Complexity Classifications for Nonmonotonic Reasoning and Enumeration

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

Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation
von

Dipl.-Math. Johannes Schmidt

geboren am 4. Juni 1984 in Hannover

2012

Referent:	Heribert Vollmer, Leibniz Universität Hannover
Korreferent:	Till Tantau, Universität zu Lübeck
Tag der Promotion:	16. Oktober 2012

Acknowledgments

I would like to thank my supervisor Nadia Creignou for the great collaboration during the three years of my PhD – it was always a pleasure and I learned a lot. I also thank my supervisor Heribert Vollmer and my coauthors for the joint research. I thank Heribert Vollmer, Joachim Reineke and Michael Holz for their great lectures that showed me the beauty and fascination of mathematics. Finally, I thank my family for their support and my sister Annalisa for her English advices. I thank in particular my wife Annika for her love and support especially in the final phase of writing.

Remerciements

Je remercie ma directrice de thèse Nadia Creignou pour la grandiose collaboration que l'on a partagé durant ces trois ans de thèse. Ça a toujours été un plaisir dans lequel j'y ai beaucoup appris. Je remercie également mon directeur de thèse Heribert Vollmer et mes coauteurs pour la recherche que l'on a effectué ensemble. Je remercie Heribert Vollmer, Joachim Reineke et Michael Holz pour leurs beaux cours qui m'ont montré la beauté et la fascination des mathématiques. Finalement je remercie ma famille pour son soutien et ma sœur Annalisa pour ses conseils en anglais. Je remercie en particulier ma femme Annika pour son amour et son soutien spécialement durant la période de rédaction.

Danksagung

Ich möchte meiner Doktormutter Nadia Creignou für die großartige Zusammenarbeit während der drei Jahre meiner Promotion danken – es war stets ein Vergnügen und ich habe viel gelernt. Ich danke auch meinem Doktorvater Heribert Vollmer und meinen Koautoren für die gemeinsame Forschung. Ich danke Heribert Vollmer, Joachim Reineke und Michael Holz für ihre tollen Vorlesungen die mir die Schönheit und Faszination der Mathematik gezeigt haben. Schließlich danke ich meiner Familie für ihre Unterstützung und meiner Schwester Annalisa für ihre Englischberatung. Ich danke besonders meiner Frau Annika für ihre Liebe und Unterstütung speziell in der finalen Phase des Schreibens.

Abstract

In this thesis we consider the computational complexity of problems from two central formalisms of nonmonotonic reasoning: abduction and argumentation. The first one is designed to formalize the process of finding explanations for some observed manifestation, the second (and more recent) one gives a theoretical framework to formalize the process of argumentation. We focus on the explanation-existence problem for abduction and on the argument-existence problem for argumentation. Considered in full propositional logic these problems are believed to be computationally costly tasks (they are often situated at the second level of the polynomial hierarchy). With the purpose of understanding sources of hardness and of identifying tractable fragments of propositional logic we consider several abduction and argumentation problems in two well-established settings allowing for complexity classifications. In the first one, *Post's Framework*, restrictions are made on the allowed connectives in the used formulæ, whereas in the second one, *Schaefer's Framework*, one considers formulæ in conjunctive normal form, where the clauses are generalized to applications of arbitrary Boolean relations to variables and one restricts the allowed type of relations. We discuss differences and common features between the explanation-existence and the argument-existence problem in function of the two chosen frameworks.

Finally, we consider *enumeration*. In particular we consider the problem of enumerating all solutions (models) of a propositional formula by non-decreasing weight in Schaefer's framework (the weight of a model being the number of variables assigned to true). We identify precisely those relations for which this enumeration problem becomes tractable (i.e., is enumerable with polynomial delay) and those relations for which it remains intractable, unless the polynomial hierarchy collapses.

Keywords: Complexity Classifications, Abduction, Argumentation

Résumé

Nous considérons dans cette thèse la complexité algorithmique de problèmes émanant de deux formalismes de raisonnement non-monotone: l'abduction et l'argumentation. Le premier est destiné à formaliser le processus de trouver des explications pour une manifestation observée, le second (et plus récent) offre un cadre théorique pour formaliser le processus de l'argumentation. Nous nous concentrons sur le problème d'existence d'une explication pour l'abduction et sur le problème d'existence d'un argument pour l'argumentation. Dans le cadre de la logique propositionnelle dans son ensemble ces problèmes sont considérés comme étant des tâches algorithmiques difficiles (ils sont souvent situés au deuxième niveau de l'hiérarchie polynomial). Notre but est d'une part de comprendre les sources de difficulté, et d'autre part d'identifier des fragments de la logique propositionnelle dans lequels ces problèmes sont résolubles efficacement. Pour cela nous considérons ces problèmes d'abduction et d'argumentation dans deux cadres bien-établis qui permettent des classifications de complexité : Le cadre de *Post* et celui de *Schaefer*. Dans le cadre de Post, des restrictions sont faites sur les connecteurs autorisés dans les formules utilisées. Dans le cadre de Schaefer, on considère les formules en forme normale conjonctive généralisée, les "clauses" sont alors des applications de relations booléennes à des variables et on restreint le type des relations autorisées. Nous discutons les points communs et les différences du point de vue de la complexité, entre le problème d'existence d'une explication et le problème d'existence d'un argument, et cela dans chacun des deux cadres présentés ci-dessus.

Finalement, nous considérons l'*énumération*. En particulier nous considérons le problème d'énumérer toutes les solutions (modèles) d'une formule propositionnelle par poids croissant dans le cadre de Schaefer (le poids d'un modèle étant le nombre de variables assignées à vrai). Nous identifions précisément les relations pour lesquelles ce problème d'énumération devient efficacement résoluble (i.e., est énumérable avec délai polynomial) et les relations pour lesquelles il reste difficile, à moins que l'hiérarchie polynomiale ne s'écrase.

Mots clés: Classifications en Complexité, Abuction, Argumentation

Zusammenfassung

In dieser Dissertation betrachten wir die Berechnungskomplexität zwei zentraler Formalismen des Nichtmonotonen Schließens: Abduktion und Argumentation. Ersterer ist dazu entworfen worden den Prozess des Erklärens von Beobachtungen zu formalisieren, der zweite (und jüngere) bietet ein theoretisches Umfeld um den Prozess des Argumentierens zu formalisieren. Wir konzentrieren uns auf das Erklärungs-Existenz Problem für Abduktion und auf das Argument-Existenz Problem für Argumentation. In vollständiger Aussagenlogik wird allgemein angenommen, dass diese Probleme berechnungstechnisch komplex sind (sie liegen oft auf dem zweiten Niveau der Polynomialzeithierarchie). Mit dem Ziel die tieferen Gründe der Komplexität, und einfachere Varianten dieser Probleme in Fragmenten der Aussagenlogik zu identifizieren, betrachten wir verschiedene Abduktions- und Argumentationsprobleme in zwei etablierten Umfeldern die Komplexitätsklassifikationen erlauben. Im ersten, *Post's Framework*, werden die in Formeln erlaubten Operatoren eingeschränkt, während man im zweiten, *Schaefer's Framework*, Formeln in verallgemeinerter konjunktiver Normalform betrachtet, wobei Klauseln zu Anwendungen beliebiger Boolescher Relationen auf Variablen verallgemeinert werden und man die erlaubten Typen von Relationen einschränkt. Wir besprechen Unterschiede und Gemeinsamkeiten zwischen dem Erklärungs-Existenz und dem Argument-Existenz Problem in Funktion der gewählten Umfelder.

Schließlich behandeln wir *Enumeration*. Speziell betrachten wir in Schaefer's Framework das Problem alle Lösungen (Modelle) einer aussagenlogischen Formel nach nicht-absteigendem Gewicht aufzuzählen (das Gewicht eines Modells ist die Anzahl der Variablen die auf wahr gesetzt sind). Wir identifizieren exakt jene Relationen für die dieses Aufzählungsproblem einfach (d.h. mit polynomiellem Delay aufzählbar) ist und solche Relationen für die es komplex bleibt (unter der Annahme, dass die Polynomialzeithierarchie nicht kollabiert).

Schlagworte: Komplexitätsklassifikationen, Abduktion, Argumentation

Contents

1 Introduction **15**

2 Preliminaries **23**
2.1 Propositional Logic 23
2.2 Complexity Theory 24
2.2.1 Complexity Classes 24
2.2.2 Reductions and complete Problems 26
2.3 Abduction and Argumentation 27

3 Classifications in Post's Framework **33**
3.1 Post's Framework 33
3.1.1 Boolean clones and Post's Lattice 34
3.1.2 Post's lattice as a Tool for Complexity Analysis . . 36
3.1.3 Parameterizing by B-formulæ 39
3.2 The complexity of Symmetric Abduction 40
3.2.1 Technical results and tools 40
3.2.2 The complexity of the Existence Problem 42
3.2.3 The complexity of the Verification Problem 50
3.3 The complexity of Positive Abduction 52
3.3.1 Technical results and tools 52
3.3.2 The complexity of the Existence Problem 53
3.3.3 The complexity of the Verification Problem 59
3.3.4 Overview of results for Abduction 60
3.4 The complexity of Argumentation 62
3.4.1 Technical results and tools 62
3.4.2 The complexity of the Existence Problem 63
3.4.3 The complexity of the Verification Problem 65

3.4.4 The complexity of Relevance and Dispensability . 69
3.4.5 Overview of results for Argumentation 72

4 Classifications in Schaefer's framework 83
4.1 Schaefer's framework 83
4.1.1 Preliminaries 84
4.1.2 Background from Universal Algebra 86
4.1.3 Schaefer's Framework and the Galois connection . 88
4.1.4 Implementation results 91
4.2 The complexity of Argumentation 96
4.2.1 Complexity of the Existence Problem 96
4.2.2 Complexity of the Verification Problem 100
4.2.3 Overview of results for Argumentation 103
4.3 Enumeration of Models 105
4.3.1 Complexity of Enumeration 105
4.3.2 Polynomial Delay Algorithms 106
4.3.3 Hardness Results 110

5 Concluding remarks 121

Chapter 1

Introduction

At the beginning of the 20th century the German mathematician David Hilbert dreamed of grounding all existing mathematical theories to a finite and complete set of axioms, and of proving that these axioms were consistent (Hilbert's program). In 1931 Kurt Gödel arguably proved that this will not be possible: he proved that any consistent and complete theory of *interesting* expressiveness will be based on an uncomputable set of axioms. This historical result was ground-breaking and influenced significantly mathematics of the 20th century. Together with the reformulation by Alan Turing this result was at the same time the first large spreading of the notion of uncomputability: the mathematical community became aware of problems that in general no computer may solve within finite time. The existence of uncomputable problems was naturally sobering since it showed the limitations of computers, it showed that computers may not compute *anything*. In the same time it was satisfying that there was at least mathematical evidence of this fact. In practice, it became clear that not every computable problem can efficiently be solved: there are computable problems that require provably at least exponential time (i.e., the function describing the running time of a solving algorithm in function of the size of the input instance is an exponential one). Problems solvable in *exponential* time are generally *not* regarded as efficiently solvable for practical use (the class of such problems is denoted EXP), since only small increments of the input size cause huge explosions of the running time. However, *polynomial* running time (leading to the class P) is commonly considered an appropriate notion of efficiency. Unfortunately,

it appears that there are many problems of practical interest for which no efficient algorithms are known, but neither is there a mathematical proof which excludes the existence of such an algorithm. The best known class of such problems is the class NP, the class of *efficiently verifiable* problems. A prominent example of an NP-problem is the *Traveling Salesman Problem*: given a list of cities, their pairwise distances and a maximal length k, the question is whether there exists a route of length at most k that visits each city exactly once and returns to the origin city. This problem belongs to NP since it is apparently easy to verify whether a given route satisfies the requirements. But finding such a route appears to be difficult, the best known algorithms have still exponential running time (that is, roughly speaking, the best known method is to systematically try all possible routes). Since every efficiently solvable problem is also efficiently verifiable we have clearly $\mathrm{P} \subseteq \mathrm{NP}$. But whether the inclusion is strict is one of the most important open questions in theoretical computer science.

An important step in the investigation of the P-NP problem was the identification of NP-complete problems, which means, problems in NP that allow efficient translations from any other problem in NP. NP-complete problems may thus be viewed as the *hardest* problems in NP. The first problem shown to be NP-complete is the problem SAT, the satisfiability problem for propositional formulæ [Coo71]. Also the Traveling Salesman Problem is NP-complete, that is, there are efficient translations from and to the SAT-problem. If for any NP-complete problem a polynomial time algorithm was found, it would imply $\mathrm{P} = \mathrm{NP}$. So, while even for the hardest problems in NP we cannot exclude the existence of an efficient algorithm, we can at least give good reasons to *believe* that a problem does not admit an efficient algorithm: prove that it is equivalent to an NP-complete problem.

We will treat in this thesis problems that might be even harder so solve than NP-complete problems: problems that are known to be efficiently verifiable only with the help of an NP-oracle. That is, a verifying algorithm may ask questions to another NP-problem (the *oracle*) and gets the correct answer instantly (this class is denoted $\mathrm{NP}^{\mathrm{NP}}$ or alternatively Σ_2^{p}). One can continue this procedure of defining the class of problems being efficiently verifiable with the help of an $\mathrm{NP}^{\mathrm{NP}}$-oracle and so on, leading to the *polynomial hierarchy*. A last important complexity class we will encounter is the class DP of differences of NP-problems. The above mentioned notion of completeness can analogously be applied to

any other class.

We will consider in this thesis problems taken from the contexts of abduction and argumentation known to be complete for the above mentioned complexity classes. Thus, most likely there are no efficient algorithms to solve them. Our purpose is to introduce parameters in these problems that may regulate their complexity in order to identify fragments of lower complexity.

Abduction Usual *deduction* is the process of deducing a *conclusion* from a *major premise* (certain rules / knowledge about the world) and a *minor premise* (cases). Consider for this the following example from [Pop73]:

major premise: *All people with tuberculosis have bumps*
minor premise: *Mr. Jones has tuberculosis*
conclusion: *Mr. Jones has bumps*

Thus, knowing that *All people with tuberculosis have bumps* and that *Mr. Jones has tuberculosis*, we may deduce that *Mr. Jones has bumps*. In contrast, *induction* is the process of generalizing from the minor premise and associated observations (the conclusion) to the rules, that is hypothesizing the major premise. In our example, knowing that *Mr. Jones has tuberculosis* and observing that *Mr. Jones has bumps* we could hypothesize that maybe *All people with tuberculosis have bumps*. At last, *abduction* is the process of finding an explanation for a given observation and some general rules, that is, hypothesizing the minor premise from a given major premise and the conclusion. Observing that *Mr. Jones has bumps* and knowing that *All people with tuberculosis have bumps* we may consider *Mr. Jones has tuberculosis* a possible explanation.

For abduction it has become common to indicate the major premise as the *knowledge base* that contains certain facts and rules about the behavior of the world. Possible minor premises are the *hypothesis* from which explanations may be formed for the *observation* or *manifestation* (the conclusion). Abduction is said to be a form of *nonmonotonic reasoning* since adding information to the knowledge base may invalidate previously valid explanations. For instance, adding in the above example the simple statement that *Mr. Jones has no tuberculosis* renders invalid the previous explanation *Mr. Jones has tuberculosis*.

Nowadays abduction is a fundamental and important form of non-

monotonic reasoning that has extensively been studied within Artificial Intelligence. As illustrates our example a possible field of application is medical diagnosis [BATJ89]. Other application areas are text analysis [HSAM93], system diagnosis [SW01], configuration problems [AFM02] and temporal knowledge bases [BL00].

There are several approaches to formalize the problem of abduction. In this thesis we focus on *logic-based abduction*. Following the formalization of Eiter and Gottlob [EG95], we are given the knowledge base Γ as a set of propositional formulæ, the hypotheses as a set of variables A, and the manifestation φ as a propositional formula. An *explanation* (or solution) for an instance (Γ, A, φ) is a set $E \subseteq A$ such that $\Gamma \cup E$ is consistent and logically implies the manifestation φ. In this thesis we will consider the following two decision problems:

1. *explanation-existence* (ABD for short)
   given (Γ, A, φ), decide whether there exist explanations;

2. *explanation-validity* (ABD-CHECK for short)
   given (Γ, A, φ, E), decide whether E is a valid explanation.

In full propositional logic the first problem is Σ_2^{p}-complete, while the second problem is DP-complete [EG95].

Argumentation According to the seminal work by Dung [Dun95], argumentation can be seen as a generalization of many forms of nonmonotonic reasoning previously developed (e.g., Reiter's Default Logic [Rei80] or Moore's Autoepistemic Logic [Moo85]). It is therefore, as previous formalisms of reasoning with incomplete information, an attempt to make machine-based reasoning more flexible, more powerful, more human like. The basic principle of those formalisms is always the same: an assumption or conclusion may be acceptable as long as no contradictory information comes up. In argumentational reasoning a statement is believable if it can be argued successfully against attacking arguments [Dun95].

As already implied, argumentation as a subject of intensive research within Artificial Intelligence is a relatively young discipline. One can identify, among others, two important lines of research: *abstract* argumentation and *logic-based* argumentation. Abstract argumentation focuses on the relation between arguments without taking their internal structure into account. A basis for considerations is an *(abstract) argumentation framework* [Dun95] that can be visualized by a directed graph whose nodes

are arguments, a directed edge (a, b) indicates that argument a attacks argument b. One uses certain semantics to find acceptable subsets of arguments by analyzing solely this graph obtained from the arguments and conflicts. The term *abstract* indicates that both the nature of arguments and the nature of the attack relation are ignored.

In this thesis we focus on *logic-based argumentation*. One goal in logic-based argumentation is to find a concrete formal representation of an argument and then to define - on top of this concept - notions such as counterarguments, rebuttals or undercuts (see [BH01]). Many proposals consider an argument as a pair (Φ, α), where Φ is the *support* which has to entail (or justify) the *claim* α. Let us illustrate this on an example from [BH08].

Argument 1

Support:	Simon Jones is a public person, so we can publicize details about his private life.
Claim:	We can publicize that Simon Jones is having an affair.

Argument 2

Support:	Simon Jones just resigned from the House of Commons; hence, he is no longer a public person.
Claim:	Simon Jones is no longer a public person.

Each argument is well-formed since its support implies its claim. Though, together they are conflicting: the claim of argument 2 contradicts the support of argument 1. That is, argument 2 *attacks* argument 1. Note that the attack relation is generally not symmetric.

To get more precise in the formalization, we define an argument as a pair (Φ, α), where the support Φ is a consistent set (or a minimal consistent set) of formulæ from a given knowledge base Δ that entails the claim α which is a formula (see, for example [Cay95, BH01, AC02, GS04, DKT06]). Different logical formalisms provide different definitions for consistency and entailment and hence give different options for defining the notion of an argument. In this thesis we formalize arguments in propositional logic. Among others, we will consider the complexity of the following central decision problems:

1. *argument-existence* (ARG for short)
   given (Δ, α), decide whether there exists a support $\Phi \subseteq \Delta$ for α;

2. *argument-validity* (ARG-CHECK for short)
   given (Φ, α), decide whether Φ is a valid support for α.

Computing the support for an argument underlies many reasoning problems in logic-based argumentation, for instance, the computation of argument trees as proposed in [BH01]. In full propositional logic the argument-existence problem is Σ_2^{p}-complete [PWA03], while the argument-validity problem is DP-complete.

Complexity Classifications We would like to draw a more detailed picture of the complexity of the presented problems from abduction and argumentation. We want to understand sources of hardness and to identify tractable variants. A *complexity classification* for a problem appears to be an appropriate method to do so: introducing a parameter to a problem, a (complete) complexity classification permits to indicate for each possible value of the parameter the precise complexity of the parameterized problem. In the literature, among others, two central frameworks have been proposed that allow for complexity classifications. We will refer to them as *Post's Framework* and *Schaefer's Framework*. Both frameworks give a formal method of obtaining fragments of propositional logic and are thus well-suited to parameterize the problems we are interested in. We will give detailed introductions to the frameworks in Chapters 3.1 (Post) and 4.1 (Schaefer).

Results Abduction has already been considered in Schaefer's Framework [CZ06, NZ08]. Our new contributions are to consider both abduction and argumentation in Post's Framework (Chapters 3.2, 3.3 and 3.4) and argumentation in Schaefer's Framework (Chapter 4.2). At last we will address in Chapter 4.3 a problem of completely different nature: Enumeration. In contrast to decision problems, Enumeration requires not to decide whether there are solutions or not, but to explicitly generate all solutions of a problem instance. We address the problem of enumerating the satisfying assignments (models) of a propositional formula in Schaefer's Framework.

Publications The results of Chapter 3 have previously been published in [CST11] (abduction) and [CSTW11] (argumentation), except for the classifications for the explanation-validity problem for abduction. The results of Chapter 4 on enumeration previously appeared in [COS11],

the ones on argumentation have been accepted for publication at *Fourth International Conference on Computational Models of Argument, 2012* (COMMA'12).

Chapter 2

Preliminaries

We assume that the reader is familiar with basic mathematic objects such as sets, functions and relations and with basic notions from theoretical computer science such as Turing machines. Furthermore we assume familiarity with propositional logic.

2.1 Propositional Logic

The symbols 0 and 1 represent the Boolean constants *false* and *true*. The set of all propositional formulæ is denoted by $\mathcal{L}$. A *model* for a formula φ is a truth assignment to the set of its variables that satisfies φ. An assignment to an ordered set of variables $(x_1, \ldots, x_n)$ will generally be identified with an n-tuple over $\{0,1\}$. We denote by $mod(\varphi)$ the set of models of φ. We say that a formula φ is *satisfiable* (*sat* for short) if $mod(\varphi) \neq \emptyset$ and *unsatisfiable* (*unsat* for short) if $mod(\varphi) = \emptyset$. Let α, β be variables or constants. We denote by $\varphi[\alpha/\beta]$ the formula obtained from φ by replacing all occurrences of α with β. If X is a set of variables, $\varphi[X/\beta]$ is the formula obtained by replacing all occurrences of all variables of X by β. For a given formula φ we denote by $\mathrm{Vars}(\varphi)$ the set of variables occurring in φ. We extend this definition on sets of formulæ Γ as $\mathrm{Vars}(\Gamma) = \bigcup_{\varphi\in\Gamma} \mathrm{Vars}(\varphi)$. We identify finite Γ with the conjunction of all the formulæ in Γ, $\bigwedge_{\varphi\in\Gamma} \varphi$. Naturally, $\Gamma[\alpha/\beta]$ then stands for $\bigwedge_{\varphi\in\Gamma} \varphi[\alpha/\beta]$. We will need the notion of *quantified (Boolean) formulæ* which extends propositional formulæ with the shorthands $\exists x\varphi(x) := \varphi(0) \vee \varphi(1)$ and $\forall x\varphi(x) := \varphi(0) \wedge \varphi(1)$, where x is a variable and φ a quantified formula. A variable x is *free*

if it does not appear in the scope of $\exists x$ or $\forall x$. A quantified formula is *closed* if it has no free variables. A closed quantified formula is *valid* if it is equivalent to 1. We will also consider propositional formulæ with only existentially quantified variables that still have free variables. For a quantifier free formula φ and a vector $\vec{x} = (x_1, \ldots, x_n)$ of n variables the models of $\exists \vec{x} \varphi$ are thus $\bigcup_{c_1,\ldots,c_n \in \{0,1\}} mod(\varphi[x_1/c_1, \ldots, x_n/c_n])$. We say that two formulæ φ and ψ are *equivalent* (written $\varphi \equiv \psi$) if every assignment $\sigma : \mathrm{Vars}(\varphi) \cup \mathrm{Vars}(\psi) \to \{0,1\}$ on the combined variable sets satisfies φ if and only if it satisfies ψ. We say that two formulæ φ and ψ are *equisatisfiable* (written $\varphi \equiv_{SAT} \psi$) if φ is satisfiable if and only if ψ is satisfiable. We write $\varphi \models \psi$ if φ implies (entails) ψ, i.e., if ψ is satisfied by any assignment $\sigma : \mathrm{Vars}(\varphi) \cup \mathrm{Vars}(\psi) \to \{0,1\}$ that satisfies φ. In an *implication* $\varphi \models \psi$, we call φ the *premise* and ψ the *conclusion*.

A *literal* l is a variable x or its negation $\neg x$. A *positive* literal is a variable x, a *negative* literal is the negation of a variable $\neg x$. Given a set of variables V, $\mathrm{Lits}(V)$ denotes the set of all literals formed upon the variables in V, i.e., $\mathrm{Lits}(V) := V \cup \{\neg x \mid x \in V\}$. A *clause* is a disjunction of literals and a *term* is a conjunction of literals. A propositional formula φ is in *conjunctive normal form* (CNF for short) if it is a conjunction of clauses. The formula φ is in *disjunctive normal form* (DNF for short) if it is a disjunction of terms. The formula φ is in *kCNF* if it is a conjunction of clauses of size exactly k, φ is in *kDNF* if it is a disjunction of terms of size exactly k.

2.2 Complexity Theory

2.2.1 Complexity Classes

We present here the complexity classes we will meet in this thesis.

We identify a decision problem with a language, i.e., its set of "yes"-instances. We call a decision problem *trivial* if it can be solved within constant time. The class P (resp. NP) is defined as the set of languages (problems) that can be solved in polynomial time by a deterministic (resp. non-deterministic) Turing machine. The class EXP is the set of languages that can be solved in exponential time on a deterministic Turing machine, where the class LOGSPACE is the set of languages that can be solved in logarithmic space by a deterministic Turing machine. The complement of NP is denoted by coNP, that is the class of languages whose set of

$$\text{LOGSPACE} \subseteq \text{P} \subsetneq \begin{matrix} \text{NP} \\ \text{coNP} \end{matrix} \subsetneq \text{DP} \subseteq \Sigma_2^{\text{p}} \subseteq \Sigma_3^{p} \subseteq \ldots \subseteq \text{EXP}$$

Figure 2.1: Known inclusions between complexity classes

"no"-instances lies in NP. The class NP can equivalently be defined as languages that can be *verified* in polynomial time. That is, NP is the class of languages A for which there is a language $B \in \text{P}$ and a polynomial p such that for any instance x holds

$$x \in A \iff \exists w, |w| \leq p(|x|) : (x, w) \in B.$$

The class DP is the set of languages of differences between NP-languages, i.e., $\text{DP} = \{A \backslash B \mid A, B \in \text{NP}\} = \{A \cap B \mid A \in \text{NP}, B \in \text{coNP}\}$. For a complexity class C and a problem A we denote C^A the class of problems that can be decided by a C-machine using an A-oracle (that is a Turing machine may ask questions of the style *is* $x \in A$*?* and gets the correct answer in constant time). For a complexity class D we define $C^D := \bigcup_{A \in D} C^A$. The classes of the polynomial hierarchy are defined as $\Sigma_{k+1}^p := \text{NP}^{\Sigma_k^p}$ with $\Sigma_0^p := \text{P}$. Thus we have $\Sigma_0^p = \text{P}$, $\Sigma_1^p = \text{NP}$ and the class $\Sigma_2^{\text{p}} = \text{NP}^{\text{NP}}$ is the set of languages that can be decided in polynomial time by a non-deterministic Turing machine that uses an oracle $A \in \text{NP}$. It clearly holds $\Sigma_k^p \subseteq \Sigma_{k+1}^p$ for all k.

Figure 2.2.1 visualizes known inclusions of the polynomial hierarchy and the all compounding class EXP. It is generally believed that the higher a problem lies in the polynomial hierarchy, the more time it will take to solve it, the *harder* it is. The problems in P are generally said to be efficiently solvable or to be *tractable*, whereas the above classes are said to represent different degrees of *intractable* problems. It is worth mentioning that up to know none of the indicated inclusions is known to be strict, except $\text{P} \subsetneq \text{EXP}$. Even the total collapse of the whole polynomial hierarchy to LOGSPACE is not excluded. For more background information, the reader is referred to [Pap94].

2.2.2 Reductions and complete Problems

In order to compare the difficulty of problems we consider *logspace many-one reductions*, defined as follows: a language A is logspace many-one reducible to some language B (written $A \leq_{\mathrm{m}}^{\log} B$) if there exists a logspace-computable function f such that $x \in A$ if and only if $f(x) \in B$. For some complexity class C, a language A is said to be C-*hard* if all languages in C are logspace many-one reducible to A. A language A is C-*complete* if $A \in C$ and A is C-hard. Complete problems of a class C are the *hardest* representatives of the class C, they do most likely not lie in lower classes. The logspace many-one reduction is reflexive and transitive. To show C-hardness of a problem A it suffices thus to reduce a problem B already known to be C-hard to A.

The satisfiability of propositional formulæ is the core of many complexity classes and therefore provides central complete problems for many classes. We list here some of the most relevant complete problems that we will use often to obtain new hardness results.

Problem:	3SAT (NP-complete according to [Coo71])
Instance:	A propositional formula φ in 3CNF.
Question:	Is φ satisfiable?

Problem:	POS-2IN3-SAT (NP-complete according to [Sch78])
Instance:	A propositional formula φ in 3CNF with only positive literals.
Question:	Is there an assignment to the variables of φ that sets in each clause exactly two variables to true?

Problem:	$\mathrm{QSAT}_{\exists,2}$ (Σ_2^{p}-complete according to [Wra77])
Instance:	A closed quantified Boolean formula of the form $\varphi = \exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m \psi$, where ψ is a propositional formula in 3DNF.
Question:	Is φ valid?

From the NP-completeness of 3SAT one quickly identifies the following canonical problem as DP-complete.

Problem:	SATUNSAT (DP-complete)
Instance:	Two propositional formulæ (φ, ψ) in 3CNF.
Question:	Is φ satisfiable and ψ unsatisfiable?

We will also use the following DP-complete problem as starting point to show DP-hardness.

Problem: CRITICAL-3SAT (DP-complete according to [PW88])
Instance: A propositional formula φ in 3CNF.
Question: Is φ unsatisfiable but removing any of its clauses makes it satisfiable?

2.3 Abduction and Argumentation

Abduction Given a knowledge base Γ and a manifestation φ, an *explanation* E is a set of literals that is consistent with Γ and that together with Γ entails the manifestation φ. That is, an explanation E has to satisfy

1. $\Gamma \cup E$ is consistent,
2. $\Gamma \cup E \models \varphi$.

Note that these conditions are equivalent to the followings:

(B1) $\Gamma \wedge E$ is satisfiable,

(B2) $\Gamma \wedge E \wedge \neg\varphi$ is unsatisfiable.

We define the *explanation-existence problem* as follows.

Problem: ABD
Instance: $\mathcal{I} = (\Gamma, A, \varphi)$, where $\Gamma \subseteq \mathcal{L}$, $A \subseteq \mathrm{Vars}(\Gamma)$ and $\varphi \in \mathcal{L}$
Question: Does there exist an explanation $E \subseteq \mathrm{Lits}(A)$?

We refer to the above defined abduction problem as *symmetric abduction*, since every variable of the hypotheses A may be taken positively or negatively to construct an explanation. An explanation E is *full* if it holds $\mathrm{Vars}(E) = A$. Note that every explanation can be extended to a full one. We will also consider *positive abduction*, where we are interested in purely positive explanations only. We call an explanation E a *positive explanation* if holds $E \subseteq A$.

Problem: P-ABD
Instance: $\mathcal{I} = (\Gamma, A, \varphi)$, where $\Gamma \subseteq \mathcal{L}$, $A \subseteq \mathrm{Vars}(\Gamma)$ and $\varphi \in \mathcal{L}$

Question: Does there exist an explanation $E \subseteq A$?

While in full propositional logic the positive and the symmetric abduction problem are equivalent, we will see that their complexity differs in fragments of propositional logic.

Example 2.3.1. *[Logic based abduction] Let the knowledge base* Γ *be given by the three formulæ*

$\text{Charly-is-lazy} \wedge \text{Charly-is-alone} \rightarrow \text{Charly-plays-truant}$,
$\text{Dad-is-shopping} \rightarrow \text{Charly-is-alone}$,
$\neg(\text{buses-are-running}) \rightarrow \text{Charly-plays-truant} \wedge \neg(\text{Dad-is-shopping})$.

Let the hypotheses be the three variables
$\{\text{buses-are-running}, \text{Dad-is-shopping}, \text{Charly-is-lazy}\}$

Imagine we observe Charly-plays-truant. *The set* $\{\text{Charly-is-lazy}, \text{Charly-is-alone}\}$ *is no valid explanation, since* Charly-is-alone *is no hypothesis. Though,*

$$E_1 = \{\neg(\text{buses-are-running})\}$$

is a valid explanation. Note that E_1 *is no positive explanation. A positive explanations would be*

$$E_2 = \{\text{Dad-is-shopping}, \text{Charly-is-lazy}\}.$$

You can observe the nonmonotonicity of abduction by adding to the knowledge base the fact $\neg(\text{Charly-is-alone})$. *This will invalidate* E_2 *as an explanation since* E_2 *is not consistent anymore with the knowledge base.*

Besides the explanation existence problem we are also interested in the decision problem of validity. We define the *explanation-validity* problem as follows.

Problem: ABD-CHECK
Instance: $\mathcal{I} = (\Gamma, A, \varphi, E)$, where $\Gamma \subseteq \mathcal{L}$, $A \subseteq \text{Vars}(\Gamma)$, $\varphi \in \mathcal{L}$ and $E \subseteq \text{Lits}(A)$
Question: Is E an explanation?

We will also consider the positive variant where we restrict our focus on purely positive explanations.

Problem: P-ABD-CHECK
Instance: $\mathcal{I} = (\Gamma, A, \varphi, E)$, where $\Gamma \subseteq \mathcal{L}$, $A \subseteq \text{Vars}(\Gamma)$, $\varphi \in \mathcal{L}$ and $E \subseteq A$
Question: Is E an explanation?

Argumentation We will denote the knowledge base by Δ, representing a large depository of information, from which arguments can be constructed for arbitrary claims.

Following [BH01], an argument is a pair (Φ, α), where Φ is a set of formulæ and α is a formula such that

1. Φ is consistent,
2. $\Phi \models \alpha$,
3. Φ is minimal with this last property, i.e., no proper subset of Φ entails α.

We say that (Φ, α) is an *argument* for α. If $\Phi \subseteq \Delta$ then it is said to be an *argument in* Δ. We call α the *claim* (or *consequent*) and Φ the *support* of the argument. Note that these three conditions are equivalent to the followings:

(C1) Φ is satisfiable,

(C2) $\Phi \wedge \neg\alpha$ is unsatisfiable (i.e., $\Phi \models \alpha$), and

(C3) for all $\varphi \in \Phi$, $(\Phi \setminus \{\varphi\}) \cup \{\neg\alpha\}$ is satisfiable (i.e., Φ is minimal).

We define the *argument-existence problem* as follows.

Problem: ARG
Instance: $\mathcal{I} = (\Delta, \alpha)$, where $\Delta \subseteq \mathcal{L}$ and $\alpha \in \mathcal{L}$.
Question: Does there exist Φ such that (Φ, α) is an argument in Δ?

Besides the decision problem for the existence of an argument we are interested in the decision problems for validity, relevance and dispensability. These problems can in particular be used to give general enumeration schemes of arguments, a central task in argumentation [CSTW11].

Problem: ARG-CHECK
Instance: $\mathcal{I} = (\Phi, \alpha)$, where $\Phi \subseteq \mathcal{L}$ and $\alpha \in \mathcal{L}$.
Question: Is (Φ, α) an argument?

Problem: ARG-REL
Instance: $\mathcal{I} = (\Delta, \alpha, \varphi)$, where $\Delta \subseteq \mathcal{L}($ and $\alpha, \varphi \in \mathcal{L}$.

Question: Does there exist an argument (Φ, α) in Δ such that $\varphi \in \Phi$?

Problem: Arg-Disp
Instance: $\mathcal{I} = (\Delta, \alpha, \varphi)$, where $\Delta \subseteq \mathcal{L}$ and $\alpha, \varphi \in \mathcal{L}$.
Question: Does there exist an argument (Φ, α) in Δ such that $\varphi \notin \Phi$?

