What Makes Propositional Abduction Tractable

Gustav Nordh, Bruno Zanuttini

What makes propositional abduction tractable

Gustav Nordh [1]

LIX, École Polytechnique
Route de Saclay
F-91 128 Palaiseau, France

Bruno Zanuttini

GREYC, UMR CNRS 6072
Université de Caen
Bd. du Maréchal Juin
F-14 032 Caen Cedex, France

Abstract

Abduction is a fundamental form of nonmonotonic reasoning that aims at finding explanations for observed manifestations. This process underlies many applications, from car configuration to medical diagnosis. We study here the computational complexity of deciding whether an explanation exists in the case when the application domain is described by a propositional knowledge base. Building on previous results, we classify the complexity for local restrictions on the knowledge base and under various restrictions on hypotheses and manifestations. In comparison to the many previous studies on the complexity of abduction we are able to give a much more detailed picture for the complexity of the basic problem of deciding the existence of an explanation. It turns out that depending on the restrictions, the problem in this framework is always polynomial-time solvable, NP-complete, coNP-complete, or Σ_2^P-complete.

Based on these results, we give an *a posteriori* justification of what makes propositional abduction hard even for some classes of knowledge bases which allow for efficient satisfiability testing and deduction. This justification is very simple and intuitive, but it reveals that no nontrivial class of abduction problems is tractable. Indeed, tractability essentially requires that the language for knowledge bases is unable to express both causal links and conflicts between hypotheses. This generalizes a similar observation by Bylander *et al.* for set-covering abduction.

Key words: Abduction, Propositional logic, Computational complexity

1 Introduction

Abduction is the fundamental reasoning process which consists of explaining observations by plausible causes taken from a given set of hypotheses. For instance, it is an abduction problem to try to derive diseases from observed symptoms, according to known rules relating both. This process was extensively studied by Peirce [6], and its importance to Artificial Intelligence was first emphasized by Morgan [38] and Pople [41].

From the application point of view, abduction has demonstrated its importance. It has been applied in particular to explanation-based diagnosis (e.g., medical diagnosis [7]), to text interpretation [29], and to planning [28]. It is also the fundamental process underlying ATMSs [13].

The formalization and resolution of abduction problems have been studied in numerous formalisms, among which set-covering [7], default logic [22], logic programming [21,36]. We are interested here in its resolution in classical propositional logic.

We adopt a complexity-theoretic point of view. More precisely, we are interested in the complexity of deciding whether an abduction problem has a solution when the underlying knowledge base is propositional. Thus our study follows Selman and Levesque's [48] and Eiter and Gottlob's [20] seminal papers. We also build on two classifications previously obtained in our framework [12,40].

Even in the simple setting of propositional logic, deciding whether an abduction problem has a solution is in general Σ_2^{P}-complete. Consequently, like for most hard computational problems, several approaches have been studied for solving it efficiently: Exhibiting tractable classes obtained by restrictions over the knowledge base [12,16,20,24,51]; heuristic approaches, in particular through computation of prime implicates [14,15,37,49] and through reducing the problem to QBF and using generic QBF solvers [19]; compilation [8,35]; and approximation [31,50].

In this paper we adopt the approach consisting of trying to find tractable restrictions over the knowledge base. Our contribution is twofold.

First, we identify the complexity of abduction, with varying restrictions over the representations of manifestations and hypotheses, for every constraint lan-

Email addresses: `nordh@lix.polytechnique.fr` (Gustav Nordh), `zanutti@info.unicaen.fr` (Bruno Zanuttini).

[1] Partially supported by the *Swedish-French Foundation* and the *National Graduate School in Computer Science* (CUGS), Sweden

guage and every clausal or equational language restricting the (propositional) knowledge base, under reasonable assumptions on the representation of the constraints. Concerning manifestations, we study the restrictions where they are expressed as a positive, negative, or unrestricted literal, clause, term, or CNF. Concerning hypotheses, we study the restrictions where they are expressed by a set of literals which is positive, negative, closed under complement, or unrestricted. To that aim, we use the now well-known Schaefer's framework [46] and Post's lattice [42]. Precisely, we proceed as follows.

- We first prove a relatively small number of tractability and hardness results for particular constraint languages.
- Using Post's classification and these results, we then derive the complexity of abduction for *any* constraint language.
- In a similar manner, and at the same time, we obtain the complexity of abduction for any clausal or equational language.

We exhibit new polynomial and new hard restrictions. We also discover that abduction is always either in P, NP-complete, coNP-complete, or Σ_2^{P}-complete, depending on the restrictions. Such a result could not be taken for granted, due to Ladner's result stating that if P $\neq$ NP, then there exist problems in NP that are neither in P nor NP-complete [33]. Moreover, the fact that some restrictions yield NP-complete, and others coNP-complete problems is surprising at first sight. It reveals in particular that abduction is a very rich problem in terms of completeness results in different complexity classes. Thus our results can be used as starting points for establishing complexity results for other problems, in particular in nonmonotonic reasoning.

