A Fine-Grained Complexity View on Propositional Abduction - Algorithms and Lower Bounds

Victor Lagerkvist,[1] Mohamed Maizia,[1,2] Johannes Schmidt[2]

[1] Department of Computer and Information Science, Linköping University, Sweden

[2] Department of Computer Science and Informatics, Jönköping University, Sweden

victor.lagerkvist@liu.se, mohamed.maizia@ju.se, johannes.schmidt@ju.se

May 16, 2025

Abstract

The *Boolean satisfiability problem* (SAT) is a well-known example of monotonic reasoning, of intense practical interest due to fast solvers, complemented by rigorous fine-grained complexity results. However, for *non-monotonic* reasoning, e.g., *abductive* reasoning, comparably little is known outside classic complexity theory. In this paper we take a first step of bridging the gap between monotonic and non-monotonic reasoning by analyzing the complexity of intractable abduction problems under the seemingly overlooked but natural parameter n: the number of variables in the knowledge base. We obtain several positive results for Σ_2^P- as well as NP- and coNP-complete fragments, which implies the first example of beating exhaustive search for a Σ_2^P-complete problem (to the best of our knowledge). We complement this with lower bounds and for many fragments rule out improvements under the *(strong) exponential-time hypothesis*.

1 Introduction

The *Boolean satisfiability* problem is a well-known NP-complete problem. Due to the rapid advance of SAT solvers, many combinatorial problems are today solved by reducing to SAT, which can then be solved with off-the-shelf solvers. SAT fundamentally encodes a form of *monotonic* reasoning in the sense that conclusions remain valid regardless if new information is added. However, the real world is *non-monotonic*, meaning that one should be able to retract a statement if new data is added which violates the previous conclusion. One of the best known examples of non-monotonic reasoning is *abductive* reasoning where we are interested in finding an explanation, vis-à-vis a hypothesis, of some observed manifestation. Abduction has many practical applications, e.g., scientific discovery [10], network security [1], computational biology [22], medical diagnosis [18], knowledge base updates [23], and explainability issues in machine learning and decision support systems [7, 8, 2, 21]. This may be especially poignant in forthcoming decades due to the continued emergence of AI in new and surprising applications, which need to be made GDPR compliant [26] and explainable. The incitement for solving abduction fast, even when it is classically intractable, thus seems highly practically motivated.

Can non-monotonic reasoning be performed as efficiently as monotonic reasoning, or are there fundamental differences between the two? The classical complexity of abduction (and many other forms of

non-monotonic reasoning) is well-known [5, 27] and suggests a difference: SAT is NP-complete, while most forms of non-monotonic reasoning, including *propositional* abduction, are generally Σ_2^P-complete. However, modern complexity theory typically tells a different story, where classical hardness results do not imply that the problems are hopelessly intractable, but rather that different algorithmic schemes should be applied. For SAT, there is a healthy amount of theoretical research complementing the advances of SAT solvers, and k-SAT for every k can be solved substantially faster than 2^n (where n is the number of variables) via the resolution-based *PPSZ* algorithm [19]. There is a complementary theory of lower bounds where the central conjecture is that 3-SAT is not solvable in $2^{o(n)}$ time (*exponential-time hypothesis* (ETH) [9]) and the *strong exponential-time hypothesis* (SETH) which implies that SAT with unrestricted clause length (CNF-SAT) cannot be solved in c^n time for any $c < 2$.

In contrast, the precise exponential time complexity of abduction is currently a blind spot, and *no* improved algorithms are known for the intractable cases. In this paper we thus issue a systematic attack on the complexity of abduction with a particular focus on the natural complexity parameter n, the number of variables in the knowledge base, sometimes supplemented by $|H|$ or $|M|$, the number elements in the hypothesis H or manifestation M. To obtain general results we primarily consider the setup where we are given a set of relations Γ (a *constraint language*) where the knowledge base of an instance is provided by a Γ-formula. We write $\mathrm{ABD}(\Gamma)$ for this problem and additionally also consider the variant where an explanation only consists of positive literals (P-$\mathrm{ABD}(\Gamma)$) since these two variants exhibit interesting differences. The classical complexity of abduction is either in P, NP-complete, coNP-complete, or Σ_2^P-complete [17], and for which intractable Γ is it possible to beat exhaustive search? According to Cygan et al., tools to precisely analyze the exponential time complexity of NP-complete problems are in its infancy [4]. For problems at higher levels of the polynomial hierarchy the situation is even more dire. Are algorithmic approaches for problems in NP still usable? Are the tools to obtain lower bounds still usable? Why are no sharp upper bounds known for problems in non-monotonic reasoning, and are these problems fundamentally different from e.g. satisfiability problems?

We successfully answer many of these questions with novel algorithmic contributions. First, in Section 3 we show why enumerating all possible subsets of the hypothesis gives a bound 2^n for ABD and the (surprisingly bad) 3^n bound for P-ABD. Hence, any notion of improvement should be measured against 2^n for ABD and 3^n for P-ABD. Generally improving the factor $2^{|H|}$ (which may equal 2^n) seems difficult but we do manage this for languages Γ where all possible models of the knowledge base can be enumerated in c^n time, for some $c \leq 2$, which we call *sparsely enumerable* languages. We succeed with this for both $\mathrm{ABD}(\Gamma)$ (Section 3.1) and P-$\mathrm{ABD}(\Gamma)$ (Section 3.2), and while the algorithms for the two different cases share ideas, the details differ in intricate ways. It should be remarked that both algorithms solve the substantially more general problem of enumerating *all* (maximal) explanations which may open up further, e.g., probabilistic, applications for abduction. The enumeration algorithms in addition to exponential time also need exponential memory, but we manage to improve the naive 3^n bound for P-$\mathrm{ABD}(\Gamma)$ to 2^n with only polynomial memory. The sparsely enumerable property is strong: it fails even for 2-SAT and it is a priori not clear if it is ever true for intractable languages. Despite this, we manage to (in Section 3.3) describe three properties implying sparse enumerability. This captures relations definable by equations $x_1 + \ldots + x_k = q \pmod p$ (EQUATIONS^k). The problem(s) (P-)$\mathrm{ABD}(\textsc{Equations})$ is Σ_2^P-complete and is, to the best of our knowledge, the first example of beating exhaustive search for a Σ_2^P-complete problem (under n). This yields improved algorithms for Σ_2^P-complete P-$\mathrm{ABD}(\mathrm{XSAT})$ (*exact satisfiability*) and NP-complete P-$\mathrm{ABD}(\mathrm{AFF}^{(\leq k)})$ (arity bounded equations over GF(2)).

In Section 3.4 we consider more restricted types of abduction problems with a particular focus on

(Type) Class	Classical complexity	Improved
EQUATIONS^k ($k \geq 1$)	Σ_2^P-C	Yes
XSAT	Σ_2^P-C	$O^*(2^{\frac{n}{2}})$
(P) AFF^k ($k \geq 1$)	NP-C	Yes
(M) k-CNF$^+$ ($k \geq 1$)	NP-C	Yes
(P) k-CNF$^-$ $\cup$ IMP ($k \geq 1$)	NP-C	Yes
(P) finite 1-valid	coNP-C	Yes

Table 1: Upper bounds for P-ABD and ABD.

(Type) Class	Assumption	Bound
(M) 2-CNF$^+$	ETH	$(\frac{\lvert H\rvert}{\lvert M\rvert})^{o(\lvert M\rvert)}$
(P) 2-CNF$^+$ $\cup$ IMP	ETH	$(\frac{\lvert H\rvert}{\lvert M\rvert})^{o(\lvert M\rvert)}$
(M) k-CNF ($k \geq 4$)	SETH	2^n
(P) k-CNF ($k \geq 4$)	SETH	1.4142^n
CNF$^-$ $\cup$ IMP, Horn	SETH	1.2599^n
(M) CNF$^+$, DualHorn	SETH	1.2599^n

Table 2: Lower bounds for P-ABD and ABD.

(P-)ABD(k-CNF$^+$) where k-CNF$^+$ contains all positive clauses of arity k. Here, the problems are only NP-complete, in which case circumventing the $2^{|H|}$ barrier appears easier. For these, and similar, problems, we construct an improved algorithm based on a novel reduction to a problem SIMPLESAT^p which can be solved by branching. For coNP, only P-ABD(Γ) becomes relevant, and we (in Section 3.5) prove a simple but general improvement whenever a finite Γ is invariant under a constant Boolean operation.

In Section 4, we prove lower bounds under (S)ETH for missing intractable cases. Let IMP $= \{\{(0,0), (0,1), (1,1)\}\}$. Under the ETH, we first prove that ABD(2-CNF$^+$) and ABD(2-CNF$^-$ $\cup$ IMP) cannot be solved in time $(\frac{|H|}{|M|})^{o(|M|)}$ under ETH, which asymptotically matches exhaustive search. For classical cases like k-CNF ($k \geq 4$) and NAE-k-SAT ($k \geq 5$) we establish sharp lower bounds of the form 2^n for ABD and 1.4142^n for P-ABD under the SETH. For (P-)ABD(CNF$^-$ $\cup$ IMP), we rule out improvements to 1.2599^n, $1.4142^{|H|}$, $2^{|M|}$, or $\left(\frac{|H|}{|M|}\right)^{|M|}$ under SETH. This transfers to Horn for (P-)ABD and to DualHorn for ABD. For ABD(2-CNF), we prove that sharp lower bounds under the SETH are unlikely unless NP $\subseteq$ P/Poly, leaving its precise fine-grained complexity as an interesting open question.

The results are summarized in Table 1 and Table 2 where (P), respectively (M), indicates that the result only holds for P-ABD, respectively ABD. Thus, put together, we obtain a rather precise picture of the fine-grained complexity of ABD(Γ) and P-ABD(Γ) for almost all classical intractable languages Γ. Notably, we have proven that even Σ_2^P-complete problems can admit improved algorithms with respect to n, and that the barrier of exhaustively enumerating all possible explanations can be broken.

Proofs of statements marked with ($\star$) can be found in the appendix, following this paper.

2 Preliminaries

We begin by introducing basic notation and terminology.

Propositional logic. We assume familiarity with propositional logic, clauses, and formulas in conjunctive/disjunctive form (CNF/DNF). We denote $\mathrm{Var}(\varphi)$ the variables of a formula φ and for a set of formulas F, $\mathrm{Var}(F) = \bigcup_{\varphi \in F} \mathrm{Var}(\varphi)$. We identify finite F with the conjunction of all formulas from F, that is $\bigwedge_{\varphi \in F} \varphi$. A *model* of a formula φ is an assignment $\sigma : \mathrm{Var}(\varphi) \mapsto \{0,1\}$ that satisfies φ. For two formulas ψ, φ we write $\psi \models \varphi$ if every model of ψ also satisfies φ. For a set of variables V, $\mathrm{Lits}(V)$ the set of all literals formed upon V, that is, $\mathrm{Lits}(V) = V \cup \{\neg x \mid x \in V\}$.

Boolean constraint languages. A *logical relation* of arity $k \in \mathbb{N}$ is a relation $R \subseteq \{0,1\}^k$, and $\mathrm{ar}(R) = k$ denotes the arity. An (R-)constraint C is a formula $C = R(x_1, \ldots, x_k)$, where R is a k-ary logical relation, and $x_1, \ldots, x_k$ are (not necessarily distinct) variables. An assignment σ *satisfies* C, if $(\sigma(x_1), \ldots, \sigma(x_k)) \in R$. A (Boolean) *constraint language* Γ is a (possibly infinite) set of logical relations, and a Γ-*formula* is a conjunction of constraints over elements from Γ. A Γ-formula ϕ is *satisfied* by an assignment σ, if σ simultaneously satisfies all constraints in it, in which case σ is also called a *model of* ϕ. We define the two constant unary Boolean relations as $\bot = \{(0)\}$ and $\top = \{(1)\}$, and the two constant 0-ary relations as $\mathsf{f} = \emptyset$ and $\mathsf{t} = \{\emptyset\}$. We say that a k-ary relation R is *represented* by a propositional CNF-formula ϕ if ϕ is a formula over k distinct variables $x_1, \ldots, x_k$ and $\phi \equiv R(x_1, \ldots, x_k)$. Note such a CNF-representation exists for any logical relation $R \subseteq \{0,1\}^k$. We write CNF for the relations corresponding to all possible clauses and Horn/DualHorn for the set relations corresponding to Horn/DualHorn clauses. Additionally, k-CNF is the subset of CNF of arity k, IMP $= \{R\}$, where $R(x,y) = \{0,1\}^2 - \{(1,0)\}$. The notation (k-)CNF$^+$ (resp. (k-)CNF$^-$) denotes the version where each clause is positive (resp. negative). We use AFF to denote the set of relations representable by systems of Boolean equations modulo 2, i.e., $x_1 + \ldots + x_k \equiv q \pmod 2$ for $q \in \{0,1\}$. As representation of each relation in a constraint language we use the defining CNF-formula unless stated otherwise.