Abd versus Arg We will observe in this thesis that the explanation-existence problem for abduction and the argument-existence problem for argumentation show quite different behavior in fragments of propositional logic. However, we want to outline here their proximity in full propositional logic. In full propositional logic the abduction problem Abd and the argumentation problem Arg are equivalent (with respect to logspace many-one reductions) since they are both complete for the second level of the polynomial hierarchy ([EG95, PWA03]). Indeed there are very simple reductions proving this equivalence. We give here exemplary the reductions between P-Abd and Arg.

1. P-Abd $\leq_{\mathrm{m}}^{\log}$ Arg: $(\Gamma, A, \varphi) \mapsto (\Delta, \alpha)$, where
   $\Delta := \{\bigwedge_{\psi \in \Gamma} \psi\} \cup A$,
   $\alpha := \varphi \wedge \bigwedge_{\psi \in \Gamma} \psi$.

2. Arg $\leq_{\mathrm{m}}^{\log}$ P-Abd: $(\Delta, \alpha) \mapsto (\Gamma, A, \varphi)$, where
   $\Delta = \{\varphi_1, \ldots, \varphi_n\}$,
   $A := \{x_1, \ldots, x_n\}$,
   $\Gamma := \{x_i \leftrightarrow \varphi_i \mid 1 \leq i \leq n\}$,
   $\varphi := \alpha$.

For fragments of propositional logic these reductions do generally not preserve the properties of the chosen fragment and are thus not suited to transfer complete complexity classifications between abduction and argumentation. Note for instance that Horn-formulæ are not preserved by the second reduction. Nevertheless we will use the idea of the first reduction to transfer certain hardness results from abduction to argumentation.

The complexity of both problems, Abd and Arg rests on two sources: finding a candidate explanation / support and verifying that it is indeed valid. This gives a straight forward procedure to prove the Σ_2^{p}-membership: guess an explanation / support and subsequently verify with the help of an NP-oracle (or coNP-oracle) that it is valid. We will see that in fragments

of propositional logic also NP-, coNP- and P-membership will occur. The NP-membership typically occurs when the verification step can be performed in P, whereas coNP-membership occurs when there is a natural candidate (that is the guessing step can be skipped). The P-membership occurs if both is the case: there is a natural candidate that can be verified in P. Since the verification step is based on satisfiability and implication, we have NP-membership whenever these tasks are tractable. A natural support for argumentation may be the whole knowledge base Δ when it is consistent. In the case of positive abduction a natural explanation may be the whole set of hypotheses A when it is guaranteed to be consistent.

The structural differences between ABD and ARG are as follows.

(a) For ABD the knowledge base Γ can be assumed to be consistent (it describes a consistent part of the worlds behavior), this does usually not change the complexity; for ARG the knowledge base Δ is usually explicitly inconsistent (otherwise no conflicting arguments can be constructed, which is central in argumentation), assuming a consistent Δ usually renders ARG easier since the test for consistency can be dropped.

(b) For ABD the knowledge base Γ is always entirely used (together with a selected subset of hypotheses) in the tests for consistency and entailment, while for ARG these tests are performed on a chosen subset Φ of the knowledge base Δ. As we will see later on, this has particular technical consequences when considering these problems in fragments of propositional logic.

Another difference between abduction and argumentation regards the minimality condition in the definition of an argument, whereas an explanation for abduction is not needed to be minimal in this thesis. Where in the argument-existence problem ARG this condition does not influence the complexity (a support does exists if and only if a minimal support exists), it matters for the verification problem ARG-CHECK. As we will see, there are fragments of propositional logic for which the argument-verification problem ARG-CHECK is potentially harder then the argument-existence problem ARG, unless the polynomial hierarchy collapses. This phenomenon does not occur in the case of abduction where we do not have to verify any minimality condition.

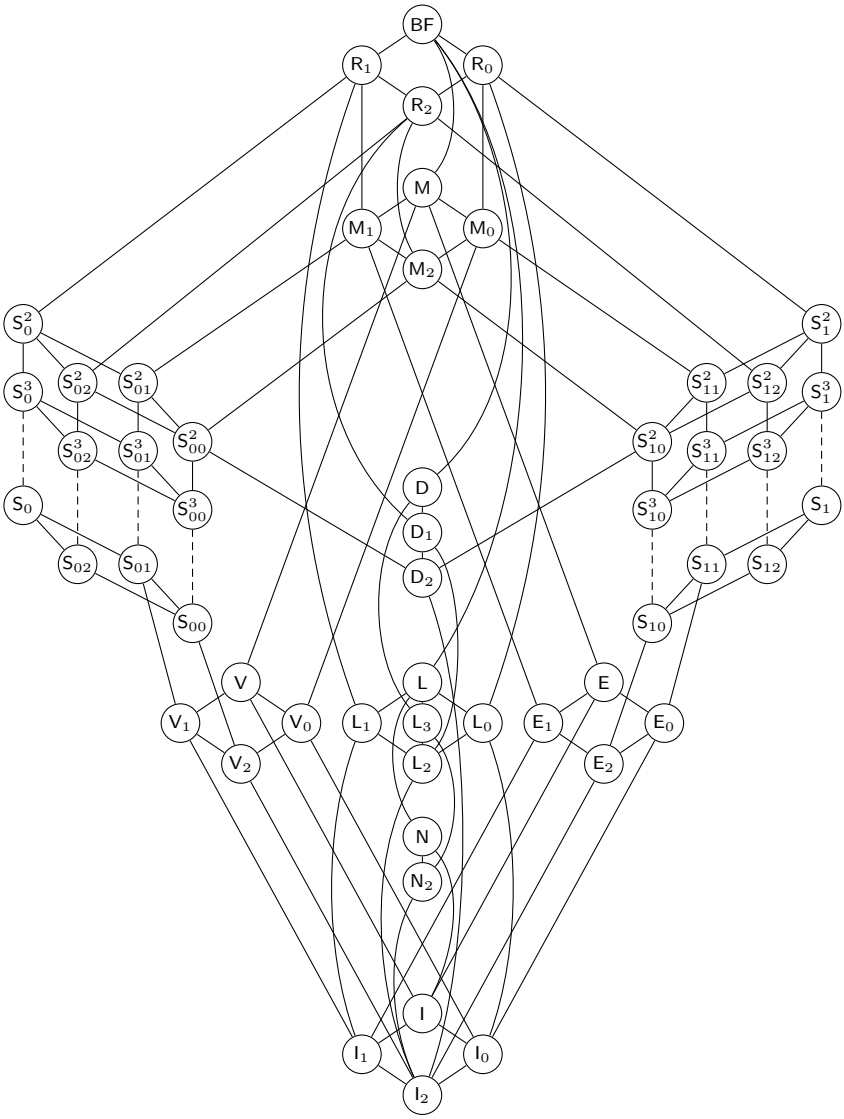

Figure 2.2: Post's lattice.

Chapter 3

Complexity Classifications in Post's Framework

3.1 Post's Framework

In this chapter the approach to obtain fragments of propositional logic is to restrict the allowed connectives in propositional formulæ. That is, instead of considering propositional formulæ over the standard Boolean connectives $\{\wedge, \vee, \neg\}$, we take the connectives from a given set B of arbitrary Boolean functions, leading to the notion of so-called *B-formulæ*. This approach has first been taken by Lewis [Lew79] who showed that $\textsc{Sat}(B)$, the satisfiability problem for B-formulæ, is NP-complete if and only if the negation of implication $(x \wedge \neg y)$ can be expressed by the given connectives B. Since then, Lewis' approach has been applied to a wide range of problems including equivalence and implication problems [Rei03, BMTV09a], satisfiability and model checking in modal and temporal logics [BHSS06, BSS+08], default logic [BMTV09b], and circumscription [Tho09].

At first sight, classifying the complexity of problems parameterized by B-formulæ for all possible sets B requires the study of infinitely many cases. It appears that the complexity usually does not depend directly on B, but on the expressiveness of B. More precisely, the complexity depends on the *clone* $[B]$ generated by B, that is, the set of Boolean functions that can be obtained from the functions in B by projection and

arbitrary composition. This reduces considerably the number of cases to study. We will make this more precise in the following.

3.1.1 Boolean clones and Post's Lattice

A *Boolean function* is an n-ary function $f : \{0,1\}^n \to \{0,1\}$. A *Boolean clone* is a set of Boolean functions that is closed under superposition, i.e., it contains all projections (that is, the functions $f(a_1,\ldots,a_n) = a_k$ for all $1 \leq k \leq n$ and $n \in \mathbb{N}$) and is closed under arbitrary composition. We will henceforth omit the term *Boolean* since we only deal with Boolean clones. Let B be a finite set of Boolean functions. We denote by $[B]$ the smallest clone containing B and call B a *base* for $[B]$. In 1941 Emil Post identified the set of all clones of Boolean functions [Pos41]. He gave a finite base for each of the clones and showed that the set of all clones ordered by inclusion together with the operations $[B \cup B']$ and $[B] \cap [B']$ forms a lattice, hence the name *Post's lattice* (see Figure 2.2 on page 32). In order to define the clones we introduce the following properties, where we give for each of them at least one example. We denote by f an n-ary Boolean function. The *dual* of a function f is the Boolean function $\mathrm{dual}(f)(x_1,\ldots,x_n) := \neg f(\neg x_1,\ldots,\neg x_n)$. The function t_k^n denotes the n-ary k-threshold function that evaluates to 1 if and only if at least k of its n inputs are set to 1:

$$t_k^n(x_1,\ldots,x_n) = 1 \iff \sum_{i=1}^{n} x_i \geq k$$

The function t_n^{n+1} in particular can be written as

$$t_n^{n+1} = \bigvee_{i=1}^{n+1} (x_1 \wedge \cdots \wedge x_{i-1} \wedge x_{i+1} \wedge \cdots \wedge x_{n+1}).$$

- f is *c-reproducing* if $f(c,\ldots,c) = c$, $c \in \{0,1\}$. The binary *and* ($\wedge$) and the binary *or* ($\vee$) are 0- and 1-reproducing, the binary exclusive or ($\oplus$) is 0-reproducing, but not 1-reproducing, whereas the unary negation ($\neg$) is neither 1- nor 0-reproducing.

- f is *monotonic* if $a_1 \leq b_1,\ldots,a_n \leq b_n$ implies $f(a_1,\ldots,a_n) \leq f(b_1,\ldots,b_n)$. Boolean functions build upon composition of only $\wedge, \vee, 0, 1$ are monotonic, like for instance $g(x,y,z) \equiv x \wedge (1 \wedge (y \vee z))$.

- f is *c-separating of degree* k if for all $A \subseteq f^{-1}(c)$ of size $|A| = k$ there exists an $i \in \{1, \ldots, n\}$ such that $(a_1, \ldots, a_n) \in A$ implies $a_i = c$, $c \in \{0, 1\}$. The $(n+1)$-ary threshold function t_n^{n+1} being true if and only if 'at least n variables are true' is 1-separating of degree n. For instance $t_2^3(x, y, z) \equiv (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ is 1-separating of degree 2.

- f is *c-separating* if f is c-separating of degree $|f^{-1}(c)|$. The implication $g(x, y) \equiv \neg x \vee y \equiv x \to y$ is 0-separating.

- f is *self-dual* if $f \equiv \text{dual}(f)$. The majority function $g(x, y, z) \equiv (x \wedge \neg y) \vee (x \wedge \neg z) \vee (\neg y \wedge \neg z)$ is self-dual.

- f is *affine* if $f \equiv x_1 \oplus \cdots \oplus x_n \oplus c$ with $c \in \{0, 1\}$. The function $g(x, y, z) \equiv x \oplus y \oplus z \oplus 1$ is affine and self-dual.

A list of all clones with definitions and finite bases is given in Table 3.1 on page 37. In the naming of the clones the semantic of single indexes is as follows. Index 2 indicates that the clone contains no constants at all. Index 0 (resp. 1) indicates that the clone contains only the constant 0 (resp. 1) but not 1 (resp. 0). Clones with no index contain both constants 0 and 1. The only exceptions to this convention are the clones D and D_1 which do not contain any constants at all. The index * stands for all valid indexes. Clones of particular importance, since they mark points in Post's lattice where the complexity of many problems changes, are:

- the clone of all Boolean functions $\mathsf{BF} = [\wedge, \neg] = [\wedge, \vee, \neg, 0, 1]$
- the monotonic clones M_*, e.g., $\mathsf{M}_2 = [\wedge, \vee]$, $\mathsf{M} = [\wedge, \vee, 0, 1]$
- the affine clones L_*, e.g., $\mathsf{L}_2 = [x \oplus y \oplus z]$, $\mathsf{L} = [x \oplus y, 0, 1]$
- the disjunctive clones V_*, e.g., $\mathsf{V}_2 = [\vee]$, $\mathsf{V} = [\vee, 0, 1]$
- the conjunctive clones E_*, e.g., $\mathsf{E}_2 = [\wedge]$, $\mathsf{E} = [\wedge, 0, 1]$
- the c-reproducing clones R_*, R_1 1-repr., R_0 0-repr., R_2 1- and 0-repr.
- the implication clone $\mathsf{S}_0 = [\to]$
- the negated-implication clone $\mathsf{S}_1 = [x \wedge \neg y]$
- the dual clones D_*, D self-dual, $\mathsf{D}_1 = \mathsf{D} \cap \mathsf{R}_2$, $\mathsf{D}_2 = \mathsf{D} \cap \mathsf{M}$
- the clones $\mathsf{S}_{00} = \mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M} = [x \vee (y \wedge z)]$, $\mathsf{S}_{10} = \mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M} = [x \wedge (y \vee z)]$ and $\mathsf{S}_{12} = \mathsf{S}_1 \cap \mathsf{R}_2 = [x \wedge (y \vee \neg z)]$

A propositional formula using only functions from B as connectives is called a *B-formula*. The set of all B-formulæ is denoted by $\mathcal{L}(B)$.

Let f be an n-ary Boolean function. A B-formula φ with $\mathrm{Vars}(\varphi) = \{x_1, \ldots, x_n\}$ is called *B-representation* of f if $f \equiv \varphi$, i.e., there is a permutation π on $\{1, \ldots, n\}$ such that $\varphi(x_1, \ldots, x_n) = 1$ if and only if $f(x_{\pi(1)}, \ldots, x_{\pi(n)}) = 1$. Such a B-representation exists for every $f \in [B]$. Yet, it may happen that a B-representation of some function uses some input variable more than once, see Example 3.1.1.

Example 3.1.1. *[Exponential blow up] Let $h(x, y) \equiv \neg(x \wedge y)$. An $\{h\}$-representation of the binary and, $\mathrm{and}(x, y) \equiv x \wedge y$, is $h(h(x, y), h(x, y))$. Observe that an $\{h\}$-representation of the n-ary and, $\mathrm{and}_n(x_1, \ldots, x_n) \equiv x_1 \wedge \cdots \wedge x_n$, based on the recursive application of the $\{h\}$-representation $h(h(x, y), h(x, y))$ of the binary and to the formula*

$$\Big(\ldots \big(((x_1 \wedge x_2) \wedge x_3) \wedge x_4 \big) \wedge \ldots \Big) \wedge x_n$$

leads to an explosion of the formula size. This is because the parentheses-depth is linear in the number of variables and the variables x, y appear twice in the $\{h\}$-representation $h(h(x, y), h(x, y))$ of the binary and. We can avoid this exponential blow up by placing the parentheses in a way such that we get a formula of logarithmic parentheses-depth, i.e.,

$$\Big(\ldots \big((x_1 \wedge x_2) \wedge (x_3 \wedge x_4) \big) \wedge \ldots \Big) \ \wedge \ \Big(\cdots \wedge \big((x_{n-3} \wedge x_{n-2}) \wedge (x_{n-1} \wedge x_n) \big) \ldots \Big).$$

3.1.2 Post's lattice as a Tool for Complexity Analysis

We denote by $\textsc{Prob}(B)$ a general problem parameterized by B-formulæ, i.e., its instances are made of B-formulæ.

Fix a set B of Boolean functions. Clearly, when showing membership of $\textsc{Prob}(B)$ to some complexity class, the membership also applies to $\textsc{Prob}(B')$ for any $B' \subseteq B$. Analogously, a hardness result for $\textsc{Prob}(B)$ also applies to all $\textsc{Prob}(B')$ such that $B \subseteq B'$. Summed up, in Post's lattice *membership spreads down* and *hardness spreads up*.

As we already mentioned, the complexity of problems parameterized by B-formulæ often is determined by the clone $[B]$. That is, the complexity does not change inside a clone. Formally stated, it often holds that

$$\textsc{Prob}(B_1) \leq_{\mathrm{m}}^{\log} \textsc{Prob}(B_2) \text{ if } B_1 \subseteq [B_2].$$

Name	Definition	Base
BF	All Boolean functions	$\{x \wedge y, \neg x\}$
R_0	$\{f \mid f$ is 0-reproducing$\}$	$\{x \wedge y, x \oplus y\}$
R_1	$\{f \mid f$ is 1-reproducing$\}$	$\{x \vee y, x \oplus y \oplus 1\}$
R_2	$\mathsf{R}_0 \cap \mathsf{R}_1$	$\{\vee, x \wedge (y \oplus z \oplus 1)\}$
M	$\{f \mid f$ is monotonic$\}$	$\{x \vee y, x \wedge y, 0, 1\}$
M_1	$\mathsf{M} \cap \mathsf{R}_1$	$\{x \vee y, x \wedge y, 1\}$
M_0	$\mathsf{M} \cap \mathsf{R}_0$	$\{x \vee y, x \wedge y, 0\}$
M_2	$\mathsf{M} \cap \mathsf{R}_2$	$\{x \vee y, x \wedge y\}$
S_0^n	$\{f \mid f$ is 0-separating of degree $n\}$	$\{x \rightarrow y, \mathrm{dual}(t_n^{n+1})\}$
S_0	$\{f \mid f$ is 0-separating$\}$	$\{x \rightarrow y\}$
S_1^n	$\{f \mid f$ is 1-separating of degree $n\}$	$\{x \wedge \neg y, t_n^{n+1}\}$
S_1	$\{f \mid f$ is 1-separating$\}$	$\{x \wedge \neg y\}$
S_{02}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2$	$\{x \vee (y \wedge \neg z), \mathrm{dual}(t_n^{n+1})\}$
S_{02}	$\mathsf{S}_0 \cap \mathsf{R}_2$	$\{x \vee (y \wedge \neg z)\}$
S_{01}^n	$\mathsf{S}_0^n \cap \mathsf{M}$	$\{\mathrm{dual}(t_n^{n+1}), 1\}$
S_{01}	$\mathsf{S}_0 \cap \mathsf{M}$	$\{x \vee (y \wedge z), 1\}$
S_{00}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \mathrm{dual}(t_n^{n+1})\}$
S_{00}	$\mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z)\}$
S_{12}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2$	$\{x \wedge (y \vee \neg z), t_n^{n+1}\}$
S_{12}	$\mathsf{S}_1 \cap \mathsf{R}_2$	$\{x \wedge (y \vee \neg z)\}$
S_{11}^n	$\mathsf{S}_1^n \cap \mathsf{M}$	$\{t_n^{n+1}, 0\}$
S_{11}	$\mathsf{S}_1 \cap \mathsf{M}$	$\{x \wedge (y \vee z), 0\}$
S_{10}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z), t_n^{n+1}\}$
S_{10}	$\mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z)\}$
D	$\{f \mid f$ is self-dual$\}$	$\{(x \wedge \neg y) \vee (x \wedge \neg z) \vee (\neg y \wedge \neg z)\}$
D_1	$\mathsf{D} \cap \mathsf{R}_2$	$\{(x \wedge y) \vee (x \wedge \neg z) \vee (y \wedge \neg z)\}$
D_2	$\mathsf{D} \cap \mathsf{M}$	$\{(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)\}$
L	$\{f \mid f$ is affine$\}$	$\{x \oplus y, 1\}$
L_0	$\mathsf{L} \cap \mathsf{R}_0$	$\{x \oplus y\}$
L_1	$\mathsf{L} \cap \mathsf{R}_1$	$\{x \oplus y \oplus 1\}$
L_2	$\mathsf{L} \cap \mathsf{R}_2$	$\{x \oplus y \oplus z\}$
L_3	$\mathsf{L} \cap \mathsf{D}$	$\{x \oplus y \oplus z \oplus 1\}$
V	$\{f \mid f$ is a disjunction or constants$\}$	$\{x \vee y, 0, 1\}$
V_0	$\mathsf{V} \cap \mathsf{R}_0$	$\{x \vee y, 0\}$
V_1	$\mathsf{V} \cap \mathsf{R}_1$	$\{x \vee y, 1\}$
V_2	$\mathsf{V} \cap \mathsf{R}_2$	$\{x \vee y\}$
E	$\{f \mid f$ is a conjunction or constants$\}$	$\{x \wedge y, 0, 1\}$
E_0	$\mathsf{E} \cap \mathsf{R}_0$	$\{x \wedge y, 0\}$
E_1	$\mathsf{E} \cap \mathsf{R}_1$	$\{x \wedge y, 1\}$
E_2	$\mathsf{E} \cap \mathsf{R}_2$	$\{x \wedge y\}$
N	$\{f \mid f$ depends on at most one variable$\}$	$\{\neg x, 0, 1\}$
N_2	$\mathsf{N} \cap \mathsf{R}_2$	$\{\neg x\}$
I	$\{f \mid f$ is a projection or a constant$\}$	$\{\mathrm{id}, 0, 1\}$
I_0	$\mathsf{I} \cap \mathsf{R}_0$	$\{\mathrm{id}, 0\}$
I_1	$\mathsf{I} \cap \mathsf{R}_1$	$\{\mathrm{id}, 1\}$
I_2	$\mathsf{I} \cap \mathsf{R}_2$	$\{\mathrm{id}\}$

Table 3.1: List of all Boolean clones with definitions and bases.

However, such a useful lemma can generally not easily be proved in advance. The canonical idea of proving such a lemma is to transform B_1-formula into B_2-formula by replacing every B_1-connective by its B_2-*representation*. As illustrates Example 3.1.1, this reduction is not necessarily polynomial: Since the B_2-representation of some function may use some input variable more than once, the formula size may grow exponentially. (The existence or not of a polynomial-size B_2-representation for any B_1-formula is a topic of independent interest, which has been addressed several times in the literature, see e.g., [KW88], [Spi71]). Nevertheless we will use the idea of this reduction very frequently, avoiding an exponential blow-up by special structures of the B_1-formulæ. When we show hardness-results for $\textsc{Prob}(B)$ we generally show hardness first only for $\textsc{Prob}([B])$ and show then in a second step that the proof can indeed be extended to show hardness also for $\textsc{Prob}(B)$, in the spirit of the above mentioned canonical reduction.

It appears that many problems parameterized by B-formulæ allow the use of some special Boolean function f 'for free': often one can prove with little effort a lemma of the form

$$\textsc{Prob}(B) \equiv_{\mathrm{m}}^{\log} \textsc{Prob}(B \cup \{f\}), \text{ for some } f \in \mathsf{BF},$$

which reduces the number of cases to consider. We will therefore frequently add some function $f \notin C$ to a clone C and consider the clone $C' = [C \cup \{f\}]$ generated out of C and f. With Post's lattice one can determine this C' quite easily: It is the lowest clone above C that contains f. In this thesis it will be the Boolean constant 1 and sometimes also the constant 0 that will be available 'for free'. The following list contains identities we will frequently use.

- $[\mathsf{S}_{00} \cup \{0,1\}] = [\mathsf{D}_2 \cup \{0,1\}] = [\mathsf{S}_{10} \cup \{0,1\}] = \mathsf{M}$
- $[\mathsf{S}_{02} \cup \{0,1\}] = [\mathsf{D}_1 \cup \{0,1\}] = [\mathsf{S}_{12} \cup \{0,1\}] = \mathsf{BF}$
- $[\mathsf{S}_1 \cup \{1\}] = [\mathsf{D} \cup \{1\}] = \mathsf{BF}$
- $[\mathsf{S}_{00} \cup \{1\}] = \mathsf{S}_{01}$
- $[\mathsf{D}_2 \cup \{1\}] = \mathsf{S}_{01}^2$
- $[\mathsf{S}_{10} \cup \{1\}] = \mathsf{M}_1$
- $[\mathsf{S}_{12} \cup \{1\}] = [\mathsf{D}_1 \cup \{1\}] = \mathsf{R}_1$

- $[\mathsf{S}_{02} \cup \{1\}] = \mathsf{S}_0$
- $[\mathsf{L}_0 \cup \{1\}] = \mathsf{L}$
- $[\mathsf{L}_3 \cup \{1\}] = \mathsf{L}$

3.1.3 Parameterizing by B-formulæ

Abduction In the above defined abduction problems we now introduce two parameters in order to consider the problems in fragments of propositional logic. The first one is a set of Boolean functions B indicating that the formulæ in the knowledge base Γ are restricted to B-formulæ.

We will also consider several restrictions of the manifestations. To indicate them, we introduce a second parameter $\mathcal{M}$ meaning that the manifestation φ is required to be

- Q (resp. PQ, NQ): a single literal (resp. positive literal, negative literal),
- C (resp. PC, NC): a clause (resp. positive clause, negative clause),
- T (resp. PT, NT): a term (resp. positive term, negative term),
- $\mathcal{L}(B)$: a B-formula.

In the above used notation the definition of $\textsc{Abd}(B, \mathrm{PQ})$ for instance is as follows.

Problem:	$\textsc{Abd}(B, \mathrm{PQ})$
Instance:	$\mathcal{I} = (\Gamma, A, q)$, where $\Gamma \subseteq \mathcal{L}(B)$, $A \subseteq \mathrm{Vars}(\Gamma)$ and q a variable
Question:	Does there exist an explanation $E \subseteq \mathrm{Lits}(A)$?

For a specified set $\mathcal{M}$ of manifestation variants we will use $\textsc{Abd}(B, \mathcal{M})$ to talk about all manifestation variants in $\mathcal{M}$ simultaneously.

Argumentation For the argumentation problems we introduce only one parameter. Even if the claim α is not necessarily present in the knowledge base Δ, it seems us the most natural variant to consider the support and the claim to be of the same type of formulæ. Contrary to abduction, the process of argumentation does not stop after having formed one argument. Hence the claim of an argument may be used to form another support for another claim.

The parameter is a set of Boolean functions B indicating that the formulæ in the knowledge base Δ and the claim α are B-formulæ. The definition of $\textsc{Arg}(B)$ for instance is as follows.

Problem:	$\textsc{Arg}(B)$
Instance:	$\mathcal{I} = (\Delta, \alpha)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha \in \mathcal{L}(B)$.
Question:	Does there exist Φ such that (Φ, α) is an argument in Δ?

3.2 The complexity of Symmetric Abduction

3.2.1 Technical results and tools

As we mentioned in Section 3.1.2, often some special Boolean functions can be used 'for free'. In the case of symmetric abduction these are the constants 0 and 1. As shows the following important lemma, we may always assume that our B contains the constant 1. In some cases also the constant 0 is available. We will explain afterwards on an example how this reduces the number cases we have to study.

Lemma 3.2.1. *Let B be a finite set of Boolean functions.*

1. *If $\mathcal{M} \in \{\mathrm{Q}, \mathrm{C}, \mathrm{T}, \mathcal{L}(B)\}$, then*

$$\textsc{Abd}(B, \mathcal{M}) \equiv_{\mathrm{m}}^{\log} \textsc{Abd}(B \cup \{1\}, \mathcal{M}),$$
$$\textsc{Abd-Check}(B, \mathcal{M}) \equiv_{\mathrm{m}}^{\log} \textsc{Abd-Check}(B \cup \{1\}, \mathcal{M}).$$

2. *If $\mathcal{M} \in \{\mathrm{Q}, \mathrm{C}, \mathrm{T}\}$ and $\vee \in [B]$, then*

$$\textsc{Abd}(B, \mathcal{M}) \equiv_{\mathrm{m}}^{\log} \textsc{Abd}(B \cup \{0\}, \mathcal{M}),$$
$$\textsc{Abd-Check}(B, \mathcal{M}) \equiv_{\mathrm{m}}^{\log} \textsc{Abd-Check}(B \cup \{0\}, \mathcal{M}).$$

Proof. One direction of the equivalences is trivial and hence omitted. To reduce $\textsc{Abd}(B \cup \{1\}, \mathcal{M})$ to $\textsc{Abd}(B, \mathcal{M})$ we transform any instance of the first problem in replacing every occurrence of 1 by a fresh variable t and adding the unit clause (t) to the knowledge base.

To reduce $\textsc{Abd}(B \cup \{0\}, \mathcal{M})$ to $\textsc{Abd}(B, \mathcal{M})$, let $\mathcal{P} = (\Gamma, A, \psi)$ be an instance of the first problem and f be a fresh variable. Since $\mathcal{M} \in \{\mathrm{Q}, \mathrm{C}, \mathrm{T}\}$, we can suppose w.l.o.g. that ψ does not contain 0. We map $\mathcal{P}$ to $\mathcal{P}' = (\Gamma', A', \psi)$, where Γ' is the B-representation of $\{\varphi[0/f] \vee f \mid \varphi \in \Gamma\}$ and $A' = A \cup \{f\}$. Note that Γ' is equivalent to $\Gamma[0/f] \vee f$. Let now E be a solution for $\mathcal{P}$, i.e., $\Gamma \wedge E$ is satisfiable and $\Gamma \wedge E \wedge \neg\psi$ is unsatisfiable. One easily verifies that $E' = E \cup \{\neg f\}$ is a solution for $\mathcal{P}'$. Conversely, let E' be a solution for $\mathcal{P}'$. Define $E = E' \backslash \{\neg f\}$. The unsatisfiability of $\Gamma' \wedge E' \wedge \neg\psi$ allows us to conclude that $\neg f \in E'$. With this and the satisfiability of $\Gamma' \wedge E'$ we obtain that $\Gamma \wedge E$ is satisfiable and $\Gamma \wedge E \wedge \neg\psi$ is unsatisfiable.

The same reductions work for $\textsc{Abd-Check}$. □

This lemma holds also for purely positive/negative queries, clauses or terms, i.e., $\mathrm{Q}, \mathrm{C}, \mathrm{T}$ can be replaced by $\mathrm{PQ}, \mathrm{PC}, \mathrm{PT}$ or $\mathrm{NQ}, \mathrm{NC}, \mathrm{NT}$, respectively. This is because the manifestation is essentially left unchanged by the reductions.

Lemma 3.2.1 together with the identities of clones mentioned in Section 3.1.2 reduces the number of cases we have to study. For instance later on we will show a hardness result for $\textsc{Abd}(B, \mathrm{PQ})$ in the three following cases:

1. $\mathsf{S}_{00} \subseteq [B]$
2. $\mathsf{D}_2 \subseteq [B]$
3. $\mathsf{S}_{10} \subseteq [B]$

In case 1 it holds that $\vee \in [B]$ and we may thus apply Lemma 3.2.1 for both constants 0 and 1. It suffices then to show hardness of $\textsc{Abd}(B \cup \{0,1\}, \mathrm{PQ})$. Since further $\mathsf{M} = [\mathsf{S}_{00} \cup \{0,1\}] \subseteq [[B] \cup \{0,1\}] = [B \cup \{0,1\}]$, showing hardness of $\textsc{Abd}(B, \mathrm{PQ})$, we may thus assume that $\mathsf{M} \subseteq [B]$.

In case 2, applying Lemma 3.2.1 for the constant 1, it suffices to show hardness of $\textsc{Abd}(B \cup \{1\}, \mathrm{PQ})$. It holds further that $\mathsf{S}^2_{01} = [\mathsf{D}_2 \cup \{1\}] \subseteq [[B] \cup \{1\}] = [B \cup \{1\}]$. Thus it we may assume that $\mathsf{S}^2_{01} \subseteq [B]$. But for those B we have $\vee \subseteq [B]$ and hence we may now apply Lemma 3.2.1 for the constant 0: It suffices to show hardness of $\textsc{Abd}(B \cup \{0\}, \mathrm{PQ})$ (for B such that $\mathsf{S}^2_{01} \subseteq [B]$). Now we have $\mathsf{M} = [\mathsf{S}^2_{01} \cup \{0\}] \subseteq [[B] \cup \{0\}] = [B \cup \{0\}]$. Thus we may finally assume that $\mathsf{M} \subseteq [B]$.

Analogously to case 2, also case 3 is reduced to the case $\mathsf{M} \subseteq [B]$ such that finally, in order to cover all three cases, it suffices to consider the case $\mathsf{M} \subseteq [B]$. We will often play this game of going up in Post's lattice by adding constants and applying Lemma 3.2.1. However, we will henceforth give less details.

In [CZ06] and [NZ08] the authors study abduction in Schaefer's framework. We will use some results from [NZ08] on affine formulæ, namely Propositions 66, 67, 69 and 70. Affine formulæ can be seen as a conjunction of $\oplus$-clauses, i.e., as a system of linear equations over the finite field GF(2) (the exclusive or $\oplus$ can be seen as addition modulo 2). The fragment of affine formulæ is the only one which appears in both Post's and Schaefer's Framework. Therefore the results on affine formulæ are the only ones we can use from [NZ08]. For instance Proposition 66 of [NZ08] translates to P-membership for $\textsc{Abd}(\mathsf{L}, \mathrm{C})$, [NZ08, Proposition 67] translates to P-membership for $\textsc{Abd}(\mathsf{L}, \mathrm{T})$, [NZ08, Proposition 70] translates to NP-hardness for P-$\textsc{Abd}(\mathsf{L}_3, \mathrm{PT})$.

3.2.2 The complexity of the Existence Problem

We will consider first the variant where the manifestation is a single query $(\mathrm{Q}, \mathrm{PQ}, \mathrm{NQ})$, then we turn to clauses $(\mathrm{C}, \mathrm{PC}, \mathrm{NC})$, terms $(\mathrm{T}, \mathrm{PT}, \mathrm{NT})$ and B-formulæ ($\mathcal{L}(B)$). We will always first classify the positive variant and then derive with little effort the classifications for the other variants.

Thus, the first problem under consideration is $\textsc{Abd}(B, \mathrm{PQ})$ where we want to explain a single positive literal.

Problem:	$\textsc{Abd}(B, \mathrm{PQ})$
Instance:	$\mathcal{I} = (\Gamma, A, q)$, where $\Gamma \subseteq \mathcal{L}(B)$, $A \subseteq \mathrm{Vars}(\Gamma)$ and q a variable
Question:	Does there exist an explanation $E \subseteq \mathrm{Lits}(A)$?

The following theorem gives a complete classification of the complexity of $\textsc{Abd}(B, \mathrm{PQ})$ in function of the authorized connectives B. The classification is visualized in Figure 3.1 on page 74.

Theorem 3.2.2. *Let B be a finite set of Boolean functions. Then the symmetric explanation-existence problem for propositional B-formulæ with a positive literal manifestation, $\textsc{Abd}(B, \mathrm{PQ})$, is*

1. Σ_2^{p}-*complete if* $\mathsf{S}_{02} \subseteq [B]$ *or* $\mathsf{S}_{12} \subseteq [B]$ *or* $\mathsf{D}_1 \subseteq [B]$,
2. NP-*complete if* $\mathsf{S}_{00} \subseteq [B] \subseteq \mathsf{M}$ *or* $\mathsf{S}_{10} \subseteq [B] \subseteq \mathsf{M}$ *or* $\mathsf{D}_2 \subseteq [B] \subseteq \mathsf{M}$,
3. *in* P *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$, *and*
4. *in* Logspace *in all other cases.*

The same classification holds for $\textsc{Abd}(B, \mathrm{Q})$, $\textsc{Abd}(B, \mathrm{PC})$ *and* $\textsc{Abd}(B, \mathrm{C})$.

We split the proof of Theorem 3.2.2 into several propositions.

Proposition 3.2.3. *Let B be a finite set of Boolean functions such that $[B] \subseteq \mathsf{E}$ or $[B] \subseteq \mathsf{N}$ or $[B] \subseteq \mathsf{V}$. Then* $\textsc{Abd}(B, \mathrm{PQ}) \in$ Logspace.

Proof. Let $\mathcal{P} = (\Gamma, A, q)$ be an instance of $\textsc{Abd}(B, \mathrm{PQ})$.