From the application point of view, our tables of complexity allow the designers of knowledge-based agents or expert systems to choose the appropriate knowledge representation language, according to the tradeoff between the expressiveness required and the constraints on resolution of abduction problems. Moreover, when a representation language that is hard for abduction must be used, the precise complexity of the corresponding problem allows to choose heuristic approaches for solving it. For instance, with an appropriate reduction an NP-complete problem can be solved by a satisfiability solver, while a Σ_2^P-complete one cannot; a more generic QBF solver (or a specialized $QBF_{\exists,2}$-solver) must be used.

Our second contribution is to identify a simple set of minimal conditions yielding NP-completeness for languages which allow for efficient deduction (and are thus good candidates for knowledge representation). For instance, when terms have to be explained, we discover that abduction is NP-hard exactly when the language for knowledge bases can both express causal links from hypotheses to individual manifestations and forbid some combinations of hypotheses. This generalizes similar observations by Bylander *et al.* [7] about set-covering abduction.

From this condition it follows that tractability can occur only in very restricted cases, i.e., when there can be no causal dependency at all or when causes can all be assumed together. For instance, in medical diagnosis this means that diseases must not rule each other out for the task to be tractable. We also argue that these conditions give intuitions about results beyond Schaefer's framework of constraint languages, and we revisit some previously known results in that manner. In this spirit, our observations allow to adopt a unified point of view on the results exhibited with many different restrictions in the literature.

The use of Post's lattice and Schaefer's framework for studying the complexity of reasoning problems is sometimes considered to be overlimitating. For instance, it does not encompass the class of all Horn formulas; this is because Horn clauses can be arbitrarily long. In this study, we adopt these powerful tools coming from complexity-theory and show how to overcome some of these limitations. Indeed, we use them to show results about infinite constraint languages as well as about finite ones, and we formulate the former in terms of classes of CNF formulas. These extensions are directly motivated by AI-applications where it is common to model the knowledge base using an infinite constraint language represented in terms of classes of CNF formulas.

To put our results in context, we briefly discuss the results from the literature on the complexity of abduction that are most relevant to our present study. Our starting point is Selman and Levesque's [48], and Eiter and Gottlob's [20] classical results. Selman and Levesque [48] proved that deciding whether an abduction problem over a Horn knowledge base has an explanation is NP-complete, even when the hypotheses are given as a set of positive literals and the manifestation is a single positive literal. Similarly, Eiter and Gottlob [20] proved that when the knowledge base is given by a general propositional formula, the problem becomes Σ_2^{P}-complete. Moreover they note that when the knowledge base is given by a definite Horn formula, the hypotheses are positive literals, and the manifestations are given by a positive term, then the problem is in P.

From these results it is clear that the complexity of the abduction problem depends heavily on the representation language of the knowledge base. This have led researchers to investigate the complexity of abduction for various restrictions on the representation language of the knowledge base. This line of research culminated with the two recent complexity classifications given in [12,40]. In these papers the complexity of deciding if there exists an explanation is classified for a very general class of restrictions on the representation language of the knowledge base, that include many restrictions that have been studied before in the literature. Although the restrictions on the representation language of the knowledge base studied in these papers are identical, the papers differ on the representation of hypotheses and manifestations. In [12] the hypotheses are given by a set of literals that are closed under complement and the manifestation is given by a single literal, while in [40] the hypotheses are allowed to be any set of literals and the manifestations are given by a term.

When comparing the results of [12] and [40] it becomes clear that even small variations on the representation of hypotheses and manifestations can have a strong impact on the complexity of the abduction problem. Hence, to understand the complexity of abduction one must also understand how restrictions on the representation of hypotheses and manifestations influence the complexity. This is partially the motivation for the present study where we systematically analyse the complexity of abduction under simultaneous restrictions on the representation of hypotheses, manifestations, *and* the knowledge base.

The paper is organized as follows. We first give some preliminaries about propositional logic, Schaefer's framework and complexity classes (Section 2) and the definition of the abduction problems we are interested in (Section 3). We then survey previous work and give an overview of our results (Section 4). The technical content of the paper follows: we give several generic reductions between abduction problems (Section 5) and recall several well-known tools for studying abduction (Section 6); we then give the complexity results (Sections 7 to 12) by distinguishing the various families of restrictions over the knowledge base. These results are summarized in Section 13, and some further restrictions on the problems are discussed in Section 14. Finally, we discuss what makes abduction hard in Section 15, and we conclude in Section 16.

2 Preliminaries

In the paper we consider restrictions over knowledge bases seen either as propositional formulas or as conjunctions of constraints taken from a fixed language. We introduce here the corresponding basic definitions and notations, define the restrictions that we will consider, and briefly recall some complexity notions.

2.1 Propositional logic

Boolean variables are usually denoted by x, possibly with subscripts and superscripts. A *literal* is either a variable (*positive* literal) or the negation of one (*negative* literal). Literals are usually denoted by ℓ. The opposite of literal ℓ is written $\overline{\ell}$, i.e., $\overline{\ell} = x$ if $\ell = \neg x$ and vice-versa.