The satisfiability problem for Γ-formulas, also known as Boolean constraint satisfaction problem, is denoted $\mathrm{SAT}(\Gamma)$. It asks, given a Γ-formula ϕ, whether ϕ admits a model.

Propositional Abduction. An instance of the abduction problem over a constraint language Γ is given by (KB, H, M), where KB is a Γ-formula, and H and M sets of variables, referred to as *hypothesis* and *manifestation*. We let $V = \mathrm{Var}(\mathrm{KB}) \cup \mathrm{Var}(H) \cup \mathrm{Var}(M)$ be the set of variables and write $n = |V|$ for its cardinality. The *symmetric abduction problem*, denoted $\mathrm{ABD}(\Gamma)$, asks whether there exists an $E \subseteq \mathrm{Lits}(H)$ such that 1) $\mathrm{KB} \wedge E$ is satisfiable, and 2) $\mathrm{KB} \wedge E \models M$. If such an E exists, it is called an *explanation* for M. If an explanation E satisfies $\mathrm{Var}(E) = H$ it is called a *full explanation*, if it satisfies $E \subseteq H$, it is called a *positive explanation*. If there exists an explanation E, it can always be extended to a full explanation (by extending E according to the satisfying assignment underlying condition 1). However, the existence of an explanation does not imply the existence of a positive explanation. The *positive abduction problem*, denoted P-$\mathrm{ABD}(\Gamma)$, asks whether there exists a positive explanation. We write (P-)$\mathrm{ABD}(\Gamma)$ if the specific abduction type is not important.

Example 1. *Consider the following example.*

$$\begin{aligned} \mathit{KB} &= \{A \wedge B \to C, D \to B, \neg E \to C \wedge \neg D\}, \\ H &= \{A, D, E\}, \quad M = \{C\} \end{aligned}$$

$E_1 = \{A, \neg D, \neg E\}$ *is a full explanation, and* $E_2 = \{A, D\}$ *is a positive explanation.*

We note at this point that the classical complexity of $\mathrm{ABD}(\Gamma)$ and $\text{P-ABD}(\Gamma)$ is fully determined for all Γ, due to Nordh and Zanuttini [17]. An illustration of these classifications is found in the supplementary material.

Algebra. We denote by $\bar{x}$ the Boolean negation operation, that is, $f(x) = \neg x$. An n-ary *projection* is an operation f of the form $f(x_1, \ldots, x_n) = x_i$ for some fixed $1 \leq i \leq n$. An operation $f : \{0,1\}^k \to \{0,1\}$ is *constant* if for all $\mathbf{x} \in \{0,1\}^k$ it holds $f(\mathbf{x}) = c$, for a $c \in \{0,1\}$. For a k-ary operation $f\colon \{0,1\}^k \to \{0,1\}$ and $X \subseteq \{0,1\}^k$ we write $f_{|X}$ for the function obtained by restricting the domain of f to X. An operation f is a *polymorphism* of a relation R if for every $t_1, \ldots, t_k \in R$ it holds that $f(t_1, \ldots, t_k) \in R$, where f is applied coordinate-wise. In this case R is called *closed* or *invariant*, under f, ard $\mathrm{Inv}(F)$ denotes the set of all relations invariant under every function in F. Dually, for a set of relations Γ, $\mathrm{Pol}(\Gamma)$ denotes the set of all polymorphisms of Γ. Sets of the form $\mathrm{Pol}(\Gamma)$ are known as *clones*, and sets of the form $\mathrm{Inv}(F)$ are known as *co-clones*. A *clone* is a set of functions closed under 1) functional composition, and 2) projections (selecting an arbitrary but fixed coordinate). For a set B of (Boolean) functions, $[B]$ denotes the corresponding clone, and B is called its *base*. There is an inverse relationship between $\mathrm{Pol}(\Gamma)$ and $\mathrm{Inv}(F)$ but we defer the details to the supplemental material.

Complexity theory. We assume familiarity with basic notions in classical complexity theory (cf. [25]) and use complexity classes P, NP, coNP, $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{P}}$. In this paper we work in the setting of *parameterized* complexity where the *complexity parameter* is the number of variables n, in either an abduction or SAT instance, or the number of vertices in a graph problem. For a *variable problem* A we let $I(A)$ be the set of instances and $\mathrm{Var}(I)$ the set of variables. If clear from the context we usually write n rather than $|\mathrm{Var}(I)|$.We define the following two types of reductions [11] (note that ordinary polynomial-time reductions do not necessarily preserve the number of variables).

Definition 2. *Be A, B two variable problems. A function $f\colon I(A) \to I(B)$ is a* many-one linear variable reduction *(LV-reduction) with parameter $C \geq 0$ if I is a yes-instance of A iff $f(I)$ is a yes-instance of B, $|\mathrm{Var}(f(I))| = C \cdot |\mathrm{Var}(I)| + O(1)$, and f can be computed in polynomial time.*

If in an LV-reduction the parameter C is 1, we speak of a *CV-reduction*, and we take us the liberty to view a reduction which actually shrinks the number of variables ($|\mathrm{Var}(f(I))| \leq |\mathrm{Var}(I)| + O(1)$) as a CV-reduction, too. We use $A \leq^{\mathrm{CV}} B$, respectively $A \leq^{\mathrm{LV}} B$ as shorthands, and sometimes write $A =^{\mathrm{CV}} B$ if $A \leq^{\mathrm{CV}} B$ and $B \leq^{\mathrm{CV}} A$. For algorithms' (exponential) running time and space usage we adopt the O^* notation which suppresses polynomial factors. A CV-reduction transfers exact exponential running time: if $A \leq^{\mathrm{CV}} B$, and B can be solved in time $O^*(c^n)$, then also A can be solved in time $O^*(c^n)$ (where n denotes the complexity parameter). LV-reductions, on the other hand, preserve *subexponential complexity*, i.e., if B can be solved in $O^*(c^n)$ time for every $c > 1$ and $A \leq^{\mathrm{LV}} B$ then A is solvable in $O^*(c^n)$ time for every $c > 1$, too. For additional details we refer the reader to [13].

3 Upper Bounds via SAT Based Approaches

We begin by investigating the possibility of solving $\mathrm{ABD}(\Gamma)$ and $\text{P-ABD}(\Gamma)$ faster, conditioned by a reasonably efficient algorithm for $\mathrm{SAT}(\Gamma)$. This assumption is necessary to beat exhaustive search since $\mathrm{SAT}(\Gamma) \leq^{\mathrm{CV}}$ (P-)$\mathrm{ABD}(\Gamma)$, which implies that if $\mathrm{SAT}(\Gamma)$ is not solvable in $O(c^n)$ time for any $c < 2$ then (P-)$\mathrm{ABD}(\Gamma)$ is not solvable in $O(c^n)$ time for any $c < 2$, either.

For a constraint language Γ we let $\Gamma^+ = \Gamma \cup \{\bot, \top\}$, i.e., Γ expanded with the two constant Boolean relations. Trivially, we have $\mathrm{SAT}(\Gamma) \leq^{\mathrm{CV}} \mathrm{SAT}(\Gamma^+)$, and we observe that if $\mathrm{Pol}(\Gamma)$ does not contain a constant

operation then we additionally have $\mathrm{SAT}(\Gamma^+) \leq^{\mathrm{CV}} \mathrm{SAT}(\Gamma)$ [12]. We obtain the following baseline bound to beat.

Theorem 3. *(⋆) Let Γ be a constraint language such that $\mathrm{SAT}(\Gamma^+)$ is solvable in $f(n)$ time for some computable function $f\colon \mathbb{N} \to \mathbb{N}$. Then*

1. $\mathrm{ABD}(\Gamma)$ *is solvable in* $2^{|H|} \cdot f(n - |H|) \cdot (|M| + 1)$ *time,*
2. $\text{P-ABD}(\Gamma)$ *is solvable in* $\Sigma_{E \subseteq H} f(n - |E|) \cdot (|M| + 1)$ *time.*

Since every $\mathrm{SAT}(\Gamma^+)$ problem is solvable in $O^*(c^n)$ time for some $c \leq 2$ we get an overall bound of $2^{|H|} \cdot O^*(c^{n-|H|}) \in O^*(2^n)$ for $\mathrm{ABD}(\Gamma)$, and $\Sigma_{E \subseteq H} O^*(c^{n-|E|}) = O^*(c^{n-|H|} \cdot (c+1)^{|H|}) \in O^*((c+1)^n) \in O^*(3^n)$ for $\text{P-ABD}(\Gamma)$.

The question is now when these baseline bounds can be beaten. As a general method we (for both ABD and P-ABD) consider the assumption that all models of the knowledge base can be enumerated sufficiently fast.

Definition 4. *The set of models of a* $\mathrm{SAT}(\Gamma)$ *instance φ is denoted* $\mathrm{Mod}(\varphi)$*. If there exists $c < 2$ such that $|\mathrm{Mod}(\varphi)| \leq c^n$ then* $\mathrm{SAT}(\Gamma)$ *is said to be* sparse*.*

We also need the corresponding computational property where we require all models to be enumerable fast.

Definition 5. *Let Γ be a constraint language. If* $\mathrm{Mod}(\varphi)$*, for every* $\mathrm{SAT}(\Gamma)$ *instance φ, can be enumerated in $O^*(c^n)$ time for some $c < 2$ then we say that* $\mathrm{SAT}(\Gamma)$ *is* sparsely enumerable*.*

3.1 Faster Algorithms for ABD

We handle $\mathrm{ABD}(\Gamma)$ first since the analysis is simpler than for $\text{P-ABD}(\Gamma)$ (in Section 3.2). Trading polynomial for exponential space we consider a faster algorithm for $\mathrm{ABD}(\Gamma)$ under the condition that $\mathrm{SAT}(\Gamma)$ is sparsely enumerable.

We first state a technical lemma, facilitating the presentation of the algorithms and reductions throughout the following sections.

Lemma 6. *(⋆) When solving an abduction instance (KB, H, M), we can W.L.O.G. assume that $M, H \subseteq \mathrm{Var}(KB)$, via a polynomial time preprocessing. This constitutes even a CV-reduction of the problem to itself.*

Now the basic idea to solve $\mathrm{ABD}(\Gamma)$ is to define an equivalence relation $\equiv_H$ over $\mathrm{Mod}(\mathrm{KB})$ by letting $f \equiv_H g$ if and only if $f_{|H} = g_{|H}$ (where H is the hypothesis set). We then construct the equivalence classes of $\equiv_H$ and discard a class when it fails to explain M. An explanation exists if there is a non-empty class where every member satisfies M. Initially, this requires exponential space, $O^*(2^n)$. However, by only storing information on whether a potential explanation E has an extension that fails to satisfy M, space usage is reduced to $O^*(2^{|H|})$. The space usage is further limited by the enumerating algorithm's runtime, $O^*(c^n)$, resulting in total space usage bounded by $O^*(\min(c^n, 2^{|H|}))$. We obtain the following theorem.

Theorem 7. *(⋆) Let Γ be a constraint language where* $\mathrm{SAT}(\Gamma)$ *is sparsely enumerable in $O^*(c^n)$ time. Then* $\mathrm{ABD}(\Gamma)$ *is solvable in $O^*(c^n)$ time and $O^*(\min(c^n, 2^{|H|}))$ space.*

3.2 Faster Algorithms for P-ABD

In this section we present improved algorithms for brute force and enumeration for positive abduction, that have the same complexities as the symmetric variants described above. Recall that the baseline bound to beat for P-ABD (from Theorem 3) is $O^*(3^n)$, and we begin by lowering this to $O^*(2^n)$ via a more sophisticated exhaustive search scheme.

Algorithm 1 Algorithm $\mathcal{A}$ for P-ABD(Γ).

Require: KB, H, M, D, δ
1: $E = D \cup \delta$
2: $G = E \cup \{\neg x \mid x \in H - E\}$
3: **if** KB $\wedge$ $G \wedge \neg M$ is satisfiable **then**
4: **return** $\bot$
5: **if** KB $\wedge$ G is satisfiable **then**
6: **return** $\top$
7: **else**
8: **if** $D = \emptyset$ **then**
9: **return** $\bot$
10: flag = $\bot$
11: **for** $x \in D$ **do**
12: flag = flag $\vee$ $\mathcal{A}$(KB, $H, M, D - \{x\}, \delta$)
13: $\delta = \delta \cup \{x\}$
14: **return** flag

Theorem 8. ($\star$) *For any constraint language* Γ*,* P-ABD(Γ) *can be solved in* $O^*(2^n)$ *time and polynomial space.*

Proof. Consider Algorithm 3. The starting parameters are $D = H$ and $\delta = \emptyset$. The recursive algorithm systematically explores all subsets of H as candidates. It starts with the base candidate $E = H$. Inside each recursive call, it first checks if the current candidate E, extended to a full candidate G, entails the manifestation (line 3). If this fails, it concludes that neither G nor any subset of G (which includes E and all its subsets) can be an explanation, and returns $\bot$. Else, it checks if G is consistent with KB. If yes, then G is obviously a (full) explanation, but the algorithm concludes that in this case even $E \subseteq G$ is a (positive) explanation. If G is not consistent with KB, the algorithm concludes that neither E is consistent with KB, and can thus not be an explanation. Then the algorithm systematically checks candidates where exactly one variable is removed from E (lines 11–14). Descending in the recursive calls (line 12) it makes sure to systematically explore *all* subsets of a candidate, thereby avoiding visiting the same subset multiple times (this is the purpose of δ). For correctness, we refer to the supplementary material. □

We now consider an algorithm based on enumerating all models of the knowledge base, similar to the symmetric abduction case, by trading polynomial for exponential space.