For $[B] = \mathsf{N}$ or E, Γ is equivalent to a set of literals, hence $\mathcal{P}$ has the empty set or $\{q\}$ as a solution if $\mathcal{P}$ possesses a solution at all. Finally notice that satisfiability of a set of N-formulæ can be tested in logarithmic space (basically check for contradicting literals, reducing nested negation on the fly).

For $[B] = \mathsf{V}$ each formula $\varphi \in \Gamma$ is equivalent to either a constant or disjunction of positive literals. The knowledge base Γ is equivalent to a positive CNF-formula, thus it is unsatisfiable if and only if it contains the empty clause or the constant 0 as a formula. It holds that (Γ, A, q) has a solution if and only if Γ contains a formula $\varphi \equiv q \vee x_1 \vee \cdots \vee x_k$ such that $\{x_1, \ldots, x_k\} \subseteq A$, and $\Gamma[x_1/0, \ldots, x_k/0]$ is satisfiable. This can be tested in logarithmic space, as satisfiability of V-formulæ and substitution of symbols (on the fly) can be performed in logarithmic space (according to [Sch05] satisfiability of monotonic formulæ is in Logspace). □

Proposition 3.2.4. *Let B be a finite set of Boolean functions such that $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$. Then* $\textsc{Abd}(B, \mathrm{PQ})$ *is in* P.

Proof. In this case, deciding whether an instance of $\textsc{Abd}(B, \mathrm{PQ})$ has a solution logspace reduces to the problem of deciding whether a propositional abduction problem in which the knowledge base is a set of linear equations over GF(2) has a solution. This has been shown to be decidable in polynomial time in [Zan03]. □

Proposition 3.2.5. *Let B be a finite set of Boolean functions such that $\mathsf{S}_{00} \subseteq [B] \subseteq \mathsf{M}$ or $\mathsf{S}_{10} \subseteq [B] \subseteq \mathsf{M}$ or $\mathsf{D}_2 \subseteq [B] \subseteq \mathsf{M}$. Then* $\textsc{Abd}(B, \mathrm{PQ})$ *is* NP-*complete.*

Proof. We first show that $\textsc{Abd}(B, \mathrm{PQ})$ is efficiently verifiable. Let $\mathcal{P} = (\Gamma, A, q)$ be an $\textsc{Abd}(B, \mathrm{PQ})$-instance and $E \subseteq \mathrm{Lits}(A)$ be a candidate for an explanation. Define Γ' as the set of formulæ obtained from Γ by replacing each occurrence of the proposition x with 0 if $\neg x \in E$, and each occurrence of the proposition x with 1 if $x \in E$. It holds that E is a solution for $\mathcal{P}$ if Γ' is satisfiable and $\Gamma'[q/0]$ is not. These tests can be performed in polynomial time, because Γ' is a set of monotonic formulæ [Lew79]. Hence, $\textsc{Abd}(B, \mathrm{PQ}) \in \mathrm{NP}$.

Next we give a reduction from the NP-complete problem Pos-2in3-Sat, i.e., the problem to decide whether there exists an assignment that satisfies exactly two propositions in each clause of a given formula in conjunctive normal form with exactly three positive propositions per clause, see [Sch78]. Let $\varphi := \bigwedge_{i \in I} c_i$ with $c_i = x_{i1} \vee x_{i2} \vee x_{i3}$, $i \in I$, be the given formula. We map φ to the following instance $\mathcal{P} = (\Gamma, A, q)$. Let q_i, $i \in I$, be fresh, pairwise distinct propositions and let $A := \mathrm{Vars}(\varphi) \cup \{q_i \mid i \in I\}$. The set Γ is defined as

$$\Gamma := \{c_i \mid i \in I\} \tag{3.1}$$

$$\cup \{x_{i1} \vee x_{i2} \vee q_i, x_{i1} \vee x_{i3} \vee q_i, x_{i2} \vee x_{i3} \vee q_i \mid i \in I\} \tag{3.2}$$

$$\cup \{\textstyle\bigvee_{i \in I} \bigwedge_{j=1}^{3} x_{ij} \vee \bigvee_{i \in I} q_i \vee q\}. \tag{3.3}$$

We show that there is an assignment that sets to true exactly two propositions in each clause of φ if and only if $\mathcal{P}$ has a solution. First, suppose that there exists an assignment σ such that for all $i \in I$, there is a permutation π_i of $\{1, 2, 3\}$ such that $\sigma(x_{i\pi_i(1)}) = 0$ and $\sigma(x_{i\pi_i(2)}) = \sigma(x_{i\pi_i(3)}) = 1$. Thus (3.1) and (3.2) are satisfied, and (3.3) is equivalent to $\bigvee_{i \in I} q_i \vee q$. From this, it is readily observed that $\{\neg x \mid \sigma(x) = 0\} \cup \{\neg q_i \mid i \in I\}$ is a solution to $\mathcal{P}$.

Conversely, suppose that $\mathcal{P}$ has an explanation E that is w.l.o.g. full. Then $\Gamma \wedge E$ is satisfiable and $\Gamma \wedge E \models q$. Let $\sigma\colon \mathrm{Vars}(\Gamma) \to \{0, 1\}$ be an assignment that satisfies $\Gamma \wedge E$. Then, for any $x \in A$, $\sigma(x) = 0$ if $\neg x \in E$, and $\sigma(x) = 1$ otherwise. Since $\Gamma \wedge E$ entails q and as the only occurrence of q is in (3.3), we obtain that σ sets to 0 each q_i and at least one proposition in each clause of φ. Consequently, from (3.2) it follows that σ sets to 1 at least two propositions in each clause of φ. Therefore, σ sets to 1 exactly two propositions in each clause of φ.

It remains to show that $\mathcal{P}$ can be transformed into an $\textsc{Abd}(B, \mathrm{PQ})$-instance for all considered B. Observe that $\vee \in [B \cup \{1\}]$ and $[\mathsf{S}_{00} \cup \{0, 1\}] = [\mathsf{D}_2 \cup \{0, 1\}] = [\mathsf{S}_{10} \cup \{0, 1\}] = \mathsf{M}$. Therefore due to Lemma 3.2.1

it suffices to consider the case $[B] = \mathsf{M}$ (see Section 3.2.1 for details). Using the associativity of $\vee$ rewrite (3.3) as an $\vee$-tree of logarithmic depth and replace all the connectives in Γ by their B-representation ($\vee, \wedge \in [B]$). □

Proposition 3.2.6. *Let B be a finite set of Boolean functions such that $\mathsf{S}_{02} \subseteq [B]$ or $\mathsf{S}_{12} \subseteq [B]$ or $\mathsf{D}_1 \subseteq [B]$. Then $\textsc{Abd}(B, \mathrm{PQ})$ is Σ_2^{p}-complete.*

Proof. Membership in Σ_2^{p} is easily seen to hold: given an instance (Γ, A, q), guess an explanation E and subsequently verify that $\Gamma \wedge E$ is satisfiable and $\Gamma \wedge E \wedge \neg q$ is not.

Observe that $\vee \in [B \cup \{1\}]$. By virtue of Lemma 3.2.1 and the fact that $[\mathsf{S}_{02} \cup \{0,1\}] = [\mathsf{S}_{12} \cup \{0,1\}] = [\mathsf{D}_1 \cup \{0,1\}] = \mathsf{BF}$, it suffices to consider the case $[B] = \mathsf{BF}$. From [EG95] it can be easily derived that the propositional abduction problem remains Σ_2^{p}-complete when the knowledge base Γ is a set of clauses. From such an instance (Γ, A, q) we build an instance of $\textsc{Abd}(B, \mathrm{PQ})$ by rewriting first each clause as an $\vee$-tree of logarithmic depth and then replacing the occurring connectives $\vee$ and $\neg$ by their B-representation, thus concluding the proof. □

Note that the problem $\textsc{Abd}(B, \mathrm{Q})$ obeys the same classification as $\textsc{Abd}(B, \mathrm{PQ})$ since all bounds, upper and lower, easily carry over.

For $\textsc{Abd}(B, \mathrm{NQ})$ the problem becomes trivial if $[B] \subseteq \mathsf{M}$ (more precisely, it cannot have a solution). For $[B] \subseteq \mathsf{L}$ $\textsc{Abd}(B, \mathrm{NQ})$ is solvable in polynomial time according to [Zan03]. For the remaining clones (i.e., for $\mathsf{S}_{02} \subseteq [B]$, $\mathsf{S}_{12} \subseteq [B]$, and $\mathsf{D}_1 \subseteq [B]$), we can again easily adapt the proofs of $\textsc{Abd}(B, \mathrm{PQ})$. This way we obtain a dichotomous classification for $\textsc{Abd}(B, \mathrm{NQ})$ into P-membership and Σ_2^{p}-complete cases; thus skipping the intermediate NP level.

Theorem 3.2.7. *Let B be a finite set of Boolean functions. Then the symmetric explanation-existence problem for propositional B-formulæ with a negative literal manifestation, $\textsc{Abd}(B, \mathrm{NQ})$, is*

1. *Σ_2^{p}-complete if $\mathsf{S}_{02} \subseteq [B]$ or $\mathsf{S}_{12} \subseteq [B]$ or $\mathsf{D}_1 \subseteq [B]$,*
2. *in P if $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$, and*
3. *trivial in all other cases.*

The same classification holds for $\textsc{Abd}(B, \mathrm{NC})$ and $\textsc{Abd}(B, \mathrm{NT})$.

We now consider the symmetric abduction problem for different variants on the manifestations: clause, term and B-formula.

ABD(B, PC) For clauses, it is obvious that we have $\textsc{Abd}(B, \mathrm{PQ}) \leq_{\mathrm{m}}^{\log} \textsc{Abd}(B, \mathrm{PC})$. Therefore, all hardness results continue to hold for $\textsc{Abd}(B, \mathrm{PC})$. It is an easy exercise to prove that all algorithms that have been developed for a single literal can be naturally extended to clauses. Therefore, the complexity classifications for the problems $\textsc{Abd}(B, \mathrm{PC})$, $\textsc{Abd}(B, \mathrm{C})$ and $\textsc{Abd}(B, \mathrm{NC})$ are exactly the same as for $\textsc{Abd}(B, \mathrm{PQ})$, $\textsc{Abd}(B, \mathrm{Q})$ and $\textsc{Abd}(B, \mathrm{NQ})$, respectively.

ABD(B, PT) Allowing for terms as manifestations increases the complexity for the clones V_* (from membership in LOGSPACE to NP-completeness). The intuitive reason for this is that knowledge base Γ as a conjunction of positive clauses, together with the positive term in the manifestation, allows to express negative literals in the knowledge base, thus simulating a 3CNF-formula.

Proposition 3.2.8. *Let B be a finite set of Boolean functions such that $\mathsf{V}_2 \subseteq [B] \subseteq \mathsf{V}$. Then $\textsc{Abd}(B, \mathrm{PT})$ is* NP*-complete.*

Proof. Let B be a finite set of Boolean functions such that $\mathsf{V}_2 \subseteq [B] \subseteq \mathsf{V}$ and let $\mathcal{P} = (\Gamma, A, t)$ be an instance of $\textsc{Abd}(B, \mathrm{PT})$. Hence, Γ is a set of B-formulæ and t is a term, $t = \bigwedge_{i=1}^{n} l_i$. Observe that E is a solution for $\mathcal{P}$ if $\Gamma \wedge E$ is satisfiable and for every $i = 1, \ldots, n$, $\Gamma \wedge E \wedge \neg l_i$ is not. Given a set $E \subseteq \mathrm{Lits}(A)$, these verifications, which require substitution of symbols and evaluation of an $\vee$-formula, can be performed in polynomial time, thus proving membership in NP.

To prove NP-hardness, we give a reduction from 3SAT. Let φ be a 3CNF-formula, $\varphi := \bigwedge_{i \in I} c_i$. Let $x_1, \ldots, x_n$ enumerate the variables occurring in φ. Let $x'_1, \ldots, x'_n$ and $q_1, \ldots, q_n$ be fresh, pairwise distinct variables. We map φ to $\mathcal{P} = (\Gamma, A, t)$, where

$$\begin{aligned} \Gamma :=& \{c_i[\neg x_1/x'_1, \ldots, \neg x_n/x'_n] \mid i \in I\} \\ & \cup \{x_i \vee x'_i, x_i \vee q_i, x'_i \vee q_i \mid 1 \leq i \leq n\}, \\ A :=& \{x_1, \ldots, x_n, x'_1, \ldots, x'_n\}, \\ t :=& q_1 \wedge \cdots \wedge q_n. \end{aligned}$$

We show that φ is satisfiable if and only if $\mathcal{P}$ has a solution. First assume that φ is satisfied by the assignment $\sigma\colon \{x_1, \ldots, x_n\} \to \{0, 1\}$. Define $E := \{\neg x_i \mid \sigma(x_i) = 0\} \cup \{\neg x'_i \mid \sigma(x_i) = 1\}$ and $\hat{\sigma}$ as the extension of σ mapping $\hat{\sigma}(x'_i) = \neg\sigma(x_i)$ and $\hat{\sigma}(q_i) = 1$ for all $1 \leq i \leq n$. Obviously, $\hat{\sigma} \models$

$\Gamma \wedge E$. Furthermore, $\Gamma \wedge E \models q_i$ for all $1 \leq i \leq n$, because any satisfying assignment of $\Gamma \wedge E$ sets to 0 either x_i or x'_i and thus $\{x_i \vee q_i, x'_i \vee q_i\} \models q_i$. Hence E is an explanation for $\mathcal{P}$.

Conversely, suppose that $\mathcal{P}$ has a full explanation E. The facts that $\Gamma \wedge E \models q_1 \wedge \cdots \wedge q_n$ and that each q_i occurs only in the clauses $x_i \vee q_i, x'_i \vee q_i$ enforce that, for every i, E contains $\neg x_i$ or $\neg x'_i$. Because of the clause $x_i \vee x'_i$, it cannot contain both. Therefore in E the value of x'_i is determined by the value of x_i and is its dual. From this it is easy to conclude that the assignment $\sigma\colon \{x_1, \ldots, x_n\} \to \{0, 1\}$ defined by $\sigma(x_i) = 0$ if $\neg x_i \in E$, and 1 otherwise, satisfies φ. Finally $\mathcal{P}$ can be transformed into an $\textsc{Abd}(B, \mathrm{PT})$-instance by replacing every occurrence of $\vee$ by its B-representation. An exponential blowup is avoided since every formula in Γ is the disjunction of at most three variables.

□

The other fragments can easily be obtained from the classification for $\textsc{Abd}(B, \mathrm{PQ})$ or are already solved in the literature.

Theorem 3.2.9. *Let B be a finite set of Boolean functions. Then the symmetric explanation-existence problem for propositional B-formulæ with a positive term manifestation,* $\textsc{Abd}(B, \mathrm{PT})$*, is*

1. Σ_2^{p}*-complete if* $\mathsf{S}_{02} \subseteq [B]$ *or* $\mathsf{S}_{12} \subseteq [B]$ *or* $\mathsf{D}_1 \subseteq [B]$*,*
2. NP*-complete if* $\mathsf{V}_2 \subseteq [B] \subseteq \mathsf{M}$ *or* $\mathsf{S}_{10} \subseteq [B] \subseteq \mathsf{M}$ *or* $\mathsf{D}_2 \subseteq [B] \subseteq \mathsf{M}$*,*
3. *in* P *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$*, and*
4. *in* Logspace *in all other cases.*

The same classification holds for $\textsc{Abd}(B, \mathrm{T})$*.*

Proof. 1. The Σ_2^{p}-hardness follows directly from Proposition 3.2.6.

2. For the clones $\mathsf{V}_2 \subseteq [B] \subseteq \mathsf{V}$, see Proposition 3.2.8. In all other clones, the NP-hardness follows from a straightforward generalization of the proof of Proposition 3.2.5.

3. Membership in P follows from [NZ08, Proposition 67].

4. Analogous to Proposition 3.2.3.

□

We observe that once again all upper and lower bounds for $\textsc{Abd}(B, \mathrm{PT})$ easily carry over to $\textsc{Abd}(B, \mathrm{T})$. Further, it is also easily seen that the problem $\textsc{Abd}(B, \mathrm{NT})$ is classified exactly as $\textsc{Abd}(B, \mathrm{NQ})$ (Therorem 3.2.7).

ABD(B, $\mathcal{L}(B)$) Allowing B-formulæ as manifestations makes the classification dichotomous: all problems become either Σ_2^{p}-complete or they are in P.

Problem: $\textsc{Abd}(B, \mathcal{L}(B))$
Instance: $\mathcal{I} = (\Gamma, A, \varphi)$, where $\Gamma \subseteq \mathcal{L}(B)$, $A \subseteq \mathrm{Vars}(\Gamma)$ and $\varphi \in \mathcal{L}(B)$
Question: Does there exist an explanation $E \subseteq \mathrm{Lits}(A)$?

The main result we need is stated by the following proposition.

Proposition 3.2.10. *Let B be a finite set of Boolean functions such that $\mathsf{S}_{00} \subseteq [B]$ or $\mathsf{S}_{10} \subseteq [B]$ or $\mathsf{D}_2 \subseteq [B]$. Then $\textsc{Abd}(B, \mathcal{L}(B))$ is Σ_2^{p}-complete.*

Proof. We prove Σ_2^{p}-hardness by giving a reduction from the Σ_2^{p}-hard problem $\textsc{QSat}_{\exists,2}$ [Wra77]. Let an instance of $\textsc{QSat}_{\exists,2}$ be given by a closed formula $\chi := \exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m \varphi$ with φ being a 3-DNF-formula. First observe that $\exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m \varphi$ is true if and only if there exists a consistent set $X \subseteq \mathrm{Lits}(\{x_1, \ldots, x_n\})$ such that $X \cap \{x_i, \neg x_i\} \neq \emptyset$, for all $1 \leq i \leq n$, and $\neg X \vee \varphi$ is (universally) valid (or equivalently $\neg\varphi \wedge X$ is unsatisfiable).

Denote by $\overline{\varphi}$ the negation normal form of $\neg\varphi$ and let $\overline{\varphi}'$ be obtained from $\overline{\varphi}$ by replacing all occurrences of $\neg x_i$ with a fresh proposition x_i', $1 \leq i \leq n$, and all occurrences of $\neg y_i$ with a fresh proposition y_i', $1 \leq i \leq m$. That is, $\overline{\varphi}' \equiv \overline{\varphi}[\neg x_1/x_1', \ldots, \neg x_n/x_n', \neg y_1/y_1', \ldots, \neg y_m/y_m']$. Thus $\overline{\varphi}' = \bigwedge_{i \in I} c_i'$, where every c_i' is a disjunction of three propositions. To χ we associate the propositional abduction problem $\mathcal{P} = (\Gamma, A, \psi)$ defined as follows:

$$
\begin{aligned}
\Gamma &:= \{c_i' \vee q \mid i \in I\} \\
&\quad \cup \{x_i \vee x_i' \mid 1 \leq i \leq n\} \cup \{y_i \vee y_i' \mid 1 \leq i \leq m\} \\
&\quad \cup \{f_i \vee x_i, t_i \vee x_i', f_i \vee t_i \mid 1 \leq i \leq n\}, \\
A &:= \{t_i, f_i \mid 1 \leq i \leq n\}, \\
\psi &:= q \vee \textstyle\bigvee_{1 \leq i \leq n} (x_i \wedge x_i') \vee \bigvee_{1 \leq i \leq m} (y_i \wedge y_i').
\end{aligned}
$$

Suppose that χ is true. Then there exists an assignment $\sigma\colon \{x_1,\dots,x_n\} \to \{0,1\}$ such that no extension $\sigma'\colon \{x_1,\dots,x_n\} \cup \{y_1,\dots,y_m\} \to \{0,1\}$ of σ satisfies $\neg\varphi$. Define X as the set of literals over $\{x_1,\dots,x_n\}$ set to 1 by σ. Defining $E := \{\neg f_i, t_i \mid x_i \in X\} \cup \{\neg t_i, f_i \mid \neg x_i \in X\}$, we obtain with abuse of notation

$$\begin{aligned}\Gamma \wedge E \wedge \neg\psi \;&\equiv_{SAT}\; \textstyle\bigwedge_{i\in I} c'_i \wedge \bigwedge_{1\le i\le n}(x_i \oplus x'_i) \wedge \bigwedge_{1\le i\le m}(y_i \oplus y'_i) \wedge \\ &\qquad \textstyle\bigwedge_{1\le i\le n,\sigma(x_i)=1} x_i \wedge \bigwedge_{1\le i\le n,\sigma(x_i)=0} x'_i \\ &\equiv_{SAT}\; \neg\varphi \wedge X,\end{aligned}$$

which is unsatisfiable by assumption. As $\Gamma \wedge E$ is satisfied by any assignment setting in addition all x_i, x'_i, $1 \le i \le n$, and all y_j, y'_j, $1 \le i \le m$, to 1, we have proved that E is an explanation for $\mathcal{P}$.

Conversely, suppose that $\mathcal{P}$ has an explanation E. Assume w.l.o.g. that E is full. Due to the clause $(f_i \vee t_i)$ in Γ, we also may assume that $|E \cap \{\neg t_i, \neg f_i\}| \le 1$ for all $1 \le i \le n$.

Setting $X := \{x_i \mid \neg f_i \in E\} \cup \{\neg x_i \mid \neg t_i \in E\}$ we now obtain $\bigwedge_{1\le i\le n} \big((f_i \vee x_i) \wedge (t_i \vee \neg x_i) \wedge (f_i \vee t_i)\big) \wedge E \equiv_{SAT} X$ and $\Gamma \wedge E \wedge \neg\psi \equiv_{SAT} \neg\varphi \wedge X$ as above. Hence, $\neg\varphi \wedge X$ is unsatisfiable, which implies the existence of an assignment $\sigma\colon \{x_1,\dots,x_n\} \to \{0,1\}$ such that no extension $\sigma'\colon \{x_1,\dots,x_n\} \cup \{y_1,\dots,y_m\} \to \{0,1\}$ of σ satisfies $\neg\varphi$. Therefore, we have proved that χ is true if and only if $\mathcal{P}$ has an explanation.

It remains to show that $\mathcal{P}$ can be transformed into an $\textsc{Abd}(B, \mathcal{L}(B))$-instance for any relevant B. Since $[\mathsf{S}_{00} \cup \{1\}] = \mathsf{S}_{01}$, $[\mathsf{S}_{10} \cup \{1\}] = \mathsf{M}_1$, $[\mathsf{D}_2 \cup \{1\}] = \mathsf{S}^2_{01}$ and $\mathsf{S}_{01} \subseteq \mathsf{S}^2_{01} \subseteq \mathsf{M}_1$, it suffices to consider the case $[B] = \mathsf{S}_{01}$ by Lemma 3.2.1. Observe that $x \vee (y \wedge z) \in [B]$. The transformation can hence be done in polynomial time by local replacements, rewriting ψ as $\bigvee_{1\le i\le n} q \vee (x_i \wedge x'_i) \vee \bigvee_{1\le i\le m} q \vee (y_i \wedge y'_i)$ and using the associativity of $\vee$.

□

The following classification is visualized in Figure 3.3 on page 76.

Theorem 3.2.11. *Let B be a finite set of Boolean functions. Then the symmetric explanation-existence problem for propositional B-formulæ with a B-formula manifestation, $\textsc{Abd}(B, \mathcal{L}(B))$, is*

1. *Σ^{p}_2-complete if $\mathsf{S}_{00} \subseteq [B]$ or $\mathsf{S}_{10} \subseteq [B]$ or $\mathsf{D}_2 \subseteq [B]$,*
2. *in P if $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$, and*

3. in LOGSPACE *in all other cases.*

Proof. 1. See Proposition 3.2.10.

2. Membership in P follows directly from [Zan03].

3. Analogous to Proposition 3.2.3.

□

Observe that there are no sets B of Boolean functions for which $\text{ABD}(B, \mathcal{L}(B))$ is NP-complete.

3.2.3 The complexity of the Verification Problem

For the argument-verification problem we consider only the basic variant where the manifestation is a single positive literal. Most results on the other variants can be derived with little effort as in the case of the explanation-existence problem.

Problem:	ABD-CHECK
Instance:	$\mathcal{I} = (\Gamma, A, q, E)$, where $\Gamma \subseteq \mathcal{L}$, $A \subseteq \text{Vars}(\Gamma)$, q a variable and $E \subseteq \text{Lits}(A)$
Question:	Is E an explanation?

We recall that to verify whether E is an explanation we have to verify the following two conditions:

(B1) $\Gamma \wedge E$ is satisfiable,

(B2) $\Gamma \wedge E \wedge \neg q$ is unsatisfiable.

This gives straightforward the DP-membership, the fist condition being checkable in NP and the second one in coNP. We will prove the following classification theorem. The classification is visualized in Figure 3.5 on page 78.

Theorem 3.2.12. *Let B be a finite set of Boolean functions. Then the symmetric explanation-validity problem for propositional B-formulæ with a positive literal manifestation,* $\text{ABD-CHECK}(B, \text{PQ})$*, is*

1. DP*-complete if* $\mathsf{S}_{02} \subseteq [B]$ *or* $\mathsf{S}_{12} \subseteq [B]$ *or* $\mathsf{D}_1 \subseteq [B]$

2. in P *if* $[B] \subseteq \mathsf{L}$

3. *in* LOGSPACE *if* $[B] \subseteq \mathsf{M}$ *or* $[B] \subseteq \mathsf{N}$

We split the proof into two propositions.

Proposition 3.2.13. *Let B be a finite set of Boolean functions such that $\mathsf{S}_{02} \subseteq [B]$ or $\mathsf{S}_{12} \subseteq [B]$ or $\mathsf{D}_1 \subseteq [B]$. Then* ABD-CHECK$(B, \mathrm{PQ})$ *is* DP*-complete.*

Proof. For the DP-membership we define

$$A = \{\ (\Gamma, A, q, E) \mid \Gamma \wedge E \text{ is sat } \} \text{ and}$$
$$B = \{\ (\Gamma, A, q, E) \mid \Gamma \wedge E \wedge \neg q \text{ is unsat } \}.$$

Obviously $A \in$ NP, $B \in$ coNP and ABD-CHECK $= A \cap B$.

To prove DP-hardness, let (φ, ψ) be an instance of the DP-complete problem SATUNSAT. Our mapping is as follows:

$$(\varphi, \psi) \ \mapsto \ (\Gamma,\ A,\ q,\ E), \text{ where}$$
$$q \text{ a fresh variable,}$$
$$\Gamma = \{\varphi\} \cup \{q \leftrightarrow \neg\psi\}$$
$$A = E = \emptyset$$

Obviously it holds

$$\left.\begin{array}{l} \varphi \text{ sat} \\ \psi \text{ unsat} \end{array}\right\} \iff \left\{\begin{array}{l} \Gamma \wedge E \ \equiv\ \varphi \wedge (q \leftrightarrow \neg\psi) \text{ sat} \\ \Gamma \wedge E \wedge \neg q \ \equiv\ \varphi \wedge \psi \wedge \neg q \text{ unsat} \end{array}\right.$$

It remains to show that $(\Gamma,\ A,\ q,\ E)$ can be transformed into an ABD-CHECK(B, PQ)-instance for all considered B. Observe that $\vee \in [B \cup \{1\}]$. By virtue of Lemma 3.2.1 and the fact that $[\mathsf{S}_{02} \cup \{0,1\}] = [\mathsf{S}_{12} \cup \{0,1\}] = [\mathsf{D}_1 \cup \{0,1\}] = \mathsf{BF}$, it suffices therefore to consider the case $[B] = \mathsf{BF}$. Thus, it remains to rewrite Γ as a set of B-formulæ for B such that $[B] = \mathsf{BF}$. In order to do so, we use the associativity of $\wedge$ and rewrite φ and $(q \leftrightarrow \neg\psi)$ as $\wedge$-trees of logarithmic depth and then replace the connectives $\vee, \wedge, \neg, \leftrightarrow$ by their B-representation.

□

Proposition 3.2.14. *The following memberships hold.*

1. ABD-CHECK$(\mathsf{N}, \mathrm{PQ}) \in$ LOGSPACE
2. ABD-CHECK$(\mathsf{M}, \mathrm{PQ}) \in$ LOGSPACE

3. Abd-Check$(\mathsf{L}, \mathrm{PQ}) \in \mathrm{P}$

Proof. We have to check the two conditions (B1) and (B2). Since we deal with N-, monotonic or affine formulæ, this can be done in Logspace (see [Sch05] and Proposition 3.2.3) or P (Gaussian elimination), respectively. □

3.3 The complexity of Positive Abduction

3.3.1 Technical results and tools

We will now study the complexity of positive abduction, in which an explanation consists of a set of positive literals.

To begin with, note that we still have the constant 1 'for free', but no corresponding property for the constant 0, as we have for symmetric abduction. The reason is that there are no negative literals anymore in an explanation which helped previously to express the constant 0.

Lemma 3.3.1. *Let B be a finite set of Boolean functions. If $\mathcal{M} \in \{\mathrm{Q}, \mathrm{C}, \mathrm{T}, \mathcal{L}(B)\}$, then*

$$\text{P-Abd}(B, \mathcal{M}) \equiv_{\mathrm{m}}^{\log} \text{P-Abd}(B \cup \{1\}, \mathcal{M}),$$
$$\text{P-Abd-Check}(B, \mathcal{M}) \equiv_{\mathrm{m}}^{\log} \text{P-Abd-Check}(B \cup \{1\}, \mathcal{M})$$

Proof. Analogously to the proof of Lemma 3.2.1. □

However, we have another useful property. For 1-reproducing or monotonic sets of formulæ, deciding the existence of a positive explanation reduces to testing whether A is one.

Lemma 3.3.2. *For $[B] \subseteq \mathsf{R}_1$ or $[B] \subseteq \mathsf{M}$, an instance (Γ, A, φ) of* P-Abd$(B, \mathcal{M})$ *has solutions if and only if A is a solution.*

Proof. Let $E \subseteq A$ be an arbitrary positive explanation for the given instance (Γ, A, φ), i.e., $\Gamma \wedge E$ is satisfiable and $\Gamma \wedge E \wedge \neg\varphi$ is unsatisfiable. With the 1-validity (respectively monotonicity) of Γ it follows immediately that $\Gamma \wedge A$ is satisfiable and that $\Gamma \wedge A \wedge \neg\varphi$ is unsatisfiable.

□

As a consequence we will see that some of the formerly NP-complete cases become tractable and that some of the formerly Σ_2^{p}-complete cases become coNP-complete.

3.3.2 The complexity of the Existence Problem

We begin our study with the variant of a positive literal manifestation.

Problem:	P-ABD(B, PQ)
Instance:	$\mathcal{I} = (\Gamma, A, q)$, where $\Gamma \subseteq \mathcal{L}(B)$, $A \subseteq \mathrm{Vars}(\Gamma)$ and q a variable
Question:	Does there exist an explanation $E \subseteq A$?

Proposition 3.3.3. *Let* $[B] \subseteq \mathsf{M}$. *Then* P-ABD$(B, \mathrm{PQ}) \in$ LOGSPACE.

Proof. According to Lemma 3.3.2 it suffices to test if A is a solution, that is, to test if $\Gamma \wedge A \wedge \neg q$ or equivalently $\Gamma \wedge A[q/0]$ is unsatisfiable and if $\Gamma \wedge A$ is satisfiable. This can be done in logarithmic space, since $\Gamma \wedge A$ and $\Gamma \wedge A[q/0]$ are monotonic formulæ [Sch05]. □

Proposition 3.3.4. *Let* $\mathsf{S}_{02} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{S}_{12} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{D}_1 \subseteq [B] \subseteq \mathsf{R}_1$. *Then* P-ABD$(B, \mathrm{PQ})$ *is* coNP*-complete.*

Proof. According to Lemma 3.3.2 it suffices to test whether A is a solution. Since $\Gamma \wedge A$ is always satisfiable, only the task of testing whether $\Gamma \wedge A \wedge \neg q$ is unsatisfiable remains. And this can be done in coNP.

Since $[\mathsf{D}_1 \cup \{1\}] = [\mathsf{S}_{12} \cup \{1\}] = \mathsf{R}_1$, $[\mathsf{S}_{02} \cup \{1\}] = \mathsf{S}_0$ and $\mathsf{S}_0 \subseteq \mathsf{R}_1$, we may assume that $\mathsf{S}_0 \subseteq [B]$ by Lemma 3.2.1. To show coNP-hardness we give a reduction from $\overline{3\mathrm{SAT}}$. Let $\varphi = \bigwedge_{i \in I} c_i$ be a 3CNF-formula. Since $[\{\rightarrow, 0\}] = \mathsf{BF}$, each clause c_i has a representation as a $\{\rightarrow, 0\}$-formula which we indicate by c_i'. Since each clause c_i consists of the disjunction of exactly three literals, each c_i' has a fixed number of occurrences of the implication connective $\rightarrow$. Let q be a fresh proposition. We map φ to $(\Gamma, \emptyset, q)$, where we define $\Gamma = \bigcup_{i \in I} c_i'[0/q]$. Note that Γ is a set of S_0-formulæ of polynomial size and 1-reproducing. Let φ be unsatisfiable. Then Γ is satisfied by the assignment setting to 1 all propositions and $\Gamma \wedge \neg q$ is unsatisfiable, because it is equivalent to $\varphi \wedge \neg q$. Summing up, $\emptyset$ is an explanation for $(\Gamma, \emptyset, q)$. Conversely, let φ be satisfiable. This implies that $\Gamma \wedge \neg q$ is satisfiable and thus $(\Gamma, \emptyset, q)$ has no explanations.

It remains to transform $(\Gamma, \emptyset, q)$ into a P-ABD(B, PQ)-instance for any relevant B. As $\to \in \mathsf{S}_0 \subseteq [B]$, this is done by replacing in Γ every occurrence of $\to$ by its B-representation. An exponential blow up may not occur since the implication connective $\to$ has a fixed number of occurrences.

□

The following classification is visualized in Figure 3.2 on page 75.

Theorem 3.3.5. *Let B be a finite set of Boolean functions. Then the positive explanation-existence problem for propositional B-formulæ with a positive literal manifestation,* P-ABD(B, PQ)*, is*

1. Σ_2^{p}*-complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$*,*
2. coNP*-complete if* $\mathsf{S}_{02} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{S}_{12} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{D}_1 \subseteq [B] \subseteq \mathsf{R}_1$*,*
3. *in* P *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$*,*
4. *in* LOGSPACE *in all other cases.*

The same classification holds for the problems P-ABD(B, Q)*,* P-ABD(B, PC) *and* P-ABD(B, C)*.*

Proof.

1. In [EG95], Eiter and Gottlob prove that the abduction problem in which the knowledge base is a set of clauses remains Σ_2^{p}-hard even if explanations are required to comprise positive literals only. A reduction from this problem can be done analogously to the one in Proposition 3.2.6.
2. See Proposition 3.3.4.
3. Membership in P follows from [NZ08, Proposition 66].
4. See Proposition 3.3.3.

□

Once again all upper and lower bounds for P-ABD(B, PQ) easily carry over to P-ABD(B, Q).

For P-ABD(B, NQ) the problem becomes trivial if $[B] \subseteq \mathsf{R}_1$ (Lemma 3.3.2). For $\mathsf{L}_0 \subseteq [B] \subseteq \mathsf{L}$ and $\mathsf{L}_3 \subseteq [B] \subseteq \mathsf{L}$, we obtain membership in P since in this case P-ABD(B, Q) is in P. For all remaining cases (i.e., for $\mathsf{D} \subseteq [B]$ and $\mathsf{S}_1 \subseteq [B]$), we obtain Σ_2^{p}-completeness from an easy adaptation of the first part in the proof of Proposition 3.3.5.

Theorem 3.3.6. *Let B be a finite set of Boolean functions. Then the positive explanation-existence problem for propositional B-formulæ with a negative literal manifestation,* P-ABD(B, NQ)*, is*

1. Σ_2^{p}*-complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$*,*
2. *in* P *if* $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$*,*
3. *trivial in all other cases.*

The same classification holds for P-ABD(B, NC)*.*

We turn now to the study of positive abduction for manifestations that are restricted to be respectively a clause, a term, or a B-formula.