A *clause* is a finite disjunction of literals, written $C = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_n$. A formula in *Conjunctive Normal Form* (*a CNF* for short) is a finite conjunction of clauses, written $KB = C_1 \wedge C_2 \wedge \cdots \wedge C_k$. A *term* is a finite conjunction of literals, written $T = \ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n$. Variables, literals, clauses and terms are considered special cases of formulas. A *(linear) equation* is an equation of the form $(x_1 \oplus x_2 \oplus \cdots \oplus x_n = a)$ with $a \in \{0, 1\}$ ($\oplus$ denotes addition modulo 2). An *affine* formula is a finite conjunction of linear equations. Observe that affine formulas are not CNFs.

If φ is a formula, $\mathit{Vars}(\varphi)$ denotes the set of all variables that occur in φ. Given a set of variables V, $\mathit{Lits}(V)$ denotes the set of all literals formed upon the variables in V, i.e., $\mathit{Lits}(V) = V \cup \{\neg x \mid x \in V\}$. If L is a set of literals, $\bigwedge L$ denotes their conjunction, and $N(L)$ denotes the set of opposite literals $\{\overline{\ell} \mid \ell \in L\}$. A clause, term, or CNF is said to be *positive* (resp. *negative*) if all the literals which occur in it are positive (resp. negative), and a clause or term is said to be *unit* if it contains exactly one literal. The empty clause is written $\bot$. Observe that $\mathit{Vars}(\bot) = \emptyset$ and that $\bot$ is always false. Similarly, $(0 = 1)$ denotes the always false linear equation with no variable.

An *assignment* μ to a set of variables V is a mapping from V to $\{0, 1\}$. When the order of the variables is clear, we write assignments as words, e.g., $\mu = 011$ if $\mu(x_1) = 0$ and $\mu(x_2) = \mu(x_3) = 1$. If KB is a formula, an assignment μ to $\mathit{Vars}(KB)$ or to a superset of $\mathit{Vars}(KB)$ is said to *satisfy* KB, or to be a *model* of KB, if it makes KB evaluate to 1 with the usual rules for the connectives. We write $\mu \models KB$.

A formula KB is said to be *satisfiable* if it has at least one model. It is said to *entail* a formula KB' if every assignment to $\mathit{Vars}(KB) \cup \mathit{Vars}(KB')$ which satisfies KB also satisfies KB', written $KB \models KB'$. If moreover $KB' \models KB$, then KB and KB' are said to be *(logically) equivalent*, written $KB \equiv KB'$.

Example 1 *The formula:*

$$KB = (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2)$$

is a CNF containing 4 clauses. We have $\mathit{Vars}(KB) = \{x_1, x_2, x_3, x_4\}$*, and thus* $\mathit{Lits}(\mathit{Vars}(KB)) = \{x_1, \neg x_1, \ldots, x_4, \neg x_4\}$*. The assignment* μ *to* $\mathit{Vars}(KB)$ *defined by* $\mu(x_1) = 0$*,* $\mu(x_2) = 1$*,* $\mu(x_3) = 0$*, and* $\mu(x_4) = 0$ *(*$\mu = 0100$ *for short) is a model of* KB*, while* 0000 *is not. The formula* $KB' = (x_1 \oplus x_2 = 1) \wedge (x_2 \oplus x_3) = 0$ *is an affine formula. Its set of models is* $\{011, 100\}$*, and it can be seen that* $KB' \models KB$*, while* $KB \not\models KB'$ *and thus,* $KB \not\equiv KB'$*.*

2.2 Classes of clauses and equations

We will first impose restrictions on the knowledge bases of abduction problems in the form of classes of clauses and linear equations. These classes are reported in Table 1 (page 11, second and third columns) and described below.

The class of all clauses is denoted by $\mathcal{C}_{CNF}$. A clause C is said to be 1-valid if it is satisfied by assigning 1 to all variables in it. Equivalently, a clause is 1-valid if it contains at least one positive literal. The class of all 1-valid clauses is denoted by $\mathcal{C}_{1v}$. Dually, $\mathcal{C}_{0v}$ is the class of all 0-valid clauses.

A clause is *Horn* if it contains at most one positive literal. The class of all Horn clauses is denoted by $\mathcal{C}_{Horn}$. A clause is 1-valid Horn (also called *definite Horn*) if it contains exactly one positive literal. The class of all 1-valid Horn clauses is denoted by $\mathcal{C}_{1v-Horn}$. Analogously, a clause C is said to be *dual-Horn* (0-valid dual-Horn) if it contains at most one (exactly one) negative literal, and the class of all dual-Horn clauses (0-valid dual-Horn clauses) is denoted by $\mathcal{C}_{dHorn}$ ($\mathcal{C}_{0v-dHorn}$). Moreover, the class of all 1-valid dual-Horn clauses is denoted by $\mathcal{C}_{1v-dHorn}$.