Theorem 9. ($\star$) *Let* Γ *be a constraint language where* SAT(Γ) *is sparsely enumerable in* $O^*(c^n)$ *time. Then* P-ABD(Γ) *is solvable in* $O^*(c^n)$ *time and* $O^*(c^n)$ *space.*

3.3 Provably Sparse Languages

We have proved that (P-)ABD can be solved faster if all models of the underlying SAT problem can be enumerated sufficiently fast. Hence, it is highly desirable to classify the SAT problems where this is indeed the case — provided that any positive, non-trivial examples actually exist. We obtain a general characterization of such languages based on three abstract properties. Here, we always assume that a $\mathrm{SAT}(\Gamma)$ instance is represented by listing all tuples in a relation.

Definition 10. *A constraint language Γ is* asymptotically sparse *if there exists $r_0 \geq 1$ and $c < 2$ such that for r-ary $R \in \Gamma$, $r_0 \leq r$ we have $|R| \leq c^r$.*

If R is an n-ary relation and $g\colon [m] \to [n]$, for some $m \leq n$, then the relation $R_g(x_1, \ldots, x_m) \equiv R(x_{g(1)}, \ldots, x_{g(m)})$ is said to be a *minor* of R.

Definition 11. *For a constraint language Γ we let $\mathrm{Min}(\Gamma) = \{M \mid M \text{ is a minor of } R \in \Gamma\}$ be the set of minors of Γ.*

Similarly, if $R \in \Gamma$ of arity $\mathrm{ar}(R) = k$ and $f\colon X \to \{0,1\}$ for some $X \subseteq [k]$ then we define the *substitution* of f over R as the relation $R_{|f} = \{\mathrm{Proj}_{[k]-X}(t) \mid t = (c_1, \ldots, c_k) \in R, i \in X \Rightarrow c_i = f(i)\}$ where $\mathrm{Proj}_{[k]-X}(t)$ denotes the $(k - |X|)$-ary tuple obtained by only keeping indices in $[k]$ outside X.

Definition 12. *A language Γ is said to be* closed under branching *if it is closed under substitutions, i.e., if $R \in \Gamma$ and $f\colon X \to \{0,1\}$ for $X \subseteq [\mathrm{ar}(R)]$ then $R_{|f} \in \Gamma$, and is closed under minors, i.e., $\mathrm{Min}(\Gamma) = \Gamma$.*

We observe that Γ is finite if and only if $\mathrm{Min}(\Gamma)$ is finite.

Example 13. *Any language Γ can be closed under branching simply by repeatedly closing it under minors and substitutions. For example, consider 3-SAT and a positive clause corresponding to the relation $R = \{0,1\}^3 - \{(0,0,0)\}$. Then $\mathrm{Min}(\{R\}) = \{R, \{0,1\}^2 - \{(0,0)\}, \top\}$. If we close R under substitutions then we obtain $\{\{0,1\}^2 - \{(0,0)\}, \top, \mathtt{f}\}$ by identifying one or more variable to 0 and $\{\{0,1\}^2, \{0,1\}, \mathtt{t}\}$ by identifying one or more variable to 1.*

Thus, while it is easy to close a language Γ under branching, this process might introduce undesirable relations of the form $\{0,1\}^k$ which do not enforce any constraints.

Definition 14. *A relation $R \subset \{0,1\}^k$ is said to be* non-trivial. *The relation $\{0,1\}^k$ is said to be* trivial.

By combining these properties we obtain a novel characterization of sparsely enumerable languages.

Theorem 15. *($\star$) Let Γ be a constraint language which (1) is asymptotically sparse, (2) is closed under branching, and (3) every $R \in \Gamma$ is non-trivial. Then $\mathrm{SAT}(\Gamma)$ is sparsely enumerable.*

We continue by proving that such languages actually exist. First, say that a k-ary Boolean relation R is *totally symmetric*, or just *symmetric*, if there exists a set $S \subseteq [k] \cup \{0\}$ such that $(x_1, \ldots, x_k) \in R$ if and only if $x_1 + \ldots + x_k \in S$. Given $S \subseteq \{0, \ldots, k\}$ we write R_S for the symmetric relation induced by S, i.e., $R_S(x_1, \ldots, x_k) \equiv (\sum x_i \in S)$.

Definition 16. *We let* EQUATIONS $= \{R_S \mid k \geq 1, p, q \leq k+1, S = \{i \in [k] \cup \{0\} \mid i \equiv q \pmod p\}$.

Thus, each relation in EQUATIONS can be defined by an equation of the form $x_1 + \ldots + x_k = q \pmod p$ for fixed p, q, k. In particular $\mathrm{AFF} \subseteq$ EQUATIONS but it also contains relations inducing Σ_2^P-complete (P-)ABD problems. Perhaps contrary to intuition, EQUATIONS is *not* closed under minors, since the resulting relations are not necessarily symmetric, but a simple work-around is to fix a finite subset of EQUATIONS and then close it under minors. We obtain the following.

Lemma 17. ($\star$) *The following statements are true.*

1. EQUATIONS *is closed under substitutions and contains only non-trivial relations.*
2. *Let* $k \geq 1$. *For* $\mathcal{E}^{\leq k} = \{R \in$ EQUATIONS $\mid \mathrm{ar}(R) \leq k\}$ $\mathrm{SAT}(\mathrm{Min}(\mathcal{E}^{\leq k}))$ *is sparsely enumerable.*

Finally, let $\mathrm{XSAT} \subseteq$ EQUATIONS be the set of relations $R_{1/k}$ representable by equations $x_1 + \ldots + x_k = 1 \pmod k + 1$, and for each $k \geq 1$ the constant relation $\bot^k = \{(0, \ldots, 0)\}$ of arity k (note also that $R_{1/1} = \top$), and, finally, the two nullary relations f and t. The resulting problem $\mathrm{SAT}(\mathrm{XSAT})$ is thus the natural generalization of the well-known NP-complete problem 1-IN-3-SAT to arbitrary arities. Also, recall that $\mathrm{AFF} \subseteq$ EQUATIONS is the set of relations representable by systems of Boolean equations modulo 2, i.e., $x_1 + \ldots + x_k \equiv q \pmod 2$ for $q \in \{0, 1\}$. We additionally write $\mathrm{AFF}^{\leq k}$ for the set of affine relations of arity at most k.

Theorem 18. ($\star$) *The following statements are true.*

1. $\mathrm{SAT}(\mathrm{XSAT})$ *is sparsely enumerable in* $O^*(\sqrt{2}^n)$ *time.*
2. $\mathrm{SAT}(\mathrm{AFF}^{\leq k})$ *is sparsely enumerable for every* $k \geq 1$.

Let us also observe that the bound for $\mathrm{SAT}(\mathrm{XSAT})$ is tight in the sense that $|\mathrm{Mod}(\varphi)| = \sqrt{2}^n$, $n = 2m$, if φ encodes inequalities between x_1 and x_2, x_3 and x_4, and so on.

3.4 Algorithms for NP-complete fragments

Symmetric abduction is NP-complete for k-CNF$^+$ languages for any $k \geq 2$, and both symmetric and positive abduction are NP-complete for languages of the form k-CNF$^-$ $\cup$ IMP, $k \geq 2$ [17]. We show in the following that these cases can be solved in improved time.

Definition 19. *Denote by* SIMPLESAT^p *the SAT-problem where we are given a formula* φ *of the following form. Here,* C *stands for a positive clause of size at most* p *and* T *stands for a negative term of any size.*

$$\varphi = \bigwedge C \wedge \bigwedge \bigvee T$$

Lemma 20. SIMPLESAT^p *can be solved in time* $O^*(c^n)$*, where* c *is the branching factor associated with a* $(1, \ldots, p)$*-branching.*

Proof. Perform a branch and reduce scheme. Variables not occurring in any positive clause can be reduced to 0 (thereby simplifying some of the negative terms). Then branch on the variables of positive clauses, with the standard $(1, \ldots, p)$-branching. □

We are now ready to show that ABD(k-CNF$^+$) can be CV-reduced to SIMPLESAT^k.

Theorem 21. *(⋆)* $\mathrm{ABD}(k\text{-}CNF^+) \leq^{\mathrm{CV}} \mathrm{SIMPLESAT}^k$. *Moreover, the* SIMPLESAT*-instance contains only variables from* H.

We show that (P-)ABD(k-CNF$^-$ ∪ IMP) can be CV-reduced to ABD(k-CNF$^+$).

Lemma 22. *(⋆)* (P-)$\mathrm{ABD}(k\text{-}CNF^- \cup IMP) \leq^{\mathrm{CV}} \mathrm{ABD}(k\text{-}CNF^+)$.

The following corollary finally states the improved results, following immediately from the previous statements.

Corollary 23. $\mathrm{ABD}(k\text{-}CNF^+)$ *and* (P-)$\mathrm{ABD}(k\text{-}CNF^- \cup IMP)$ *can be solved in improved time. That is, in time* $O^*(c^{|H|})$*, for a* $c < 2$*, stemming from the branching vector* $(1, \ldots, k)$.

3.5 Algorithms for coNP-complete fragments

In the case of positive abduction coNP-complete cases arise [17] when Γ is 1-valid. Under certain additional assumptions, P-$\mathrm{ABD}(\Gamma)$ can then be solved in improved time.

Theorem 24. *Let* Γ *be a 1-valid constraint language. If* $\mathrm{SAT}(\Gamma^+)$ *can be decided in* $O^*(c^n)$ *time for* $c \leq 2$ *then* P-$\mathrm{ABD}(\Gamma)$ *can be decided in* $O^*(c^n)$ *time.*

Proof. First, note that the 1-valid property is responsible for coNP-membership: there is an explanation iff H is an explanation. Then KB $\wedge H$ is always consistent (since 1-valid). Thus, we only need to check whether KB $\wedge$ H $\models$ M. This implication can be decided by invoking the given $\mathrm{SAT}(\Gamma^+)$ algorithm $|M|$ times: KB $\wedge H \models M$ iff for each $m \in M$ the Γ^+-formula KB $\wedge H \wedge \neg m$ is unsatisfiable, and we thus only increase the $O^*(c^n)$ complexity by a polynomial factor. □

An application example is a knowledge base in k-CNF (k ≥ 3) where each clause contains at least 1 positive literal (this ensures 1-validity). The underlying constraint language Γ is still expressive enough to render P-$\mathrm{ABD}(\Gamma)$ coNP-hard [17]. Then, any Γ^+-formula is expressible as k-CNF formula (without additional variables), so $\mathrm{SAT}(\Gamma^+)$ can be solved in improved time via the standard $(1, \ldots, k)$-branching. With Theorem 24 we conclude that P-$\mathrm{ABD}(\Gamma)$ can be solved in improved time.

4 Lower bounds and Reductions

We continue by matching our new upper bounds with lower bounds. We base our lower bounds on ETH and its stronger variant SETH (recall the definition in Section 1).

4.1 ETH Based Lower Bounds

Our aim in this section is to prove the following theorem.

Theorem 25. $\mathrm{ABD}(2\text{-}CNF^+)$ *and* $\mathrm{ABD}(2\text{-}CNF^- \cup IMP)$ *cannot be solved in time* $(\frac{|H|}{|M|})^{o(|M|)}$ *under ETH.*

Proof. We provide a CV-reduction from the k-*colored clique* problem to $\mathrm{ABD}(\text{2-CNF}^- \cup \mathrm{IMP})$, i.e., given a graph $G = (V, E)$ where the vertices are colored with k different colors, decide if a clique containing a vertex of each color exists. It is known that k-colored clique cannot be solved in time $(\frac{n}{k})^{o(k)}$ under ETH, where n is the number of vertices, and k is the number of colors [16].

