status:	Married, since 2018

Education

1997–2009	Matriculation and Intermediate Studies Board of Intermediate and Secondary Education Faisalabad, Pakistan
2009–2013	Bachelor of Honours in Mathematics, Excellent (3.83/4.00) University of Sargodha, Pakistan
2014–2018	Master of Science in Mathematics, Good (4.0/5.0) University of Helsinki, Finland
2016–2017	Subject Teacher Education (STEP), Excellent (5.0/5.0) University of Helsinki, Finland

Employment

12/2010–06/2014	Private Tutor for Mathematics Grade 10–12 and Bachelor students
11/2017–06/2018	Substitute Teacher for Physics and Math Etelä Tapiolan Lukio (High School) Espoo, Finland
09/2018–present	Research Assistant, Ph.D., Leibniz Universität Hannover

LIST OF PUBLICATIONS

[1] Johannes Klaus Fichte, Markus Hecher, Yasir Mahmood, and Arne Meier. Decomposition-guided reductions for argumentation and treewidth. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1880–1886, 2021. doi:10.24963/ijcai.2021/259.

[2] Juha Kontinen, Arne Meier, and Yasir Mahmood. A parameterized view on the complexity of dependence and independence logic. *Journal of Logic and Computation*, 11 2022. exac070. arXiv:https://academic.oup.com/logcom/advance-article-pdf/doi/10.1093/logcom/exac070/47041664/exac070.pdf, doi:10.1093/logcom/exac070.

[3] Juha Kontinen, Arne Meier, and Yasir Mahmood. A parameterized view on the complexity of dependence logic. In *Logical Foundations of Computer Science - International Symposium, LFCS 2022, Deerfield Beach, FL, USA, January 10-13, 2022, Proceedings*, pages 125–142, 2022. doi:10.1007/978-3-030-93100-1_9.

[4] Yasir Mahmood and Arne Meier. Parameterised complexity of model checking and satisfiability in propositional dependence logic. In Andreas Herzig and Juha Kontinen, editors, *Foundations of Information and Knowledge Systems - 11th International Symposium, FoIKS 2020, Dortmund, Germany, February 17-21, 2020, Proceedings*, volume 12012 of *Lecture Notes in Computer Science*, pages 157–174. Springer, 2020. doi:10.1007/978-3-030-39951-1_10.

[5] Yasir Mahmood and Arne Meier. Parameterised complexity of model checking and satisfiability in propositional dependence logic. *Ann. Math. Artif. Intell.*, 90(2):271–296, 2022. doi:10.1007/s10472-021-09730-w.

[6] Yasir Mahmood, Arne Meier, and Johannes Schmidt. Parameterised complexity of abduction in schaefer's framework. In *Logical Foundations of Computer Science - International Symposium, LFCS 2020, Deerfield Beach, FL, USA, January 4-7, 2020, Proceedings*, pages 195–213, 2020. doi:10.1007/978-3-030-36755-8_13.

[7] Yasir Mahmood, Arne Meier, and Johannes Schmidt. Parameterized complexity of abduction in schaefer's framework. *J. Log. Comput.*, 31(1):266–296, 2021. doi:10.1093/logcom/exaa079.

[8] Yasir Mahmood, Arne Meier, and Johannes Schmidt. Parameterized complexity of logic-based argumentation in schaefer's framework. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6426–6434, 2021. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16797.

[9] Yasir Mahmood and Jonni Virtema. Parameterised complexity of propositional logic in team semantics. *CoRR*, abs/2105.14887, 2021. URL: https://arxiv.org/abs/2105.14887, arXiv:2105.14887.

LIST OF TALKS

The author has presented his (joint) work in front of researchers at various workshops and conferences. A complete list of the talks by the author is given below.

1. Dagstuhl Seminar on Logics for Dependence and Independence, 2019 Wadern.
   Title: *Initial Steps into the Parametrised Complexity of Dependence Logic*

2. Nordic Complexity Workshop, 2019 Jena.
   Title: *Parameterized Complexity of Propositional Dependence Logic*

3. 78. Workshop on Algorithms and Complexity, Theorietag 2019 Berlin.
   Title: *Parameterized Complexity of Abduction in Schaefer's Framework*

4. International Symposium on Logical Foundations of Computer Science, 2020 FL USA.
   Title: *Parameterized Complexity of Abductive Reasoning in Schaefer's Framework*

5. Symposium on Foundations of Information and Knowledge Systems, 2020 Dortmund.
   Title: *Parameterized Complexity of Propositional Dependence Logic*

6. 79. Workshop on Algorithms and Complexity, Theorietag 2020 (Virtual).
   Title: *Parameterised Complexity of Argumentation*

7. Association for the Advancement of Artificial Intelligence, AAAI 2021 (Virtual).
   Title: *Parameterised Complexity of Argumentation in Schaefer's Framework* [2]

8. International Joint Conference on Artificial Intelligence, IJCAI 2021 (Virtual).
   Title: *Decomposition-Guided Reductions for Argumentation and Treewidth* [3]

9. Workshop on Logics of Dependence and Independence, LoDE 2021 (Virtual).
   Title: *Parameterised Complexity of Propositional Logic in Team Semantics*

[2] Recorded talk jointly with Arne Meier & Johannes Schmidt
[3] Recorded talk jointly with Arne Meier, Markus Hecher & Johannes Fichte

10. Workshop on Logics of Dependence and Independence, LoDE 2021 (Virtual).
    Title: *Parameterized Complexity of Model Checking in Dependence Logic*

11. 80. Workshop on Algorithms and Complexity, Theorietag 2021 (Virtual).
    Title: *Parameterized Complexity of Team-Based Logics*

12. International Symposium on Logical Foundations of Computer Science, 2022 (Virtual).
    Title: *A Parameterized view on the Complexity of Dependence Logic*

13. CIRM Conference on Logic, Databases and Complexity, 2022 Marseille.
    Title: *Counter-Claims Augmented Argumentation Frameworks*

14. 82. Workshop on Algorithms and Complexity, Theorietag 2022 Frankfurt.
    Title: *Dung's Argumentation Frameworks Augmented with Counter-Claims*