A clause is said to be *bijunctive* if it contains at most two literals, and *implicative* if it contains zero (empty clause) or one literal, or is of the form $(\neg x_1 \vee x_2)$. The classes of all bijunctive and of all implicative clauses are denoted by $\mathcal{C}_{bij}$ and $\mathcal{C}_{impl}$, respectively. A clause is said to be IHS-$B-$ if it is implicative or negative, and the class of all IHS-$B-$ clauses is denoted by $\mathcal{C}_{IHSB-}$. Similarly, a clause is said to be IHS-$B+$ if it is implicative or positive, and the class of all IHS-$B+$ clauses is denoted by $\mathcal{C}_{IHSB+}$. Moreover, a clause is said to be *of width k* if it contains at most k literals. The classes of all IHS-$B-$ and IHS-$B+$ clauses of width k are denoted $\mathcal{C}_{IHSB-/k}$ and $\mathcal{C}_{IHSB+/k}$, respectively.

A clause is said to be *essentially negative* if it is negative or unit positive. The class $\mathcal{C}_{neg,=}$ contains all essentially negative clauses, as well as the equality relations $(x_1 = x_2)$. Observe that such equality relations are not clauses, thus we slightly abuse language. The reason why we allow the equality relation is that it makes the class $\mathcal{C}_{neg,=}$ equivalent in expressiveness to a relational clone, and thus, in particular, stable with respect to the complexity of abduction (see Section 2.3). Nevertheless, we also discuss the case when this relation is not allowed in the paper. Dually to $\mathcal{C}_{neg,=}$, the class $\mathcal{C}_{pos,=}$ contains all essentially positive clauses and the equality relations.

Recall that a *linear equation* is an equation of the form $(x_1 \oplus \cdots \oplus x_n = a)$, $a \in \{0,1\}$. The class of all linear equations is denoted by $\mathcal{E}_{aff}$. Similarly to the case of clauses we say that a linear equation is 1-valid (resp. 0-valid) if it is satisfied by assigning 1 (resp. 0) to all variables in it. The classes of all 1-valid and all 0-valid linear equations are denoted by $\mathcal{E}_{1v-aff}$ and $\mathcal{E}_{0v-aff}$, respectively.

A linear equation is *of width* 2 if it contains at most two variables. The class of all linear equations of width 2 is denoted by $\mathcal{E}_{\mathit{aff/2}}$.

Finally, the class $\mathcal{C}_{\mathit{unit},=}$ contains all unit clauses and equality relations (the same remark as for $\mathcal{C}_{\mathit{neg},=}$ applies). The class $\mathcal{C}_{\mathit{1v-unit},=}$ contains the unit positive clauses and the equality relations, and the class $\mathcal{C}_{\mathit{0v-unit},=}$ contains the unit negative clauses and the equality relations.

2.3 Constraint languages and Post's lattice

In this section we introduce the notions of constraint languages and relational clones. We also describe Post's classification of all Boolean relational clones, usually referred to as *Post's lattice.* A more detailed introduction to Post's lattice can be found in the survey articles [2,3].

Constraints generalize the notions of clauses and linear equations, and constraint languages generalize the notion of classes of clauses and equations. The set of all n-tuples of elements from $\{0,1\}$ is denoted by $\{0,1\}^n$. Such tuples are denoted as sequences or as words, e.g., $(0,1,1)$ or 011. Any subset of $\{0,1\}^n$ is called an *n-ary relation* on $\{0,1\}$. A (finite) *constraint language* over $\{0,1\}$ is an arbitrary (finite) set of (finitary) relations over $\{0,1\}$.

If Γ is a constraint language, then a *constraint* over Γ is an application of an n-ary relation $R \in \Gamma$ to an n-tuple of variables, written $R(x_1,\ldots,x_n)$ (possibly with repeated variables). A *formula over* Γ (or Γ*-formula*) is a finite conjunction of constraints over Γ, written $KB = R_1(x_{11},\ldots,x_{1n_1}) \wedge \cdots \wedge R_k(x_{k1},\ldots,x_{kn_k})$. A constraint over Γ is considered a special case of Γ-formula. Like for propositional formulas, we write $\mathit{Vars}(KB)$ for the set of all variables occurring in KB.

If $C = R(x_1,\ldots,x_n)$ is a constraint, an assignment μ to $\{x_1,\ldots,x_n\}$ or to a superset of $\{x_1,\ldots,x_n\}$ is said to *satisfy* C, or to be a *model* of C, if the n-tuple $(\mu(x_1),\mu(x_2),\ldots,\mu(x_n))$ is in R. If KB is a Γ-formula, then μ is said to *satisfy* KB if it satisfies all its constraints. The notions of satisfiability, entailment and equivalence are defined like for propositional formulas, and similarly for entailment and equivalence between a propositional formula and a Γ-formula.