For the reduction, assume an arbitrary instance of a k-colored clique problem over a graph $G = (V, E)$, and where $C = c_i \mid i \in [1 \dots k]$ is the set of colors. For each color c_i we add a manifestation m_i to M. For each vertex v_i colored with the color c_i we add to KB the clause $(\neg v_i \vee m_i)$. For each two vertices v_i and v_j that are not connected by an edge, we add the clause $(\neg v_i \vee \neg v_j)$. This completes the reduction. Since the number of variables in the abduction problem is equal to the number of vertices plus the number of colors, this is a CV-reduction. If we can choose a clique that contains at least one vertex from each color without choosing two vertices that are not connected, then for each chosen vertex v_i, we set $v_i = \top$ in the abduction instance as part of the solution. This will entail all m_i's without causing a contradiction in KB. The correctness proof from the other side is exactly the same.

Last, by Lemma 22 we additionally obtain that $\mathrm{ABD}(\text{2-CNF}^- \cup \mathrm{IMP}) \leq^{\mathrm{CV}} \mathrm{ABD}(\text{2-CNF}^+)$. □

Note that this lower bound is given with respect to $|H|$ and $|M|$ and not n, and it is identical to the running time of the brute force algorithms of these problems. Indeed, in the worst case, we try all possible combinations of hypotheses for all manifestations. If a hypothesis appears for two manifestations at the same time, it only reduces the need to choose an additional hypothesis for the second one. The worst case is then when all hypotheses are split up among manifestations. The number of combinations is thus $(\frac{|H|}{|M|})^{|M|}$, making the algorithm close to optimal.

4.2 Lower Bounds for (P-)ABD(4-CNF) under SETH

Here, we prove a strong lower bound under SETH which shows that we should only expect small improvements for 4-CNF, and, more generally, k-CNF for any $k \geq 4$.

Theorem 26. $(\star)$ *Under SETH, there is no* $c < 1$ *such that* $\mathrm{ABD}(\textit{4-CNF})$ *is solvable in* 2^{cn} *time.*

The following lemma gives the slightly worse lower bound for positive abduction variant of 4-CNF.

Lemma 27. $(\star)$ *Under SETH, there is no* $c < 1$ *such that* $\text{P-}\mathrm{ABD}(\textit{4-CNF})$ *can be solved in time* 1.4142^{cn}.

We remark that this leaves 3-CNF as an interesting open case. We have no algorithms that run in less than 2^n so far, and no known lower bounds under SETH.

4.3 Languages closed under complement

We analyze languages closed only under complement, i.e., languages Γ such that $\mathrm{Pol}(\Gamma) = [\bar{x}]$. Well-known examples of such languages include *not-all-equal* satisfiability. For this class of languages we manage to relate them to languages closed only under projections, in a very precise sense, and show that one without loss of generality can concentrate solely on the latter class.

Theorem 28. *(*$\star$*) Let* Γ *be a constraint language such that Pol*$(\Gamma) = [\bar{x}]$. *Then* $\text{(P-)}\mathrm{ABD}(\Gamma \cup \{\bot, \top\}) =^{\mathrm{CV}} \text{(P-)}\mathrm{ABD}(\Gamma)$.

For the next theorem, let k-NAE be the set of all k-ary relations that forbid exactly two complementary assignments (determined by a sign pattern). The full details can be found in the supplementary material.

Theorem 29. *(⋆)* (P-)ABD(*k-CNF*) $\leq^{\text{CV}}$ (P-)ABD(*(k+1)-NAE*).

Thus, in general we should not expect to solve complementative abduction problems faster than 2^n (via Theorem 26).

4.4 Lower Bounds for Horn and CNF$^+$ Abduction

Last, we prove strong lower bounds for Horn and CNF$^+$. These lower bounds do not conclusively rule out algorithms faster than 2^n but we stress that sharp lower (under SETH) and upper bounds of the form c^n are rather uncommon in the literature, and an improved upper bound would likely need to use completely new ideas.

Lemma 30. *(⋆) Under SETH, there is no $c < 1$ such that (P-)*ABD(*CNF$^-$ ∪ IMP*) *can be solved in time* 1.2599^{cn} *or* $1.4142^{c|H|}$ *or* $2^{c|M|}$ *or* $\left(\frac{|H|}{|M|}\right)^{c|M|}$.

Since 2-CNF$^-$ ∪ IMP is a special case of Horn, we obtain the same lower bounds for Horn knowledge bases. Furthermore, using the reduction from Lemma 22, we obtain a CV-reduction from (P-)ABD(CNF$^-$ ∪ IMP) to ABD(CNF$^+$), and CNF$^+$ in turn is a special case of DualHorn. We therefore obtain the following Corollary.

Corollary 31. *Under SETH, there is no $c < 1$ such that (P-)*ABD*(Horn) as well as* ABD(*CNF$^+$*) *and* ABD(*DualHorn*) *can be solved in time* 1.2599^{cn} *or* $1.4142^{c|H|}$ *or* $2^{c|M|}$ *or* $\left(\frac{|H|}{|M|}\right)^{c|M|}$.

An interesting question that might occur is whether or not we can have SETH lower bounds for the problems in Section 4.1. Those problems instead have weaker lower bounds from ETH. The following lemma will show that we are unlikely to obtain such bounds with the same method we obtained them for the problems in this Section (4.4).

Lemma 32. *(⋆) If CNF-SAT* $\leq^{\text{LV}}$ ABD(*2-CNF*) *then NP ⊆ P/Poly.*

Proof. We provide a short proof sketch. From [6], we know that CNF-SAT does not admit a kernel of size $f(n)$ where f is a polynomial, i.e., an equisatisfiable instance with e.g. a quadratic number of clauses (unless NP ⊆ P/Poly). To prove the claim it suffices to have a reduction from ABD(2-CNF) to CNF-SAT that only has a polynomial number of clauses in terms of n. Assume an arbitrary ABD(2-CNF) instance. For every $m_i \in M$, for all clauses of the type: $(\ell_j \vee m_i), \ldots, (\ell_k \vee m_i)$ where $\ell_i \ldots \ell_k$ are a literals, we delete these clauses and add the following CNF clause $(\ell_i \vee \ldots \vee \ell_k)$, and otherwise keep everything unchanged. □

It is thus unlikely to obtain SETH lower bounds for ABD(2-CNF) from CNF-SAT under LV- or CV-reductions.

5 Concluding Remarks

We demonstrated that non-monotonic reasoning, and in particular propositional abduction, for many non-trivial cases *do* admit improvements over exhaustive search. We find it particularly interesting that even Σ_2^P-complete problems fall under the scope of our methods. Might it even be the case that Σ_2^P is not such an imposing barrier as classical complexity theory tells us? Nevertheless, despite many positive and negative results, there are still open cases remaining and many interesting directions for future research.

Faster Enumeration? We proved that finite subsets of AFF and EQUATIONS are susceptible to enumeration. It is easy to see that $\mathrm{Pol}(\mathrm{AFF})$ contains the *Maltsev* operation $x - y + z(\pmod 2)$, while EQUATIONS is exactly the set of symmetric relations invariant under a *partial Maltsev* operation [14]. Is it a coincidence that all of our positive enumeration cases can be explained by partial Maltsev operation, or could universal algebra be applied even further? For example, it is straightforward to show that if a language is *not* preserved by partial Maltsev, then it can not be sparsely enumerable. Extending this further, if one allows e.g. a polynomial-time preprocessing, could it even be the case that a Boolean (possibly non-symmetric) language is sparsely enumerable if and only if it is invariant under partial Maltsev?

2- and 3-CNF. While (P-)$\mathrm{ABD}(4\text{-CNF})$ is unlikely to admit improved algorithms, (P-)$\mathrm{ABD}(k\text{-CNF})$ for $k \leq 3$ is wide open. These languages are not sparsely enumerable so they do not fall under the scope of the enumeration algorithms, yet, it appears highly challenging to prove sharp lower bounds for them (and recall that CNF-SAT does not admit an LV-reduction to (P-)$\mathrm{ABD}(2\text{-CNF})$ unless NP $\subseteq$ P/Poly). As a possible starting point one could consider instances with only a linear number m of clauses in the knowledge base, or, the more restricted case when each variable may only occur in a fixed number of constraints. Could instances of this kind be solved with enumeration?

Other Parameters? Related to the above question one could more generally ask when (P-)$\mathrm{ABD}(\Gamma)$ admits an improved algorithm with complexity parameter m, which we observe in general can be much larger than n. Do any of the algorithmic results carry over, and can lower bounds be obtained? For the related quantified Boolean formula problems, Williams [28] constructed an $O(1.709^m)$ time algorithm, so one could be cautiously optimistic about analyzing abduction with m.

Appendix

For completeness' sake we include the full preliminaries, although most of these notions have already been defined in the main paper. Section 2 contains all missing proofs.

1 Preliminaries

Propositional logic We assume familiarity with propositional logic. A *literal* is a variable x or its negation $\neg x$. A *clause* is a disjunction of literals and a *term* is a conjunction of literals. A formula φ is in *conjunctive normal form* (CNF) if it is a conjunction of clauses, and in *disjunctive normal form* (DNF) if it is a disjunction of terms. We denote $\mathrm{Var}(\varphi)$ the variables of a formula φ. Analogously, for a set of formulas F, $\mathrm{Var}(F)$ denotes $\bigcup_{\varphi \in F} \mathrm{Var}(\varphi)$. We identify finite F with the conjunction of all formulas from F, that is $\bigwedge_{\varphi \in F} \varphi$. A mapping $\sigma : \mathrm{Var}(\varphi) \mapsto \{0,1\}$ is called an *assignment* to the variables of φ. A *model* of a formula φ is an assignment to $\mathrm{Var}(\varphi)$ that satisfies φ. For two formulas ψ, φ we write $\psi \models \varphi$ if every model of ψ also satisfies φ. For a set of variables V, $|V|$ denotes the *cardinality* of V, and $\mathrm{Lits}(V)$ the set of all literals formed upon V, that is, $\mathrm{Lits}(V) = V \cup \{\neg x \mid x \in V\}$.

Boolean constraint languages A *logical relation* of arity $k \in \mathbb{N}$ is a relation $R \subseteq \{0,1\}^k$, and $\mathrm{ar}(R) = k$ denotes the arity. An (R-)constraint C is a formula $C = R(x_1, \ldots, x_k)$, where R is a k-ary logical relation, and $x_1, \ldots, x_k$ are (not necessarily distinct) variables. An assignment σ *satisfies* C, if $(\sigma(x_1), \ldots, \sigma(x_k)) \in R$. A (Boolean) *constraint language* Γ is a (possibly infinite) set of logical relations, and a Γ-*formula* is a conjunction of constraints over elements from Γ. Eventually, a Γ-formula ϕ is *satisfied* by an assignment σ, if σ simultaneously satisfies all constraints in it. In such a case σ is also called a *model of* ϕ. We define the two constant unary Boolean relations as $\bot = \{(0)\}$ and $\top = \{(1)\}$, and the two constant 0-ary relations as $\mathsf{f} = \emptyset$ and $\mathsf{t} = \{\emptyset\}$.

We say that a k-ary relation R is *represented* by a propositional CNF-formula ϕ if ϕ is a formula over k distinct variables $x_1, \ldots, x_k$ and $\phi \equiv R(x_1, \ldots, x_k)$. Note such a CNF-representation exists for any logical relation $R \subseteq \{0,1\}^k$ and hence for any R-constraint. Any Γ-formula (a conjunction of Γ-constraints) can thus be seen as a propositional formula and admits a CNF-representation.

We note at this point that many classical fragments of propositional logic such as e.g. Horn, dualHorn, affine, k-CNF, can be modeled via the notion of Γ-formulas. For instance, $\Gamma = \{\{0,1\}^2 - \{(0,0)\}, \{0,1\}^2 - \{(0,1)\}, \{0,1\}^2 - \{(1,0)\}, \{0,1\}^2 - \{(1,1)\}\}$ models exactly 2-CNF formulas.

We use the shorthand k-CNF to denote the corresponding finite constraint language (arity bounded by k). IMP denotes the constraint language $\{R\}$, where $R(x,y) = \{0,1\}^2 - \{(1,0)\}$. We use the shorthands CNF, Horn, DualHorn to denote the corresponding infinite constraint languages (unbounded arity). The notation (k-)CNF$^+$ (resp. (k-)CNF$^-$) denotes the version where each clause is positive (resp. negative). We use AFF to denote the set of relations representable by systems of Boolean equations modulo 2, i.e., $x_1 + \ldots + x_k \equiv q \pmod 2$ for $q \in \{0,1\}$. As representation of each relation in a constraint language we use the defining CNF-formula.

The satisfiability problem for Γ-formulas, also known as Boolean constraint satisfaction problem, is denoted $\mathrm{SAT}(\Gamma)$. It asks, given a Γ-formula ϕ, whether ϕ admits a model. It is well-known that $\mathrm{SAT}(\Gamma)$ is decidable in polynomial time if Γ is Horn, or dualHorn, or 2-CNF, or affine, or 1-valid, or 0-valid, but in all other cases is NP-complete [24].

