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

Victor Lagerkvist[2], **Mohamed Maizia**[1,2] and **Johannes Schmidt**[1]
[1] Department of Computer Science and Informatics, Jönköping University, Jönköping, Sweden
[2] Department of Computer and Information Science, Linköping University, Linköping, Sweden
victor.lagerkvist@liu.se, mohamed.maizia@ju.se, mohamed.maizia@liu.se, johannes.schmidt@ju.se

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 [Inoue *et al.*, 2009], network security [Alberti *et al.*, 2005], computational biology [Ray *et al.*, 2006], medical diagnosis [Obeid *et al.*, 2019], knowledge base updates [Sakama and Inoue, 2003], and explainability issues in machine learning and decision support systems [Ignatiev *et al.*, 2019; Ignatiev, 2020; Brarda *et al.*, 2021; Racharak and Tojo, 2021]. 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 [Sovrano *et al.*, 2020] 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 [Eiter and Gottlob, 1995; Thomas and Vollmer, 2010] 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 [Paturi *et al.*, 2005]. 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) [Impagliazzo and Paturi, 2001]) 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 [Nordh and Zanuttini, 2008], 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 [2016]. 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 $\text{P-ABD}(\Gamma)$ (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 $\text{P-ABD}(\Gamma)$ 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 $x_1 + \ldots + x_k = q \pmod{p}$ ($\textsc{Equations}^k$). The problem(s) $\text{(P-)ABD}(\textsc{Equations})$ 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 $\text{P-ABD}(\mathrm{XSAT})$ (*exact satisfiability*) and NP-complete $\text{P-ABD}(\mathrm{AFF}^{(\leq k)})$ (arity bounded equations over GF(2)).

(Type) Class	Classical complexity	Improved
$\textsc{Equations}^k$ $(k \geq 1)$	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.

In Section 3.4 we consider more restricted types of abduction problems with a particular focus on $\text{(P-)ABD}(k\text{-CNF}^+)$ 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 $\textsc{SimpleSAT}^p$ which can be solved by branching. For coNP, only $\text{P-ABD}(\Gamma)$ 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.

(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.

In Section 4, we prove lower bounds under (S)ETH for missing intractable cases. Let $\mathrm{IMP} = \{\{(0,0),(0,1),(1,1)\}\}$. Under the ETH, we first prove that $\mathrm{ABD}(2\text{-CNF}^+)$ and $\mathrm{ABD}(2\text{-CNF}^- \cup \mathrm{IMP})$ 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 $\text{(P-)ABD}(\text{CNF}^- \cup \text{IMP})$, 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 $\mathrm{ABD}(2\text{-CNF})$, we prove that sharp lower bounds under the SETH are unlikely unless $\mathrm{NP} \subseteq \mathrm{P/Poly}$, 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 $\mathrm{ABD}(\Gamma)$ and $\text{P-ABD}(\Gamma)$ 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 technical report [Lagerkvist et al.*, 2025].*

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 $\perp = \{(0)\}$ and $\top = \{(1)\}$, and the two constant 0-ary relations as $\mathtt{f} = \emptyset$ and $\mathtt{t} = \{\emptyset\}$. 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.*

$$KB = \{A \wedge B \to C, D \to B, \neg E \to C \wedge \neg D\},$$
$$H = \{A, D, E\}, \qquad M = \{C\}$$

$E_1 = \{A, \neg D, \neg E\}$ *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 P-$\mathrm{ABD}(\Gamma)$ is fully determined for all Γ, due to Nordh and Zanuttini [2008]. An illustration of these classifications is found in the technical report [Lagerkvist *et al.*, 2025].

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 technical report.

Complexity theory. We assume familiarity with basic notions in classical complexity theory (cf. [Sipser, 1997]) 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 [Jonsson *et al.*, 2017] (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 [Jonsson *et al.*, 2021b].

3 Upper Bounds via SAT Based Approaches

We begin by investigating the possibility of solving $\mathrm{ABD}(\Gamma)$ and P-$\mathrm{ABD}(\Gamma)$ 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 $\mathrm{SAT}(\Gamma^+) \leq^{\mathrm{CV}} \mathrm{SAT}(\Gamma)$ [Jonsson *et al.*, 2021a]. We obtain the following baseline bound to beat.