Example 2 (continued) *The set* $R = \{0100, 0110, 0111, 1000, 1001\}$ *is a* 4*-ary relation, and* $R' = \{01,10\}$ *and* $R'' = \{00,11\}$ *are binary relations. Thus* $\Gamma = \{R, R', R''\}$ *is a constraint language. Then* $R(x_1,x_2,x_3,x_4) \wedge R''(x_3,x_4)$ *is a* Γ*-formula. It can be seen that this formula entails KB of Example 1. Now* $C = R(x_1,x_2,x_3,x_3)$ *is a* Γ*-constraint (over three variables) satisfied by exactly* $010, 011, 100$*.*

The unary relations $\mathrm{F} = \{0\}$ and $\mathrm{T} = \{1\}$ have a special role for abduction problems, as well as the binary relations $\mathrm{R}_= = \{00, 11\}$ and $\mathrm{R}_{\neq} = \{01, 10\}$.

Relations and formulas are linked to each other by the following definition.

Definition 3 (representation) *An n-ary relation R is said to be* represented *by a formula φ if $Vars(\varphi) = \{x_1, \ldots, x_n\}$ and $\varphi \equiv R(x_1, \ldots, x_n)$.*

Thus, slightly abusing notation, we will identify a clause (or equation) C on variables $x_1, \ldots, x_k$ with the k-ary relation consisting of its set of satisfying assignments. In particular, we will often identify the literal x (resp. $\neg x$) and the unary constraint $\mathrm{T}(x)$ (resp. $\mathrm{F}(x)$), so that if KB is a Γ-formula and L is a set of literals, then $KB \wedge \bigwedge L$ is considered a $\Gamma \cup \{\mathrm{T}, \mathrm{F}\}$-formula.

We also say that a relation is *Horn* if it can be represented by a Horn CNF. A constraint language Γ is said to be *Horn* if every relation in Γ is Horn. Thus, slightly abusing notation, we write $\mathcal{C}_{Horn}$ both for the class of all Horn clauses and for the (infinite) constraint language containing all the relations represented by Horn clauses. We use the same shorthands for the other classes of clauses and equations.

Example 4 (continued) *Relation R in Example 2 is represented by formula KB in Example 1. Relation $R' = \{01, 10\}$ is represented by the formula $(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$, and thus it is bijunctive. Observe that it is also represented by the formula $x_1 \oplus x_2 = 1$, and thus it is affine (of width 2).*

Central to our approach is the notion of a *relational clone*. Intuitively, the relations in the relational clone $\langle \Gamma \rangle$ are those which can be simulated by existentially quantified conjunctions of constraints over $\Gamma \cup \{\mathrm{R}_=\}$. We will show (in Section 5.2) that the complexity of abduction is stable under such simulations, from which it follows that the complexity for all (finite) constraint languages is determined by the complexity for one (finite) language in each relational clone.

Definition 5 *Let Γ be a constraint language. The* relational clone *of Γ, written $\langle \Gamma \rangle$, is the set of all relations R such that $R(x_1, \ldots, x_n)$ is logically equivalent to $\exists V KB$ (where n is the arity of R) for some set of variables V disjoint from $\{x_1, \ldots, x_n\}$ and some $\Gamma \cup \{\mathrm{R}_=\}$-formula KB with $Vars(KB) = \{x_1, \ldots, x_n\} \cup V$.*

Example 6 *Let R be the 4-ary relation represented by the CNF $(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4)$, and recall that F is the unary relation $\{0\}$. Let R' be the ternary relation represented by the formula $(x_1 \vee x_2) \wedge (x_2 \oplus x_3 = 0)$. It is easily seen that $R'(x_1, x_2, x_3) \equiv \exists x_4 R(x_1, x_1, x_2, x_4) \wedge \mathrm{F}(x_4) \wedge \mathrm{R}_=(x_2, x_3)$, and thus $R' \in \langle \{R, \mathrm{F}\} \rangle$.*

Definition 7 *A constraint language* Γ *is said to be a* relational clone *if* $\Gamma = \langle\Gamma\rangle$.

Emil Post gave a remarkable classification of all classes of Boolean functions which are closed under composition and projection [42]. Such classes of functions are referred to as *clones* and, as a result of Post's classification, the inclusion structure (under set inclusion) among Boolean clones is completely known. Later it was shown that there is a bijection between clones and relational clones [26,43] and that the inclusion structure (under set inclusion) among the relational clones follows from the inclusion structure among the clones. The classification of Boolean clones/relational clones is called *Post's lattice* and is presented in Figure 1 in terms of relational clones. The lines in the lattice represent set inclusion, i.e., a line from a relational clone ICl_1 to a relational clone ICl_2 lying above ICl_1 in the lattice, means that $ICl_1 \subseteq ICl_2$. The dotted lines represent infinite chains of relational clones (e.g., $IS_0^2, IS_0^3, \ldots, IS_0^n, \ldots$ for the rightmost line).

We give a definition of each relational clone that is relevant to our study in Table 1, taken from [11]. The other relational clones are not relevant in the sense that their complexity for abduction is always the same as one of its super-clones and one of its sub-clones.