Propositional Abduction An instance of the abduction problem is given by (KB, H, M), where KB denotes the knowledge base (or *theory*), H is the set of hypotheses, and M is the set of manifestations. In the propositional case KB is given as a (set of) propositional formula(s), and both H and M are sets of variables. If not noted otherwise, we denote by V the total set of variables in an abduction instance, that is, $V = \mathrm{Var}(\mathrm{KB}) \cup \mathrm{Var}(H) \cup \mathrm{Var}(M)$, and $n = |V|$ denotes its cardinality. The *symmetric abduction problem*, denoted ABD, asks whether there exists an $E \subseteq \mathrm{Lits}(H)$ such that 1) $\mathrm{KB} \wedge E$ is satisfiable, and 2) $\mathrm{KB} \wedge E \models M$. If such an E exists, it is called an *explanation* for M. If an explanation E satisfies $\mathrm{Var}(E) = H$ it is called a *full explanation*, if it satisfies $E \subseteq H$, it is called a *positive explanation*. If there exists an explanation E, it can always be extended to a full explanation (by extending E according to the satisfying assignment underlying condition 1). However, the existence of an explanation does not imply the existence of a positive explanation. The *positive abduction problem*, denoted P-ABD, asks whether there exists a positive explanation.

Example 33. *Propositional abduction.*

$$\begin{aligned} \mathit{KB} = \{&\mathit{Charly\text{-}lazy} \wedge \mathit{Charly\text{-}alone} \rightarrow \mathit{Charly\text{-}truant},\\ &\mathit{Dad\text{-}shopping} \rightarrow \mathit{Charly\text{-}alone},\\ &\neg\mathit{Buses\text{-}running} \rightarrow \mathit{Charly\text{-}truant} \wedge \neg\mathit{Dad\text{-}shopping}\}\\ H = \{&\mathit{Buses\text{-}running}, \mathit{Dad\text{-}shopping}, \mathit{Charly\text{-}lazy}\}\\ M = \{&\mathit{Charly\text{-}truant}\} \end{aligned}$$

The set $\{\mathit{Charly\text{-}lazy}, \mathit{Charly\text{-}alone}\}$ *would explain the manifestation Charly-truant, but is invalid, since Charly-alone is no hypothesis. However,* $E_1 = \{\neg\mathit{Buses\text{-}running}\}$ *is a valid explanation. Note that* E_1 *is neither a positive nor a full explanation.* $E_2 = \{\neg\mathit{Buses\text{-}running}, \neg\mathit{Dad\text{-}shopping}, \mathit{Charly\text{-}lazy}\}$ *is a full explanation, and* $E_3 = \{\mathit{Dad\text{-}shopping}, \mathit{Charly\text{-}lazy}\}$ *is a positive explanation.*

We define the symmetric abduction problem for Γ-formulas, denoted $\mathrm{ABD}(\Gamma)$, exactly as ABD, but the knowledge base KB is given as a Γ-formula. $\mathrm{P\text{-}ABD}(\Gamma)$ is defined analogously and we write $\mathrm{(P\text{-})ABD}(\Gamma)$ if the specific abduction type is not important. The classical complexity of $\mathrm{ABD}(\Gamma)$ and $\mathrm{P\text{-}ABD}(\Gamma)$ is fully determined for all Γ, due to [17]. An illustration of these classifications is found in Figure 1 and Figure 2.

Clones, Co-clones, Post's lattice, polymorphisms We denote by $\bar{x}$ the Boolean negation operation, that is, $f(x) = \neg x$. An n-ary *projection* is an operation f of the form $f(x_1, \ldots, x_n) = x_i$ for some fixed $1 \leq i \leq n$. An operation $f : \{0,1\}^k \rightarrow \{0,1\}$ is *constant* if for all $\mathbf{x} \in \{0,1\}^k$ it holds $f(\mathbf{x}) = c$, for a $c \in \{0,1\}$. For a k-ary operation $f\colon \{0,1\}^k \rightarrow \{0,1\}$ and $X \subseteq \{0,1\}^k$ we write $f_{|X}$ for the function obtained by restricting the domain of f to X. An operation f is a *polymorphism* of a relation R if for every $t_1, \ldots, t_k \in R$ it holds that $f(t_1, \ldots, t_k) \in R$, where f is applied coordinate-wise. In this case R is called *closed* or *invariant*, under f, ard $\mathrm{Inv}(F)$ denotes the set of all relations invariant under every function in F. Dually, for a set of relations Γ, $\mathrm{Pol}(\Gamma)$ denotes the set of all polymorphisms of Γ. Sets of the form $\mathrm{Pol}(\Gamma)$ are known as *clones*, and sets of the form $\mathrm{Inv}(F)$ are known as *co-clones*. A *clone* is a set of functions closed under 1) functional composition, and 2) projections (selecting an arbitrary but fixed coordinate). For a set B of (Boolean) functions, $[B]$ denotes the corresponding clone, and B is called its *base*. In the Boolean domain clones and their inclusion structure are fully determined, commonly known as *Post's lattice* [20]. Interestingly, the relationship between clones and co-clones constitutes a *Galois connection* [15].

Theorem 34. *Let Γ, Γ' be constraint languages. Then Inv(Pol(Γ')) $\subseteq$ Inv(Pol(Γ)) if and only if Pol(Γ) $\subseteq$ Pol(Γ')*

Post's lattice therefore also completely describes co-clones and their inclusion structure. A depiction of Post's lattice as co-clones is found in figure 1. Informally explained, every vertex corresponds to a co-clone while the edges model the containment relation. Co-clones are equivalently characterized by constraint languages (= sets of Boolean relations) closed under primitive positive first-order definitions (pp-definitions), that is, definitions of the form

$$R(x_1, \ldots, x_n) := \exists y_1, \ldots, y_m \,.\, R_1(\mathbf{x}_1) \wedge \cdots \wedge R_k(\mathbf{x}_k)$$

where each $R_i \in \Gamma \cup \{\mathrm{Eq}\}$ and each $\mathbf{x}_i$ is a tuple over $x_1, \ldots, x_n, y_1, \ldots, y_m$, and $\mathrm{Eq} = \{(0,0),(1,1)\}$. The corresponding closure operator is denoted $\langle\cdot\rangle$, that is, for a constraint language Γ, $\langle\Gamma\rangle$ is the co-clone $\mathrm{Inv}(\mathrm{Pol}(\Gamma))$ and Γ is called its *base*. Quantifier-free pp-definitions (qfpp-definitions) are defined analogously, but existential quantification is disallowed.

Complexity, algorithms, reductions We assume familiarity with basic notions in classical complexity theory (cf. [25]) and use complexity classes P, NP, coNP, $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{P}}$. Note that hardness results (e.g. NP-hardness) in this context are obtained via polynomial many-one reductions. This type of reduction is, however, not suitable to transfer exact exponential running time, and neither subexponential running time. To consider exponential time algorithms, a *complexity parameter* needs to be specified. For this paper, we consider as parameter the *total number of variables* in an instance (e.g. an abduction or SAT instance). For a variable problem A we denote $I(A)$ the set of instances and $\mathrm{Var}(I)$ denotes the set of variables. If clear from the context we usually write n rather than $|\mathrm{Var}(I)|$. Thus, formally, we work in the setting of *parameterized* complexity where the complexity parameter k is n, the number of variables in the knowledge base in an abduction problem, or the number of vertices in a graph problem. We define the following two types of reductions [11].

Definition 35. *Be A, B two variable problems. A function $f\colon I(A) \to I(B)$ is a* many-one linear variable reduction *(LV-reduction) with parameter $C \geq 0$ if:*

1. *I is a yes-instance of A iff $f(I)$ is a yes-instance of B,*
2. *$|\mathrm{Var}(f(I))| = C \cdot |\mathrm{Var}(I)| + O(1)$, and*
3. *f can be computed in polynomial time.*

If in an LV-reduction the parameter C is 1, we speak of a *CV-reduction*, and we take us the liberty to view a reduction which actually shrinks the number of variables ($|\mathrm{Var}(f(I))| \leq |\mathrm{Var}(I)| + O(1)$) as a CV-reduction, too. We use $A \leq^{\mathrm{CV}} B$, respectively $A \leq^{\mathrm{LV}} B$ as shorthands, and sometimes write $A =^{\mathrm{CV}} B$ if $A \leq^{\mathrm{CV}} B$ and $B \leq^{\mathrm{CV}} A$. For algorithms' (exponential) running time and space usage we adopt the O^* notation which suppresses polynomial factors. A CV-reduction transfers exact exponential running time: if $A \leq^{\mathrm{CV}} B$, and B can be solved in time $O^*(c^n)$, then also A can be solved in time $O^*(c^n)$ (where n denotes the complexity parameter). LV-reductions, on the other hand, preserve *subexponential complexity*, i.e., if B can be solved in $O^*(c^n)$ time for every $c > 1$ and $A \leq^{\mathrm{LV}} B$ then A is solvable in $O^*(c^n)$ time for every $c > 1$, too. For additional details we refer the reader to [13].

2 Missing Proofs

2.1 Proof of Theorem 3

Proof. Let (KB, H, M) be an instance of $\text{ABD}(\Gamma)$. We first observe that we without loss of generality may assume that $\text{Var}(E) = H$. We then exhaustively enumerate all full explanations $E \subseteq \text{Lits}(H)$ ($2^{|H|}$ time since we do not have to consider E containing both a variable and its complement) and use the algorithm for $\text{SAT}(\Gamma^+)$ to

1. check whether $\text{KB} \wedge E$ is satisfiable, and
2. check whether $\text{KB} \wedge E \wedge \neg m_i$ is unsatisfiable for each $m_i \in M$.

Since $\text{Var}(E) = H$, each instance $\text{KB} \wedge E$ effectively only has $n - |H|$ variables, we therefore get the bound $2^{|H|} \cdot f(n - |H|) \cdot (|M| + 1)$. Note that checking satisfiability of $\text{KB} \wedge E$ can be done by the presumed algorithm for $\text{SAT}(\Gamma^+)$ by forcing variables in E constant values via $\bot$- and $\top$-constraints.

For P-$\text{ABD}(\Gamma)$ the analysis is similar but where in each invocation of the algorithm for $\text{SAT}(\Gamma^+)$ we have $n - |E|$ variables. □

2.2 Proof of Lemma 6

Proof. Assume there is an $m \in M - \text{Var}(\text{KB})$. If $m \notin H$, there are no explanations. An algorithm returns False and terminates, a reduction maps to a fixed negative instance. If $m \in H$, then m can trivially be explained (by itself). It can thus be removed from both H and M, since it does not influence the existence of explanations.

Now assume there is an $h \in H - \text{Var}(\text{KB})$. If $h \notin M$, it can be removed from H, since it does not influence the existence of explanations. If $h \in M$, it can be removed from both H and M, since it does not influence the existence of explanations.

Since no variables are added, the described transformations constitute a CV-reduction. □

2.3 Proof of Theorem 7

Proof. Consider an instance (KB, H, M) of $\text{ABD}(\Gamma)$. Recall that by Lemma 6 we can W.L.O.G. assume that $M, H \subseteq \text{Var}(\text{KB})$. Let $|\text{Var}(\text{KB})| = n$.

The algorithm is presented in Algorithm 2 where we assume that $\text{modelGenerator}(\text{KB})$ can be used to enumerate all solutions to KB via the $.\text{next}()$ notation. For soundness, we first recall that there exists an explanation $E \subseteq \text{Lits}(H)$ if and only if there exists full explanation. The algorithm enumerates all models σ of KB and in line 6 extracts a full explanation candidate E from σ. This E serves as identifier of σ's equivalence class. An equivalence class is discarded if it fails (via the representative σ) to explain M. This case is detected in line 8. For the complexity, we assume that all models of $\text{SAT}(\Gamma)$ can be enumerated in $O^*(c^n)$ time, which bounds the total number of iterations. Each step *after* line 5, that is, lines 6-12, can be done in polynomial time, and all checks can be implemented with standard data structures. Note that the check in line 8 amounts to simply evaluating M on the assignment σ. □

Algorithm 2 Algorithm to solve $\mathrm{ABD}(\Gamma)$ based on enumeration.

Require: KB, H, M

generator = modelGenerator(KB)
discarded = $\emptyset$
potentialExp = $\emptyset$
while generator.notEmpty() **do**
  σ = generator.next()
  $E = \{x \in H \mid \sigma(x) = 1\} \cup \{\neg x \mid x \in H, \sigma(x) = 0\}$
  if $E \notin$ discarded **then**
    if $\sigma \not\models M$ **then**
      discarded = discarded $\cup \{E\}$
      potentialExp = potentialExp $- \{E\}$
    else
      potentialExp = potentialExp $\cup \{E\}$
 # now potentialExp contains all full explanations
if potentialExp $\neq \emptyset$ **then**
  return $\top$
else
  return $\bot$

2.4 Proof of Theorem 8

Algorithm 3 Algorithm $\mathcal{A}$ for P-$\mathrm{ABD}(\Gamma)$.