P-ABD$(B$, PC$)$ Analogously to the symmetric case the algorithms can be extended to clauses. Thus, P-ABD(B, PC) is classified as P-ABD(B, PQ). Similarly the classifications for P-ABD(B, C) and P-ABD(B, NC) are the same as classifications for P-ABD(B, Q) and P-ABD(B, NQ).

P-ABD$(B$, PT$)$ The classification for positive terms is identical to the one for a single positive literal, except for the affine clones L_0, L_3, and L. For these, the complexity of P-ABD(B, PT) jumps from membership in P to NP-completeness.

Proposition 3.3.7. *Let* $\mathsf{L}_0 \subseteq [B] \subseteq \mathsf{L}$ *or* $\mathsf{L}_3 \subseteq [B] \subseteq \mathsf{L}$*. Then* P-ABD$(B, \mathrm{PT})$ *is* NP*-complete.*

Proof. Let (Γ, A, t) with $t = \bigwedge_{i \in I} x_i$ be an instance of P-ABD(B, PT). To check whether a given $E \subseteq A$ is an explanation, we have to test the satisfiability of $\Gamma \wedge E$ and the unsatisfiability of $\Gamma \wedge E \wedge \neg x_i$ for every $i \in I$. These tasks are equivalent to solving systems of linear equations, which can be done in polynomial time. As for the hardness by Lemma 3.2.1 it suffices to consider the case $\mathsf{L}_3 \subseteq [B]$, since $\mathsf{L}_3 \subseteq \mathsf{L} = [\mathsf{L}_0 \cup \{1\}]$. The hardness for $\mathsf{L}_3 \subseteq [B]$ follows easily with [NZ08, Propositions 69, 70]. □

Theorem 3.3.8. *Let B be a finite set of Boolean functions. Then the positive explanation-existence problem for propositional B-formulæ with a positive term manifestation,* P-ABD(B, PT)*, is*

1. Σ_2^{p}*-complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$*,*

2. coNP-*complete if* $\mathsf{S}_{02} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{S}_{12} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{D}_1 \subseteq [B] \subseteq \mathsf{R}_1$,
3. NP-*complete if* $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$,
4. *in* P *if* $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$,
5. *in* LOGSPACE *in all other cases.*

The same classification holds for P-ABD(B, T).

Proof. 1. Follows from the first item of Proposition 3.3.5.

2. Both membership and hardness follow from Proposition 3.3.4.
3. See Proposition 3.3.7.
4. Since $\mathsf{L}_1 \subseteq \mathsf{R}_1$, according to Lemma 3.3.2, it suffices to check whether A is a solution. This task reduces to solving systems of linear equations which is in P.
5. Analogously to Proposition 3.3.3.

□

Once again all upper and lower bounds for P-ABD(B, PT) carry over to P-ABD(B, T).

For P-ABD(B, NT) the problem becomes trivial if $[B] \subseteq \mathsf{R}_1$ (Lemma 3.3.2). For $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$, we obtain NP-completeness with the hardness being obtained from an easy reduction from P-ABD(B, PT): according to Lemma 3.2.1 is suffices to consider $B \cup \{1\}$. As we have $x \oplus y \in \mathsf{L} = [[B] \cup \{1\}] = [B \cup \{1\}]$, we can simply transform the given positive term into a negative one. For the remaining cases (i.e., $\mathsf{D} \subseteq [B]$ and $\mathsf{S}_1 \subseteq [B]$), we obtain Σ_2^{p}-completeness from an adaption of the first part in the proof of Proposition 3.3.5.

Theorem 3.3.9. *Let B be a finite set of Boolean functions. Then the positive explanation-existence problem for propositional B-formulæ with a negative term manifestation,* P-ABD(B, NT), *is*

1. Σ_2^{p}-*complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$,
2. NP-*complete if* $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$,
3. *trivial in all other cases.*

P-ABD(B, $\mathcal{L}(B)$) The complexity of P-ABD$(B, \mathcal{L}(B))$ differs from the complexity of P-ABD(B, PQ) for the clones either (a) above E or V and below M or (b) above L_0 or L_3 and below L. For the former the complexity increases to coNP-completeness, whereas for the latter we obtain only membership in NP; the exact complexity of positive abduction when both the knowledge base and the manifestation are represented by non-1-reproducing affine formulæ remains open.

Problem: P-ABD$(B, \mathcal{L}(B))$
Instance: $\mathcal{I} = (\Gamma, A, \varphi)$, where $\Gamma \subseteq \mathcal{L}(B)$, $A \subseteq \mathrm{Vars}(\Gamma)$ and $\varphi \in \mathcal{L}(B)$
Question: Does there exist an explanation $E \subseteq A$?

Proposition 3.3.10. *Let B be a finite set of Boolean functions such that $\mathsf{L}_0 \subseteq [B] \subseteq \mathsf{L}$ or $\mathsf{L}_3 \subseteq [B] \subseteq \mathsf{L}$. Then* P-ABD$(B, \mathcal{L}(B)) \in$ NP.

Proof. Let $E \subseteq A$ be a potential solution. The test for satisfiability of $\Gamma \wedge E$ and the test for unsatisfiability of $\Gamma \wedge E \wedge \neg\varphi$ are equivalent to solving two systems of linear equations, which can be done in polynomial time. □

Proposition 3.3.11. *Let B be a finite set of Boolean functions such that $\mathsf{S}_{00} \subseteq [B] \subseteq \mathsf{M}$ or $\mathsf{S}_{10} \subseteq [B] \subseteq \mathsf{M}$ or $\mathsf{D}_2 \subseteq [B] \subseteq \mathsf{M}$. Then* P-ABD$(B, \mathcal{L}(B))$ *is* coNP*-complete.*

Proof. We will first prove coNP-membership. According to Lemma 3.3.2, it suffices to test whether A is a solution. This can be done by first verifying that $\Gamma \wedge A$ is satisfiable, and afterwards verifying that $\Gamma \wedge A \wedge \neg\varphi$ is unsatisfiable. As Γ is a set of monotonic formulæ, deciding the satisfiability of $\Gamma \wedge A$ can be done in logarithmic space [Sch05]; and deciding whether $\Gamma \wedge A \wedge \neg\varphi$ is unsatisfiable is in coNP.

To establish coNP-hardness we give a reduction from $\overline{\text{3SAT}}$. Let φ be an instance of $\overline{\text{3SAT}}$ with $\mathrm{Vars}(\varphi) = \{x_1, \ldots, x_n\}$. Let φ' be obtained from φ by replacing all occurrences of $\neg x_i$ with a fresh proposition x'_i, $1 \leq i \leq n$. That is, $\varphi' \equiv \varphi[\neg x_1/x'_1, \ldots, \neg x_n/x'_n]$. Thus $\varphi' = \bigwedge_{i \in I} c'_i$ where every c'_i is a disjunction of three propositions. To φ we associate the propositional abduction problem $\mathcal{P} = (\Gamma, \emptyset, \psi)$ defined as follows:

$$\Gamma := \{c'_i \vee q \mid i \in I\} \cup \{x_i \vee x'_i \mid 1 \leq i \leq n\},$$
$$\psi := q \vee \textstyle\bigvee_{1 \leq i \leq n}(x_i \wedge x'_i).$$

Observe that

$$\Gamma \wedge \neg\psi \;\equiv\; \varphi \wedge \neg q \wedge \bigwedge_{i=1}^{n} (x_i \oplus x_i'). \tag{3.4}$$

Suppose that $\varphi \in \overline{3\text{SAT}}$, i.e., φ is unsatisfiable. From (3.4) it follows that $\Gamma \wedge \neg\psi$ is unsatisfiable. As Γ is satisfiable, $\emptyset$ is a solution for $\mathcal{P}$.

Suppose conversely that $\emptyset$ is a solution for $\mathcal{P}$. Then $\Gamma \wedge \neg\psi \equiv \varphi \wedge \neg q \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i')$ is unsatisfiable. Since q and the x_i' do not occur in φ, we obtain the unsatisfiability of φ. Thus, $\varphi \in \overline{3\text{SAT}}$.

The transformation of $\mathcal{P}$ into an P-ABD$(B, \mathcal{L}(B))$-instance for any relevant B can be done in exactly the same way as in the proof of Proposition 3.2.10. □

The following classification is visualized in Figure 3.4 on page 77.

Theorem 3.3.12. *Let B be a finite set of Boolean functions. Then the positive explanation-existence problem for propositional B-formulæ with a B-formula manifestation,* P-ABD$(B, \mathcal{L}(B))$, *is*

1. Σ_2^{p}*-complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$,
2. coNP*-complete if* $\mathsf{S}_{02} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{S}_{12} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{D}_1 \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{S}_{00} \subseteq [B] \subseteq \mathsf{M}$ *or* $\mathsf{S}_{10} \subseteq [B] \subseteq \mathsf{M}$ *or* $\mathsf{D}_2 \subseteq [B] \subseteq \mathsf{M}$
3. *in* NP *if* $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$,
4. *in* P *if* $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$,
5. *in* LOGSPACE *in all other cases.*

Proof.

1. Follows from the first item of Proposition 3.3.5.
2. See Proposition 3.3.11 and 3.3.4.
3. See Proposition 3.3.10.
4. See the fourth item of Proposition 3.3.8.
5. For $[B] \subseteq \mathsf{V}$, a B-formula is a positive clause. Thus the result follows from Theorem 3.3.5. For $[B] \subseteq \mathsf{N}$ and $[B] \subseteq \mathsf{E}$, see Proposition 3.2.3.

□

3.3.3 The complexity of the Verification Problem

As in the case of symmetric abduction, for the argument-verification problem we consider only the basic variant where the manifestation is a single positive literal.

Problem: P-ABD-CHECK
Instance: $\mathcal{I} = (\Gamma, A, q, E)$, where $\Gamma \subseteq \mathcal{L}$, $A \subseteq \mathrm{Vars}(\Gamma)$, q a variable and $E \subseteq A$
Question: Is E an explanation?

The following classification is visualized in Figure 3.6 on page 79.

Theorem 3.3.13. *Let B be a finite set of Boolean functions. Then the positive explanation-validity problem for propositional B-formulæ with a positive literal manifestation,* P-ABD-CHECK(B, PQ) *is*

1. DP-*complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$,
2. coNP-*complete if* $\mathsf{S}_{02} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{S}_{12} \subseteq [B] \subseteq \mathsf{R}_1$ *or* $\mathsf{D}_1 \subseteq [B] \subseteq \mathsf{R}_1$,
3. *in* P *if* $[B] \subseteq \mathsf{L}$, *and*
4. *in* LOGSPACE *if* $[B] \subseteq \mathsf{M}$ *or* $[B] \subseteq \mathsf{N}$.

Proof. 1. By virtue of Lemma 3.3.1 and the fact that $[\mathsf{S}_1 \cup \{1\}] = [\mathsf{D} \cup \{1\}] = \mathsf{BF}$, it suffices thus to consider the case $[B] = \mathsf{BF}$. One easily verifies that for this case exactly the same proof as in Proposition 3.2.13 applies (since $A = E = \emptyset$).

2. Due to the fact that $\Gamma \wedge E$ is always satisfiable ($B \subseteq \mathsf{R}_1$), exactly the same proof as in Proposition 3.3.4 for P-ABD(B, PQ) applies.

3. We have to check the following two conditions.

   (B1) $\Gamma \wedge E$ is satisfiable,
   (B2) $\Gamma \wedge E \wedge \neg q$ is unsatisfiable.

   Since we deal with affine formulæ, this can be done in P via Gaussian elimination.

4. We have to check the same two conditions as mentioned in the last item. This can be done in LOGSPACE since we deal with N- or monotonic formulæ.

□

3.3.4 Overview of results for Abduction

The following two tables give an overview of the results for the studied symmetric and positive abduction problems. The small numbers on the right side in the table cells refer to the corresponding theorem / proposition. The number is omitted for trivial results.

$\mathcal{M}$	E_*	N_*	V_*	L_*	$\mathsf{D}_2, \mathsf{S}_{*0} \subseteq [B] \subseteq \mathsf{M}$	$\mathsf{D}_1, \mathsf{S}_{*2} \subseteq [B] \subseteq \mathsf{BF}$
NQ, NC, NT	trivial	trivial	trivial	$\in$ P 3.2.7	$\in$ L 3.2.7	Σ_2^{P}-c 3.2.7
PQ, PC, Q, C	$\in$ L 3.2.3	$\in$ L 3.2.3	$\in$ L 3.2.3	$\in$ P 3.2.4	NP-c 3.2.5	Σ_2^{P}-c 3.2.6
PT, T	$\in$ L 3.2.9	$\in$ L 3.2.9	NP-c 3.2.8	$\in$ P 3.2.9	NP-c 3.2.9	Σ_2^{P}-c 3.2.9
$\mathcal{L}(B)$	$\in$ L 3.2.11	$\in$ L 3.2.11	$\in$ L 3.2.11	$\in$ P 3.2.11	Σ_2^{P}-c 3.2.10	Σ_2^{P}-c 3.2.10

Table 3.2: The complexity of ABD, where $*$-subscripts on clones denote all valid completions, L abbreviates LOGSPACE, and the suffix "-c" indicates completeness for the respective complexity class.

$\mathcal{M}$	$\mathsf{E}_*, \mathsf{N}_*, \mathsf{V}_*$	$\mathsf{L}_1, \mathsf{L}_2$	$\mathsf{L}_0, \mathsf{L}_3, \mathsf{L}$	$\mathsf{D}_2, \mathsf{S}_{*0} \subseteq [B] \subseteq \mathsf{M}$	$\mathsf{D}_1, \mathsf{S}_{*2} \subseteq [B] \subseteq \mathsf{R}_1$	$\mathsf{D}, \mathsf{S}_1 \subseteq [B] \subseteq \mathsf{BF}$
NQ, NC	trivial	trivial	$\in$ P 3.3.6	trivial	trivial	Σ_2^{P}-c 3.3.6
NT	trivial	trivial	NP-c 3.3.9	trivial	trivial	Σ_2^{P}-c 3.3.9
PQ, PC, Q, C	$\in$ L 3.3.3	$\in$ P 3.3.5	$\in$ P 3.3.5	$\in$ L 3.3.3	coNP-c 3.3.4	Σ_2^{P}-c 3.3.5
PT, T	$\in$ L 3.3.8	$\in$ P 3.3.8	NP-c 3.3.7	$\in$ L 3.3.8	coNP-c 3.3.8	Σ_2^{P}-c 3.3.8
$\mathcal{L}(B)$	$\in$ L 3.3.12	$\in$ P 3.3.12	$\in$ NP 3.3.10	coNP-c 3.3.11	coNP-c 3.3.12	Σ_2^{P}-c 3.3.12

Table 3.3: The complexity of P-ABD, where $*$-subscripts on clones denote all valid completions, L abbreviates LOGSPACE, and the suffix "-c" indicates completeness for the respective complexity class.

In the case of symmetric abduction our results show, for instance, that when the knowledge base's formulæ are restricted to be represented as positive clauses (column V_*, Table 3.2), then the abduction problem for single literal manifestations is very easy (solvable in LOGSPACE); this still holds if the manifestations are represented by positive clauses. But its complexity jumps to NP-completeness if we change the restriction on the manifestations to allow for positive terms.

Considering the case that all monotonic functions can be simulated (column $\mathsf{D}_2, \mathsf{S}_{*0} \subseteq [B] \subseteq \mathsf{M}$, Table 3.2), the abduction problem is NP-complete for manifestations represented by literals, clauses, or terms. Here allowing manifestation represented by a monotonic formula, causes the jump to Σ_2^{P}-completeness. This increase in the complexity of the problem

can be intuitively explained as follows. The complexity of the abduction rests on two sources: finding a candidate explanation and checking that it is indeed a solution. The NP-complete cases that occur in our classification hold for problems in which the verification can be performed in polynomial time. If both the knowledge base and the manifestation are represented by monotonic formulæ, verifying a candidate explanation is coNP-complete.

It comes as no surprise that the complexity of P-ABD$(B, \mathcal{M})$ is lower than or equal to the complexity of ABD$(B, \mathcal{M})$ in most cases (except for the affine clones, see column $\mathsf{L}_0, \mathsf{L}_3, \mathsf{L}$, Table 3.3). We have seen in Lemma 3.3.2 that for monotonic or 1-reproducing knowledge bases only one candidate needs to be considered. In these cases the complexity of the abduction problem is determined by the verification of the candidate. This explains the appearance of coNP-complete cases in our classification (columns $\mathsf{D}_2, \mathsf{S}_{*0} \subseteq [B] \subseteq \mathsf{M}$ and $\mathsf{D}_1, \mathsf{S}_{*2} \subseteq [B] \subseteq \mathsf{R}_1$, Table 3.3). For the affine clones (i.e., $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$), on the other hand, the tractability of ABD$(B, \mathcal{M})$ relies on Gaussian elimination. This method fails when restricting the hypotheses to be positive while the manifestation is also an affine formula: the exact complexity of P-ABD$(B, \mathcal{L}(B))$ remains open for $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$.

3.4 The complexity of Argumentation

3.4.1 Technical results and tools

Similar to abduction, also for the argumentation problems we may often use a constant 'for free', facilitating the complexity analysis. The following lemma states that when the considered B can express conjunction, we may suppose that B contains the constant 1.

Lemma 3.4.1. *Let* Arg-$\mathfrak{P}$ *denote any of the problems* Arg, Arg-Check, Arg-Rel *or* Arg-Disp. *Let B be a finite set of Boolean functions such that $\wedge \in [B]$, i.e., $\mathsf{E}_2 \subseteq [B]$. Then* Arg-$\mathfrak{P}(B \cup \{1\}) \leq_{\mathrm{m}}^{\log}$ Arg-$\mathfrak{P}(B)$.

Proof. Let $\mathcal{I}$ be the given instance. We map $\mathcal{I}$ to the instance $\mathcal{I}'$ obtained by replacing each formula ψ occurring in $\mathcal{I}$ by $\psi[1/t] \wedge t$, where t be a fresh variable. □

With a little more effort, one can also simulate the constant 1 for the problems Arg(B) and Arg-Rel(B) when B can express all self-dual and monotonic functions.

Lemma 3.4.2. *Let B be a finite set of Boolean functions such that $\mathsf{D}_2 \subseteq [B]$. Then* Arg$(B \cup \{1\}) \leq_{\mathrm{m}}^{\log}$ Arg(B) *and* Arg-Rel$(B \cup \{1\}) \leq_{\mathrm{m}}^{\log}$ Arg-Rel(B).

Proof. Let $g(x, y, z) := (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$. The function g is a base of D_2 and evaluates to true if and only if at least two of the variables are set to true. Given an instance (Δ, α) of Arg$(B \cup \{1\})$, we define an instance (Δ', α') of Arg(B) by $\Delta' := \{\psi[1/t] \mid \psi \in \Delta\} \cup \{t\}$ and $\alpha' = g(\alpha[1/t], t, q)$, where t and q are fresh variables. We claim that there is an argument for α in Δ if and only if there is an argument for α' in Δ'.

Let Φ be an argument for α in Δ. Consider $\Phi' := \{\psi[1/t] \mid \psi \in \Phi\} \cup \{t\}$. Observe that $\Phi' \equiv \Phi \wedge t$. Thus Φ' is satisfiable and $\Phi' \models \alpha$, hence $\Phi' \models \alpha[1/t] \wedge t$, as t does not occur in α. Therefore, we obtain $\Phi' \models g(\alpha[1/t], t, q)$. Moreover, either Φ' or $\Phi' \setminus \{t\}$ is minimal with this property. Indeed, suppose that there exists a $\psi' \in \Phi'$ with $\psi' = \psi[1/t]$ for some $\psi \in \Phi$ such that $\Phi' \setminus \{\psi'\} \models g(\alpha[1/t], t, q)$. Then $\Phi' \setminus \{\psi'\} \models \alpha[1/t] \wedge t$ as q does not occur in Φ', and hence $\Phi \setminus \{\psi\} \models \alpha$, contradictory to the minimality of Φ.

Conversely, with similar arguments it is easy to see that if Φ' is an argument for α' in Δ', then $\Phi := \{\psi[t/1] \mid \psi \in \Phi', \psi \neq t\}$ is an argument for α in Δ: as q does not occur in Φ', $\Phi' \models \alpha'$ implies that $\Phi' \models \alpha[1/t] \wedge t$.

This proves correctness of the reduction from $\textsc{Arg}(B \cup \{1\})$ to $\textsc{Arg}(B)$. The analogous result for Arg-Rel follows from the same arguments as above, mapping the additional component φ to $\varphi' := \varphi[1/t]$. □

Observe that the reduction of Lemma 3.4.2 will not work for Arg-Check: one would have to decide whether to map Φ to Φ' or to $\Phi' \setminus \{t\}$ to ensure minimality, which requires the ability to decide whether $\Phi' \setminus \{t\} \models t$ in Logspace. Further observe that an analogous statement of Lemma 3.4.1 for the constant 0 cannot be obtained in the obvious way. Replacing every formula ψ by $\psi[0/f] \vee f$ for a fresh variable f fails since such a reduction does not preserve consistency.

We will give once more an example that shows how these lemmas reduce the number of cases to study. We will show hardness results for $\textsc{Arg-Rel}(B)$ (Theorem 3.4.9) in the same three cases as in the example we gave in Section 3.2.1 for abduction:

1. $\mathsf{S}_{00} \subseteq [B]$
2. $\mathsf{D}_2 \subseteq [B]$
3. $\mathsf{S}_{10} \subseteq [B]$

Since we have this time only the constant 1 and not the constant 0, we cannot reduce these three cases to the case $\mathsf{M} \subseteq [B]$ but to the case $\mathsf{S}_{00} \subseteq [B]$:

Since in cases 2 and 3 either $\mathsf{D}_2 \subseteq [B]$ or $\mathsf{E}_2 \subseteq [B]$, according to Lemmas 3.4.1 and 3.4.2 we have $\textsc{Arg-Rel}(B \cup \{1\}) \leq_{\mathrm{m}}^{\log} \textsc{Arg-Rel}(B)$. Therefore, it is sufficient to prove hardness for $\textsc{Arg-Rel}(B \cup \{1\})$. Observe that if $\mathsf{D}_2 \subseteq [B]$ then $\mathsf{S}_{01}^2 = [\mathsf{D}_2 \cup \{1\}] \subseteq [[B] \cup \{1\}] = [B \cup \{1\}]$ and if $\mathsf{S}_{10} \subseteq [B]$ then $\mathsf{M}_1 = [\mathsf{S}_{10} \cup \{1\}] \subseteq [B \cup \{1\}]$. Since $\mathsf{S}_{00} \subseteq \mathsf{S}_{01}^2$ and $\mathsf{S}_{00} \subseteq \mathsf{M}_1$, we have in both cases $\mathsf{S}_{00} \subseteq [B \cup \{1\}]$. So, finally in order to prove that $\textsc{Arg-Rel}(B)$ is hard in the case where $\mathsf{S}_{00} \subseteq [B]$ or $\mathsf{D}_2 \subseteq [B]$ or $\mathsf{S}_{10} \subseteq [B]$ it is sufficient to prove that the hardness holds for B such that $\mathsf{S}_{00} \subseteq [B]$.

3.4.2 The complexity of the Existence Problem

We commence our study of the introduced argumentation problems with the argument-existence problem $\textsc{Arg}(B)$.

Problem: $\textsc{Arg}(B)$
Instance: $\mathcal{I} = (\Delta, \alpha)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha \in \mathcal{L}(B)$.
Question: Does there exist Φ such that (Φ, α) is an argument in Δ?

The following classification is visualized in Figure 3.4 on page 77.

Theorem 3.4.3. *Let B be a finite set of Boolean functions. Then the argument-existence problem for propositional B-formulæ, $\textsc{Arg}(B)$, is*

1. Σ_2^{p}*-complete if* $\mathsf{D} \subseteq [B]$ *or* $\mathsf{S}_1 \subseteq [B]$,
2. coNP*-complete if* $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ *with* $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ *and* $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$,
3. *in* NP *if* $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$,
4. *in* P *if* $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$, *and*
5. *in* Logspace *if* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{E}$ *or* $[B] \subseteq \mathsf{N}$.

The same classification holds for $\textsc{Arg-Disp}(B)$.

Proof. The general argumentation problem has been shown to be Σ_2^{p}-complete by [PWA03] via a reduction from $\textsc{QSat}_{\exists,2}$. An instance of this problem is a quantified formula $\exists X \forall Y \beta$, and one may assume that the formula β is in 3DNF, *i.e.*, $\beta = \bigvee_{j=1}^{p} t_j$ with exactly three literals by term. The authors map such an instance $\exists X \forall Y \beta$ to (Δ, α), where $\Delta := \{x \leftrightarrow 0, x \leftrightarrow 1 \mid x \in X\}$, and $\alpha := \beta$. We use this reduction to obtain Σ_2^{p}-completeness for $\textsc{Arg}(B)$ if $[B] = \mathsf{BF}$. We insert parentheses in β and obtain a formula of logarithmic parentheses-depth only. We can now substitute all occurring connectives with their B-representations and obtain this way in logarithmic space an instance of $\textsc{Arg}(B)$ which is equivalent to the original (Δ, α).

As $\wedge \in \mathsf{S}_1$ and $[\mathsf{S}_1 \cup \{1\}] = \mathsf{BF}$, we obtain Σ_2^{p}-completeness for the case $\mathsf{S}_1 \subseteq [B]$ according to Lemma 3.4.1. For the case $\mathsf{D} \subseteq [B]$, we obtain Σ_2^{p}-completeness by Lemma 3.4.2, since $\mathsf{D}_2 \subseteq \mathsf{D}$ and $[\mathsf{D} \cup \{1\}] = \mathsf{BF}$.

For $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ with $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ and $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$, membership in coNP follows from the facts that satisfiability is in P ([Lew79]), that Δ (or, in the case of M, Δ restricted to its satisfiable elements) is the only candidate to be checked, and that entailment is in coNP ([BMTV09a]). To prove the coNP-hardness of $\textsc{Arg}(B)$, we give a reduction from the

implication problem for B-formulæ, which is coNP-hard if $[B]$ contains one of the clones S_{00}, S_{10}, D_2. Let (ψ, α) be a pair of B-formulæ. We map this instance to $(\{\psi\}, \alpha)$ if ψ is satisfiable (which is easy to decide) and to a trivial positive instance otherwise.

For $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$, membership in NP follows from the fact that in this case Arg-Check is in P. Due to the trivial satisfiability of B-formulæ for $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$ (1-valid formulæ are always satisfiable, hence once again Δ is the only candidate to be checked), we can improve the upper bound for $\textsc{Arg}(B)$ with $[B] \in \{\mathsf{L}_1, \mathsf{L}_2\}$ to membership in P.

In all other cases, Logspace-membership follows from the fact that both problems, satisfiability and entailment, are contained in Logspace (see [BMTV09a]) for B-formulæ. □

We will see in Section 3.4.4 that the argument-dispensability problem Arg-Disp obeys the same classification as Arg.

3.4.3 The complexity of the Verification Problem

Our next problem under consideration is the argument-validity problem.

Problem: Arg-Check(B)
Instance: $\mathcal{I} = (\Phi, \alpha)$, where $\Phi \subseteq \mathcal{L}(B)$ and $\alpha \in \mathcal{L}(B)$.
Question: Is (Φ, α) an argument?

The argument-validity problem is in DP. To see this, consider the following languages A, B.

$$\begin{aligned} A &= \{(\Delta, \Phi, \alpha) \mid \Phi \text{ is satisfiable}, \forall \varphi \in \Phi : \Phi \setminus \{\varphi\} \not\models \alpha\}; \\ B &= \{(\Delta, \Phi, \alpha) \mid \Phi \models \alpha\}. \end{aligned}$$

It holds $A \in$ NP and $B \in$ coNP and Arg-Check $= A \cap B$.

The next two propositions give the central cases where we have a matching lower bound, i.e., we provide those B for which DP-hardness holds.

Proposition 3.4.4. *Let* $\mathsf{S}_{00} \subseteq [B]$. *Then* Arg-Check$(B)$ *is* DP*-complete.*

Proof. To prove DP-hardness we establish a reduction from Critical-3Sat. Let $\psi = \bigwedge_{j=1}^{m} C_j$ be an instance of Critical-3Sat, and $\mathrm{Vars}(\psi) = \{x_1, \ldots, x_n\}$. Let $u, x'_1, \ldots, x'_n$ be fresh, pairwise distinct variables. Let further $C'_j := C_j[\neg x_i / x'_i \mid 1 \leq i \leq n]$ for $1 \leq j \leq m$ and $\psi' := \bigwedge_{j=1}^{m} C'_j$. We map ψ to (Φ, α), where we define

$$\Phi = \{C'_j \mid 1 \leq j \leq m\} \text{ and}$$
$$\alpha = \bigvee_{i=1}^{n} u \vee (x_i \wedge x'_i).$$

Since $x \vee y$ and $x \vee (y \wedge z)$ are functions of S_{00}, α and all C'_j's are S_{00}-formulæ. These are by definition 1-reproducing. Therefore, Φ and α are satisfiable. For $1 \leq k \leq m$, let Φ_k, ψ_k, ψ'_k denote the respective set of clauses where we deleted the k^{th} clause. Note that always $\Phi \equiv \psi'$ and $\Phi_k \equiv \psi'_k$.

We may assume that each variable of ψ appears both as positive and as negative literal and further that each literal has at least two occurrences in two different clauses (this can easily be derived from the proof of DP-hardness in [PW88]). These two properties will assure that we have for all $k \in \{1, \ldots, m\}$

$$\text{Vars}(\psi') = \text{Vars}(\psi'_k) = \{x_i, x'_i \mid 1 \leq i \leq n\}, \tag{3.5}$$
$$\text{Vars}(\Phi) = \text{Vars}(\Phi_k) = \{x_i, x'_i \mid 1 \leq i \leq n\} \cup \{u\}. \tag{3.6}$$

Suppose now that $\psi \in$ Critical-3Sat, i.e., ψ is unsatisfiable and ψ_k is satisfiable for all $k \in \{1, \ldots, m\}$. We show that Φ entails α. Since $\psi \wedge \bigwedge_{i=1}^{n}(x_i \oplus x'_i) \equiv \psi' \wedge \bigwedge_{i=1}^{n}(x_i \oplus x'_i)$ is unsatisfiable, and $\psi' \equiv \Phi$ is monotonic, all models of Φ have to set both x_i and x'_i to 1 for at least one $i \in \{1, \ldots, n\}$. Since $\alpha[u/0] \equiv \bigvee_{i=1}^{n}(x_i \wedge x'_i)$, we therefore have $\Phi \models \alpha[u/0]$. Obviously also $\Phi \models \alpha[u/1]$, thus we have $\Phi \models \alpha$.

It remains to prove that Φ is minimal. Since for each $k \in \{1, \ldots, m\}$ $\psi_k \wedge \bigwedge_{i=1}^{n}(x_i \oplus x'_i) \equiv \psi'_k \wedge \bigwedge_{i=1}^{n}(x_i \oplus x'_i)$ is satisfiable, no $\psi'_k \equiv \Phi_k$ entails $\alpha[u/0] \equiv \bigvee_{i=1}^{n}(x_i \wedge x'_i)$. *A fortiori* no Φ_k entails α.

Conversely suppose that $(\Phi, \alpha) \in$ Arg-Check. Then Φ entails α and, since u does not occur in Φ, we have also that Φ entails $\alpha[u/0]$. Thus we have $\psi' \models \bigvee_{i=1}^{n}(x_i \wedge x'_i)$, which implies that ψ is unsatisfiable. By the minimality of Φ we know that no Φ_k entails α. Since $\Phi_k \models \alpha[u/1]$, we conclude that $\Phi_k \not\models \alpha[u/0]$, which implies that $\psi'_k \wedge \bigwedge_{i=1}^{n}(\neg x_i \vee \neg x'_i)$ is satisfiable. As ψ'_k is monotonic, we obtain that also $\psi'_k \wedge \bigwedge_{i=1}^{n}(x_i \oplus x'_i)$ and hence ψ_k itself is satisfiable.

We finally transform (Φ, α) into a B-instance for all B such that $\mathsf{S}_{00} \subseteq [B]$ by replacing every connective by its B-representation. This transformation works in logarithmic space for Φ since it consists in clauses

of size 3. It also does for α if we first represent it as an $\vee$-tree of depth logarithmic in n. □

Proposition 3.4.5. *Let B be a finite set of Boolean functions such that $\mathsf{D}_2 \subseteq [B]$. Then* Arg-Check$(B)$ *is* DP*-complete.*

Proof. We give a reduction from Critical-3Sat similar to Proposition 3.4.4. For $k \in \mathbb{N}$, we define g_k as the $(k+1)$-ary function verifying

(a) $g_k(z_1, \ldots, z_k, 0) \equiv \bigwedge_{i=1}^{k} z_i$ and

(b) $g_k(z_1, \ldots, z_k, 1) \equiv \bigvee_{i=1}^{k} z_i$.

Note that for every $k \in \mathbb{N}$, g_k is monotonic and self-dual, and thus contained in D_2. By abuse of notation, given a positive clause $C = (c_1 \vee c_2 \vee c_3)$ and a variable x, $g_3(C, x)$ stands for $g_3(c_1, c_2, c_3, x)$. Let $\psi = \bigwedge_{j=1}^{m} C_j$ be an instance of Critical-3Sat with $C_j = (l_j^1 \vee l_j^2 \vee l_j^3)$ and $\mathrm{Vars}(\psi) = \{x_1, \ldots, x_n\}$. Let further $u, v, x_1', \ldots, x_n'$ be fresh, pairwise distinct variables and $C_j' := C_j[\neg x_i / x_i' \mid 1 \leq i \leq n]$ for $1 \leq j \leq m$.

We map ψ to (Φ, α), where we define

$$\Phi = \{g_3(C_j', u) \mid 1 \leq j \leq m\}, \text{ and}$$
$$\alpha = g_n\big((g_2(x_i, x_i', v))_{1 \leq i \leq n},\ u\big).$$

Obviously α and the formulæ in Φ are D_2-formulæ and thus satisfiable. Let $\psi' := \bigwedge_{j=1}^{m} C_j'$ and for $1 \leq k \leq m$, let Φ_k, ψ_k, ψ_k' denote the respective set of formulæ (clauses) where we deleted the k^{th} formula (clause).

As in the previous proposition, we may assume that each variable of ψ appears both as positive and as negative literal and that further each literal has at least two occurrences in two different clauses. These two properties will assure that we have for all $k \in \{1, \ldots, m\}$

$$\mathrm{Vars}(\psi') = \mathrm{Vars}(\psi_k') = \{x_i, x_i' \mid 1 \leq i \leq n\}, \tag{3.7}$$
$$\mathrm{Vars}(\Phi) = \mathrm{Vars}(\Phi_k) = \{x_i, x_i' \mid 1 \leq i \leq n\} \cup \{u\}. \tag{3.8}$$

Suppose now that $\psi \in$ Critical-3Sat, i.e., ψ is unsatisfiable and ψ_k is satisfiable for all $k \in \{1, \ldots, m\}$. We show that Φ entails α for all possible values of u, v, where we consider in detail only the case where $v = 0$. The case $v = 1$ is analogous.

(i) $(u,v) = (1,0)$: $\Phi[u/1] \equiv \psi'$ and $\alpha[u/1, v/0] \equiv \bigvee_{i=1}^{n}(x_i \wedge x_i')$. Since $\psi \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i') \equiv \psi' \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i')$ is unsatisfiable and ψ' is monotonic, all models of ψ' have to set both x_i and x_i' to 1 for at least one $i \in \{1, \ldots, n\}$. Therefore, $\Phi[u/1] \models \alpha[u/1, v/0]$.

(ii) $(u,v) = (0,0)$: $\Phi[u/0] \equiv \bigwedge_{i=1}^{n}(x_i \wedge x_i')$ and $\alpha[u/0, v/0] \equiv \bigwedge_{i=1}^{n}(x_i \wedge x_i')$. Obviously $\Phi[u/0] \models \alpha[u/0, v/0]$.