As it turns out from the results in [11], most relational clones Γ correspond to a class $\mathcal{C}$ of clauses or equations, in the sense that every relation in Γ can be represented by a conjunction of clauses (equations) from $\mathcal{C}$, and every clause (equation) from $\mathcal{C}$, viewed as a relation, is in Γ. In particular, $\Gamma = \langle\mathcal{C}\rangle$.

Thus we define relational clones in the following manner. For each entry in the table, the relational clone named in the first column is the set of all relations which can be represented by a conjunction of clauses or equations as in the third column. For instance, the class BR is the relational clone of all relations which can be represented by a conjunction of clauses (that is, the class of all relations); IE_2 is the relational clone containing all Horn relations; IL_0 is the class of all relations which can be represented by a conjunction of linear equations all with 0 as their right member, and so on. When there is a standard name for the relational clone or class of formulas, it is given in the fourth column.

Only two relational clones cannot be defined in this manner. A relation is said to be *complementive* if for every tuple $(\mu_1, \ldots, \mu_n)$ in it, the tuple $(\mu_1 \oplus 1, \ldots, \mu_n \oplus 1)$ is also in it. Then IN_2 denotes the relational clone containing all complementive relations, and IN denotes that containing all complementive and 0-valid (and thus also 1-valid) relations.

Given a relational clone $\langle\Gamma\rangle$ it is easy to locate $\langle\Gamma \cup \{\mathrm{F}\}\rangle$ and $\langle\Gamma \cup \{\mathrm{T}\}\rangle$ in Post's lattice. This will be useful several times, so we state this easy fact for

R. cl.	**Class**	**Clauses/equations**	**Name**
General case			
BR	$\mathcal{C}_{CNF}$	all clauses	—
Complementive			
IN_2	—	see page 10	complementive
IN	—	see page 10	—
1-valid and 0-valid			
II_1	$\mathcal{C}_{1v}$	clauses containing at least one positive literal	1-valid
II_0	$\mathcal{C}_{0v}$	clauses containing at least one negative literal	0-valid
Horn and dual Horn			
IE_2	$\mathcal{C}_{Horn}$	clauses with at most one positive literal	Horn
IE_1	$\mathcal{C}_{1v-Horn}$	clauses with exactly one positive literal	definite Horn
IE_0	—	$(\neg x_1 \vee \cdots \vee \neg x_n)$, $n \geq 1$, $(x_1 \vee \neg x_2 \vee \cdots \vee \neg x_n)$, $n \geq 2$	—
IE	—	$(x_1 \vee \neg x_2 \vee \cdots \vee \neg x_n)$, $n \geq 2$	—
IV_2	$\mathcal{C}_{dHorn}$	clauses with at most one negative literal	dual Horn
IV_1	$\mathcal{C}_{1v-dHorn}$	$(\neg x_1 \vee x_2 \vee \cdots \vee x_n)$, $n \geq 2$, $(x_1 \vee \cdots \vee x_n)$, $n \geq 1$	—
IV_0	$\mathcal{C}_{0v-dHorn}$	clauses with exactly one negative literal	definite dual Horn
IV	—	$(\neg x_1 \vee x_2 \vee \cdots \vee x_n)$, $n \geq 2$	—
Bijunctive and IHS-B			
ID_2	$\mathcal{C}_{bij}$	clauses containing at most 2 literals	bijunctive
IM_2	$\mathcal{C}_{impl}$	$(\neg x_1 \vee x_2)$, (x_1), $(\neg x_1)$, $\perp$	implicative
IM	—	$(\neg x_1 \vee x_2)$	—
IS_{10}	$\mathcal{C}_{IHSB-}$	(x_1), $(\neg x_1 \vee x_2)$, $(\neg x_1 \vee \cdots \vee \neg x_n)$, $n \geq 0$	IHS-$B-$
IS_{10}^k	$\mathcal{C}_{IHSB-/k}$	(x_1), $(\neg x_1 \vee x_2)$, $(\neg x_1 \vee \cdots \vee \neg x_n)$, $k \geq n \geq 0$	IHS-$B-$ of width k
IS_{12}	$\mathcal{C}_{neg,=}$	(x_1), $(\neg x_1 \vee \cdots \vee \neg x_n)$, $n \geq 0$, $(x_1 = x_2)$	essentially negative
IS_{11}	—	$(\neg x_1 \vee x_2)$, $(\neg x_1 \vee \cdots \vee \neg x_n)$, $n \geq 0$	—
IS_{11}^k	—	$(\neg x_1 \vee x_2)$, $(\neg x_1 \vee \cdots \vee \neg x_n)$, $k \geq n \geq 0$	—
IS_1^k	—	$(x_1 = x_2)$, $(\neg x_1 \vee \cdots \vee \neg x_n)$, $n \leq k$	—
IS_{00}	$\mathcal{C}_{IHSB+}$	$(\neg x_1)$, $(\neg x_1 \vee x_2)$, $(x_1 \vee \cdots \vee x_n)$, $n \geq 0$	IHS-$B+$
IS_{00}^k	$\mathcal{C}_{IHSB+/k}$	$(\neg x_1)$, $(\neg x_1 \vee x_2)$, $(x_1 \vee \cdots \vee x_n)$, $k \geq n \geq 0$	IHS-$B+$ of width k
IS_{02}	$\mathcal{C}_{pos,=}$	$(\neg x_1)$, $(x_1 \vee \cdots \vee x_n)$, $n \geq 0$, $(x_1 = x_2)$	essentially positive
IS_{01}	—	$(\neg x_1 \vee x_2)$, $(x_1 \vee \cdots \vee x_n)$, $n \geq 0$	—
IS_{01}^k	—	$(\neg x_1 \vee x_2)$, $(x_1 \vee \cdots \vee x_n)$, $k \geq n \geq 0$	—
IS_0^k	—	$(x_1 = x_2)$, $(x_1 \vee \cdots \vee x_n)$, $n \leq k$	—
Affine			
IL_2	$\mathcal{E}_{aff}$	all linear equations	affine
IL_0	$\mathcal{E}_{0v-aff}$	$(x_1 \oplus \cdots \oplus x_n = 0)$, $n \geq 0$	—
IL_1	$\mathcal{E}_{1v-aff}$	$(x_1 \oplus \cdots \oplus x_n = a)$, $n \geq 0$, $a = n \pmod 2$	—
IL_3	—	$(x_1 \oplus \cdots \oplus x_n = a)$, n even, $a \in \{0,1\}$	—
IL	—	$(x_1 \oplus \cdots \oplus x_n = 0)$, n even	—
ID_1	$\mathcal{E}_{aff/2}$	$(0 = 1)$, $(x_1 = a)$, $(x_1 \oplus x_2 = a)$, $a \in \{0,1\}$	affine of width 2
ID	—	$(0 = 1)$, $(x_1 \oplus x_2 = a)$, $a \in \{0,1\}$	—
Unit			
IR_2	$\mathcal{C}_{unit,=}$	(x), $(\neg x)$, $(x_1 = x_2)$	—
IR_1	$\mathcal{C}_{1v-unit,=}$	(x), $(x_1 = x_2)$	—
IR_0	$\mathcal{C}_{0v-unit,=}$	$(\neg x)$, $(x_1 = x_2)$	—
IBF	—	$(x_1 = x_2)$	—