Require: KB, H, M, D, δ

$E = D \cup \delta$
$G = E \cup \{\neg x \mid x \in H - E\}$
if KB $\wedge\, G \wedge \neg M$ is satisfiable **then**
  return $\bot$
if KB $\wedge\, G$ is satisfiable **then**
  return $\top$
else
  if $D = \emptyset$ **then**
    return $\bot$
  flag = $\bot$
  for $x \in D$ **do**
    flag = flag $\vee\, \mathcal{A}$(KB, $H, M, D - \{x\}, \delta$)
    $\delta = \delta \cup \{x\}$
  return flag

Proof. Consider Algorithm 3. The starting parameters are $D = H$ and $\delta = \emptyset$.

The recursive algorithm systematically explores all subsets of H as candidates. It starts with the base candidate $E = H$. Inside each recursive call, it first checks if the current candidate E, extended to a full

candidate G, entails the manifestation (line 3). If this fails, it concludes that neither G nor any subset of G (which includes E and all its subsets) can be an explanation, and returns $\perp$. Else, it checks if G is consistent with KB. If yes, then G is obviously a (full) explanation, but the algorithm concludes that in this case even $E \subseteq G$ is a (positive) explanation. In claim 37 below we argue why this is correct. If G is not consistent with KB, the algorithm concludes that neither E is consistent with KB, and can thus not be an explanation. In claim 36 below we argue why this is correct. Then the algorithm systematically checks candidates where exactly one variable is removed from E (lines 11-13). Descending in the recursive calls (line 12) it makes sure to systematically explore *all* subsets of a candidate, thereby avoiding visiting the same subset multiple times (this is the purpose of δ).

Claim 36. *If in line 5 of the algorithm KB $\wedge$ G is unsatisfiable, then so is KB $\wedge$ E.*

Proof. We use an inductive argument. At the initial call we have that $E = G = H$, so the statement is obviously true. Now, consider the case where E is a strict subset of H. By construction of the algorithm we know that any strict superset F of E must have failed as explanation candidate (otherwise no recursive call would have descended to the current point). Since a failure due to line 3 stops descending, the failure (of F) must in this case have been due to line 5. We state this as the following observation.

$$\text{If } F \supset E, \text{ then KB} \wedge F \text{ is unsatisfiable.} \tag{1}$$

Now assume for contradiction that KB $\wedge$ E is satisfiable. Then there must be a witnessing satisfying assignment σ. We define $C = \{x \mid x \in H - E, \sigma(x) = 1\}$ and conclude that KB $\wedge$ $E \wedge C$ must be satisfiable (by σ). If C is non-empty, then $E \cup C$ is a strict superset of E, thus by observation 1, KB $\wedge$ $E \wedge C$ must be unsatisfiable, a contradiction. If C is empty, σ assigns 0 to all variables in $H - E$. Therefore, σ satisfies KB $\wedge$ G, a contradiction.

□

Claim 37. *If in line 3 of the algorithm KB $\wedge$ $G \wedge \neg M$ is unsatisfiable, then so is KB $\wedge$ $E \wedge \neg M$.*

Proof. As in claim 36 we obtain the same observation 1. The remaining reasoning proceeds exactly analogously, only with KB $\wedge \neg M$, instead of KB. □

It remains to observe the claimed time and space complexity. In the worst case the algorithm goes through all subsets of H, and checks satisfiability in the remaining $V - H$ variables. Its time complexity is thus $O^*(2^{|H|}) \cdot O^*(2^{|V-H|}) = O^*(2^n)$. The descending depth is linear and no set can grow to exponential size. The overall space usage is therefore polynomial. □

2.5 Proof of Theorem 9

Proof. Similar to the enumeration scheme of Algorithm 2, we use a generator with the $.\mathrm{next}()$ notation to enumerate all models of the knowledge base KB. But this time we require a specific order. For an assignment σ we define the *weight with respect to* H as $w_H(\sigma) = |\{x \mid x \in H, \sigma(x) = 1\}|$. We require the generator to respect the order of non-increasing $w_H(\sigma)$. Algorithm 4 assumes that L is a given list of all models of KB in this order, and that the generator created in line 1 follows this order. Note that L can be constructed in time and space $O^*(c^n)$ by generating all models with the given enumeration algorithm in time $O^*(c^n)$, storing them, and then sorting them in time $O^*(c^n \cdot log(c^n)) = O^*(c^n)$.

The effect of this order is that the algorithm will encounter *subset-maximal candidates* first. A positive explanation candidate $E \subseteq H$ is called *subset-maximal candidate* if KB $\wedge$ E is satisfiable and E is subset-maximal with this property. It is readily observed that to decide whether there are any positive explanations, it is sufficient to check all subset-maximal candidates. As we explain in the following, this is exactly what the algorithm does.

After line 1 the algorithm proceeds exactly as Algorithm 2, except that in line 6 a *positive* explanation is extracted (instead of a *full* explanation) and that in line 13-15 it is made sure that if a candidate E is discarded, also all (immediate) subsets are discarded. Note that if a subset-maximal candidate E fails to explain M (detected in line 8), then also any strict subset of E will fail. It is thus *legitimate* to discard all subsets of a discarded E. But it is even *necessary* for the algorithm's correctness: it assures that the algorithm continues to only consider subset-maximal candidates. *If* the algorithm *did* hit a non-subset-maximal candidate E (that is not listed in the discarded set), it could miss a witness of E failing to explain M. That is, it could miss a model σ of KB $\wedge$ E such that $\sigma \not\models M$, since this witness might already have been 'consumed' with a superset of E.

□

Algorithm 4 Enumeration based algorithm for P-ABD(Γ).

Require: L, H, M
1: generator = Generator(L)
2: discarded = $\emptyset$
3: potentialExp = $\emptyset$
4: **while** generator.notEmpty() **do**
5: σ = generator.next()
6: $E = \{x \in H \mid \sigma(x) = 1\}$
7: **if** $E \notin$ discarded **then**
8: **if** $\sigma \not\models M$ **then**
9: discarded = discarded $\cup \{E\}$
10: potentialExp = potentialExp $- \{E\}$
11: **else**
12: potentialExp = potentialExp $\cup \{E\}$
13: **if** $E \in$ discarded **then**
14: *# discard all immediate subsets*
15: discarded = discarded $\cup \{E - \{x\} \mid x \in E\}$
16: *# now potentialExp contains all subset-maximal positive explanations*
17: **if** potentialExp $\neq \emptyset$ **then**
18: return $\top$
19: **else**
20: return $\bot$

2.6 Proof of Theorem 15

Proof. Let r_0 be the integer such that for every $r \geq r_0$ it holds that $|R| \leq c^r$ for every r-ary $R \in \Gamma$ (for a fixed $c < 2$). Such r_0 exists since Γ is asymptotically sparse. We first explain how to exhaustively branch

until we reach sufficiently low arity $< r_0$.

Now let (V, C) be an instance of $\mathrm{SAT}(\Gamma)$ with maximum arity $r \geq r_0$. For a partial truth assignment $f\colon X \to \{0,1\}$, $X \subseteq V$, we write $C_{|f} = \{c_{|f} \mid c \in C\}$ where $c = R(\mathbf{x})$ and $c_{|f} = R_{|f}(\mathbf{x}')$ with the scope $\mathbf{x}'$ obtained by removing every variable in X from $\mathbf{x}$. Each such $R_{|f} \in \Gamma$ since we assume that Γ is closed under branching.

Let A be the following branching algorithm, which, given a set of constraints C, for every constraint $R(x_1, \ldots, x_k) \in C$, $\mathrm{ar}(R) = k$, $r_0 \leq k \leq r$, branches as follows:

- $A((C - \{R(x_1, \ldots, x_k)\})_{|f_1})$,

$\vdots$

- $A((C - \{R(x_1, \ldots, x_k)\})_{|f_{|R|}})$,

where f_i is the partial truth assignment corresponding to $t_i \in R$. Since Γ is closed under minors we may assume that the variables in the constraint $R(x_1, \ldots, x_k)$ are all distinct. Hence, in every branch we eliminate k variables and in total there are $|R| \leq c^k$ branches. In the worst case this gives a bound of $|R|^{\frac{n}{k}} \leq c^{k\frac{n}{k}} \leq c^n$.

Note that if we apply branching again to a constraint $R'_{|f_i}(y_1, \ldots, y_{k'})$ for some $1 \leq i \leq |R|$, then we may again assume that all variables $y_1, \ldots, y_{k'}$ are distinct (since Γ is closed under minors), and, importantly, that $|R| \leq c^{k'}$, and we may simply apply the branching algorithm A again to get the required bound.

We keep this up until we in a branch reach an instance C_{r_0} where every constraint has arity $< r_0$. Pick a constraint $R''(z_1, \ldots, z_{k''})$ where $k'' < r_0$. We observe that $|R''| < 2^{k''} < 2^{r_0}$ where the first equation holds via the assumption that R'' is non-trivial, and the worst possible bound is then smaller than $(2^{r_0} - 1)^{\frac{n}{r_0}} = (2^{r_0} - 1)^{\frac{1}{r_0}^n} = d^n < 2^n$ since $r_0 \geq 1$ is constant. Put together, the total number of models is dominated by either c^n or d^n, both better than 2^n. □

2.7 Proof of Lemma 17

Proof. First, let R_S be a k-ary relation induced by $p, q \leq k + 1$, i.e., the sum of all arguments is congruent to q modulo p. Clearly, inserting 0 or 1 in the equation simply mean that we have to adjust p, q, and k, as necessary. Similarly, $R_S \subset \{0,1\}^k$ since $S \subset [k] \cup \{0\}$ (each equation has at least one weight which is not allowed).

Second, we only have to establish that all relations in $\mathrm{Min}(\mathcal{E}^{\leq k})$ are non-trivial and closed under substitutions. We provide a short proof sketch since it follows from basic properties of equations. Thus, assume that we are given a minor of an equation in $\mathcal{E}^{\leq k}$, and then assign one or more variables constant values. Then the resulting equation can be obtained from the original equation by first assigning the corresponding variables constant values, and then taking a minor of the resulting relation. To additionally see that every relation is non-trivial, we simply observe that an equation of the form $x_1 + \ldots + x_\ell \equiv q \pmod p$, $\ell \leq k$, cannot become trivially satisfied by identifying variables. □

2.8 Proof of Theorem 18

Proof. We begin by proving that XSAT satisfies the necessary assumptions. First, consider a relation $R \in \mathrm{XSAT}$ of arity k. Then either $R = R_{1/k}$ in which case $|R_{1/k}| = k$, or $R = \bot^k$ in which case $|\bot^k| = 1$, so the language is clearly asymptotically sparse. Second, for an assignment $f\colon X \to \{0,1\}$, $X \subseteq [k]$, for which

$R_{|f} \neq \emptyset$, note that if $f(i) = 0$ for every $i \in X$, then $R_{|f} = R_{1/k'}$ for $k' = k - |X|$. Similarly, if $f(i) = 1$ for some $i \in X$, then $R_{|f} = \bot^{k'}$ for some $k' = k - |X|$. Relations of the form $\bot^k_{|f}$ are also simple to handle. Third, let $g\colon [k] \to [k']$, and consider the minor defined by $R_{1/k}(x_{g(1)}, \ldots, x_{g(m)})$. A simple case analysis shows that if $g(i) = g(j)$ for $i \neq j$ To then obtain the concrete bound $O^*(\sqrt{2}^n)$ we simply observe that when branching on a k-ary constraint we obtain a branching vector of

$$(\overbrace{k, k, \ldots, k}^{k \text{ branches}})$$

and by solving the resulting recurrence equation it is easy to see that the worst possible case is when $k = 2$, giving the bound $O^*(\sqrt{2}^n)$.