Theorem 3. ($\star$) *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. ($\star$) *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. ($\star$) *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 $\text{P-ABD}(\Gamma)$.

```
Require: KB, $H, M, D, \delta$
 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, $\text{P-ABD}(\Gamma)$ can be solved in $O^*(2^n)$ time and polynomial space.*

Proof. Consider Algorithm 1. 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 technical report. ☐

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* $\mathrm{SAT}(\Gamma)$ *is sparsely enumerable in* $O^*(c^n)$ *time. Then* $\mathrm{P\text{-}ABD}(\Gamma)$ *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* $\mathrm{Min}(\Gamma) = \{M \mid M$ *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* $\{\{0,1\}^2 - \{(0,0)\}, \top, \mathsf{f}\}$ *by identifying one or more variable to* 0 *and* $\{\{0,1\}^2, \{0,1\}, \mathsf{t}\}$ *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* $\textsc{Equations} = \{R_S \mid k \geq 1, p, q \leq k+1, S = \{i \in [k] \cup \{0\} \mid i \equiv q \pmod{p}\}\}$.

Thus, each relation in $\textsc{Equations}$ can be defined by an equation of the form $x_1 + \ldots + x_k = q \pmod{p}$ for fixed p, q, k. In particular $\mathrm{AFF} \subseteq \textsc{Equations}$ but it also contains relations inducing Σ_2^P-complete (P-)ABD problems. Perhaps contrary to intuition, $\textsc{Equations}$ 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 $\textsc{Equations}$ and then close it under minors. We obtain the following.

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

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

Finally, let $\mathrm{XSAT} \subseteq \textsc{Equations}$ 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 $\perp^k = \{(0, \ldots, 0)\}$ 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 $\mathrm{AFF} \subseteq \textsc{Equations}$ 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, $k \geq 2$ [Nordh and Zanuttini, 2008]. We show in the following that these cases can be solved in improved time.

Definition 19. *Denote by* $\textsc{SimpleSAT}^p$ *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. $(\star)$ ABD(*k-CNF*$^+$)$\leq^{\text{CV}}$ SIMPLESAT^k. *Moreover, the* SIMPLESAT*-instance contains only variables from* H.

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

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

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

Corollary 23. ABD(*k-CNF*$^+$) *and* (P-)ABD(*k-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 [Nordh and Zanuttini, 2008] when Γ is 1-valid. Under certain additional assumptions, P-ABD(Γ) can then be solved in improved time.

Theorem 24. *Let* Γ *be a 1-valid constraint language. If* SAT(Γ^+) *can be decided in* $O^*(c^n)$ *time for* $c \leq 2$ *then* P-ABD(Γ) *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 SAT(Γ^+) 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 $\geq$ 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-ABD(Γ) coNP-hard [Nordh and Zanuttini, 2008]. Then, any Γ^+-formula is expressible as k-CNF formula (without additional variables), so SAT(Γ^+) can be solved in improved time via the standard $(1, \ldots, k)$-branching. With Theorem 24 we conclude that P-ABD(Γ) 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. ABD(*2-CNF*$^+$) *and* ABD(*2-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 ABD(2-CNF$^-$ $\cup$ 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 [Lokshtanov *et al.*, 2011].

For the reduction, assume an arbitrary instance of a k-colored clique problem over a graph $G = (V, E)$, and where $C = c_i \mid i \in [1 \ldots k]$ 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 ABD(2-CNF$^-$ $\cup$ IMP) $\leq^{\text{CV}}$ ABD(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* ABD(*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* P-ABD(*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 $\text{Pol}(\Gamma) = [\bar{x}]$. 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(Γ) $= [\bar{x}]$. Then* $(\text{P-})\text{ABD}(\Gamma \cup \{\bot, \top\}) =^{\text{CV}} (\text{P-})\text{ABD}(\Gamma)$.

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 technical report.

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

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. *($\star$) Under SETH, there is no $c < 1$ such that (P-)*ABD*(CNF$^-$ $\cup$ 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$^-$ $\cup$ 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$^-$ $\cup$ 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. ($\star$) *If CNF-SAT $\leq^{\text{LV}}$* ABD*(2-CNF) then NP $\subseteq$ P/Poly.*

Proof. We provide a short proof sketch. From [Fortnow and Santhanam, 2011], 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 $\subseteq$ 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: $(\ell_j \lor m_i), \ldots, (\ell_k \lor m_i)$ where $\ell_i \ldots \ell_k$ are a literals, we delete these clauses and add the following CNF clause $(\ell_i \lor \ldots \lor \ell_k)$, and otherwise keep everything unchanged. $\square$

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 Pol(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 [Lagerkvist and Wahlström, 2022]. 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-)ABD(4-CNF) is unlikely to admit improved algorithms, (P-)ABD(k-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-)ABD(2-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-)ABD(Γ) 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 [2002] constructed an $O(1.709^m)$ time algorithm, so one could be cautiously optimistic about analyzing abduction with m.

Acknowledgements

The authors are supported by the Swedish research council under grant VR-2022-03214.

References

[Alberti *et al.*, 2005] 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.

[Brarda *et al.*, 2021] 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.

[Cygan *et al.*, 2016] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, and Jesper Nederlof et al. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016.

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

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

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

[Ignatiev, 2020] 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.

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

[Inoue *et al.*, 2009] 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.

[Jonsson *et al.*, 2017] Peter Jonsson, Victor Lagerkvist, Gustav Nordh, and Bruno Zanuttini. Strong partial clones and the time complexity of SAT problems. *J. Comput. Syst. Sci.*, 84:52 – 78, 2017.

[Jonsson *et al.*, 2021a] Peter Jonsson, Victor Lagerkvist, and Biman Roy. Fine-grained time complexity of constraint satisfaction problems. *ACM Trans. Comput. Theory.*, 13(1), 2021.

[Jonsson *et al.*, 2021b] 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.

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

[Lagerkvist *et al.*, 2025] Victor Lagerkvist, Mohamed Maizia, and Johannes Schmidt. A fine-grained complexity view on propositional abduction - algorithms and lower bounds. *CoRR*, abs/2505.10201, 2025.

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

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

[Obeid *et al.*, 2019] 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.

[Paturi *et al.*, 2005] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for *k*-sat. *J. ACM*, 52(3):337–364, 2005.

[Racharak and Tojo, 2021] 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.

[Ray *et al.*, 2006] 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.

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

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

[Sovrano *et al.*, 2020] 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.

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

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