Table 1
Classes of clauses and equations

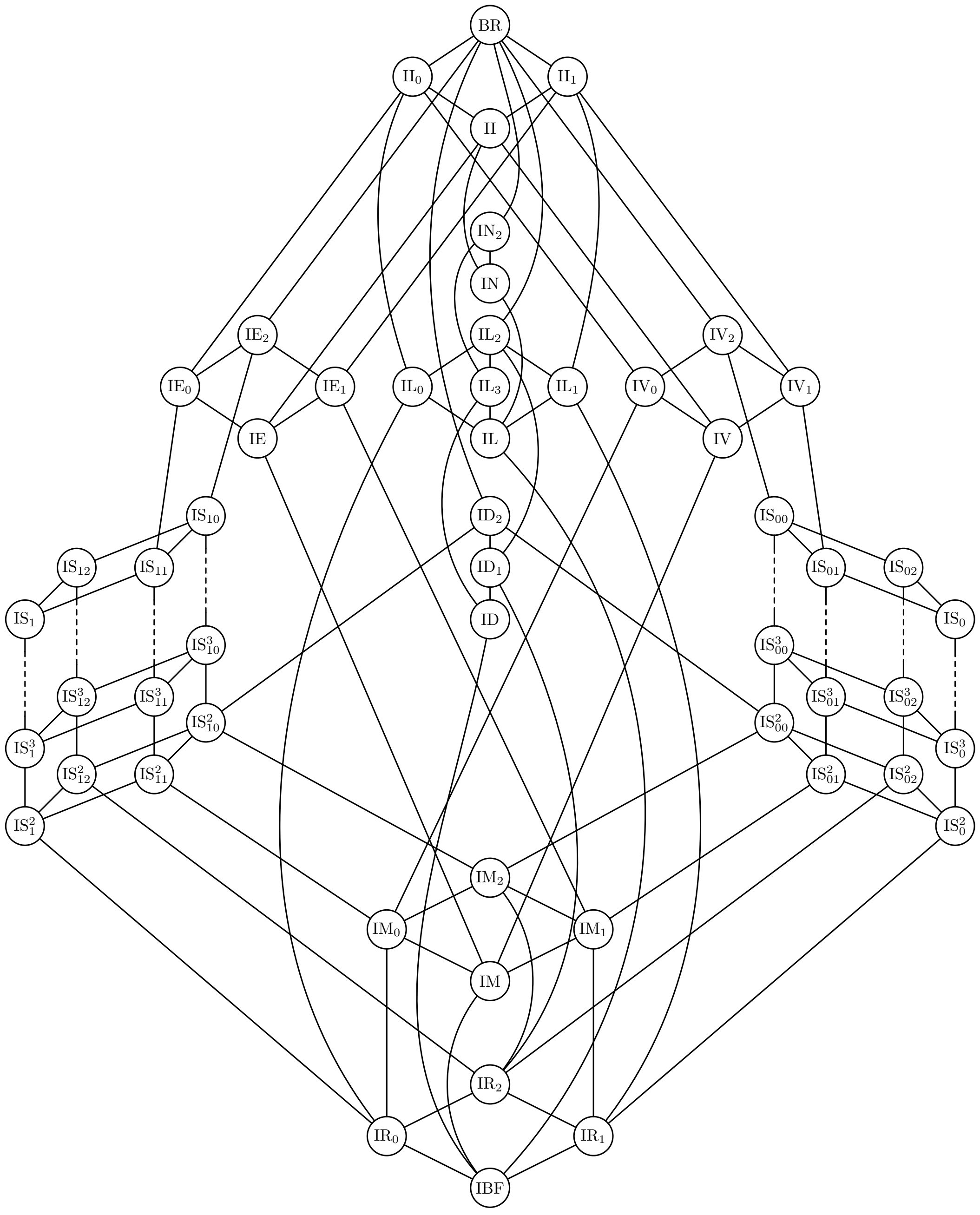

Fig. 1. Post's lattice of Boolean relational clones in the form of a Hasse diagram.

future reference.

Proposition 8 $\langle \Gamma \cup \{\mathrm{F}\} \rangle$ *is the least upper bound (in Post's lattice) of* $\langle \Gamma \rangle$ *and* IR_0*.* $\langle \Gamma \cup \{\mathrm{T}\} \rangle$ *is the least upper bound of* $\langle \Gamma \rangle$ *and* IR_1*.*

2.4 Complexity notions

We assume that the reader is familiar with the basic notions of complexity theory, but we briefly recall the following. P is the class of decision problems solvable in deterministic polynomial time. NP is the class of decision problems solvable in nondeterministic polynomial time. $\Sigma_2^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}}$ is the class of decision problems solvable in nondeterministic polynomial time with access to an NP-oracle. A problem is NP-hard (Σ_2^{P}-hard) if every problem in NP (Σ_2^{P}) is polynomial-time reducible to it. A problem is NP-complete (Σ_2^{P}-complete) if it is in NP and NP-hard (resp. in Σ_2^{P} and Σ_2^{P}-hard). Throughout the paper we assume that P, NP, and Σ_2^{P} are pairwise distinct.

coNP is the dual complexity class of NP. That is, a problem is in coNP (resp. coNP-complete) if its complement is in NP (resp. NP-complete).

If a problem Π can be reduced to a problem Π' under polynomial-time many-one reductions, then we write $\Pi \leq_{\mathrm{P}} \Pi'$. If $\Pi \leq_{\mathrm{P}} \Pi'$ and $\Pi' \leq_{\mathrm{P}} \Pi$, then we write $\Pi \equiv_{\mathrm{P}} \Pi'$. We write $\Pi \leq_{\mathrm{P}} \Pi_1, \Pi_2$ if $\Pi \leq_{\mathrm{P}} \Pi_1$ and $\Pi \leq_{\mathrm{P}} \Pi_2$, and dually for $\Pi_1, \Pi_2 \leq_{\mathrm{P}} \Pi$.

3 The abduction problem