Second, for $\mathrm{AFF}^{\leq k}$, we observe that it is asymptotically sparse since it is finite, every relation is non-trivial since $\mathrm{AFF}^{\leq k} \subseteq$ EQUATIONS, and that it is closed under branching follows from basic properties of affine equations. However, in contrast to EQUATIONS, it is closed under minors: an equation is true if and only if the total parity is even or odd, and identifying two or more variables does not affect this relationship. □

2.9 Proof of Theorem 21

Proof. Let (KB, H, M) be an instance of $\mathrm{ABD}(\text{k-CNF}^+)$. Note that there exists an explanation if and only if there exists a purely negative explanation. Denote by $\mathcal{C}$ the set of clauses in KB, that is, $\mathrm{KB} = \bigwedge_{C \in \mathcal{C}} C$, where each clause C is positive and of size k. One further verifies that to determine whether there are any explanations, one only needs to consider two types of clauses: (1) clauses that contain only variables from H (seen as a "constraint" on how to select variables from H as negative literals), and (2) clauses that contain exactly one variable $m \in M$ and variables from H (that is, $\neg h_1, \ldots, \neg h_{k-1}$ is a candidate to explain m, via the clause $(h_1 \vee \cdots \vee h_{k-1} \vee m)$). Therefore, we define the two sets of clauses

$$F = \{(h_1 \vee \cdots \vee h_{k-1} \vee m) \in \mathcal{C} \mid h_1, \ldots, h_{k-1} \in H, m \in M\},$$
$$G = \{(h_1 \vee \cdots \vee h_k) \in \mathcal{C} \mid h_1, \ldots, h_k \in H\},$$

and disregard henceforth all other clauses. For each $m \in M$ define the formula

$$D_m = \bigvee_{(h_1 \vee \cdots \vee h_{k-1} \vee m) \in F} (\neg h_1 \wedge \cdots \wedge \neg h_{k-1})$$

Note that D_m is a negative DNF formula, that is, a disjunction of negative terms. We map (KB, H, M) to the $\text{SIMPLE}\mathrm{SAT}^k$ instance

$$\varphi = \bigwedge_{(h_1 \vee \cdots \vee h_k) \in G} (h_1 \vee \cdots \vee h_k) \wedge \bigwedge_{m \in M} D_m$$

Note that φ is of required form and that $\mathrm{Var}(\varphi) \subseteq H$. It is easy to verify that there is a one-to-one correspondence between negative explanations and models of φ. □

2.10 Proof of Lemma 22

Proof. Consider the instance (KB, H, M) where KB contains clauses of type $(\neg x_1 \vee \cdots \vee \neg x_k)$ and $(x \to y)$. Recall that by Lemma 6 we may W.L.O.G assume that $M, H \subseteq \text{Var}(\text{KB})$. Note further that we also may assume that $H \cap M = \emptyset$: define $X = H \cap M$, remove X from both H and M, introduce fresh variables h, m, add h to H, add m to M, and add to KB the clauses $(h \to m)$ and $\bigwedge_{x \in X}(h \to x)$. This constitues a CV-reduction of the problem to itself.

For a variable $h \in H$ denote by $\text{cons}(h)$ the consequences of h in KB, that is, all single literals that can be derived by applying exhaustively resolution on $\text{KB} \wedge h$. We map (KB, H, M) to (KB', H, M), where $\text{KB}' = \varphi_1 \wedge \varphi_2$, with

$$\varphi_1 = \bigwedge_{\substack{m \in M, h \in H \\ \text{s.t. } m \in \text{cons}(h)}} (h \vee m)$$

$$\varphi_2 = \bigwedge_{\substack{h_1, \ldots, h_k \in H \text{ s.t.} \\ \text{KB} \wedge M \wedge h_1 \wedge \cdots \wedge h_k \models \emptyset}} (h_1 \vee \cdots \vee h_k)$$

It is easy to verify that there is a one-to-one correspondence between explanations, obtained by flipping the literals of an explanation. □

2.11 Proof of Theorem 26

Proof. We consider the problem of deciding the truth value of a quantified Boolean formula $\forall X \exists Y . \Phi(X, Y)$ where Φ is in 3-CNF, i.e., whether there for all possible assignments to the set of variables X exists an assignment to the set of variables Y such that $\Phi(X, Y)$ under those assignments is true. Then, from [3] we know that this problem can not be solved in c^n time for any $c < 2$ under SETH.

the complement of this problem is $\exists X \forall Y \Phi'(X, Y)$ where $\Phi' = \neg\Phi$ is in 3-DNF can also not be done in time less than 2^n under SETH.

It suffices to have a CV reduction from $\forall X \exists Y \Phi(X, Y)$ where Φ is in 3-DNF to $\text{ABD}(4\text{-CNF})$, inspired by the reduction from [5].

Let $\exists x_1, x_2 \ldots \forall y_1, y_2 \ldots \Phi$ be in 3-DNF form. Define the following $\text{ABD}(4\text{-CNF})$ instance: (KB, H, M) where $H = \{x_i\}$, $M = \{y_i\} \cup \{s\}$, and $\text{KB} = \{\Phi \to s \wedge y_1 \wedge y_2 \ldots\} \wedge \{s \to y_1 \wedge y_2 \ldots\}$. First, we show that KB can be expressed in 4-CNF. Her, Φ is in 3-DNF, $\{\Phi \to s \wedge y_1 \ldots\}$, and can be written as $\{\neg\Phi \vee \{s \wedge y_1 \ldots\}\}$, $\{\neg\Phi\}$, which is in 3-CNF, and distributing the $\vee$ makes 4-CNF clauses. The second part $\{s \to y_1 \wedge y_2 \ldots\}$ follows the same reasoning to make it into 2-CNF.

We prove the correctness of our reduction. From left to right: assume that the QBF formula is true. Then there is $X \subseteq H$ such that for all Y, Φ is true. The theory KB is satisfiable for X by putting all y_i and s to true. The entailment of Y is easy to see, as putting any y_i or s to $\bot$ makes the theory evaluate to false if ϕ is true.

Now, from right to left: we assume the abduction problem does have a solution, and show that the QBF formula is true. Let $X \subseteq H$ be a solution to the abduction problem. X satisfies ϕ when all y_i and s are

true, since KB must be evaluated to true when the both the manifestations and solutions of the abduction problem are positive. Now let us assume s is false. By entailment KB must be unsatisfiable, meaning that $\{\Phi \rightarrow s \vee y_i \ldots\}$ must be false. This can only be the case if Φ is $\top$, and this applies for all Y, thus completing the proof. □

2.12 Proof of Lemma 27

Proof. We reduce a regular ABD(4-CNF) instance into a P-ABD(4-CNF) as follows: $(\text{KB}, H, M) \rightarrow (\text{KB}', H', M)$ where $\text{KB}' = \text{KB} \cup \{x \leftrightarrow \neg x' \mid x \in H\}$ and $H' = H \cup \{x'\}$.

It is easy to see that the second problem has a positive solution if and only if the first one has any solution. The inequalities between the x and x' variables can be written in 2-CNF, thus KB$'$ is in 4-CNF. We notice that the number of variables has increased linearly, but remains less than twice the number of variables of the original problem, thus, P-ABD(4-CNF) cannot be solved faster than $2^{n/2} = 1.4142^n$ □

2.13 Proof of Theorem 28

Proof. We present the proof for P-ABD(Γ) but the construction is exactly the same for ABD(Γ). Note that P-ABD(Γ) $\leq^{\text{CV}}$ P-ABD($\Gamma \cup \{\bot, \top\}$) trivially holds, so we only need to prove the other direction. For this, we first claim that $R_{\neq} = \{(0,1),(1,0)\}$ is qfpp-definable by Γ. Choose a relation $R \in \Gamma$ of arity k such that $\emptyset \subset R \subset \{0,1\}^k$ and such that R does not contain either of the two constant tuples $(0,\ldots,0)$ and $(1,\ldots,1)$. Such a relation must exist since otherwise Pol(Γ) $\supset [\bar{x}]$ (note that $(0,\ldots,0) \in R$ but $(1,\ldots,1) \notin R$ is not possible since $\overline{(0,\ldots,0)} = (1,\ldots,1)$). Second, choose a non-constant tuple $t = (a_1,\ldots,a_k) \in R$, and construct two sets of indices I_0 and I_1 such that $i \in I_b$ if and only if $a_i = b$ for $b \in \{0,1\}$. If we then define the relation $S(x,y) \equiv R(x_1,\ldots,x_k)$ where $x_i = x$ if $i \in I_0$, and $x_i = y$ if $i \in I_1$ it follows (1) that $(0,1) \in S$ due to the assumptions on the tuple t, and (2) that $\bar{(t)} = (1,0) \in S$ since Pol(Γ) $= [\bar{x}]$. Finally, if $(0,0) \in S$ or $(1,1) \in S$ then $(0,\ldots,0) \in R$ or $(1,\ldots,1) \in R$, contradicting our original assumption. We conclude that $S = R_{\neq}$ and for simplicity assume that $R_{\neq} \in \Gamma$ to simplify the notation in the remaining proof.

Second, let (ϕ, H, M) be an instance of P-ABD($\Gamma \cup \{\bot, \top\}$). Partition ϕ into $\phi_1 \cup \phi_2$ where ϕ_1 contains only constraints from Γ, and ϕ_2 only constraints from $\{\bot, \top\}$. We introduce two fresh variables V_0 and V_1 and create the instance (ϕ', H', M') of P-ABD(Γ) where:

1. $\phi' = \phi_1 \wedge \bigwedge_{\bot(x) \in \phi_2} R_{\neq}(x, V_1) \wedge \bigwedge_{\top(y) \in \phi_2} R_{\neq}(y, V_0) \wedge R_{\neq}(V_0, V_1)$,
2. $H' = H \cup \{V_1\}$, and
3. $M' = M \cup \{V_1\}$.

Assume that $E \subseteq H$ is an explanation for (ϕ, H, M). Then $\phi' \wedge (E \cup \{V_1\})$ is satisfiable and explains $M \cup \{V_1\}$. For the other direction, assume that $E' \subseteq H'$ is an explanation for (ϕ', H', M'). Since $V_1 \in M'$ we either have $V_1 \in E$, or $y \in E$, for some $\top(y) \in \phi_2$, which via the constraints $\bigwedge_{\bot(x) \in \phi_2} R_{\neq}(x, V_1)$ and $R_{\neq}(V_0, V_1)$ ensures that $E' \cap H$ is an explanation for (ϕ, H, M). □

2.14 Proof of Theorem 29

We need a little bit of additional notation for the next theorem. Let $s = (s_1,\ldots,s_k) \in [0,1]^k$ be a k-ary *sign-pattern*, let $0_s = (a_1,\ldots,a_k) \in \{0,1\}^k$ be the unique tuple such that the sum $(s_1 \oplus a_1) + \ldots (s_k \oplus a_k) = 0$,

and 1_s the corresponding tuple whose sum is k. Finally, let $R^s_{\text{NAE}} = \{0,1\}^k - \{0_s, 1_s\}$. It is easy to see that one for every k-clause $(\ell_1 \vee \ldots \vee \ell_k)$ can construct a sign-pattern and thus a corresponding R^s_{NAE} relation. We let k-NAE be the set of all such relations of arity k.

Proof. Again, we present the proof for P-ABD(Γ) but the construction is exactly the same for ABD(Γ). Let (ϕ, H, M) be an instance of P-ABD($k-CNF$). We introduce two fresh variables V_0, V_1, the constraint $R_{\neq}(V_0, V_1)$ (recall the proof of Theorem 28, and construct $\phi' = \{R^s_{\text{NAE}}(x_1, \ldots, x_k, V_0) \mid (\ell_1 \vee \ldots \ell_k) \in \phi\}$ where the ith component of the sign-pattern s is 0 if ℓ_i is positive, and equal to 1 otherwise. Let $E' = E \cup \{V_1\}$ and $M' = M \cup \{V_1\}$. The correctness then follows via similar arguments as for satisfiability (see, e.g., Lemma 47 in [11]). □

2.15 Proof of Lemma 30

Proof. We give a reduction from CNF-SAT which at most triplicates the number of variables ($2^{\frac{1}{3}} > 1.2599$). The second bound follows since the reduction satisfies $|H| = 2 \cdot |\operatorname{Var}(\varphi)|$ (and $2^{\frac{1}{2}} > 1.4142$). The last two bounds follow since in the reduction $|M| = |\operatorname{Var}(\varphi)|$ and $\frac{|H|}{|M|} = 2$.