It remains to prove that Φ is minimal. Since $\psi \in$ CRITICAL-3SAT, $\psi_k \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i') \equiv \psi_k' \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i')$ is satisfiable and hence $\Phi_k[u/1] \equiv \psi_k'$ does not entail $\alpha[u/1, v/0] \equiv \bigvee_{i=1}^{n}(x_i \wedge x_i')$, i.e., $\Phi_k \not\models \alpha$.

Conversely suppose that $(\Phi, \alpha) \in$ ARG-CHECK. Then $\Phi \models \alpha$ and, since v does not occur in Φ, we have also $\Phi[u/1] \models \alpha[u/1, v/0]$. Thus we have $\psi' \models \bigvee_{i=1}^{n}(x_i \wedge x_i')$, which implies that ψ is unsatisfiable. By the minimality of Φ, we obtain that no Φ_k entails α. One easily verifies that in the cases $(u,v) \in \{(0,0),(0,1),(1,1)\}$ Φ_k still entails α (use (3.8) for (u,v) = (0,0)). Thus we have that $\Phi_k[u/1] \not\models \alpha[u/1, v/0]$, which implies that $\psi_k' \wedge \bigwedge_{i=1}^{n}(\neg x_i \vee \neg x_i')$ is satisfiable. As ψ_k' is monotonic, we obtain that $\psi_k' \wedge \bigwedge_{i=1}^{n}(x_i \oplus x_i')$ and hence ψ_k itself is satisfiable, too.

Finally, we transform (Φ, α) into a B-instance for all B such that $\mathsf{D}_2 \subseteq [B]$ in replacing all occurrences of g_k by its B-representation. This transformation works in logarithmic space, because we may assume the function g_n to be a g_2-tree of depth logarithmic in n. □

The full picture for ARG-CHECK(B) is given in the forthcoming theorem, where we also provide the results for the 'easier' clones. The classification is visualized in Figure 3.7 on page 80.

Theorem 3.4.6. *Let B be a finite set of Boolean functions. Then the argument-validity problem for propositional B-formulæ,* ARG-CHECK(B)*, is*

1. DP-*complete if* $\mathsf{S}_{00} \subseteq [B]$ *or* $\mathsf{S}_{10} \subseteq [B]$ *or* $\mathsf{D}_2 \subseteq [B]$,
2. *in* P *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$,
3. *in* LOGSPACE *if* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{E}$ *or* $[B] \subseteq \mathsf{N}$.

Proof. For DP-completeness, according to Propositions 3.4.4 and 3.4.5 it remains only to deal with the case $\mathsf{S}_{10} \subseteq [B]$. Since $\mathsf{D}_2 \subseteq \mathsf{M}_1 = [\mathsf{S}_{10} \cup \{1\}] \subseteq [B \cup \{1\}]$, we obtain that ARG-CHECK($B \cup \{1\}$) is DP-hard

by Proposition 3.4.5. As $\wedge \in [B]$, we may apply Lemma 3.4.1 and obtain the DP-hardness of ARG-CHECK(B).

In the case $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$ the sets Φ, $\Phi \cup \{\neg\alpha\}$, and $(\Phi \setminus \{\varphi\}) \cup \{\neg\alpha\}$ for all $\varphi \in \Phi$ can be easily transformed into systems of linear equations. Thus checking the three conditions (C1)–(C3) as given on page 29 comes down to solving a polynomial number of systems of linear equations. This can be done in polynomial time using Gaussian elimination. For $[B] \subseteq \mathsf{V}$, for $[B] \subseteq \mathsf{E}$, and for $[B] \subseteq \mathsf{N}$ this check can be done in logarithmic space, as in this case the satisfiability of sets of B-formulæ can be determined in logarithmic space (see [Sch05] and Proposition 3.2.3). □

3.4.4 The complexity of Relevance and Dispensability

Note that the argument-dispensability problem ARG-DISP,

Problem: ARG-DISP(B)
Instance: $\mathcal{I} = (\Delta, \alpha, \varphi)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha, \varphi \in \mathcal{L}(B)$.
Question: Does there exist an argument (Φ, α) in Δ such that $\varphi \notin \Phi$?

obeys the same classification as ARG, since it is equivalent to ARG via reductions that do not influence the type of the formulæ from Δ and α:

1. ARG $\leq_{\mathrm{m}}^{\log}$ ARG-DISP: $(\Delta, \alpha) \mapsto (\Delta \cup \{t\}, \alpha, t)$.
2. ARG-DISP $\leq_{\mathrm{m}}^{\log}$ ARG: $(\Delta, \alpha, \varphi) \mapsto (\Delta \setminus \{\varphi\}, \alpha)$.

The remaining decision problem to analyse is ARG-REL(B) which turns out to be the most difficult in terms of complexity.

Problem: ARG-REL(B)
Instance: $\mathcal{I} = (\Delta, \alpha, \varphi)$, where $\Delta \subseteq \mathcal{L}(B)$ and $\alpha, \varphi \in \mathcal{L}(B)$.
Question: Does there exist an argument (Φ, α) in Δ such that $\varphi \in \Phi$?

Proposition 3.4.7. *Let B be a finite set of Boolean functions such that $\mathsf{S}_{00} \subseteq [B]$. Then* ARG-REL($B$) *is Σ_2^{p}-complete.*

Proof. To see that ARG-REL(B) is contained in Σ_2^{p}, observe that, given an instance $(\Delta, \alpha, \varphi)$, we can guess a set $\Phi \subseteq \Delta$ such that $\varphi \in \Phi$ and

verify conditions (C1)–(C3) as given on page 29 in polynomial time using an NP-oracle.

To prove Σ_2^{p}-hardness, we provide a reduction from the problem $\mathrm{QSAT}_{\exists,2}$. An instance of this problem is a quantified formula $\exists X \forall Y \beta$ where $\beta = \bigvee_{j=1}^{p} t_j$ with exactly three literals by term. Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. Let u, v, $x'_1, \ldots, x'_n, y'_1, \ldots, y'_m$ be fresh variables. We transform $\exists X \forall Y \beta$ to $(\Delta, \alpha, \varphi)$, where

$$\Delta := \{x_i, x'_i \mid 1 \leq i \leq n\} \cup \{v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i)\} \cup \{u\},$$

$$\alpha := \beta' \wedge v \wedge \big(\bigvee_{i=1}^{n} (x_i \wedge x'_i) \vee u \big),$$

$$\varphi := u,$$

with $\beta' = \bigvee_{j=1}^{p} t'_j$ and $t'_j := t_j[\neg x_1/x'_1, \ldots, \neg x_n/x'_n, \neg y_1/y'_1, \ldots, \neg y_m/y'_m]$ for all $1 \leq j \leq p$.

We show that $\exists X \forall Y \beta$ is valid if and only if $(\Delta, \alpha, \varphi) \in \textsc{Arg-Rel}(\{\wedge, \vee\})$. If $\exists X \forall Y \beta$ is valid, then there exists an assignment $\sigma \colon X \to \{0,1\}$ such that $\sigma \models \beta$. Consequently, for $\Phi := \{x_i \mid \sigma(x_i) = 1\} \cup \{x'_i \mid \sigma(x_i) = 0\} \cup \{u, v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i)\}$, we obtain $\Phi \models \alpha$. As Φ is consistent, it thus remains to show that u is relevant, i.e., that $\Phi \setminus \{u\} \not\models \alpha$. This follows from the fact that $\Phi \setminus \{u\}$ is satisfied by any assignment σ' extending σ with $\sigma'(u) := 0$ and $\sigma'(x'_i) := 1 - \sigma(x_i)$, while such a σ' does not entail $\bigvee_{i=1}^{n} (x_i \wedge x'_i) \vee u$ and hence $\sigma' \not\models \alpha$.

For the converse direction, let Φ be a support for α such that $u \in \Phi$. Since $\Phi \models \alpha$ we conclude that $v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i) \in \Phi$ and hence $\Phi = \mathcal{X} \cup \{v \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i)\} \cup \{u\}$, for some $\mathcal{X} \subseteq \{x_i, x'_i \mid 1 \leq i \leq n\}$. From $\Phi \models \alpha$ also follows that $\Phi \models \beta'$. From the minimality of Φ we conclude that in particular $\Phi \setminus \{u\} \not\models \alpha$. And therefore $\Phi \not\models \bigvee_{i=1}^{n} (x_i \wedge x'_i)$. That is $\Phi \wedge \bigwedge_{i=1}^{n} (\neg x_i \vee \neg x'_i)$ is satisfiable and since Φ is monotonic, consequently also $\Phi \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i)$ is satisfiable. Summed up, we know that $\gamma := \mathcal{X} \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i) \wedge \bigwedge_{i=1}^{m} (y_i \vee y'_i)$ is satisfiable and $\gamma \models \beta'$. Hence, *a fortiori*, $\gamma' := \mathcal{X} \wedge \bigwedge_{i=1}^{n} (x_i \oplus x'_i) \wedge \bigwedge_{i=1}^{m} (y_i \oplus y'_i)$ is satisfiable and $\gamma' \models \beta'$. Define now $\sigma_X(x_i) = 1$ if $x_i \in \mathcal{X}$, $\sigma_X(x_i) = 0$ otherwise. Obviously any extension of σ_X to Y satisfies β and therefore $\exists X \forall Y \beta$ is valid.

It remains to transform $(\Delta, \alpha, \varphi)$ into an $\textsc{Arg-Rel}(B)$-instance for all B such that $\mathsf{S}_{00} \subseteq [B]$. As both $\wedge$ and $\vee$ are associative, we can insert parentheses into $(\Delta, \alpha, \varphi)$ such that we can represent each formula

as binary $\{\wedge, \vee\}$-tree of logarithmic depth. Let f be a fresh variable and let h be the boolean function in S_{00} defined by $h(f, x, y) \equiv f \vee (x \wedge y)$. We further transform our instance into $(\Delta', \alpha' \vee f, \varphi')$, where $\Delta', \alpha', \varphi'$ are obtained by replacing each occurrence of $x \wedge y$ by $h(f, x, y)$. One easily verifies that $(\Delta', \alpha' \vee f, \varphi')$ is in $\textsc{Arg-Rel}(\{\vee, h\})$ if and only if $(\Delta, \alpha, \varphi) \in \textsc{Arg-Rel}(\{\wedge, \vee\})$. We finally replace $\vee$ and h by their B-representation. □

Proposition 3.4.8. *Let B be a finite set of Boolean functions such that $[B] \subseteq \mathsf{V}$ or $[B] \subseteq \mathsf{E}$ or $[B] \subseteq \mathsf{N}$. Then $\textsc{Arg-Rel}(B)$ is in $\textsc{Logspace}$.*

Proof. We assume the representation of V-, E-, or N-formulæ as respectively positive clauses, positive terms, or literals. Let us first consider $\textsc{Arg-Rel}(B)$ for $[B] \subseteq \mathsf{E}$. It is easy to observe that a set of positive terms Δ entails a positive term α if and only if $\mathrm{Vars}(\alpha) \subseteq \mathrm{Vars}(\Delta)$. We claim that Algorithm 1 decides $\textsc{Arg-Rel}(B)$.

Algorithm 1 Algorithm for $\textsc{Arg-Rel}(B)$ with $[B] \subseteq \mathsf{E}$.

Require: a set Δ of positive terms and positive terms α, φ with $\varphi \in \Delta$.
1: **for all** $x \in \mathrm{Vars}(\varphi)$ **do**
2: $\quad\Delta_x := \{\varphi\} \cup \{\tau \in \Delta \mid x \notin \mathrm{Vars}(\tau)\}$
3: $\quad$**if** $\Delta_x \models \alpha$ **then**
4: $\quad\quad$accept
5: $\quad$**end if**
6: **end for**
7: reject

Algorithm 1 can be implemented using only a logarithmic amount of space if we do not construct Δ_x entirely but rather check the condition in line 3 directly: $\Delta_x \models \alpha$ holds if and only if $\mathrm{Vars}(\alpha) \subseteq \mathrm{Vars}(\varphi) \cup \mathrm{Vars}(\{\tau \in \Delta \mid x \notin \mathrm{Vars}(\tau)\})$.

To prove correctness, notice that Algorithm 1 accepts only if there exists a $\Delta_x \subseteq \Delta$ such that $\Delta_x \models \alpha$ and $\Delta_x \setminus \{\varphi\} \not\models \alpha$. Thus Δ_x contains a support Φ such that $\varphi \in \Phi$. Conversely, let Φ be a support such that $\varphi \in \Phi$. Since $\Phi \models \alpha$ and $\Phi \setminus \{\varphi\} \not\models \alpha$, there is at least one $x_i \in (\mathrm{Vars}(\varphi) \cap \mathrm{Vars}(\alpha)) \setminus \mathrm{Vars}(\Phi \setminus \{\varphi\})$. For this x_i the algorithm constructs $\Delta_{x_i} := \{\varphi\} \cup \{\tau \in \Delta \mid x_i \notin \mathrm{Vars}(\tau)\}$. Obviously $\Phi \subseteq \Delta_{x_i}$ and therefore $\Delta_{x_i} \models \alpha$ which causes the algorithm to accept.

Next, consider $\textsc{Arg-Rel}(B)$ for $[B] \subseteq \mathsf{V}$. Observe that a set of positive clauses C entails a positive clause α if and only if there is a clause $c \in C$

such that $\mathrm{Vars}(c) \subseteq \mathrm{Vars}(\alpha)$. Thus if there is a support Φ with $\varphi \in \Phi$ then it is the singleton $\{\varphi\}$. Given $(\Delta, \alpha, \varphi)$ as an instance of Arg-Rel(V), it hence suffices to check whether $\mathrm{Vars}(\varphi) \subseteq \mathrm{Vars}(\alpha)$, which can be done in Logspace.

Finally Arg-Rel(B) for $[B] \subseteq \mathsf{N}$ is in Logspace, since each B-formula can be transformed into a single literal. □

We obtain the following complexity classification for Arg-Rel. The classification is visualized in Figure 3.8 on page 81.

Theorem 3.4.9. *Let B be a finite set of Boolean functions. Then the argument-relevance problem for propositional B-formulæ,* Arg-Rel(B)*, is*

1. Σ_2^{p}*-complete if* $\mathsf{S}_{00} \subseteq [B]$ *or* $\mathsf{D}_2 \subseteq [B]$ *or* $\mathsf{S}_{10} \subseteq [B]$*,*
2. *in* NP *if* $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$*,*
3. *in* Logspace *if* $[B] \subseteq \mathsf{V}$ *or* $[B] \subseteq \mathsf{E}$ *or* $[B] \subseteq \mathsf{N}$*.*

Proof. Applying Lemma 3.4.1 and Lemma 3.4.2 as shown in Section 3.4.1, for (1) it suffices to show hardness for B such that $\mathsf{S}_{00} \subseteq [B]$. This has been done in Proposition 3.4.7. Item (2) follows from the fact that for $[B] \subseteq \mathsf{L}$, Arg-Check(B) is in P (Proposition 3.4.6) and (3) has been shown in Proposition 3.4.8. □

3.4.5 Overview of results for Argumentation

It turns out that Arg and Arg-Disp have the same complexity classification as the positive abduction problem with a B-formula manifestation P-Abd($B, \mathcal{L}(B)$).

The following table gives an overview of the results for the studied argumentation problems. The small numbers on the right side in the table cells refer to the corresponding theorem / proposition.

	$\mathsf{N}_*, \mathsf{V}_*, \mathsf{E}_*$	$\mathsf{L}_1, \mathsf{L}_2$	$\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3$	$\mathsf{D}_2, \mathsf{S}_{*0} \subseteq [B] \subseteq \mathsf{M}, \mathsf{R}_1$	$\mathsf{D}, \mathsf{S}_1 \subseteq [B] \subseteq \mathsf{BF}$
Arg-Check	$\in$ L 3.4.6	$\in$ P 3.4.6	$\in$ P 3.4.6	DP-c 3.4.4, 3.4.5	DP-c 3.4.4, 3.4.5
Arg, Arg-Disp	$\in$ L 3.4.3	$\in$ P 3.4.3	$\in$ NP 3.4.3	coNP-c 3.4.3	Σ_2^{p}-c 3.4.3
Arg-Rel	$\in$ L 3.4.8	$\in$ NP 3.4.9	$\in$ NP 3.4.9	Σ_2^{p}-c 3.4.7	Σ_2^{p}-c 3.4.7

Table 3.4: The complexity of the argumentation problems. The $*$-subscripts on clones denote all valid completions, L abbreviates Logspace, and the suffixe "-c" indicates completeness.

Our results show, for instance, that when the knowledge base's formulæ are restricted to be represented as positive clauses, positive terms or single literals (column $\mathsf{V}_*, \mathsf{E}_*, \mathsf{N}_*$) then all considered problems are very easy. While for affine formulæ (columns $\mathsf{L}_1, \mathsf{L}_2$ and $\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3$) all problems become more involved, with P- and NP-membership.

Notably are the sets B of Boolean connectives where $\mathcal{X} \subseteq [B] \subseteq \mathcal{Y}$ with $\mathcal{X} \in \{\mathsf{S}_{00}, \mathsf{S}_{10}, \mathsf{D}_2\}$ and $\mathcal{Y} \in \{\mathsf{M}, \mathsf{R}_1\}$. This case typically applies to monotonic formulæ in which no negation is involved. Such sets B give coNP-completeness for Arg(B), while Arg-Rel(B) remains complete for Σ_2^{p}. It may come as a surprise that in these cases verifying an argument is potentially harder than deciding the existence of an argument (Arg-Check is DP-complete, Arg is only coNP-complete). This is due to the fact that when verifying an argument, the minimality condition of a support has to be checked, whereas this condition is of no importance for the argument existence problem.

The results obtained for the L-cases are partial. This corresponds to the case where individual formulæ are linear equations over the two-elements field. The fact that Arg(B) with $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}_1$ is in P relies on Gaussian elimination knowing that in this case we only have to check whether $\Phi \models \alpha$, that is whether the linear system $\Phi \wedge \neg\alpha$ is satisfiable. For the corresponding problems Arg-Rel(B), in which minimality plays a role, we only have an NP upper-bound, so far. In fact, the exact classification of the problems into tractable and intractable cases remains open for affine sets of Boolean connectives in the following cases: Arg(B) with $[B] \in \{\mathsf{L}, \mathsf{L}_0, \mathsf{L}_3\}$ and Arg-Rel(B) with $\mathsf{L}_2 \subseteq [B] \subseteq \mathsf{L}$.

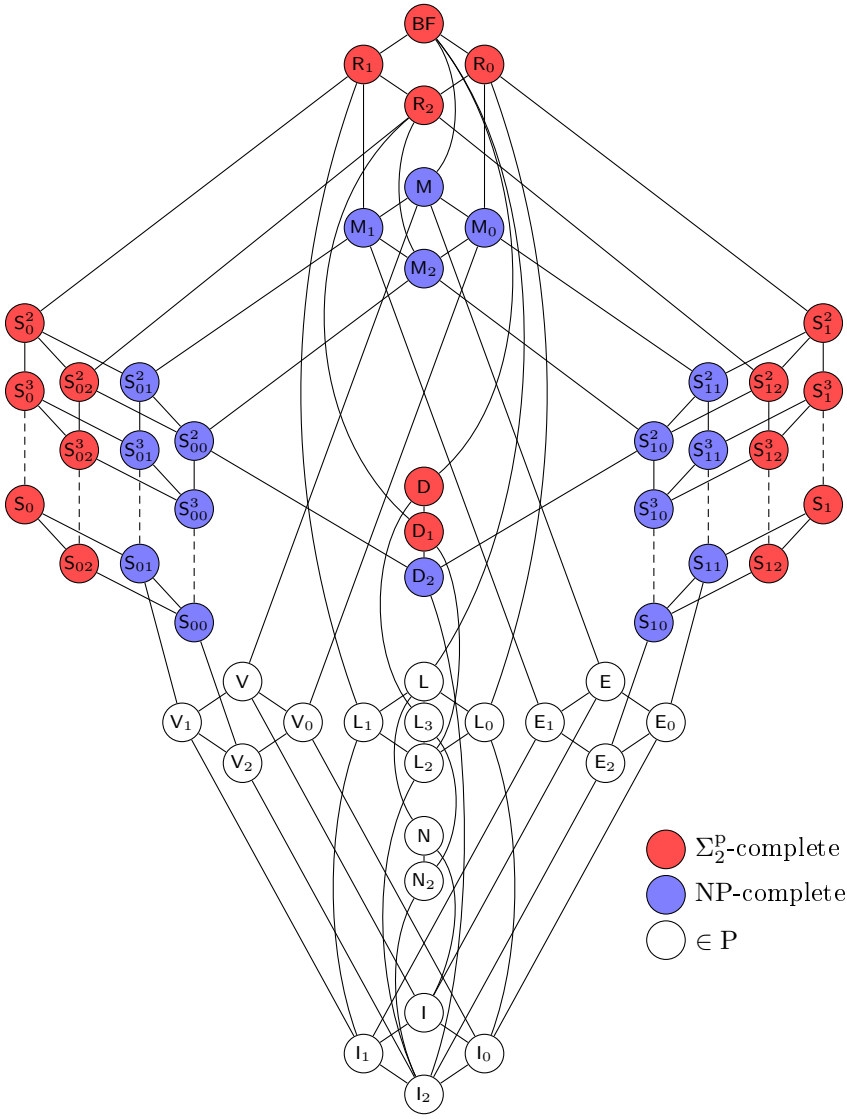

Figure 3.1: The complexity of $\textsc{Abd}(B, \mathrm{PQ})$.

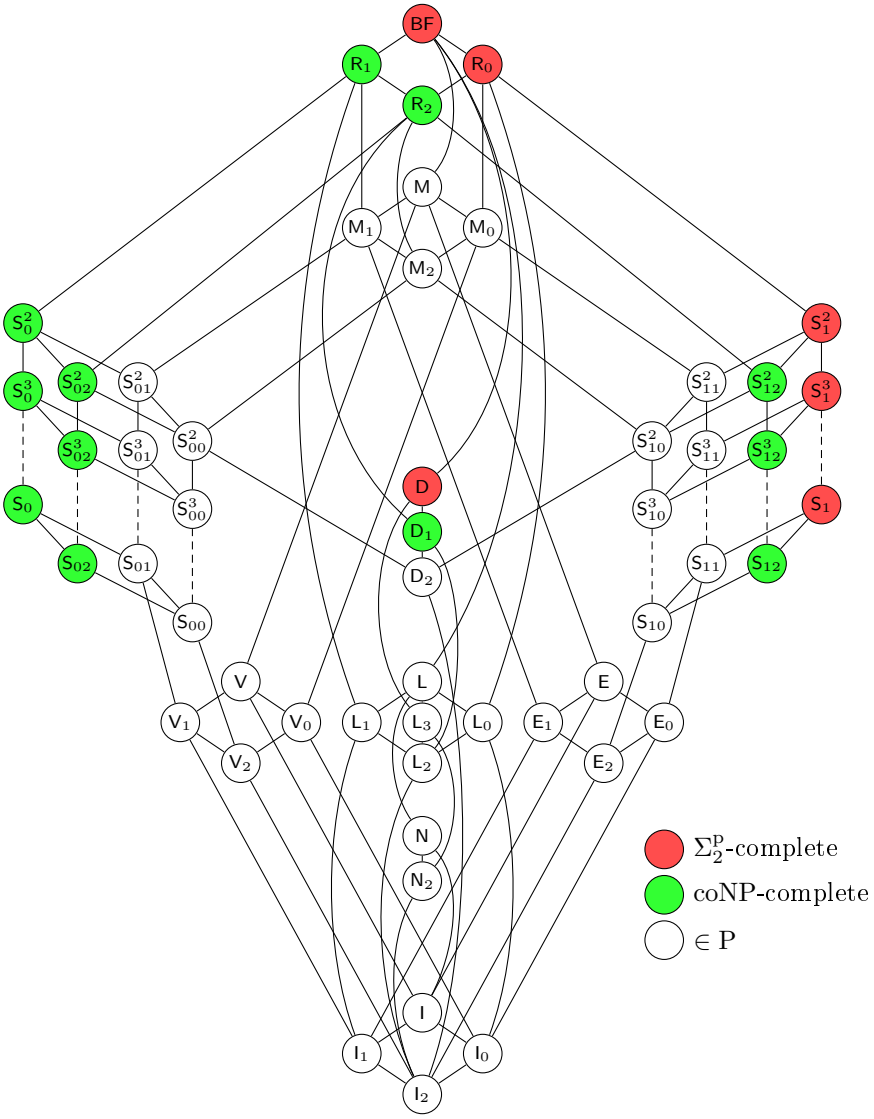

Figure 3.2: The complexity of P-ABD(B, PQ).

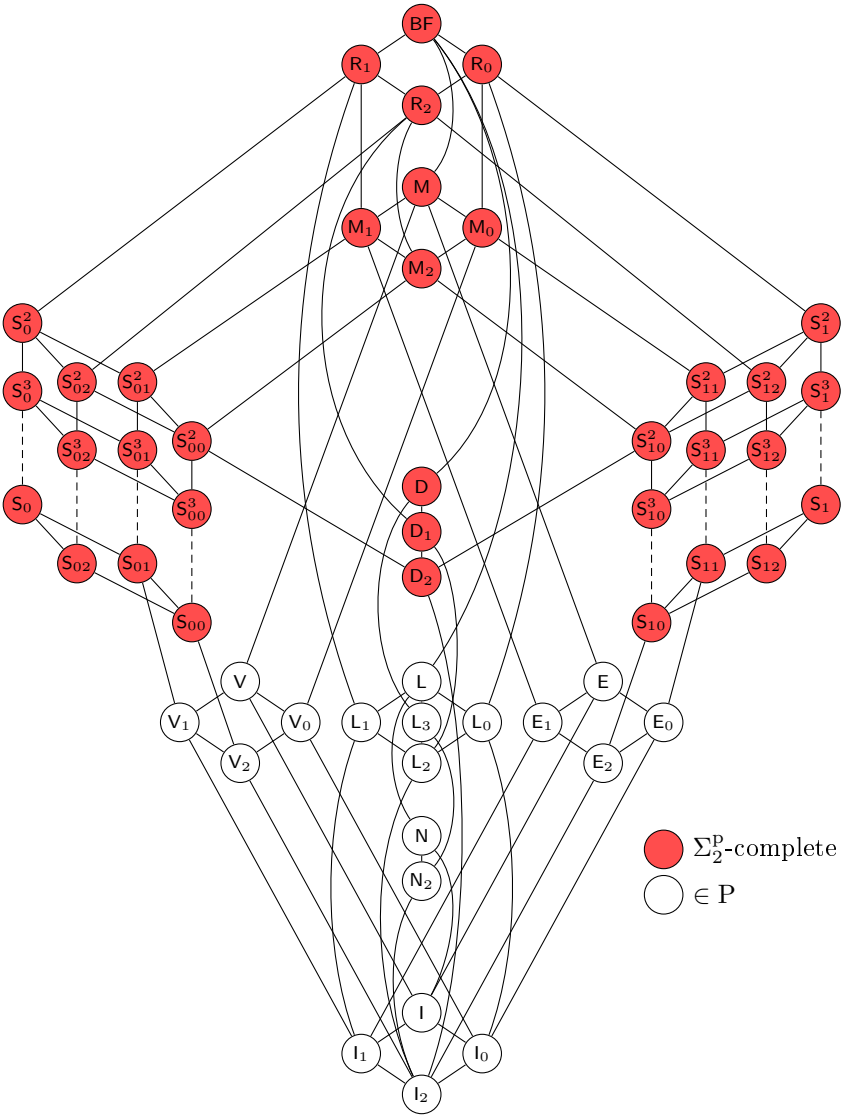

Figure 3.3: The complexity of $\textsc{Abd}(B, \mathcal{L}(B))$.

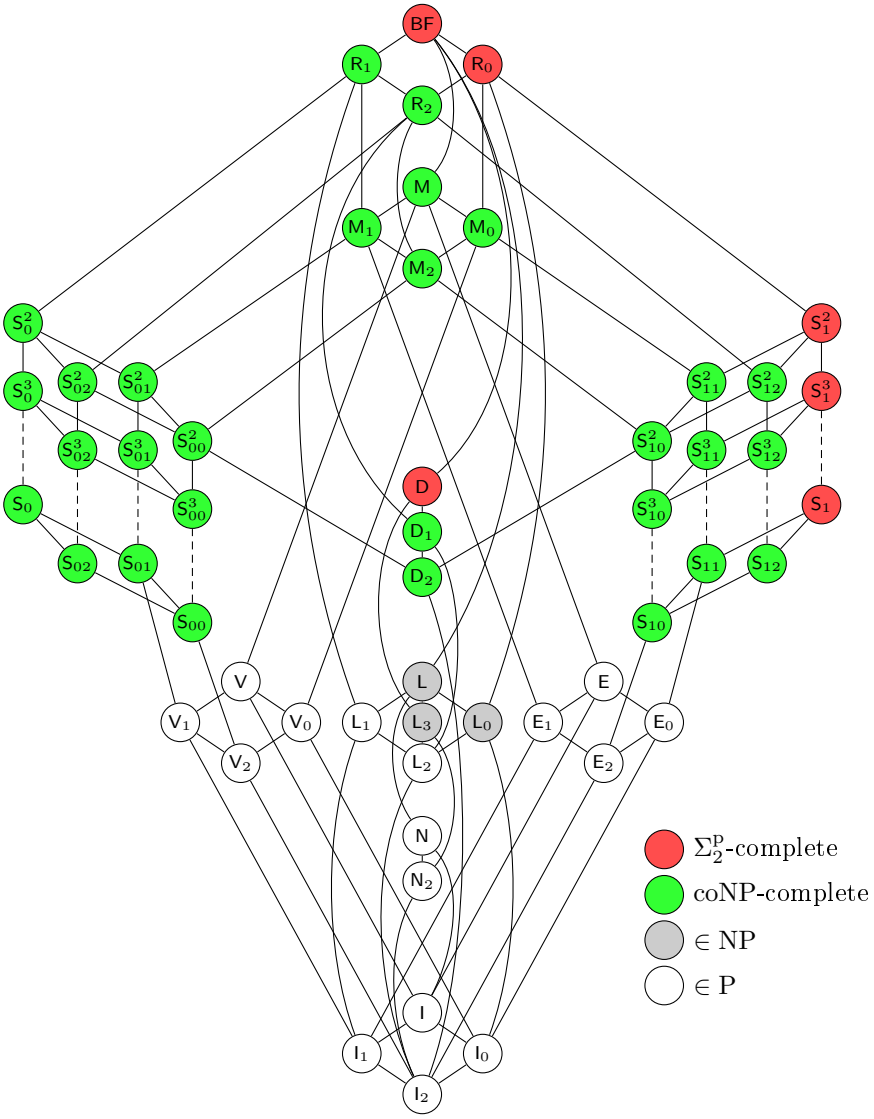

Figure 3.4: The complexity of P-ABD$(B, \mathcal{L}(B))$ and ARG(B).

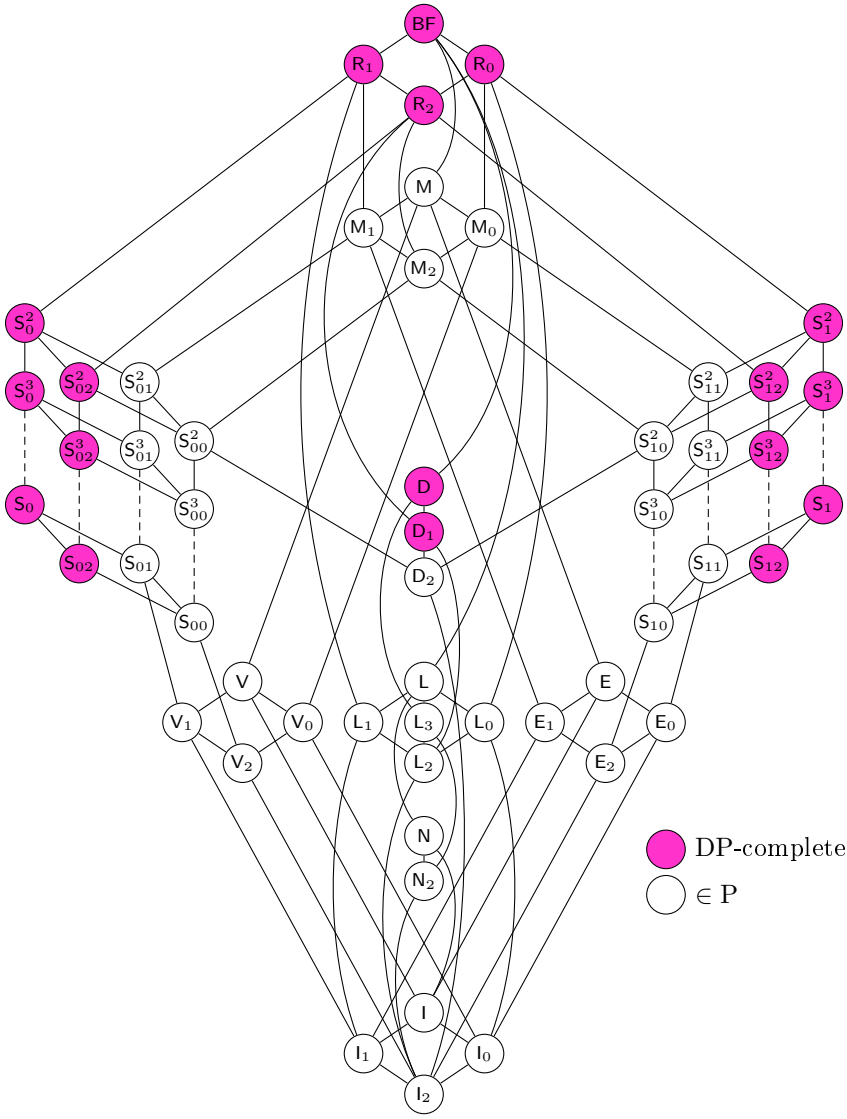

Figure 3.5: The complexity of ABD-CHECK(B, PQ).

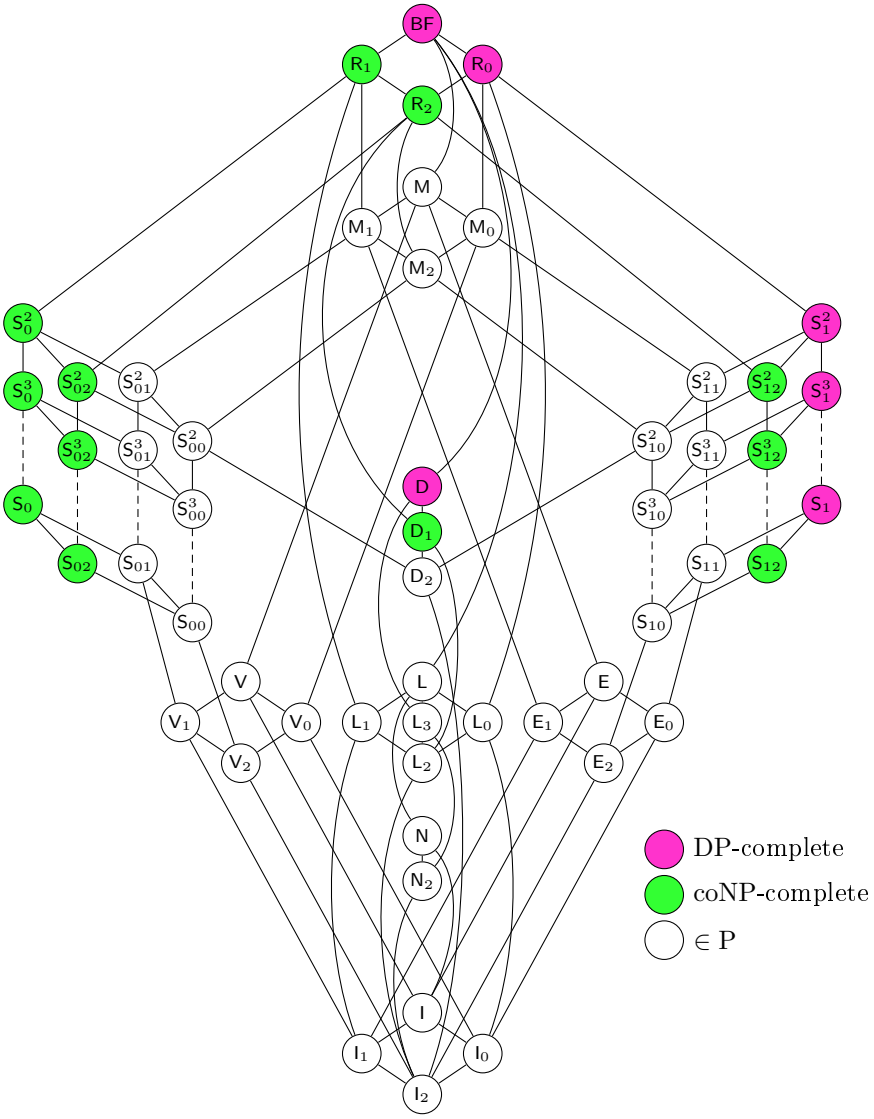

Figure 3.6: The complexity of P-ABD-CHECK(B, PQ).