Be $\varphi = \bigwedge_{j \in J} C_j$ an instance of CNF-SAT. For each variable $x \in \operatorname{Var}(\varphi)$ introduce two fresh variables, denoted x' and m_x. Denote by C'_j the clause obtained from C_j where each positive literal x is replaced by $\neg x'$. That is, C'_j contains negative literals only. Define the abduction instance as (KB, H, M), where

$$\begin{aligned}
\text{KB} =& \bigwedge_{j \in J} C'_j \\
& \wedge \bigwedge_{x \in \operatorname{Var}(\varphi)} (\neg x \vee \neg x') \wedge (x \rightarrow m_x) \wedge (x' \rightarrow m_x) \\
H =& \{x, x' \mid x \in \operatorname{Var}(\varphi)\} \\
M =& \{m_x \mid x \in \operatorname{Var}(\varphi)\}
\end{aligned}$$

□

2.16 Proof of Lemma 32

Proof. From [6], we know that CNF-SAT does not admit a kernel of size $f(n)$ where f is a polynomial, i.e., an equisatisfiable instance with e.g. a quadratic number of clauses. To prove the claim it suffices to have a reduction from ABD(2-CNF) to CNF-SAT that only has a polynomial number of clauses in terms of n. Assume an arbitrary ABD(2-CNF) instance. For every $m_i \in M$, for all clauses of the type: $(\ell_j \vee m_i), \ldots, (\ell_k \vee m_i)$ where $\ell_i \ldots \ell_k$ are a literals, we delete these clauses and add the following CNF clause $(\ell_i \vee \ldots \vee \ell_k)$, and otherwise keep everything unchanged. This completes the reduction. The correctness is straightforward: for every manifestation, we have to choose at least one of the literals that appear with it in a clause, in order to entail it. The ABD(2-CNF) instance can only have up to n^2 clauses. The resulting CNF-SAT instance from this reduction thus only has a number of clauses equal to the number of $m_i \in M$, plus the remaining 2-CNF clauses, thus at most n^2 clauses. If an LV-reduction of the form CNF-SAT $\leq^{\text{LV}}$ ABD(2-CNF) exists, together with the reduction from this proof, we would have a reduction from an arbitrary CNF-SAT instance into a CNF-SAT with only n^2 clauses, which is not possible unless NP $\subseteq$ $P/Poly$. □

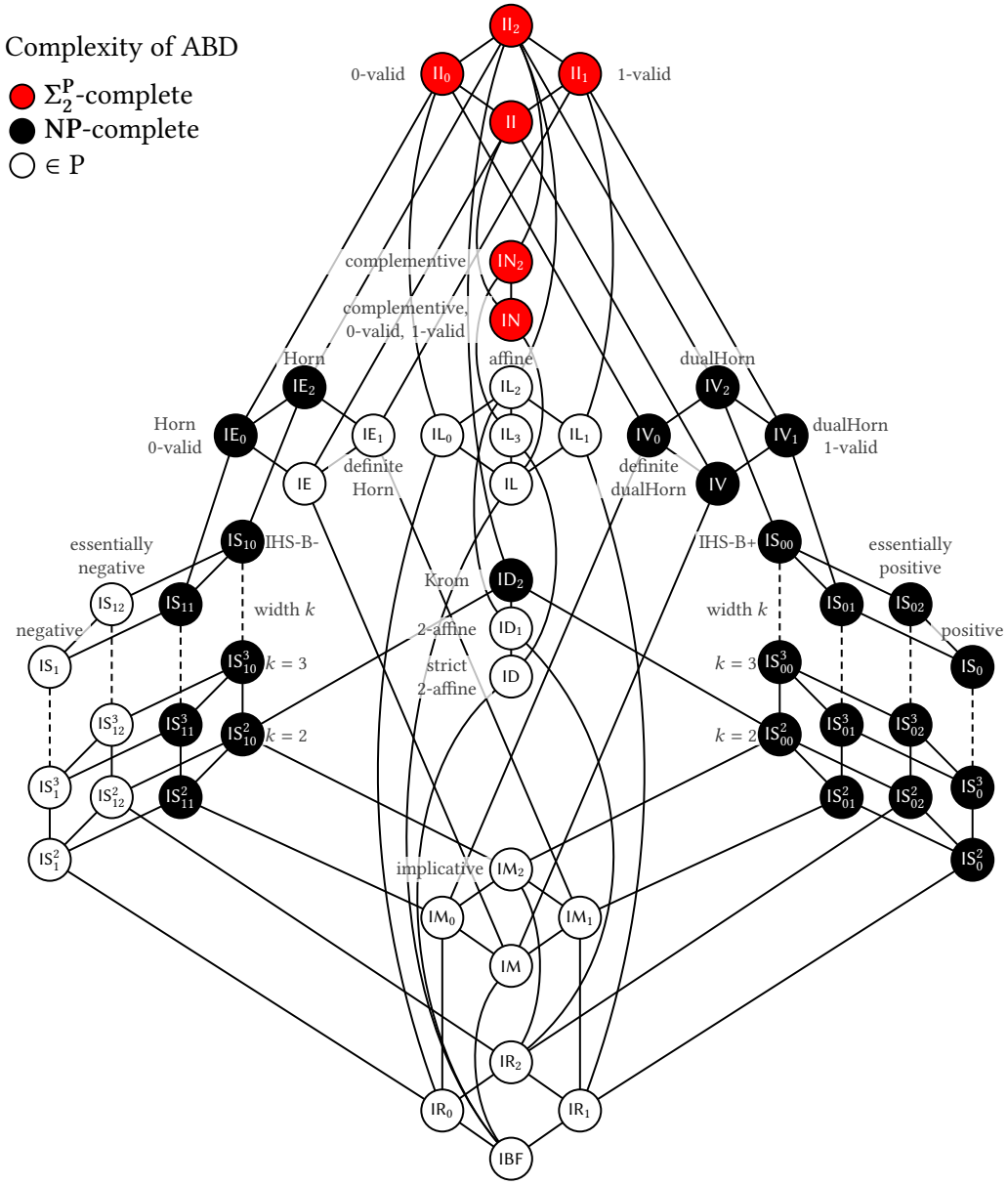

Figure 1: Classical complexity of ABD according to Nordh and Zanuttini (2008), illustrated on Post's lattice.

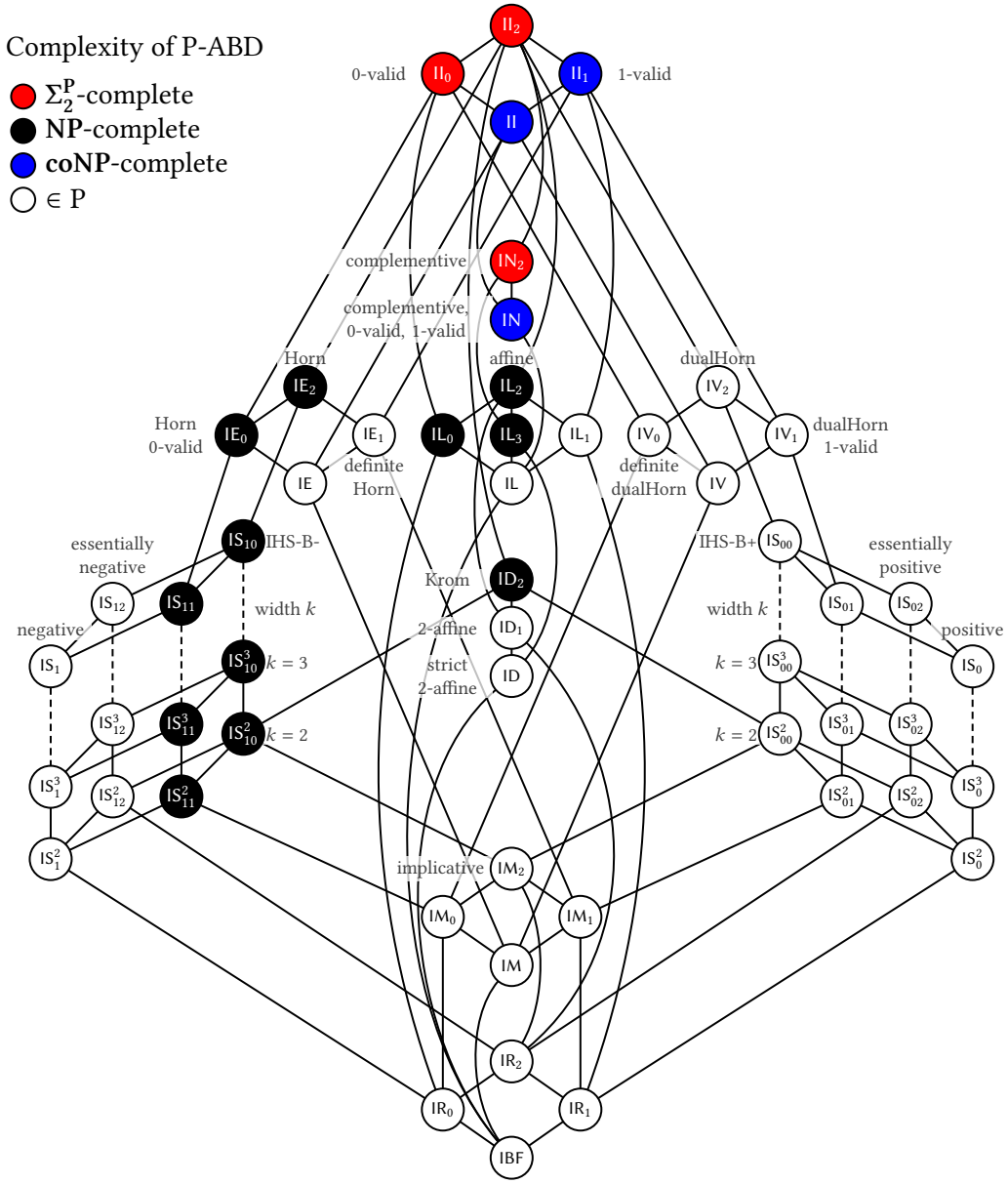

Figure 2: Classical complexity of P-ABD according to Nordh and Zanuttini (2008), illustrated on Post's lattice.

References

[1] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. Security protocols verification in abductive logic programming: A case study. In *Engineering Societies in the Agents World VI, 6th Internat. Workshop, ESAW'05*, pages 106–124, 2005.

[2] Martin E. Buron Brarda, Luciano H. Tamargo, and Alejandro Javier García. Using argumentation to obtain and explain results in a decision support system. *IEEE Intell. Syst.*, 36(2):36–42, 2021.

[3] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. On the exact complexity of evaluating quantified k-cnf. *Algorithmica*, 65(4):817–827, 2013.

[4] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, and J. Nederlof et al. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016.

[5] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *Journal of the ACM (JACM)*, 42(1):3–42, 1995.

[6] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct pcps for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.

[7] A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *Proc. 33rd AAAI Conf. on Artificial Intelligence (AAAI'19)*, pages 1511–1519, 2019.

[8] Alexey Ignatiev. Towards trustable explainable AI. In Christian Bessiere, editor, *Proc. 29th Internat. Joint Conf. on Artificial Intelligence (IJCAI'20)*, pages 5154–5158. ijcai.org, 2020.

[9] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci*, 62(2):367 – 375, 2001.

[10] Katsumi Inoue, Taisuke Sato, Masakazu Ishihata, Yoshitaka Kameya, and Hidetomo Nabeshima. Evaluating abductive hypotheses using an EM algorithm on bdds. In *Proc. 21st Internat. Joint Conf. on Artificial Intelligence (IJCAI'09)*, pages 810–815, 2009.

[11] P. Jonsson, V. Lagerkvist, G. Nordh, and B. Zanuttini. Strong partial clones and the time complexity of SAT problems. *J. Comput. Syst. Sci.*, 84:52 – 78, 2017.

[12] P. Jonsson, V. Lagerkvist, and B. Roy. Fine-grained time complexity of constraint satisfaction problems. *ACM Trans. Comput. Theory.*, 13(1), 2021.

[13] Peter Jonsson, Victor Lagerkvist, Johannes Schmidt, and Hannes Uppman. The exponential-time hypothesis and the relative complexity of optimization and logical reasoning problems. *Theor. Comput. Sci.*, 892:1–24, 2021.

[14] V. Lagerkvist and M. Wahlström. The (coarse) fine-grained structure of NP-hard SAT and CSP problems. *ACM Trans. Comput. Theory.*, 14(1), 2022.

[15] D. Lau. *Function Algebras on Finite Sets: Basic Course on Many-Valued Logic and Clone Theory (Springer Monographs in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[16] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

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

[18] Mariam Obeid, Zeinab Obeid, Asma Moubaiddin, and Nadim Obeid. Using description logic and abox abduction to capture medical diagnosis. In *Proc. 32nd Internat. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2019*, pages 376–388, 2019.

[19] R. Paturi, P. Pudlák, M. Saks, and F. Zane. An improved exponential-time algorithm for k-sat. *J. ACM*, 52(3):337–364, 2005.

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

[21] Teeradaj Racharak and Satoshi Tojo. On explanation of propositional logic-based argumentation system. In *Proc. 13th Internat. Conf. on Agents and Artificial Intelligence (ICAART'21)*, pages 323–332, 2021.

[22] Oliver Ray, Athos Antoniades, Antonis C. Kakas, and Ioannis Demetriades. Abductive logic programming in the clinical management of HIV/AIDS. In *Proc. 17th European Conf. on Artificial Intelligence*, pages 437–441, 2006.

[23] Chiaki Sakama and Katsumi Inoue. An abductive framework for computing knowledge base updates. *Theory Pract. Log. Program.*, 3(6):671–713, 2003.

[24] Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978.

[25] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[26] Francesco Sovrano, Fabio Vitali, and Monica Palmirani. Modelling GDPR-compliant explanations for trustworthy AI. In *Proc. 9th Internat. Conf. on Electronic Government and the Information Systems Perspective (EGOVIS'20)*, pages 219–233, 2020.

[27] Michael Thomas and Heribert Vollmer. Complexity of non-monotonic logics. *Bull. EATCS*, 102:53–82, 2010.

[28] R. Williams. Algorithms for quantified boolean formulas. In *Proc. 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'02)*, page 299–307, 2002.