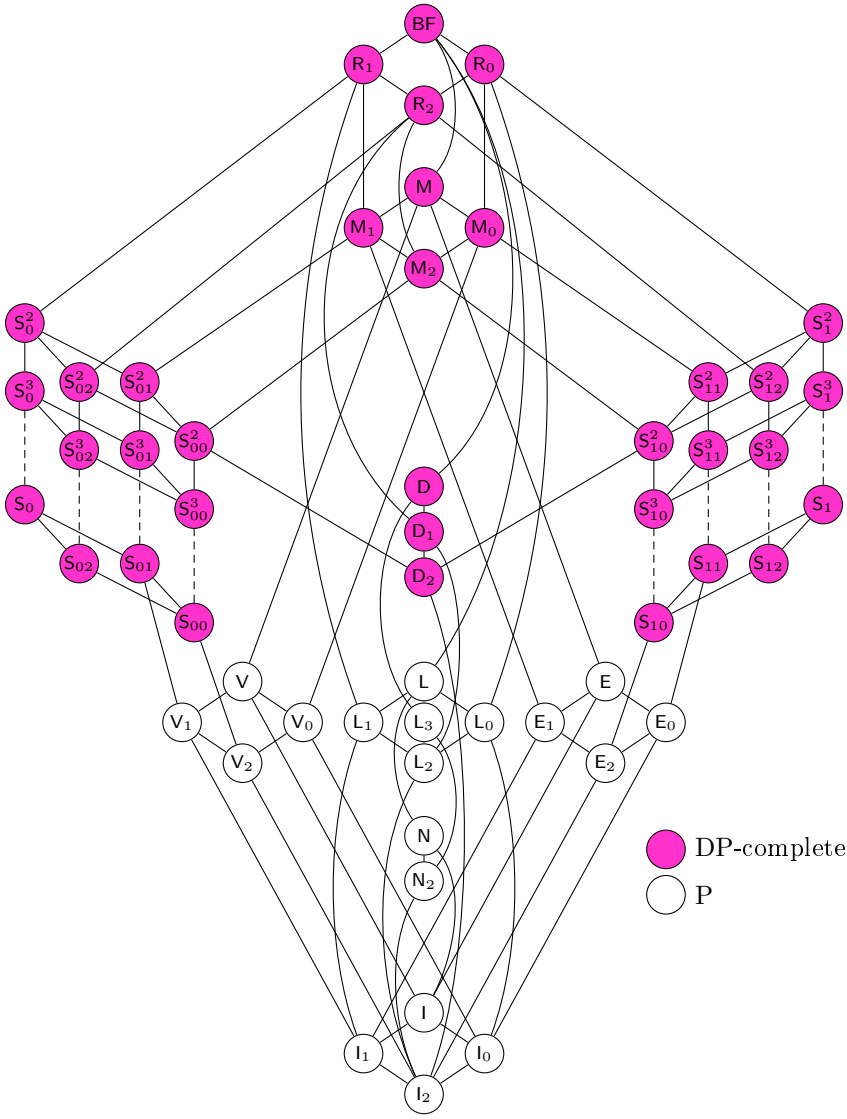

Figure 3.7: The complexity of ARG-CHECK(B).

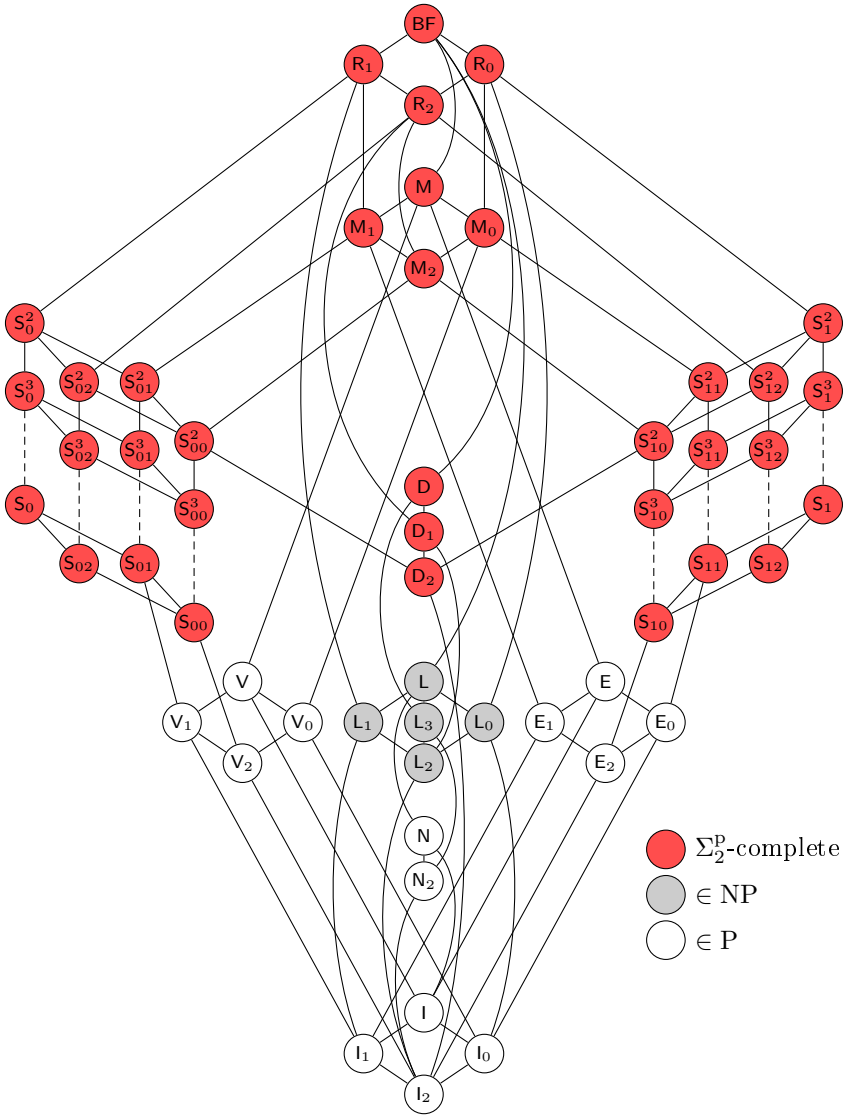

Figure 3.8: The complexity of ARG-REL(B).

Chapter 4

Complexity Classifications in Schaefer's framework

4.1 Schaefer's framework

In Schaefer's framework one considers formulæ in *generalized conjunctive normal* form. That is, starting from a CNF-formula one generalizes its clauses to applications of arbitrary Boolean relations to variables. One obtains fragments of propositional logic by restricting the set of Boolean relations S the clauses must be built of, leading to so-called *S-formulæ*. In contrast to Post's framework, no nesting is allowed. The first systematic complexity analysis of the satisfiability problem for S-formulæ was carried out by T.J. Schaefer in his 1978 paper *The complexity of Satisfiability Problems* [Sch78]. Schaefer obtained a remarkable *Dichotomy Theorem*: The satisfiability problem for S-formulæ, $\textsc{Sat}(S)$ for short, is either NP-complete or in P. The result was astonishing since it means that all infinitely many intermediate degrees between P and NP are skipped (assuming P $\neq$ NP). In the meanwhile Schaefer's approach has been applied to many problems from various contexts (e.g., Model checking for circumscription [NJ04], Default Logic [Sch07b], Equivalence and Implication [BHRV02, BBC+07, SS08, Sch07a], Abduction [CZ06, NZ08]) not restricted to pure decision problems but also to counting [CH96], [BBC+07], enumeration [CH97], and optimization (e.g., [Cre95, KS96, KSW97]).

In the following we will introduce the notion of S-formulæ formally.

Then we will discuss some standard tools from universal algebra that have often been useful in order to obtain classifications in Schaefer's Framework and that we will also build on.

4.1.1 Preliminaries

Constraint languages and S-formulæ A *logical relation* of arity k is a relation $R \subseteq \{0,1\}^k$. In this thesis we will only consider nonempty relations. By abuse of notation we do not make a difference between a relation and its predicate symbol. We will use T and F as the two unary constant relations $T = \{1\}$ and $F = \{0\}$. A *constraint*, C, is a formula $C = R(x_1, \ldots, x_k)$, where R is a logical relation of arity k and the x_i's are (not necessarily distinct) variables. For instance the two constraints $T(x)$ and $F(x)$ stand for the two unit clauses (x) and $(\neg x)$, respectively. If u and v are two variables, then $C[v/u]$ denotes the constraint obtained from C in replacing each occurrence of v by u. If V is a set of variables, then $C[V/u]$ denotes the result of substituting u to every occurrence of every variable of V in C. An assignment m of truth values to the variables *satisfies* the constraint C if $\big(m(x_1), \ldots, m(x_k)\big) \in R$. A *constraint language* S is a finite set of logical relations. An *S-formula* φ is a conjunction of constraints using only logical relations from S and is hence a quantifier-free first-order formula.

An S-formula φ is satisfied by an assignment $m : \mathrm{Vars}(\varphi) \to \{0,1\}$ if m satisfies all constraints in φ simultaneously (such a satisfying assignment is also called a *model* of φ). Assuming a canonical order on the variables we can regard models as tuples in the obvious way and we do not distinguish between a formula φ and the logical relation R_φ it defines, i.e., the relation consisting of all models of φ.

The satisfiability problem for S-formulæ is defined as follows.

Problem:	SAT(S)
Instance:	An S-formula φ.
Question:	Is φ satisfiable?

Let us illustrate the generalization to S-formulæ on an example. Define $S_1 = \{R_0, R_1, R_2, R_3\}$ by the following four relations:

$R_0 = \{0,1\}^3 \backslash \{(0,0,0)\}$ $\quad$ $R_1 = \{0,1\}^3 \backslash \{(1,0,0)\}$
$R_2 = \{0,1\}^3 \backslash \{(1,1,0)\}$ $\quad$ $R_3 = \{0,1\}^3 \backslash \{(1,1,1)\}$

An S_1-formula is given by

$$\varphi = R_0(x_1, x_2, x_3) \wedge R_2(x_2, x_3, x_4).$$

A CNF-representation of φ is given by

$$\varphi = R_0(x_1, x_2, x_3) \wedge R_2(x_2, x_3, x_4) \equiv (x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4).$$

One may observe that with the relations $\{R_0, R_1, R_2, R_3\}$ one can indeed express any clause of three variables. That is, any 3CNF-formula can be written as an S_1-formula. Therefore, the problem $\textsc{Sat}(S_1)$ describes nothing else than the problem 3SAT. Similarly, we can describe the problem 2SAT with the three relations

$R_4 = \{0,1\}^2 \backslash \{(0,0)\}$ $R_5 = \{0,1\}^2 \backslash \{(1,0)\}$ $R_6 = \{0,1\}^2 \backslash \{(1,1)\}$,

and we can describe the satisfiability problem for Horn-formuæ with clauses of length at most three by the six relations

$R_2 = \{0,1\}^3 \backslash \{(1,1,0)\}$ $R_5 = \{0,1\}^2 \backslash \{(1,0)\}$ $R_7 = \{0\}$
$R_3 = \{0,1\}^3 \backslash \{(1,1,1)\}$ $R_6 = \{0,1\}^2 \backslash \{(1,1)\}$ $R_8 = \{1\}$.

Throughout the text we refer to different types of Boolean relations following Schaefer's terminology [Sch78]. We say that a Boolean relation R is

- *Horn* (resp. *dual-Horn*) if R can be defined by a CNF formula which is Horn (resp. dual-Horn);
- *bijunctive* if it can be defined by a 2CNF formula;
- *affine* if it can be defined by an *affine* formula, i.e., conjunctions of XOR-clauses (consisting of an XOR of some variables plus maybe the constant 1) — such a formula may also be seen as a system of linear equations over GF[2];
- *width-2-affine* if it can be defined by an *affine* formula, with XOR-clauses of at most 2 variables;
- *0-valid* (resp. *1-valid*) if $R(0, \ldots, 0) = 1$ (resp. $R(1, \ldots, 1) = 1$);
- *ε-valid* if R is either 0-valid, or 1-valid or both;
- *complementive* if for all $m \in R$ we have also $\overline{m} \in R$, where $\overline{m}$ denotes the dual assignment of m defined by $\overline{m}(x) = 1 - m(x)$.

Finally a constraint language S is Horn (resp. dual-Horn, bijunctive, affine, width-2-affine, 0-valid, 1-valid, ε-valid, complementive) if every relation in S is Horn (resp. dual-Horn, bijunctive, affine, width-2-affine, 0-valid, 1-valid, ε-valid, complementive). We say that a constraint language is *Schaefer* if S is either Horn, dual-Horn, bijunctive, or affine.

4.1.2 Background from Universal Algebra

When investigating the computational complexity of problems parameterized by S-formulæ one of the most successful techniques to obtain reductions has been the application of tools from universal algebra. A Galois connection relates the expressive power of a constraint language S to its set of polymorphisms $pol(S)$ and gives a natural procedure to transform S-formulæ into S'-formulæ if $pol(S') \subseteq pol(S)$. In order to introduce this a bit more formally we first need the following definition.

Definition 4.1.1. *Let S be a constraint language. The* relational clone *(or* co-clone*) $\langle S \rangle$ is the smallest set of Boolean relations such that*

- *$\langle S \rangle$ contains the equality relation and all relations in S, and*
- *$\langle S \rangle$ is closed under primitive positive definitions, i.e., if φ is a $\langle S \rangle$-formula and $R(x_1, \ldots, x_n) \equiv \exists y_1 \ldots y_m \varphi(x_1, \ldots, x_n, y_1, \ldots y_m)$, then $R \in \langle S \rangle$.*

In other words, $\langle S \rangle$ is the set of relations that can be expressed as an $S \cup \{=\}$-formula with existentially quantified variables. Note that while a constraint language S is finite by definition, $\langle S \rangle$ is always infinite. The process of expressing a relation R in terms of an S-formula (or an $S \cup \{=\}$-formula) is also referred to as *implementing* R. The following crucial observation justifies to regard $\langle S \rangle$ as the *expressive power* of S.

Lemma 4.1.2. *If $S \subseteq \langle S' \rangle$, then* $\textsc{Sat}(S) \leq_{\mathrm{m}}^{\log} \textsc{Sat}(S')$.

Proof. Let φ be an S-formula. We transform it into an S'-formula by the following procedure:

- replace in φ every constraint by its equivalent $S' \cup \{=\}$-formula.
- delete all existential quantifiers.
- delete all equality clauses and replace variables that were forced to the same value by a chain of equality clauses, by a common new variable.

One observes that this transformation preserves the satisfiability of the formula φ. Note that the last and most costly step of the transformation is essentially an instance of the undirected graph reachability problem which is solvable in logspace [Rei05]. □

In other words, with respect to logspace-many-one reductions, the complexity of $\textsc{Sat}(S)$ does not change within a co-clone. As a consequence, one has only to consider the co-clones when classifying the complexity of $\textsc{Sat}(S)$. This will hold for any problem $\textsc{Prob}(S)$ parameterized by S-formulæ that obeys this property, i.e., that $\textsc{Prob}(S) \leq_{\mathrm{m}}^{\log} \textsc{Prob}(S')$ if $S \subseteq \langle S' \rangle$. Fortunately, all co-clones are known. In order to reveal them we need a last definition.

Definition 4.1.3. *Let $f : \{0,1\}^m \to \{0,1\}$ be a Boolean function and let $R \subseteq \{0,1\}^n$ be a Boolean relation. The function f* preserves *the relation R if for all $x_1, \ldots, x_m \in R$, where $x_i = (x_i[1], \ldots, x_i[n])$ we have*

$$\Big(f(x_1[1], \ldots, x_m[1]), \ldots, f(x_1[n], \ldots, x_m[n])\Big) \in R.$$

So f preserves R if the coordinate-wise application of f to any sequence of m vectors in R always results in a vector that is still in R. We say in this case also that R is *invariant* under f or that f is a *polymorphism* of R. We denote the set of all polymorphisms of R by $pol(R)$ and for a set of relations S we denote $pol(S)$ the set of all Boolean functions that preserve all relations in S. It is not difficult to observe that the set $pol(S)$ forms a Boolean clone. For a set of Boolean functions B we denote $inv(B)$ the set of all Boolean relations that are invariant under all functions in B. As shown first in [Gei68, BKKR69], the operators inv and pol constitute a Galois correspondence between the lattice of sets of Boolean functions and the lattice of sets of Boolean relations. This one-to-one correspondence between clones and co-clones gives us automatically the full list and structure of the co-clones: It can be read from Post's Lattice, only the inclusion relation between co-clones now points downwards. Hence, for a Boolean clone C, $inv(C)$ denotes the corresponding co-clone. It holds in particular the following.

Proposition 4.1.4. *Let S be a set of Boolean relations and B a set of Boolean functions. Then it holds:*

- $inv(pol(S)) = \langle S \rangle$

- $pol(inv(B)) = [B]$

Thus, we have $S \subseteq \langle S' \rangle$ if and only if $pol(S') \subseteq pol(S)$. Hence, Lemma 4.1.2 can equivalently be stated as follows.

Lemma 4.1.5. *If* $pol(S') \subseteq pol(S)$, *then* $\textsc{Sat}(S) \leq_{\mathrm{m}}^{\log} \textsc{Sat}(S')$.

The Galois correspondence gives us useful characterizations of the properties of relations we defined above. Indeed they describe exactly the properties of certain co-clones. The following characterizations are now part of the folklore and can be found in [Hor51, Sch78].

$pol(R) \supseteq \mathsf{V}_2$	$\Leftrightarrow$	$R \in inv(\mathsf{V}_2)$	$\Leftrightarrow$	R is dual-Horn
$pol(R) \supseteq \mathsf{E}_2$	$\Leftrightarrow$	$R \in inv(\mathsf{E}_2)$	$\Leftrightarrow$	R is Horn
$pol(R) \supseteq \mathsf{L}_2$	$\Leftrightarrow$	$R \in inv(\mathsf{L}_2)$	$\Leftrightarrow$	R is affine
$pol(R) \supseteq \mathsf{D}_1$	$\Leftrightarrow$	$R \in inv(\mathsf{D}_1)$	$\Leftrightarrow$	R is width-2-affine
$pol(R) \supseteq \mathsf{D}_2$	$\Leftrightarrow$	$R \in inv(\mathsf{D}_2)$	$\Leftrightarrow$	R is bijunctive
$pol(R) \supseteq \mathsf{N}_2$	$\Leftrightarrow$	$R \in inv(\mathsf{N}_2)$	$\Leftrightarrow$	R is complementive
$pol(R) \supseteq \mathsf{N}$	$\Leftrightarrow$	$R \in inv(\mathsf{N})$	$\Leftrightarrow$	R is compl., 0- and 1-valid
$pol(R) \supseteq \mathsf{I}$	$\Leftrightarrow$	$R \in inv(\mathsf{I})$	$\Leftrightarrow$	R is 0-valid and 1-valid
$pol(R) \supseteq \mathsf{I}_0$	$\Leftrightarrow$	$R \in inv(\mathsf{I}_0)$	$\Leftrightarrow$	R is 0-valid
$pol(R) \supseteq \mathsf{I}_1$	$\Leftrightarrow$	$R \in inv(\mathsf{I}_1)$	$\Leftrightarrow$	R is 1-valid
$pol(R) \supseteq \mathsf{I}_2$	$\Leftrightarrow$	$R \in inv(\mathsf{I}_2)$	$\Leftrightarrow$	R is any relation

4.1.3 Complexity Classifications in Schaefer's Framework and the Galois connection

Schaefer's famous dichotomous classification of the satisfiability problem $\textsc{Sat}(S)$ originally has been proved *by hand*. This means, he did not refer to the existence of implementation results thanks to the Galois connection, but explicitly gave the implementations. Later, much shorter and more elegant proofs have been established making use of the Galois connection [BCRV04, CV08]. Among the above mentioned problems that have been classified in Schaefer's Framework, not all allow to prove easily the crucial property of Lemma 4.1.2, i.e., that

$$\textsc{Prob}(S) \leq_{\mathrm{m}}^{\log} \textsc{Prob}(S') \text{ if } S \subseteq \langle S' \rangle\,. \tag{4.1}$$

The procedure in the proof of Lemma 4.1.2 of transforming S-formulæ into S'-formulæ preserves satisfiability. But it changes the number of models and variables in a quite unpredictable way, and therefore it will

not preserve more complicated aspects such as the weight or questions that regard inclusions between sets of models such as implication. However, in cases where the crucial property (4.1) could not easily be proved and a complete classification for a problem PROB was obtained *by hand*, it often could be read afterwards from the complete classification that (4.1) indeed holds. In these cases it has become common to say that the Galois connection holds *a posteriori*, where in the case where (4.1) can be used to obtain the classification, the Galois connection is said to hold *a priori*. Of course this has not to be taken as a mathematical definition: in both cases (4.1) holds.

There exist refined Galois connections established by Schnoor [SS08] avoiding the introduction of existential quantifiers (or even equality constraints) considering partial clones and partial polymorphisms. Such a connection is thus suited to preserve more complicated properties than satisfiability. When such a refined connection is usually easily shown to hold even *a priori*, the difficulties of the successful application lie here in the fact that partial clones are much less understood than usual clones: Post's Lattice draws a complete picture of the structure of usual clones, where the lattice of partial clones is not completely known.

The Galois connection has successfully been applied to abduction [NZ08] which compromises implication. Though only on variants where the manifestation (i.e., the conclusion of an implication) has a fixed form, i.e., the manifestation does not depend on the constraint language S. In this case the transformation of Lemma 4.1.2 applies only to the premise of the implication and works out well, delivering the Galois connection even *a priori*. To the authors knowledge, classifications of the variant of abduction where also the manifestation is an S-formula have not yet been attacked. In this case the Galois connection seems not to hold *a priori*, though it might hold *a posteriori*.

We will classify in Section 4.2 the argumentation problems ARG and ARG-CHECK in Schaefer' Framework. For these problems the Galois connection seems not to hold *a priori* since here we require the claim (i.e., the conclusion of an implication) to be an S-formula which lets fail the transformation of Lemma 4.1.2. Though, as we will see, it will hold *a posteriori*. In Section 4.3 we will consider the problem of enumerating all models of an S-formula by non-decreasing weight. For the hardness results we will establish in this context, we will have to show hardness of problems involving the weight of models, and therefore here neither the Galois connection seems not to hold *a priori*.

In our complexity classifications we will obtain hardness results either *by hand* (ARG and enumeration) or we will establish in the case of ARG-CHECK a Galois connection of an intermediate problem (where the claim's formula type is independent from the knowledge base's one) for constraint languages that are not Schaefer (here equality constraints may be expressed). We will state in the following section important implementation results. That is, we give conditions stating when certain key relations can be expressed by a constraint language without the introduction of existential quantifiers and equality constraints. These results will be crucial in order to obtain reductions.

Before this, we briefly state the classifications for the satisfiability problem SAT, its variant of nontrivial satisfiability

Problem:	$\text{SAT}^*(S)$
Instance:	An S-formula φ.
Question:	Does there exist a nontrivial model of φ, i.e., a model m such that $m \neq \vec{0}$ and $m \neq \vec{1}$?

the implication problem

Problem:	$\text{IMP}(S)$
Instance:	Two S-formulæ φ, ψ.
Question:	Does hold $\varphi \models \psi$?

and the positive abduction problem with a positive literal manifestation

Problem:	$\text{P-ABD}(S, \text{PQ})$
Instance:	$\mathcal{I} = (\varphi, A, q)$, where φ an S-formula, $A \subseteq \text{Vars}(\varphi)$ and q a variable
Question:	Does there exist an explanation $E \subseteq A$?

Our classifications will be built on these results.

Theorem 4.1.6. *([Sch78]) Let S be a constraint language. The satisfiability problem for S-formulæ, $\text{SAT}(S)$, is*

1. *in* P *if S is* Schaefer, 0-valid *or* 1-valid,
2. NP-*complete otherwise.*

Theorem 4.1.7. *([CH97]) Let S be a constraint language. The nontrivial satisfiability problem for S-formulæ, $\text{SAT}^*(S)$, is*

1. *in* P *if* S *is* Schaefer,

2. NP*-complete otherwise.*

Theorem 4.1.8. *([SS08]) Let* S *be a constraint language. The implication problem for* S*-formulæ,* $\textsc{Imp}(S)$*, is*

1. *in* P *if* S *is* Schaefer,

2. NP*-complete otherwise.*

Theorem 4.1.9. *([NZ08]) Let* S *be a constraint language. The positive abduction problem for* S*-formulæ with a positive literal manifestation,* $\text{P-}\textsc{Abd}(S, \text{PQ})$*, is*

1. Σ_2^{p}*-complete if* $inv(\mathsf{I}_0) \subseteq \langle S \rangle \subseteq inv(\mathsf{I}_2)$ *or* $inv(\mathsf{N}_2) \subseteq \langle S \rangle \subseteq inv(\mathsf{I}_2)$*,*

2. coNP*-complete if* $inv(\mathsf{N}) \subseteq \langle S \rangle \subseteq inv(\mathsf{I}_1)$*,*

3. NP*-complete if* $inv(\mathsf{E}_0) \subseteq \langle S \rangle \subseteq inv(\mathsf{E}_2)$*,*

4. *in* P *otherwise.*

From the last theorem we will use in particular the Σ_2^{p}-complete case when $inv(\mathsf{N}_2) \subseteq \langle S \rangle \subseteq inv(\mathsf{I}_2)$. Equivalently stated, this is when S is neither Schaefer, nor 1-valid, nor 0-valid. The complexity of $\text{P-}\textsc{Abd}(S, \text{PQ})$ is visualized in Figure 4.3 on page 117.

4.1.4 Implementation results

We concentrate here important implementation results that will be of technical but crucial use in this chapter. In the proofs R (resp. R') will denote a relation of arity k (resp. k') and $V = \{x_1, \ldots, x_k\}$ (resp. $V' = \{y_1, \ldots, y_{k'}\}$) a set of k (resp. k') distinct variables. We denote by C the $\{R\}$-constraint $C = R(x_1, \ldots, x_k)$ and by C' the $\{R'\}$-constraint $C' = R'(y_1, \ldots, y_{k'})$. The first four items of the following lemma can certainly be found in the literature as folklore.

Lemma 4.1.10. *Let* S *be a set of relations.*

1. *If* S *is* 0-valid *and not* 1-valid *(resp.* 1-valid *and not* 0-valid*), then there exists an* S*-formula equivalent to* $F(x)$ *(resp.* $T(x)$*).*

2. *If S is* complementive, *but neither* 1-valid, *nor* 0-valid, *then there exists an S-formula equivalent to* $(x \neq y)$.

3. *If S is not* complementive, *but* 1-valid *and* 0-valid, *then there exists an S-formula equivalent to* $(x \rightarrow y)$.

4. *If S is neither* complementive, *nor* 1-valid, *nor* 0-valid, *then there exists an S-formula equivalent to* $(x \wedge \neg y)$.

5. *If S is* 0-valid *and* affine *but neither* Horn *nor* 1-valid, *then there exists an S-formula equivalent to* $\neg w \wedge (x \oplus y \oplus z = 0)$.

6. *If S is* 0-valid, 1-valid, affine *but not* Horn, *then there exists an S-formula equivalent to* $(w \oplus x \oplus y \oplus z = 0)$.

7. *If S is* 0-valid, dual-Horn *but neither* affine *nor* 1-valid, *then there exists an S-formula equivalent to* $\neg t \wedge (u \rightarrow v)$.

Proof. 1. Let $R \in S$ be a relation that is 0-valid and not 1-valid (resp. 1-valid and not 0-valid). Obviously $C[V/x]$ is equivalent to $F(x)$ (resp. $T(x)$).

2. Let $R \in S$ be a relation which is complementive, but neither 1-valid nor 0-valid. Choose any $m \in R$. Let $V_1 = \{x \mid x \in V,\ m(x) = 1\}$ and $V_0 = \{x \mid x \in V,\ m(x) = 0\}$. Observe that both sets are nonempty. Consider the $\{R\}$-constraint $M(x, y) = C[V_1/x, V_0/y]$. Its set of models is $\{01, 10\}$, therefore it is equivalent to $(x \neq y)$.

3. Let $R \in S$ be a relation which is not complementive, but 1-valid and 0-valid. Since R is not complementive there exists an $m \in R$ such that $\overline{m} \notin R$. Let $V_1 = \{x \mid x \in V,\ m(x) = 1\}$ and $V_0 = \{x \mid x \in V,\ m(x) = 0\}$. Observe that both sets are nonempty. Consider the $\{R\}$-constraint $M(x, y) = C[V_0/x, V_1/y]$. Its set of models is $\{00, 11, 01\}$, therefore it is equivalent to $(x \rightarrow y)$.

4. We know that in S there is at least one relation that is not 1-valid and at least one relation that is not 0-valid.

   Suppose first that there is a relation R_0 which is not 1-valid but 0-valid and a relation R_1 which is not 0-valid but 1-valid. With these constraints one can express respectively $\mathrm{T}(x)$ and $\mathrm{F}(y)$, according to the first item. Their conjunction provides the desired formula.

Otherwise we know that there is a relation $R \in S$ that is neither 0-valid nor 1-valid. Let $m \in R$, $V_1 = \{x \mid x \in V,\ m(x) = 1\}$ and $V_0 = \{x \mid x \in V,\ m(x) = 0\}$. Observe that both sets are nonempty. Consider the $\{R\}$-constraint $N(x, y) = C[V_1/x, V_0/y]$. Its set of models is nonempty and contained in $\{01, 10\}$. If its set of models is $\{01\}$ or $\{10\}$, we are done. Therefore, suppose in the following that $mod(N(x, y)) = \{01, 10\}$, that is $N(x, y) \equiv (x \neq y)$.

Further we know that there exists a relation $R' \in S$ which is not complementive. Since R' is not complementive there exists an $m' \in R'$ such that $\overline{m'} \notin R'$. Let $V'_1 = \{x \mid x \in V',\ m'(x) = 1\}$ and $V'_0 = \{x \mid x \in V',\ m'(x) = 0\}$. Consider the $\{R'\}$-constraint $M(x, y) = C'[V'_1/x, V'_0/y]$. Its set of models contains 10 and is included in $\{10, 00, 11\}$. Finally it is easy to see that the $\{R, R'\}$-formula $M(x, y) \wedge N(x, y)$ is equivalent to $(x \wedge \neg y)$.

5. Let $R \in S$ be a relation which is 0-valid, affine and not Horn. Since R is not Horn there exist m_1 and m_2 in R such that $m_1 \wedge m_2 \notin R$. Since R is 0-valid and affine, we have $m_1 \oplus m_2 \in R$. For $i, j \in \{0, 1\}$, set $V_{i,j} = \{x \mid x \in V, m_1(x) = i \wedge m_2(x) = j\}$. Observe that $V_{0,1} \neq \emptyset$ (respectively, $V_{1,0} \neq \emptyset$), otherwise $m_1 \wedge m_2 = m_2$ (respectively, $m_1 \wedge m_2 = m_1$), contradicting the fact that $m_1 \wedge m_2 \notin R$. Moreover $V_{1,1} \neq \emptyset$, otherwise $m_1 \wedge m_2 = \vec{0}$, a contradiction. Consider the $\{R\}$-constraint

$$M(w, x, y, z) = C[V_{0,0}/w,\ V_{0,1}/x,\ V_{1,0}/y,\ V_{1,1}/z].$$

According to the above remark the three variables x, y and z effectively occur in this constraint. Let us examine the set of models of M assigning 0 to w: it contains 0011 (since $m_1 \in R$), 0101 (since $m_2 \in R$), 0110 (since $m_1 \oplus m_2 \in R$) and 0000 (since R is 0-valid). But it does not contain 0001 (since by assumption $m_1 \wedge m_2 \notin R$). Thus it does not contain 0111 either. Indeed, otherwise it would contain $0011 \oplus 0101 \oplus 0111$ (since R is affine), which is equivalent to 0001, a contradiction. From this one can prove that it contains neither 0010 nor 0100 (since $0000 \oplus 0011 \oplus 0010 = 0001$ and $0110 \oplus 0101 \oplus 0100 = 0111$). Let us consider

$$\phi(w, x, y, z) = F(w) \wedge M(w, x, y, z).$$

The $\{R, F\}$-formula ϕ is equivalent to $\neg w \wedge (x \oplus y \oplus z = 0)$ and by the first item expressible by an S-formula.

6. Let $R \in S$ a relation that is not Horn. Observe that R, which is 0-valid, 1-valid and affine, is necessarily complementive, i.e., for all $m \in R$ we have also $\vec{1} \oplus m \in R$. We can mimic the analysis made in the previous item and consider

$$M(w, x, y, z) = C[V_{0,0}/w,\, V_{0,1}/x,\, V_{1,0}/y,\, V_{1,1}/z].$$

Thus, the formula $\phi(w, x, y, z) = M(w, x, y, z)$ verifies $\phi(w, x, y, z) \equiv (w \oplus x \oplus y \oplus z = 0)$.

7. Note that by the first item $F(\cdot)$ is expressible by an S-formula. Let $R \in S$ a relation which is 0-valid, dual-Horn and not affine. Since R is 0-valid and not affine there exist two distinct tuples m_1 and m_2 in R such that $m_1 \oplus m_2 \notin R$. Since R is dual-Horn, we have $m_1 \vee m_2 \in R$. For $i, j \in \{0, 1\}$, let $V_{i,j} = \{x \mid x \in V, m_1(x) = i \wedge m_2(x) = j\}$. Observe that $V_{1,1} \neq \emptyset$, otherwise $m_1 \vee m_2 = m_1 \oplus m_2$, a contradiction. Moreover, since $m_1 \neq m_2$ either $V_{0,1}$ or $V_{1,0}$ is nonempty. Suppose first that they are both nonempty. Consider the $\{R\}$-constraint

$$M(w, x, y, z) = C[V_{0,0}/w,\, V_{0,1}/x,\, V_{1,0}/y,\, V_{1,1}/z].$$

The three variables x, y and z effectively appear in this constraint. Let us examine the set of models of M assigning 0 to w: it contains 0011 (since $m_1 \in R$), 0101 (since $m_2 \in R$), 0111 (since $m_1 \vee m_2 \in R$) and 0000 (since R is 0-valid), but does not contain 0110 (since by assumption $m_1 \oplus m_2 \notin R$). The membership of 0100, 0010, 0001 is open:

- If it does not contain 0100, then consider the formula $\phi(t, u, v) := F(t) \wedge M(t, u, v, v)$. Its set of models is $\{001, 011, 000\}$ and therefore, $\phi(t, u, v) \equiv \neg t \wedge (u \rightarrow v)$.
- If it contains 0100, then it does not contain 0010. Indeed otherwise, since R is dual-Horn it would also contain 0110, which provides a contradiction. Thus consider the formula $\phi(t, u, v) := F(t) \wedge M(t, v, u, v)$. Its set of models is $\{001, 011, 000\}$ and therefore, $\phi(t, u, v) \equiv \neg t \wedge (u \rightarrow v)$.

If for instance $V_{0,1} = \emptyset$, then one has to consider

$$M(w, y, z) = C[V_{0,0}/w, V_{1,0}/y, V_{1,1}/z].$$

In this case $\phi(t,u,v) := F(t) \wedge M(t,u,v)$ is equivalent to $\neg t \wedge (u \to v)$. □

We have now a lemma stating that in the non-Schaefer case the equality constraint can be expressed using existentially quantified variables. This result will be an important tool when classifying the problem ARG-CHECK.

Lemma 4.1.11. *Let S be a constraint language. If S is not* Schaefer*, then there exists an S-formula $\varphi(x,y,\vec{z})$ such that $\exists \vec{z}\varphi(x,y,\vec{z})$ is equivalent to $(x=y)$.*

Proof. We make a case distinction according to whether S is 0-valid and / or 1-valid.

First let S be 0-valid and 1-valid. Since S is not Schaefer it contains at least one nontrivial relation R, that is a 0-valid and 1-valid relation $R \in S$ such that there is an $m \notin R$. Let $V_1 = \{x \mid x \in V, m(x) = 1\}$ and $V_0 = \{x \mid x \in V, m(x) = 0\}$. Observe that both sets are nonempty. Consider the $\{R\}$-constraint $M(x,y) = C[V_0/x, V_1/y]$. Its set of models contains $\{00, 11\}$ and does not contain $\{01\}$. Therefore $M(x,y) \wedge M(y,x)$ is equivalent to $(x=y)$.

Second let S be 0-valid but not 1-valid (the case 1-valid but not 0-valid can be treated analogously). Since S is not Schaefer there exists a relation $R \in S$ that is not Horn. Thus there exist m_1 and m_2 in R such that $m_1 \wedge m_2 \notin R$. For $i,j \in \{0,1\}$, set $V_{i,j} = \{x \mid x \in V,\ m_1(x) = i \wedge m_2(x) = j\}$. Observe that the sets $V_{0,1}, V_{1,0}, V_{1,1}$ are not empty since otherwise $m_1 \wedge m_2 = m_2 \in R$ or $m_1 \wedge m_2 = m_1 \in R$ or $m_1 \wedge m_2 = \vec{0} \in R$, respectively. Consider the $\{R\}$-constraint $M(u,v,x,y) = C[V_{0,0}/u, V_{0,1}/v, V_{1,0}/x, V_{1,1}/y]$. It contains $\{0011, 0101, 0000\}$ (since R contains m_1 and m_2 and is 0-valid) but it does not contain 0001 (since $m_1 \wedge m_2 \notin R$). Consider the $\{R,F\}$-formula

$$M'(x,y,f) = M(f,f,x,y) \wedge M(f,f,y,x) \wedge F(f).$$

One verifies that it is equivalent to $F(f) \wedge (x=y)$ and hence $\exists f\, M'(x,y,f) \equiv (x=y)$. Note that by the first item of Lemma 4.1.10 $F(\cdot)$ is expressible by an S-formula and therefore so is M'.

At last let S be neither 0-valid nor 1-valid. It suffices here to show that we are able to express disequality, $(x \neq y)$, since $(x=y) \equiv \exists z(x \neq z) \wedge (z \neq y)$. If S is complementive we conclude with Lemma 4.1.10, second item. Therefore suppose now that S is not complementive. Let $R \in S$ be a relation that is not Horn. Thus there are $m_1, m_2 \in R$ such that $m_1 \wedge m_2 \notin$

R. We define $V_{i,j}$ as in the previous case and conclude analogously that $M_1(u,x,y,v) = C[V_{0,0}/u, V_{0,1}/x, V_{1,0}/y, V_{1,1}/v]$ contains $\{0011, 0101\}$ but not 0001. Further, let $R' \in S$ be a relation that is not dual-Horn. Thus there are $m_3, m_4 \in R'$ such that $m_3 \vee m_4 \notin R'$. For $i,j \in \{0,1\}$, set $V'_{i,j} = \{x \mid x \in V',\ m_3(x) = i \wedge m_4(x) = j\}$. Observe that the sets $V'_{0,1}$ and $V'_{1,0}$ are nonempty. Set $M_2(u,x,y,v) = C'[V'_{0,0}/u, V'_{0,1}/x, V'_{1,0}/y, V'_{1,1}/v]$. It contains $\{0011, 0101\}$ (since $m_3, m_4 \in R'$) but it does not contain 0111 (since $m_3 \vee m_4 \notin R'$). Finally consider the $\{R, R', (t \wedge \neg f)\}$-formula

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

One verifies that it is equivalent to $(x \neq y) \wedge (t \wedge \neg f)$. Due to the fourth item of Lemma 4.1.10 $(t \wedge \neg f)$ is expressible as an S-formula, and therefore so is $M(x,y,f,t)$. Finally observe that $\exists t, f\ M(x,y,f,t)$ is equivalent to $(x \neq y)$. □

4.2 The complexity of Argumentation

We will now consider the argumentation problems ARG and ARG-CHECK with a parameter S indicating the constraint language the formulæ must be built of. The resulting parameterized problems are as follows.

Problem:	ARG(S)
Instance:	$\mathcal{I} = (\Delta, \alpha)$, where Δ a set of S-formulæ and α an S-formula.
Question:	Does there exist Φ such that (Φ, α) is an argument in Δ?

Problem:	ARG-CHECK(S)
Instance:	$\mathcal{I} = (\Phi, \alpha)$, where Φ a set of S-formulæ and α an S-formula.
Question:	Is (Φ, α) an argument?

4.2.1 Complexity of the Existence Problem

Recall that the argument-existence problem ARG in general is Σ_2^{p}-complete. In Post's Framework we identified, besides polynomial cases, coNP-complete cases. The existence of NP-complete cases is not clear since we only could prove NP-membership for some affine cases, without a lower bound. In

Schaefer's Framework we obtain a complete classification which is clearly tetrachotomous.

Proposition 4.2.1. *Let S be a constraint language which is* Schaefer*, but neither* 1-valid*, nor* 0-valid*. Then* $\textsc{Arg}(S)$ *is* NP*-complete.*

Proof. The NP-membership follows from the fact that since S is Schaefer $\textsc{Sat}(S)$ and $\textsc{Imp}(S)$ are in P and thus a guessed argument can be verified in polynomial time. For the hardness proof we make a case distinction according to whether S is complementive or not. Suppose first that every relation in S is complementive. We prove the following sequence of reductions:

$$3\textsc{Sat} \leq_{\mathrm{m}}^{\log} \textsc{Arg}(\{x \neq y\}) \leq_{\mathrm{m}}^{\log} \textsc{Arg}(S).$$

The last reduction holds following Lemma 4.1.10, second item. For the first reduction let $\varphi = \bigwedge_{i=1}^{k} C_i$ be an instance of 3Sat where $\mathrm{Vars}(\varphi) = \{x_1, \ldots, x_n\}$. Let $c_1, \ldots, c_k$, $x'_1, \ldots, x'_n$, f be fresh variables. We map φ to (Δ, α) where

$$\begin{aligned}
\Delta = {} & \textstyle\bigcup_{j=1}^{n} \{x_j \neq f, x'_j \neq f\} \cup \\
& \{\textstyle\bigwedge_{j=1}^{n} (x_j \neq x'_j)\} \cup \\
& \textstyle\bigcup_{i,j} \{x_j \neq c_i \mid \neg x_j \in C_i\} \cup \\
& \{x'_j \neq c_i \mid x_j \in C_i\}, \\
\alpha = {} & \textstyle\bigwedge_{i=1}^{k} (c_i \neq f) \wedge \bigwedge_{j=1}^{n} (x_j \neq x'_j).
\end{aligned}$$

One can check that φ is satisfiable if and only if there exists a $\Phi \subseteq \Delta$ such that (Φ, α) is an argument. Intuitively, x'_j plays the role of $\neg x_j$, for every j at most one of the constraints $x_j \neq f$ and $x'_j \neq f$ can appear in the support of an argument, thus allowing to identify true literals, while for every i the constraints $x_j \neq c_i$ and $x'_j \neq c_i$ are used to certify that the clause C_i is satisfied.

Second, let us suppose that S is not complementive. We prove the following:

$$\textsc{Pos-2in3-Sat} \leq_{\mathrm{m}}^{\log} \textsc{Arg}(\{x \wedge \neg y\}) \leq_{\mathrm{m}}^{\log} \textsc{Arg}(S).$$

The last reduction follows by Lemma 4.1.10, fourth item. For the first one we start from the NP-complete problem Pos-2in3-Sat in which the instance is a set of positive 3-clauses and the question is to decide whether there exists a truth assignment such that each clause contains exactly two

true variables. Let $\varphi = \bigwedge_{i=1}^{k}(x_i \vee y_i \vee z_i)$ be an instance of the first problem and let $c_1, \ldots, c_k$, f be fresh variables. We map φ to (Δ, α) where

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

Observe that every formula in Δ can be written as $\{x \wedge \neg y\}$-formula. One can check that there is a truth assignment such that each clause C_i contains exactly two variables set to true if and only if (Δ, α) admits an argument. Observe that for every i such an argument contains exactly one of the three formulæ involving c_i, thus providing a desired satisfying assignment. □

Proposition 4.2.2. *Let S be a constraint language which is neither* Schaefer, *nor* 1-valid, *nor* 0-valid. *Then* $\textsc{Arg}(S)$ *is* Σ_2^{p}*-complete.*

Proof. We give a reduction from P-$\textsc{Abd}(S, \mathrm{PQ})$ which is Σ_2^{p}-complete according to [NZ08]. We make a case distinction according to whether S is complementive or not.

Suppose first that every relation in S is complementive. We show:

$$\text{P-}\textsc{Abd}(S, \mathrm{PQ}) \leq_{\mathrm{m}}^{\log} \textsc{Arg}(S \cup \{x \neq y\}) \leq_{\mathrm{m}}^{\log} \textsc{Arg}(S).$$

The last reduction follows by Lemma 4.1.10, second item. For the first one we map (φ, A, q) an instance of P-$\textsc{Abd}(S, \mathrm{PQ})$ to (Δ, α), where we introduce a fresh variable f and define

$$\begin{aligned}\Delta &= \{\varphi\} \cup \{(h \neq f) \mid h \in A\}, \\ \alpha &= (q \neq f).\end{aligned}$$

To see that P-$\textsc{Abd}(S, \mathrm{PQ})$ has solutions if and only if $\textsc{Arg}(S \cup \{x \neq y\})$ has solutions observe that all formulæ occurring in the transformed instance are complementive: it suffices therefore to observe correctness for $(\Delta[f/0], \alpha[f/0])$.

In the case where S is not complementive we show

$$\text{P-}\textsc{Abd}(S, \mathrm{PQ}) \leq_{\mathrm{m}}^{\log} \textsc{Arg}(S \cup \{x \wedge \neg y\}) \leq_{\mathrm{m}}^{\log} \textsc{Arg}(S).$$

The last reduction follows by Lemma 4.1.10, fourth item. For the first one we map (φ, A, q) an instance of the first problem to (Δ, α), where we introduce two fresh variables t, f and define

$$\Delta = \{\varphi\} \cup \{h \wedge \neg f \mid h \in A\} \cup \{t \wedge \neg f\},$$
$$\alpha = (q \wedge \neg f) \wedge (t \wedge \neg f).$$

Observe that Δ is made of S- and $\{x \wedge \neg y\}$-formulæ. It is easy to check that (φ, A, q) is a positive instance of the abduction problem if and only if there exists a support for α in Δ: If E is an explanation for (φ, A, q), define $\Phi := \{\varphi\} \cup \{h \wedge \neg f \mid h \in E\} \cup \{t \wedge \neg f\}$ to get a (not necessarily minimal) support for α. Conversely, if $\Phi \subseteq \Delta$ is a support for α, define $E := \{h \in A \mid (h \wedge \neg f) \in \Phi\}$ to get an explanation for (φ, A, q). □

We are now in a position to state the classification. It is visualized in Figure 4.2 on page 116.

Theorem 4.2.3. *Let S be a constraint language. The argument-existence problem* Arg(S) *is*

1. Σ_2^{p}*-complete if S is not* Schaefer *and not ε-valid,*
2. coNP*-complete if S is not* Schaefer *and ε-valid,*
3. NP*-complete if S is* Schaefer *and not ε-valid,*
4. *in* P *if S is* Schaefer *and ε-valid.*

The same classification holds for the argument-dispensability problem, Arg-Disp.

Proof. 1. Follows from Proposition 4.2.2.

2. One easily observes that, due to the fact that S is 1-valid or 0-valid, an instance (Δ, α) of Arg(S) has a solution if and only if Δ implies α. This condition can be checked in coNP since Imp(S) is in coNP.

   To prove coNP-hardness we give a reduction from the coNP-complete problem Imp(S). We map (φ, ψ) an instance of the first problem to $(\{\varphi\}, \psi)$.

3. Follows from Proposition 4.2.1.

4. One easily observes that, due to the fact that S is 1-valid or 0-valid, an instance (Δ, α) of $\textsc{Arg}(S)$ has a solution if and only if Δ implies α. This condition can be checked in polynomial time since S is Schaefer and thus $\textsc{Imp}(S)$ is in P.

□

Note that Arg-Disp obeys the same classification as Arg since the translations we gave in Post's Framework on page 69 still apply (the translations do not change the type of the formulæ from Δ or α).

4.2.2 Complexity of the Verification Problem

As mentioned previously, in order to completely classify the verification problem $\textsc{Arg-Check}(S)$, we will now introduce a technical variant, $\textsc{Arg-Check}(S_1, S_2)$, in which we can differentiate the restrictions put on the knowledge base from the ones put on the claim. Thus an instance (Δ, α) of $\textsc{Arg-Check}(S_1, S_2)$ is made of Δ a set of S_1-formulæ and α an S_2-formula. This way we may apply a Galois connection (the transformation of Lemma 4.1.2) to the knowledge base Δ without applying it to the claim, where existential quantifiers cause problems. Note that in the case of the verification of an argument, where we are given (Φ, α) with $\Phi \subseteq \Delta$, identifying variables that are connected by equality constraints does not necessarily preserve minimality of the support. It is therefore not clear how to get rid of equality constraints in the knowledge base.

As a consequence the Galois connection can be of use, but only if applied to the knowledge base (and not the claim), and in the non-Schaefer case in which equality constraints can be expressed according to Lemma 4.1.11. We state this formally in the following Lemma.

Lemma 4.2.4. *Let S', S be two constraint languages. If S is not* Schaefer *and $S' \subseteq \langle S \rangle$ then*

$$\textsc{Arg-Check}(S', S) \leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S).$$

Proof. Let (Δ, α) be an instance of the first problem, where $\Delta = \{\delta_i \mid i \in I\}$. The claim α remains unchanged. In Δ replace each δ_i by its equivalent $S \cup \{=\}$-formula with existential quantifiers. Since S is not Schaefer the equality relation is expressible as an S-formula with existential quantifiers according to Lemma 4.1.11. So in a second step replace the equality constraints by their S-representations. Finally drop all existential quantifiers. □

We will apply this tool together with the following expressibility results we get from the Galois correspondence.

Lemma 4.2.5. *Let S be a constraint language which is neither Schaefer nor complementive (that is, $S \not\subseteq inv(\mathsf{L}_2)$, $S \not\subseteq inv(\mathsf{E}_2)$, $S \not\subseteq inv(\mathsf{V}_2)$, $S \not\subseteq inv(\mathsf{D}_2)$, and $S \not\subseteq inv(\mathsf{N}_2)$, or, equivalently, $inv(\mathsf{I}) \subseteq \langle S \rangle$).*

- *If S is both* 0-valid *and* 1-valid *(i.e., $S \subseteq inv(\mathsf{I})$), then $\langle S \rangle$ contains all relations that are both* 0-valid *and* 1-valid *(since $\langle S \rangle = inv(\mathsf{I})$),*
- *else, if S is* 0-valid *(i.e., $S \subseteq inv(\mathsf{I}_0)$), then $\langle S \rangle$ contains all relations that are* 0-valid *(since $\langle S \rangle = inv(\mathsf{I}_0)$),*
- *else, if S is* 1-valid *(i.e., $S \subseteq inv(\mathsf{I}_1)$), then $\langle S \rangle$ contains all relations that are* 1-valid *(since $\langle S \rangle = inv(\mathsf{I}_1)$),*
- *else $\langle S \rangle$ contains all relations (since $\langle S \rangle = inv(\mathsf{I}_2)$).*

Our main theorem for Arg-Check(S) is as follows. The classification is visualized in Figure 4.1 on page 115.

Theorem 4.2.6. *Let S be a constraint language. The argument-validity problem* Arg-Check(S) *is*

1. *in* P *if S is* Schaefer,
2. DP-*complete if S is not* Schaefer.

Recall that the argument verification problem is in DP since there are are languages A, B with $A \in$ NP and $B \in$ coNP such that Arg-Check $= A \cap B$.

$$A = \{(\Delta, \Phi, \alpha) \mid \Phi \text{ is satisfiable}, \forall \varphi \in \Phi : \Phi \setminus \{\varphi\} \not\models \alpha\};$$
$$B = \{(\Delta, \Phi, \alpha) \mid \Phi \models \alpha\}.$$

We split the proof of Theorem 4.2.6 into three propositions.

Proposition 4.2.7. *Let S be a constraint language that is* Schaefer. *Then* Arg-Check(S) *is in* P.

Proof. Use that Sat(S) and Imp(S) are in P. □

Proposition 4.2.8. *Let S be a constraint language which is neither* Schaefer *nor* complementive. *Then* Arg-Check(S) *is* DP-*complete.*

Proof. For the hardness we give a reduction from Critical-3Sat, a DP-complete problem according to [PW88]. We we make a case distinction according to whether S is 0-valid and/or 1-valid. Throughout the proof we denote by $\varphi = \bigwedge_{j \in J} C_j$ an instance of Critical-3Sat.

Suppose first that S is both 0-valid and 1-valid. We prove for some well-chosen constraint language $S' \subseteq \langle S \rangle$ the following sequence of reductions:

$$\begin{aligned} \textsc{Critical-3Sat} &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S', \{x \to y\}) \\ &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S', S) \\ &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S). \end{aligned}$$

For the first reduction we associate with φ the instance (Φ, α) where $\Phi = \{C_j \vee (f \to t) \mid j \in J\}$ and $\alpha = (f \to t)$ with f, t fresh variables. It is easy to see that φ is a critical instance if and only if (Φ, α) is an argument. The second reduction follows by Lemma 4.1.10, third item. The third one follows from Lemma 4.2.4, observing that the formulæ in Φ are constraints built upon a finite set S' of relations which are 1-valid and 0-valid and thus $S' \subseteq \langle S \rangle$ according to Lemma 4.2.5.

Suppose now that S is 1-valid and not 0-valid. The other case (0-valid and not 1-valid) can be treated analogously. We show for some well-chosen constraint language $S' \subseteq \langle S \rangle$ that

$$\begin{aligned} \textsc{Critical-3Sat} &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S', \{\mathrm{T}\}) \\ &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S', S) \\ &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S). \end{aligned}$$

For the first reduction we associate with φ the instance (Φ, α) where $\Phi = \{C_j \vee u \mid j \in J\}$ and $\alpha = u$, where u is a fresh variable. It is easy to see that φ is a critical instance if and only if (Φ, α) is an argument. The second reduction follows by Lemma 4.1.10, first item. The third reduction follows from Lemma 4.2.4, observing that the formulæ in Φ are constraints built upon a finite set S' of relations which are 1-valid and thus $S' \subseteq \langle S \rangle$ according to Lemma 4.2.5.

Finally suppose that S is neither 1-valid nor 0-valid. We show for

some well-chosen constraint language $S' \subseteq \langle S \rangle$ that

$$\begin{aligned} \textsc{Critical-3Sat} &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S', \{x \wedge \neg y\}) \\ &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S', S) \\ &\leq_{\mathrm{m}}^{\log} \textsc{Arg-Check}(S). \end{aligned}$$

For the first reduction we associate with φ the instance (Φ, α) where $\Phi = \{(C_j \vee u) \wedge \neg v \mid j \in J\}$ and $\alpha = u \wedge \neg v$ for u, v fresh variables. It is easy to see that φ is a critical instance if and only if (Φ, α) is an argument. The second reduction follows by Lemma 4.1.10, fourth item. The third one follows from Lemma 4.2.4, observing that the formulæ in Φ are constraints built upon a finite set of relations S' and thus $S' \subseteq \langle S \rangle$ according to Lemma 4.2.5. □

To finish the proof of Theorem 4.2.6 it remains to deal with constraint languages that are not Schaefer but complementive.

Proposition 4.2.9. *Let S be a constraint language which is not* Schaefer *but is* complementive. *Then* $\textsc{Arg-Check}(S)$ *is* DP*-complete.*

Proof. We prove that $\textsc{Arg-Check}(S \cup \{T\}) \leq_{\mathrm{m}}^{\mathrm{p}} \textsc{Arg-Check}(S)$. This will prove hardness for $\textsc{Arg-Check}(S)$ since $S \cup \{T\}$ is neither Schaefer nor complementive (because of T) and therefore $\textsc{Arg-Check}(S \cup \{T\})$ is a DP-complete problem according to Proposition 4.2.8

So, let (Φ, α) be an instance of $\textsc{Arg-Check}(S \cup \{T\})$. In all formulæ replace variables occurring in a T-constraint by t and delete all T-constraints. Thus we obtain (Φ', α') an instance of $\textsc{Arg-Check}(S)$. The key to observe that this reduction is correct is that S is complementive: it suffices thus to observe correctness for the case $t = 1$ which is obvious. □

4.2.3 Overview of results for Argumentation

Our results show that the frontier between hard and easy problems for Arg-Check is the same as for the implication problem Imp. As a consequence, the classical tractable fragments of the satisfiability problem, i.e., Horn-, dual-Horn-, affine- and 2CNF-formulæ, render also tractable the argument-validity problem Arg-Check. Though, the argument-existence problem Arg remains intractable in these cases (NP-complete). We have to add the property of ε-validity to get to the tractable fragments of Arg.

The classification for ARG illustrates very clearly which conditions cause the complexity to drop. The coNP-complete cases occur when there is a natural candidate to form a support, the remaining complexity comes from the implication problem. The NP-complete cases occur when satisfiability and implication is easy, but we have still to guess a candidate support. Finally, the P-membership occurs when both conditions are true.

As in Post's Framework there are fragments (for instance in the case of 0-valid-non-Schaefer relations) for which verifying an argument is potentially harder than deciding the existence of an argument (ARG-CHECK is DP-complete, ARG is only coNP-complete). This is due to the minimality condition which is relevant for the verification but not for the existence of an argument.

Note finally that stating the obtained classifications for ARG and ARG-CHECK in terms of co-clone inclusions, we observe that the Galois connection holds with respect to polynomial many-one reductions, i.e., the complexity does not change within co-clones.

4.3 Enumeration of Models

4.3.1 Complexity of Enumeration

Enumeration requires generating *all* solutions of a problem instance *without duplicates*. We will study in this section the complexity of enumerating the models of an S-formula.

Problem:	EnumSat(S)
Instance:	An S-formula φ
Question:	enumerate all models of φ

In order to do so, we first have to introduce appropriate notions for measuring the complexity of enumeration. The classical notion of efficiency, namely *polynomial time*, seems not appropriate for enumeration: as in the case of EnumSat(S), the number of solutions to be enumerated is usually exponential in the size of the input instance.

Therefore new notions of efficiency have been developed for enumeration. The first general complexity notions for enumeration problems are stated by Johnson *et al.* in [JPY88]. The authors present three basic notions of efficiency. The first one is called *output polynomial* - the total running time is polynomial in the input *and* the output. A slightly stronger notion is *incremental polynomial* - for each *new* solution the time is polynomial in the input and the output *so far*. The third and even stronger notion is *polynomial delay* - the time between two successive solutions is polynomial in the input. In addition one can demand the space to be polynomial. A restriction of different nature is that we can prescribe the order in which we want the solutions to be output. This is a fundamental new aspect of enumeration. It turns out that the order affects heavily the complexity. Johnson *et al.* demonstrate this fact with the problem of enumerating all maximal independent sets of a graph: They show that all maximal independent sets can be enumerated in lexicographical order with polynomial delay, but that there is no polynomial delay algorithm for the reverse lexicographical order, unless P = NP.

The complexity of EnumSat(S) has already been classified in the case where no specific order is imposed. Creignou and Hébrard [CH97] obtained a dichotomous classification theorem: there is a polynomial delay algorithm if and only if the constraint language S is Schaefer, unless P = NP. The algorithms underlying this result are based on the so-called *self-reducibility* property of the satisfiability problem. In the case of the

satisfiability problem this means that one may construct the models recursively determining the value of a variable x_i by a call to the decision problem SAT that is reduced by the value of x_i. As a consequence the algorithms developed by Creignou and Hébrard are able to enumerate in lexicographic order (or the reverse) and require polynomial space.

We will consider here a variant, ENUMSAT$_w(S)$, where we want to enumerate the models of an S-formula by non-decreasing weight, the weight of a model being the number of variables assigned to 1.

Problem:	ENUMSAT$_w(S)$
Instance:	An S-formula φ
Question:	enumerate all models of φ by non-decreasing weight $\delta(m)$, where $\delta(m) = \lvert\{x \in \mathrm{Vars}(\varphi) \mid m(x) = 1\}\rvert$

The order of non-decreasing weight requires different algorithmic techniques than the ones used by Creignou and Hébrard. Further, we will see that this order increases the complexity: several previously tractable cases become intractable. In fact, we will establish two different polynomial delay algorithms for constraint languages that are Horn or width-2-affine. We will show that for all other constraint languages there is no polynomial delay algorithm, unless P = NP. In the following we first treat the tractable cases and then treat the hard cases.

4.3.2 Polynomial Delay Algorithms

We will develop in this section two polynomial delay algorithms. The first one for the width-2-affine case is based on a reduction to the SUBSETSUM problem for which we develop a polynomial delay algorithm based on dynamic programming and self-reducibility.

Proposition 4.3.1. *If S is* width-2-affine*, then there is a polynomial-space polynomial-delay algorithm that generates all models of an S-formula by non-decreasing weight.*

Proof. Let S be width-2-affine and let φ be an S-formula. Without loss of generality we can suppose that φ does not contain unitary clauses. Then each clause of φ expresses either the equality or the inequality between two variables. Using 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 each two classes are either independent

or they must have contrary truth values. We call a pair (A, B) of classes with contrary truth values *cluster*, B may be empty. It follows easily that any two clusters are independent and thus to obtain a model of φ, we choose for each cluster (A, B) either $A = 1, B = 0$ or $A = 0, B = 1$. We suppose in the following that φ is satisfiable (otherwise, we will detect a contradiction while constructing the clusters). Let $n \geq 1$ be the number of clusters, then the number of models will be 2^n. The weight contribution of each cluster to a model is either $|A|$ or $|B|$, where $|A| = |B|$ may occur. We represent a model by an n-tuple $s \in \{0,1\}^n$, indicating for each cluster which of the two assignments is taken. In the case $|A| \neq |B|$ we indicate by 0 the light assignment and by 1 the heavy assignment. Surely $(0, 0, \ldots, 0)$ will represent a model of minimal weight, and $(1, 1, \ldots, 1)$ will represent a model of maximal weight. For enumeration we may consider only the weight difference $\big||A| - |B|\big|$ of each cluster, since we can subtract the weight of a minimal model. Setting $(w_1, \ldots, w_n)$ to these weight differences of the clusters, we reduce our problem to the following enumeration problem:

Problem:	SUBSETSUM
Instance:	A sequence of non-negative integers $(w_1, \ldots, w_n) \in \mathbb{N}^n$
Question:	generate all n-tuples $s \in \{0,1\}^n$ by non-decreasing weight $\delta(s)$, where $\delta(s) = \Sigma_{i=1}^n s_i \cdot w_i$

To solve this enumeration problem we make use of the fact that in our case the sum of the weights $W := \Sigma_{i=1}^n w_i$ is linearly bounded by the number of variables of the original formula φ. This allows a strategy of dynamic programming to compute in polynomial time a matrix $A \in \{0,1\}^{(n+1,\, W+1)}$ such that $A(i,k) = 1$ if and only if with the weights $w_1, \ldots, w_i$ one can construct the sum k, where $0 \leq i \leq n$, $0 \leq k \leq W$. The matrix A is constructed by first setting $A(0,0) = 1$ and $A(0,k) = 0$ for all $k \geq 1$, and then filling the other fields row by row according to the rule $A(i,k) = 1$ if and only if $A(i-1,k) = 1$ or $A(i-1,k-w_i) = 1$. Thus the computation of A takes time $O(n \cdot W)$. After this precomputation, for each k for which there is at least one solution of weight k we enumerate all such solutions by constructing the solution strings from ϵ (the empty string) recursively (we use here in fact the self-reducibility of the decision problem *is there a solution of weight k?*).

The reader may convince himself or herself that Algorithm 2 enumerates all solutions s of the SUBSETSUM problem by non-decreasing weight

Algorithm 2 Algorithm for SUBSETSUM.

```
MAIN(w_1, ..., w_n)
 1: compute A ∈ {0,1}^{(n+1, W+1)}
 2: for k = 0 to W do
 3:    if A(n,k) = 1 then
 4:       CONSTRUCTSOLUTIONS(n, k, ε)   /* enumerate all solutions of
          weight k */
 5:    end if
 6: end for

CONSTRUCTSOLUTIONS(i, j, s)
 1: if i = 0 then
 2:    output s
 3: else
 4:    if A(i − 1, j − w_i) = 1 then
 5:       CONSTRUCTSOLUTIONS(i − 1, j − w_i, 1 ∘ s) /* ∘ stands for the
          concatenation operator */
 6:    end if
 7:    if A(i − 1, j) = 1 then
 8:       CONSTRUCTSOLUTIONS(i − 1, j, 0 ∘ s)
 9:    end if
10: end if
```

$\delta(s)$, implementing the above described method. Since both n and W are linearly bounded by the number of variables of φ, Algorithm 2 has a quadratic precomputation time and a linear delay thereafter. The translations between our original problem and SUBSETSUM can be performed in polynomial time. We finally observe that the quadratic space requirement can be improved to linear space, since for each column k of the matrix A, we have only to store at which row i we pass from 0 to 1. □

Our second algorithm for Horn-formulæ is of different nature. It is based on a technique first introduced by Johnson *et al.* in [JPY88] that uses a priority queue to control the output of the solutions.

Proposition 4.3.2. *If S is* Horn*, then there is a polynomial delay algorithm that generates all models of an S-formula by non-decreasing weight.*

Proof. Let S be Horn and let φ be an S-formula. Then φ is equivalent to a conjunction of Horn clauses. We will use a priority queue Q to

respect the order of non-decreasing weight and to avoid duplicates. The command Q.enqueue(s, k) enqueues an element s with an integer key-value k (a weight). The queue sorts by non-decreasing key-value and inserts an element s only if it is not yet present in the queue.

For notational convenience we represent a model by the set of variables it sets to 1. We use the well-known fact that for Horn formulæ the intersection of all models is the unique *minimal model* which is polynomial time computable. For a satisfiable Horn formula φ we indicate the minimal model by $mm(\varphi)$. Note that for a set of variables $V \subseteq \mathrm{Vars}(\varphi)$ the formula $\varphi \wedge V := \varphi \wedge \bigwedge_{v \in V} v$ is still representable as a Horn formula and thus, if $\varphi \wedge V$ is satisfiable, also $mm(\varphi \wedge V)$ can be computed in polynomial time.

Algorithm 3 Algorithm for Horn – Sat.

Require: φ a Horn formula
1: **if** φ unsatisfiable **then**
2: return 'no'
3: **end if**
4: Q = newPriorityQueue
5: $m := mm(\varphi)$
6: Q.enqueue(m, $|m|$)
7: **while** Q not empty **do**
8: $m :=$ Q.dequeue
9: output m
10: **for all** $x \in \mathrm{Vars}(\varphi) \setminus m$ **do**
11: **if** $\varphi \wedge m \wedge x$ satisfiable **then**
12: $m' := mm(\varphi \wedge m \wedge x)$
13: Q.enqueue(m', $|m'|$)
14: **end if**
15: **end for**
16: **end while**

We claim that Algorithm 3 enumerates the models of a given Horn formula with polynomial delay, by non-decreasing weight. The polynomial delay is easily seen. By definition of the priority queue and by the fact that the models m' generated out of m in line 11 are always of bigger weight than m itself, it is also easily seen that the models are output in the right order and that no model is output twice. To prove that no model is omitted, it suffices to show that for every model $m' \neq mm(\varphi)$ there exists a

submodel $m \subsetneq m'$ such that in line 11 the algorithm generates m' out of m. That is, there must be an $x \in m' \setminus m$ such that $m' = mm(\varphi \wedge m \wedge x)$. Consider for this the set $H := \{m \mid m \text{ a model of } \varphi \text{ and } m \subsetneq m'\}$. The set H is not empty since it contains at least the minimal model $mm(\varphi)$. A maximal element m of H fulfills our needs, since it satisfies $m' = mm(\varphi \wedge m \wedge x)$ for any $x \in m' \setminus m$.

Let us finally stress that in contrast to Algorithm 2, Algorithm 3 potentially runs in exponential space. □

4.3.3 Hardness Results

In this section we investigate the case where our constraint language S is neither Horn nor width-2-affine. Clearly, in order to enumerate the models of an S-formula by non-decreasing weight, it is a necessary condition to be able to find the lightest model efficiently. As we will prove, this is not a sufficient condition, we need also to be able to find the second one efficiently. So let us introduce the following problems.

Problem:	MIN-ONES(S)
Instance:	an S-formula φ, an integer W
Question:	Is there a model of φ of weight $\leq W$?

Problem:	MIN-ONES*(S)
Instance:	an S-formula φ, an integer W
Question:	Is there a model of φ, different from all-0, of weight $\leq W$?

From the classification obtained in [KSW97] for the corresponding optimization problem, one can deduce the following.

Proposition 4.3.3. *(Minimum ones satisfiability [KSW97]) If S is* 0-valid *or* Horn *or* width-2-affine*, then the problem* MIN-ONES(S) *is in* P*, otherwise* MIN-ONES(S) *is* NP*-complete.*

Our main contribution in this section is the following hardness result, which obviously proves that when S is neither Horn nor width-2-affine, there is no polynomial delay algorithm that enumerates all models of an S-formula in order of non-decreasing weight, unless P = NP.

Proposition 4.3.4. *Let S be a set of relations which is neither* Horn *nor* width-2-affine*. Then* MIN-ONES*(S) *is* NP*-complete.*

Proof. If S is not Schaefer, then $\text{SAT}^*(S)$ is NP-complete [Sch78] and hence so is the problem $\text{MIN-ONES}^*(S)$. If S is not 0-valid, then, since it is neither Horn nor width-2-affine, the result follows from the NP-completeness of $\text{MIN-ONES}(S)$ (Proposition 4.3.3). Therefore, it remains to study sets S that are Schaefer and 0-valid but that are neither Horn nor width-2-affine. There are three cases to analyze.

- S is bijunctive and 0-valid but neither Horn nor width-2-affine.
- S is affine and 0-valid but neither Horn nor width-2-affine.
- S is dual-Horn and 0-valid but neither Horn nor width-2-affine.

Observe that a 2CNF formula which is 0-valid is also Horn. So the first case does not occur. Besides, one can easily prove that a 0-valid affine relation which is not Horn cannot be width-2-affine (this can also be read from Post's lattice using the Galois correspondence between clones and co-clones). Therefore the proof of the proposition will be completed when we successively prove the NP-completeness of $\text{MIN-ONES}^*(S)$ for any set S such that:

1. S is affine and 0-valid but not Horn, or
2. S is dual-Horn and 0-valid but neither affine nor Horn.

The NP-completness of $\text{MIN-ONES}^*(S)$ for any set S fulfilling the description 1 or 2 above is settled, respectively, by the forthcoming Proposition 4.3.6 and Proposition 4.3.7. For the reader's convenience, a scheme of the proof is displayed in Figure 4.4 on page 118. □

Case 1: affine, 0-valid, not horn

We treat in this section constraint languages that are 0-valid and affine but not Horn. We will prove that for such a constraint language S, finding a non-all-0 model of minimal weight of an S-formula is NP-hard. In order to do so, we first prove the following basic hardness result and then derive the desired result using the implementation results of Lemma 4.1.10.

Lemma 4.3.5. $\text{MIN-ONES}^*(x \oplus y \oplus z = 0)$ *and* $\text{MIN-ONES}^*(w \oplus x \oplus y \oplus z = 0)$ *are NP-complete.*

Proof. Consider a homogeneous linear system over the finite field GF(2). Finding the non-all-0 solution with minimum weight of such a system is known to be NP-hard (see [Bar97, Theorem 4.1]). In order to prove the lemma we have to show that this problem remains hard when restricted to systems that have three (resp. four) variables by equation. Let S be a homogeneous linear system over GF(2). Suppose that S has n variables, $x_1, \ldots, x_n$. In order to reduce the number of variables in each equation we introduce auxiliary variables. If there is an equation $x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_k} = 0$ for some $k \geq 4$, we introduce a new variable y_{i_1,i_2} and replace the original equation by the two equations $y_{i_1,i_2} \oplus x_{i_1} \oplus x_{i_2} = 0$ and $y_{i_1,i_2} \oplus x_{i_3} \oplus \ldots \oplus x_{i_k} = 0$. We repeat this process until all equations have three variables. The satisfiability is preserved during this transformation. The number of auxiliary variables is bounded from above by the number of occurrences of variables in the original system. In order to keep the information on the weight of the solutions we need to introduce enough copies of the original variables, which make the auxiliary variables neglectable. Let N be the number of occurrences of variables in S. Let f be a fresh variable that will play the role of the constant 0. For each $i = 1, \ldots, n$, we introduce N copy-variables $x_i^1, \ldots, x_i^N$ of x_i and add the equations $x_i \oplus x_i^j \oplus f = 0$ for $j = 1, \ldots N$. Finally we add the equation $f \oplus f \oplus f = 0$, i.e., $f = 0$ (this will ensure that $x_i = x_i^j$ for all j). There is a one-to-one correspondence between the non-trivial solutions of S and the non-trivial solutions of the so-obtained system S'. Moreover S has a non-trivial solution of weight at most W if and only if S' has a non-trivial solution of weight at most $W(N+1)+N$. Since the system S' can be seen as an $(x \oplus y \oplus z = 0)$-formula we have thus proved the NP-hardness of Min-Ones$^*(x \oplus y \oplus z = 0)$.

Let us now reduce Min-Ones$^*(x \oplus y \oplus z = 0)$ to Min-Ones$^*(w \oplus x \oplus y \oplus z = 0)$. Let S be a homogeneous linear system over n variables such that each equation has exactly three variables. Let w and w_i for $i = 1, \ldots, n+1$ be fresh variables. Transform S into S' as follows: transform every equation $x \oplus y \oplus z = 0$ into $w \oplus x \oplus y \oplus z = 0$ and add the $n+1$ equations $w \oplus w \oplus w \oplus w_i = 0$ for $i = 1, \ldots, n+1$. Solutions of S' assigning 0 to w coincide with the solutions of S. Moreover any solution of S' assigning 1 to w has weight at least $n+1$. Therefore, S has a non-trivial solution of weight at most W ($W \leq n$) if and only if S' has a non-trivial solution of weight at most W. This completes the proof. □

Proposition 4.3.6. *If S is* 0-valid *and* affine *but not* Horn*, then the problem* Min-Ones$^*(S)$ *is* NP*-complete.*

Proof. If S is not 1-valid, then the fifth item of Lemma 4.1.10 allows a reduction from MIN-ONES$^*(x \oplus y \oplus z = 0)$ to MIN-ONES$^*(S)$ (replace each constraint $(x \oplus y \oplus z = 0)$ by the S-formula equivalent to $\neg w \wedge (x \oplus y \oplus z = 0)$, where w is a fresh variable). If S is 1-valid, then the sixth item of Lemma 4.1.10 allows a reduction from MIN-ONES$^*(w \oplus x \oplus y \oplus z = 0)$ to MIN-ONES$^*(S)$. In both cases one can conclude with Lemma 4.3.5. □

Case 2 : dual-Horn, 0-valid, not affine, not Horn

In this section we deal with constraint languages that are 0-valid and dual-Horn but neither affine nor Horn. The method of proof is not the same as in the previous section. We base here on the hardness of MIN-ONES$(S \cup \{T\})$. The clue is to get rid of the T-constraints and to pass to MIN-ONES$^*(S)$.

Proposition 4.3.7. *If S is* 0-valid *and* dual-Horn *but neither* affine *nor* Horn*, then the problem* MIN-ONES$^*(S)$ *is* NP*-complete.*

Proof. According to Proposition 4.3.3, the problem MIN-ONES$(S \cup \{T\})$ is NP-complete. We reduce MIN-ONES$(S \cup \{T\})$ to MIN-ONES$^*(S)$. Let φ be an $(S \cup \{T\})$-formula, $\varphi = \psi \wedge \bigwedge_{x \in V} T(x)$ where ψ is an S-formula. Let t be a fresh variable and consider

$$\varphi' = \psi[V/t] \wedge \bigwedge_{x \in \mathrm{Vars}(\varphi) \setminus V} x \to t.$$

Observe that the only solution that assigns 0 to t in φ' is the all-0 one.

Therefore it is clear that φ has a solution of weight at most W $(W \geq |V|)$ if and only if φ' has a non-trivial solution of weight at most $W - |V| + 1$.

It remains to show that φ' is expressible as an S-formula which comes down to showing that the implication is expressible. If S is 1-valid the implication is expressible by the third item of Lemma 4.1.10. If S is not 1-valid it follows by the seventh item of Lemma 4.1.10 that we may express the implication introducing an additional variable that will always take the value 0, which does not further disturb. □

We can finally state the classification theorem for ENUMSAT$_w(S)$.

Theorem 4.3.8. *There is a polynomial delay algorithm to enumerate the models of an S-formula by non-decreasing weight if and only if S is* Horn *or* width-2-affine*, unless* P = NP*.*

Note that by duality we can thus enumerate the models of an S-formulae by *non-increasing* weight if and only if S is dual-Horn or width-2-affine.

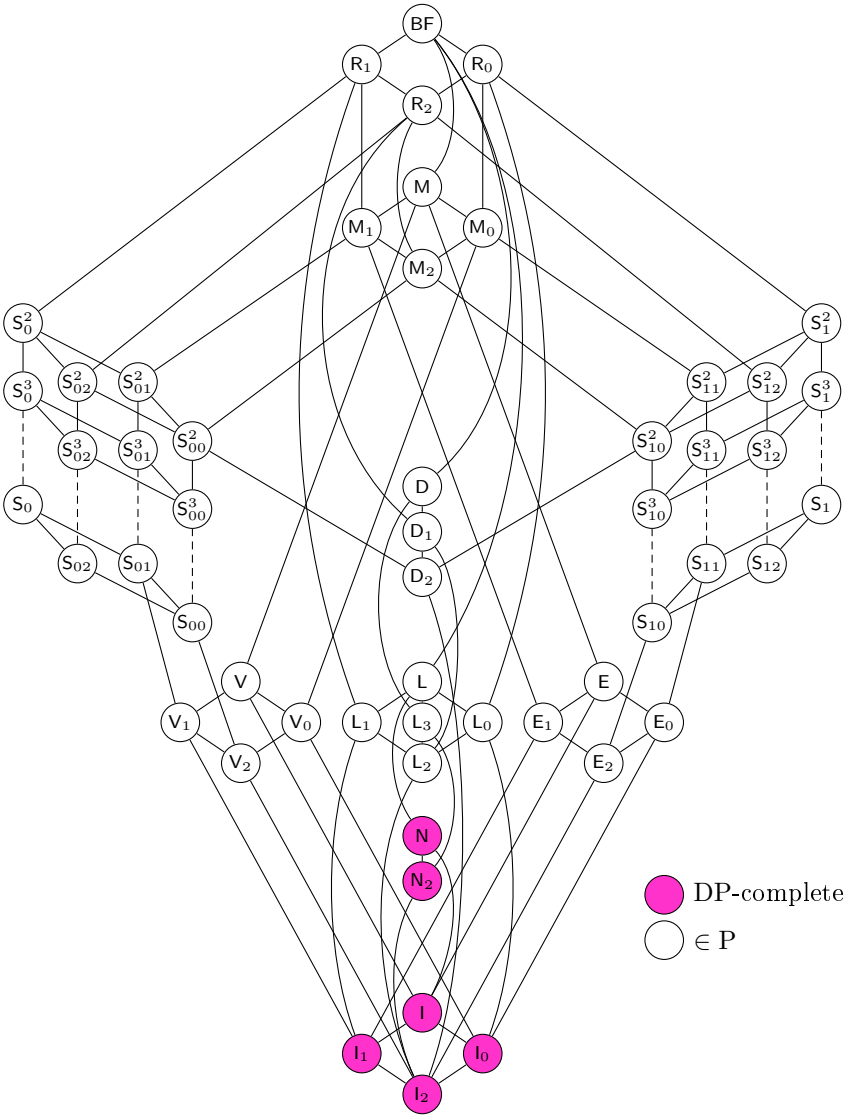

Figure 4.1: The complexity of ARG-CHECK(S).

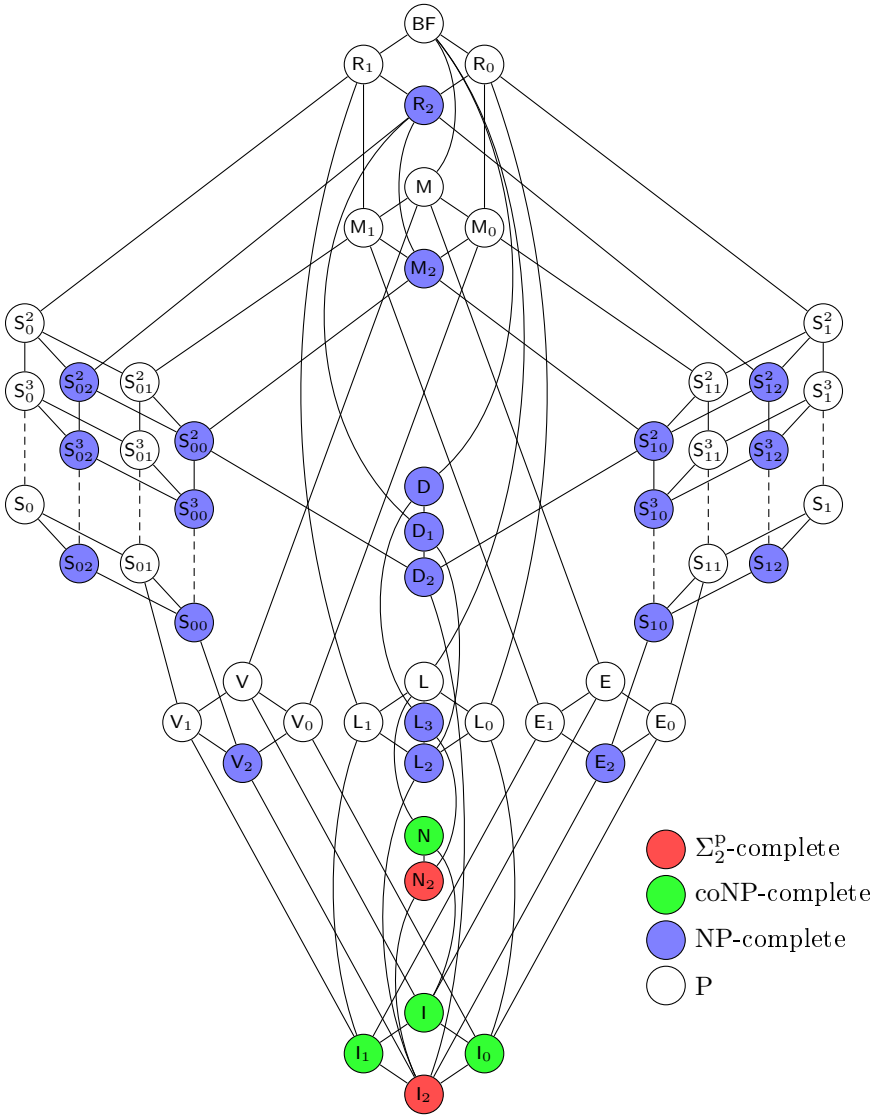

Figure 4.2: The complexity of ARG(S).

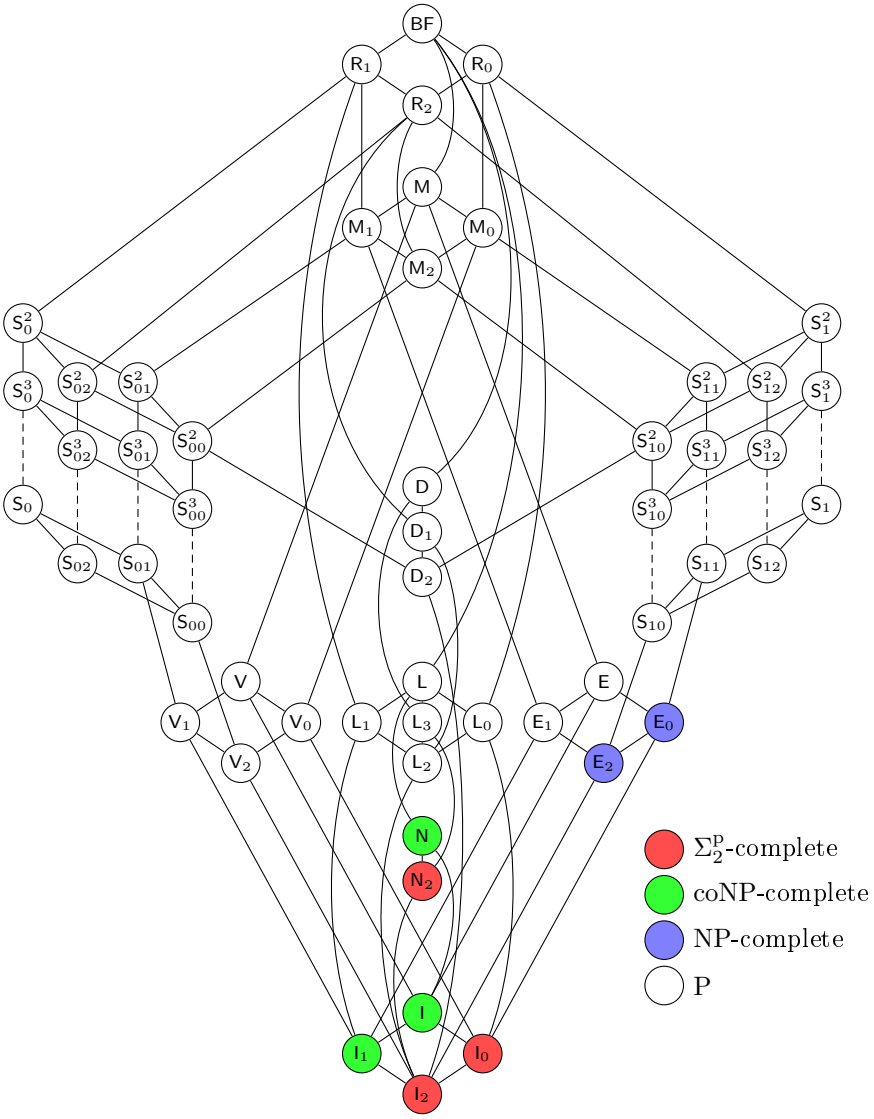

Figure 4.3: The complexity of P-ABD(S, PQ).

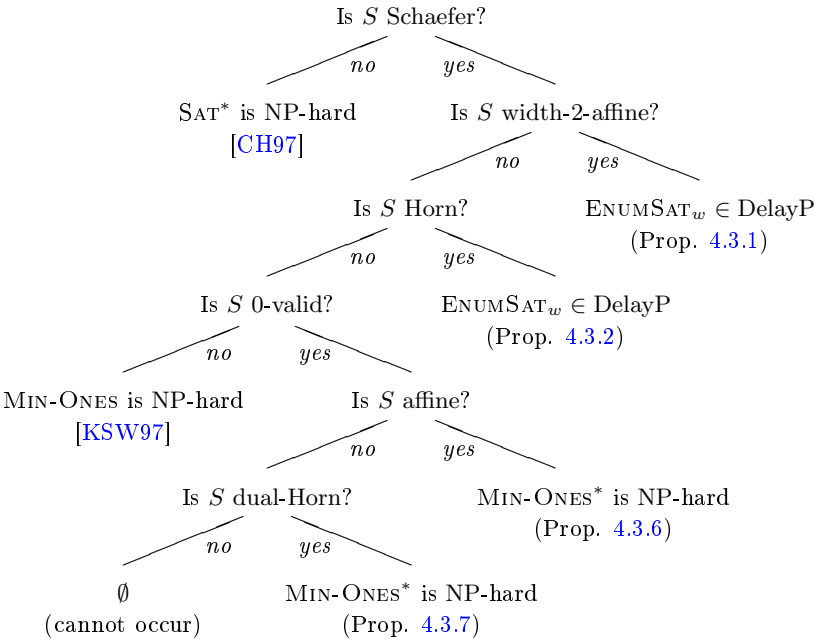

Figure 4.4: Scheme of the proof of Proposition 4.3.4

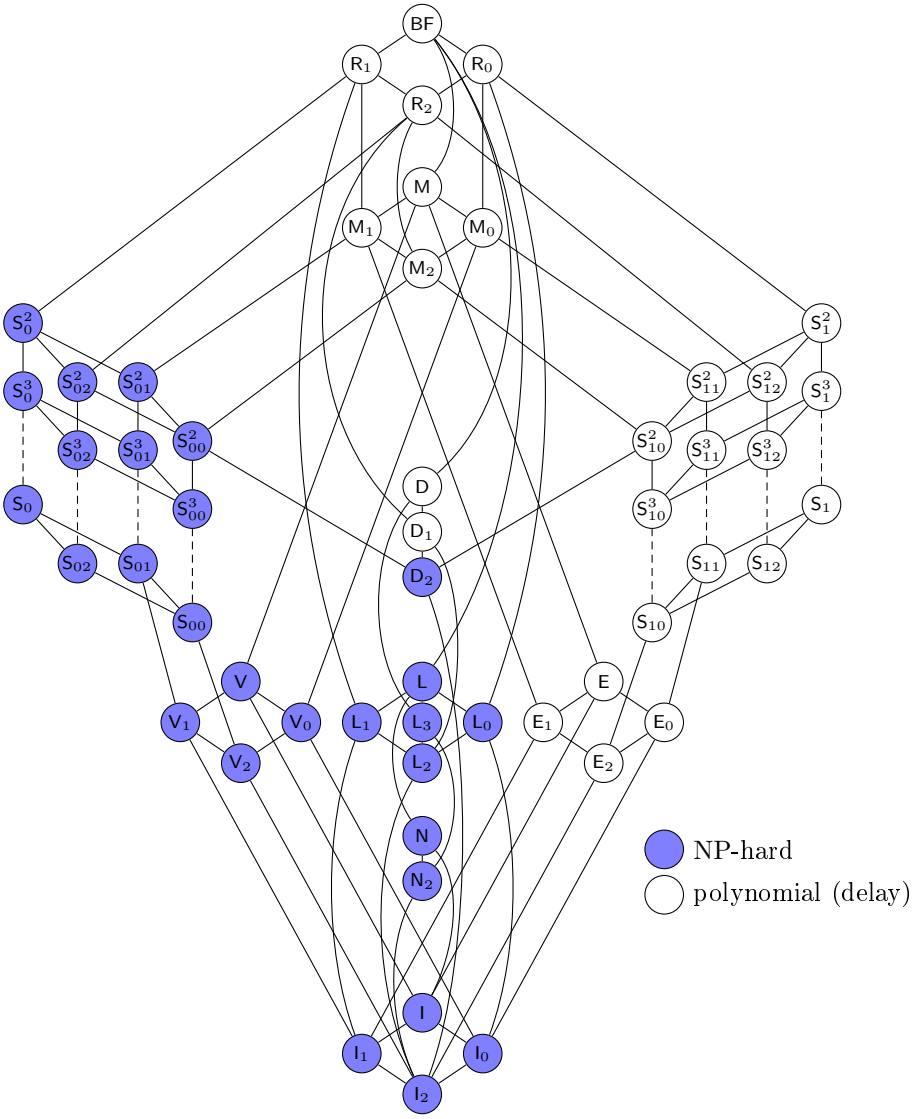

Figure 4.5: The complexity of $\textsc{EnumSat}_w(S)$ and $\textsc{Min-Ones}^*(S)$.

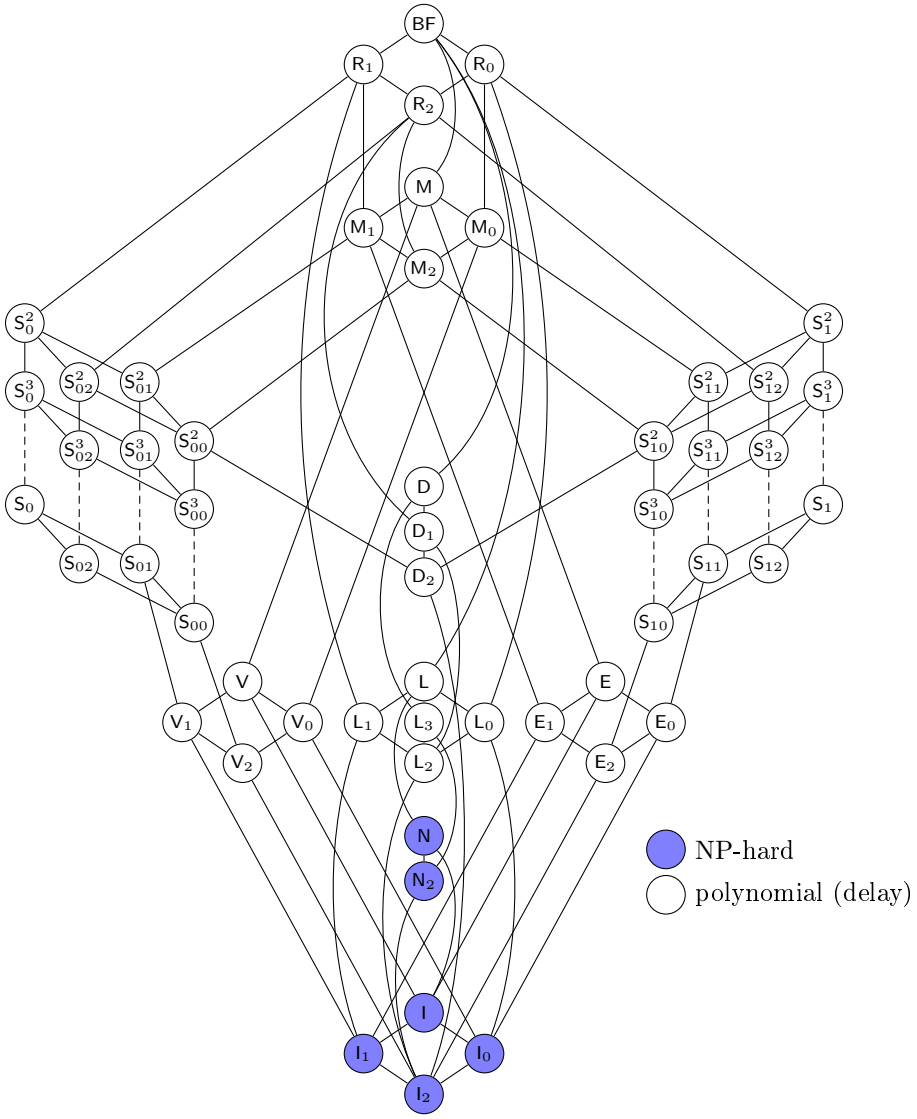

Figure 4.6: The complexity of EnumSat(S) and Sat*(S).

Chapter 5

Concluding remarks

We studied in this thesis systematically the complexity of propositional abduction and logic-based argumentation in Post's framework. In the case of abduction we only considered the existence and validity problem. We did not consider relevance and dispensability questions as we did for argumentation. In order to draw a more complete picture of the complexity of abduction it would thus be desirable to investigate also these problems. In addition, as motivated in [EG95], also for abduction subset-minimal solutions may be of interest. Demanding an explanation to be subset-minimal may increase the complexity for validity as well as for relevance and dispensability. The author estimates that classifications for all these problems can be obtained with relatively little effort, making use of similar tools and techniques as we used in this thesis.

We have to mention that some of the classifications we obtained are not complete. For positive abduction for the affine fragment $[B] \in \{\mathsf{L}_3, \mathsf{L}_0, \mathsf{L}\}$ when also the manifestation is a B-formula we only get NP-membership without being able to establish NP-completeness, neither to show P-membership. The cases of abduction where we could establish P-membership for affine formulæ rely on Gaussian elimination. This method fails for the mentioned case and there is no obvious alternative. Also in the case of Arg-Rel the precise complexity of the whole affine fragment remains open for similar reasons. It is worth noticing that a similar case, the circumscriptive inference of an affine formula from a set of affine formulæ, remained unclassified in [Tho09].

In Schaefer's Framework we studied the existence, validity and dis-

pensability problem for argumentation. A complete classification for the relevance problem appears to be much more involving and has to be let for future work (here neither the Galois connection seems not to hold a priori). For abduction, relevance and dispensability questions have not yet been studied in Schaefer's Framework. For manifestations such as single literals, clauses and terms a Galois connection should hold a priori which could render a classification feasible. In the case where the manifestation is also an S-formula, abduction has not yet been studied at all. It will be a challenge for future work to address this variant, since here the useful Galois connection seems not to hold a priori.

We would like to point out that the area of enumeration complexity is an open field. Although there are in the meanwhile many results on particular enumeration problems, the attempt to develop a theoretical framework to deal with the complexity of enumeration is quite recent (see [Bag06, Cou08, DG07, BDGO08, Str10, Sch09]). In particular there are up to now no common notions of reductions in order to establish completeness results. It could be an important step to identify complete problems that might be suitable to separate complexity classes of the hierarchy as proposed by Johnson *et al.* in [JPY88]. The hierarchy is only partially known to be strict under common complexity hypotheses from classical complexity theory.

We gave in our investigations a sort of reduction from $\textsc{EnumSat}_w(S)$ to SubSetSum for width-2-affine constraint languages. However, the reduction is basically an isomorphism. Further, we showed that SubSetSum is solvable in polynomial delay and polynomial space when the weights are polynomially bounded. However, in general (i.e., without bounds on the weights) the applied method does not work. The problem is still solvable within polynomial delay with a similar method as we applied to Horn-formulæ using a priority queue. Consequently, as in the Horn case, the algorithm does not work within polynomial space. Future work might investigate whether this exponential space can be avoided by better algorithmic methods or whether it is inherent in these problems.

Bibliography

[AC02] L. Amgoud and C. Cayrol. A reasoning model based on the production of acceptable arguments. *Ann. Math. Artif. Intell.*, 34(1-3):197–215, 2002.

[AFM02] J. Amilhastre, H. Fargier, and P. Marquis. Consistency restoration and explanations in dynamic CSPs. *Artif. Intell.*, 135(1-2):199–234, 2002.

[Bag06] G. Bagan. MSO queries on tree decomposable structures are computable with linear delay. In *Proc. of the Annual Conference of the European Association for Computer Science Logic (CSL)*, pages 167–181, 2006.

[Bar97] A. Barg. Complexity issues in coding theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(46), 1997.

[BATJ89] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson. Some results concerning the computational complexity of abduction. In *Proc. 1st KR*, pages 44–54, 1989.

[BBC+07] M. Bauland, E. Böhler, N. Creignou, S. Reith, H. Schnoor, and H. Vollmer. The complexity of problems for quantified constraints. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(023), 2007.

[BCRV04] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part II: Constraint satisfaction problems. *SIGACT News*, 35(1):22–35, 2004.

[BDGO08] G. Bagan, A. Durand, E. Grandjean, and F. Olive. Computing the jth solution of a first-order query. *RAIRO Theoretical Informatics and Applications*, 42:147–164, 2008.

[BH01] P. Besnard and A. Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, 2001.

[BH08] P. Besnard and A. Hunter. *Elements of Argumentation*. MIT Press, 2008.

[BHRV02] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. In *CSL*, pages 412–426, 2002.

[BHSS06] M. Bauland, E. Hemaspaandra, H. Schnoor, and I. Schnoor. Generalized modal satisfiability. In *Proc. 23rd STACS*, volume 3884 of *LNCS*, pages 500–511, 2006.

[BKKR69] V.G. Bodnarchuk, L.A. Kalužnin, V.N. Kotov, and B.A. Romov. Galois theory for Post algebras I, II. *Cybernetics*, 5:243–252, 531–539, 1969.

[BL00] M. Bouzid and A. Ligeza. Temporal causal abduction. *Constraints*, 5(3):303–319, 2000.

[BMTV09a] O. Beyersdorff, A. Meier, M. Thomas, and H. Vollmer. The complexity of propositional implication. *Information Processing Letters*, 109(18):1071–1077, 2009.

[BMTV09b] O. Beyersdorff, A. Meier, M. Thomas, and H. Vollmer. The complexity of reasoning for fragments of default logic. In *Proc. 12th SAT*, volume 5584 of *LNCS*, pages 51–64, 2009.

[BSS+08] M. Bauland, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The complexity of generalized satisfiability for linear temporal logic. In *Logical Methods in Computer Science*, volume 5, 2008.

[Cay95] C. Cayrol. On the relation between argumentation and nonmonotonic coherence-based entailment. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995)*, pages 1443–1448. Morgan Kaufmann, 1995.

[CH96] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996.

[CH97] N. Creignou and J. J. Hébrard. On generating all solutions of generalized satisfiability problems. *Informatique Théorique et Applications*, 31(6):499–511, 1997.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM Press, 1971.

[COS11] N. Creignou, F. Olive, and J. Schmidt. Enumerating all solutions of a Boolean CSP by non-decreasing weight. In *Proc. of 14th International Conference on Theory and Applications of Satisfiability Testing (SAT'2011)*, Lecture notes in computer science, 2011.

[Cou08] B. Courcelle. Linear delay enumeration and monadic second-order logic. *Discrete Applied Mathematics*, 2008.

[Cre95] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *J. Comput. Syst. Sci.*, 51(3):511–522, 1995.

[CST11] N. Creignou, J. Schmidt, and M. Thomas. Complexity classifications for propositional abduction in Post's framework. *Journal of Logic and Computation*, 2011.

[CSTW11] N. Creignou, J. Schmidt, M. Thomas, and S. Woltran. Complexity of logic-based argumentation in Post's framework. *Argument & Computation*, 2(2-3):107–129, 2011.

[CV08] N. Creignou and H. Vollmer. Boolean constraint satisfaction problems: when does Post's lattice help? In N. Creignou, Ph. G. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, volume 5250, pages 3–37. Springer Verlag, Berlin Heidelberg, 2008.

[CZ06] N. Creignou and B. Zanuttini. A complete classification of the complexity of propositional abduction. *SIAM J. Comput.*, 36(1):207–229, 2006.

[DG07] A. Durand and E. Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Transactions on Computational Logic*, 8(4), 2007.

[DKT06] P. M. Dung, R. Kowalski, and F. Toni. Dialectical proof procedures for assumption-based admissible argumentation. *Artif. Intell.*, 170(2):114–159, 2006.

[Dun95] P. M. 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.

[EG95] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.

[Gei68] D. Geiger. Closed Systems of Functions and Predicates. *Pacific Journal of Mathematics*, 27(1):95–100, 1968.

[GS04] A. García and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[Hor51] A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.

[HSAM93] J. R. Hobbs, M. E. Stickel, D. E. Appelt, and P. A. Martin. Interpretation as abduction. *Artif. Intell.*, 63(1-2):69–142, 1993.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[KS96] S. Khanna and M. Sudan. The optimization complexity of constraint satisfaction problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(28), 1996.

[KSW97] S. Khanna, M. Sudan, and D. Williamson. A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction. In *Proceedings 29th Symposium on Theory of Computing*, pages 11–20. ACM Press, 1997.

[KW88] M. Karchmer and A. Wigderson. Monotone Circuits for Connectivity Require Super-logarithmic Depth. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 539–550. ACM, 1988.

[Lew79] H. Lewis. Satisfiability problems for propositional calculi. *Mathematical Systems Theory*, 13:45–53, 1979.

[Moo85] R. C. Moore. Semantical considerations on modal logic. *Artificial Intelligence*, 25:75–94, 1985.

[NJ04] G. Nordh and P. Jonsson. An algebraic approach to the complexity of propositional circumscription. In *LICS*, pages 367–376, 2004.

[NZ08] G. Nordh and B. Zanuttini. What makes propositional abduction tractable. *Artif. Intell.*, 172(10):1245–1284, 2008.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pop73] H. E. Pople. On the mechanization of abductive logic. In *IJCAI*, pages 147–152, 1973.

[Pos41] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[PW88] C. H. Papadimitriou and D. Wolfe. The complexity of facets resolved. *J. Comput. Syst. Sci.*, 37(1):2–13, 1988.

[PWA03] S. Parsons, M. Wooldridge, and L. Amgoud. Properties and complexity of some formal inter-agent dialogues. *J. Log. Comput.*, 13(3):347–376, 2003.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Rei03] S. Reith. On the complexity of some equivalence problems for propositional calculi. In *Proc. 28th MFCS*, volume 2747 of *LNCS*, pages 632–641, 2003.

[Rei05] O. Reingold. Undirected ST-connectivity in log-space. In *STOC*, pages 376–385, 2005.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th STOC*, pages 216–226, 1978.

[Sch05] H. Schnoor. The complexity of the Boolean formula value problem. Technical report, Theoretical Computer Science, University of Hannover, 2005.

[Sch07a] H. Schnoor. *Algebraic Techniques for Satisfiability Problems.* Phd thesis, Leibniz Universität Hannover, Fakultät für Elektrotechnik und Informatik, 2007.

[Sch07b] I. Schnoor. *The Weak Base Method for Constraint Satisfaction.* Phd thesis, Leibniz Universität Hannover, Fakultät für Elektrotechnik und Informatik, 2007.

[Sch09] J. Schmidt. Enumeration: Algorithms and complexity. Preprint, available at http://www.thi.uni-hannover.de/fileadmin/forschung/arbeiten/schmidt-da.pdf, 2009.

[Spi71] P. M. Spira. On Time-Hardware Complexity Tradeoffs for Boolean Functions. In *Proceedings of the 4th Hawaii International Symposium on System Sciences*, pages 525–527, 1971.

[SS08] H. Schnoor and I. Schnoor. Partial polymorphisms and constraint satisfaction problems. In *Complexity of Constraints*, pages 229–254, 2008.

[Str10] Y. Strozecki. Enumeration complexity and matroid decomposition. Phd thesis, 2010.

[SW01] M. Stumptner and F. Wotawa. Diagnosing tree-structured systems. *Artif. Intell.*, 127(1):1–29, 2001.

[Tho09] M. Thomas. The complexity of circumscriptive inference in Post's lattice. In *Proc. 10th LPNMR*, volume 5753 of *Lecture Notes in Computer Science*, pages 290–302, 2009.

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

[Zan03] B. Zanuttini. New polynomial classes for logic-based abduction. *J. Artif. Intell. Res.*, 19:1–10, 2003.

Lebenslauf

Persönliche Daten

Name	Johannes Schmidt
Geburt	4. Juni 1984 in Hannover

Schulausbildung

1990 - 1994	Gundschule Hinter der Burg in Springe
1995 - 1996	Orientierungsstufe Süd in Springe
1997 - 2003	Otto-Hahn-Gymnasium in Springe
Juni 2003	Abitur

Studium und Promotion

2004 - 2008	Studium des Fachs "Mathematik mit der Studienrichtung Informatik" an der Leibniz Universität Hannover
2007	Studienarbeit, Titel *Algorithms for Exact Cover problems*, Entwicklung und Implementierung für bestimmte Anwendungsfälle
2008 - 2009	Erasmusaufenthalt an der Université d'Aix-Marseille 2 zwecks Diplomarbeit, Titel *Enumeration: Algorithms and Complexity*
April 2009	Abschluss des Studiums als *Diplom-Mathematiker* an der Leibniz Universität Hannover
2009 - 2012	Promotion in theoretischer Informatik in *co-tutelle* an der Aix-Marseille Université und an der Leibniz Universität Hannover

Außerberufliches

2003 - 2004	Zivildienst beim Deutschen Roten Kreuz in Springe
10.04.2012	Hochzeit mit Annika Moscati