We define here the various abduction problems which we study. In order to clarify the presentation, we first define the general abduction problem, without any restriction on hypotheses and manifestations, and then its restrictions.

3.1 General abduction problem

The abduction problem with restrictions on the knowledge base only is defined as follows. Recall that classes of clauses or equations are identified to constraint languages, so that the following definition encompasses them.

Problem 9 (Abd(Γ)) *Let Γ be a constraint language. An instance P of the abduction problem* $\textsc{Abd}(\Gamma)$ *is a tuple (V, H, M, KB), where*

- *V is a set of variables,*
- *$H \subseteq Lits(V)$ is the set of hypotheses,*
- *M is a propositional formula (the manifestation), with $Vars(M) \subseteq V$, and*
- *KB is a Γ-formula, with $Vars(KB) \subseteq V$.*

The question is whether there exists an explanation for P, i.e., a set $E \subseteq H$

such that $KB \wedge \bigwedge E$ is satisfiable and $KB \wedge \bigwedge E$ entails M.

We need to make assumptions on how the relations in the Γ-formula for the KB are represented. When the constraint language Γ is not specified by giving a class of equations or clauses, then we naturally assume that all the relations in all constraints in the Γ-formula are given in extension. That is, by listing all the tuples belonging to each relation explicitly. When Γ is specified by giving a class of equations or clauses, then we choose to let the Γ-formula be represented by a system of equations or CNF-formula, respectively. For complexity matters, the size of $P = (V, H, M, KB)$ is defined to be the total number of occurrences of variables in it, and additionally in the case where the Γ-formula is given in extension, the number of tuples in the relations in the Γ-formula.

As an illustration, consider, e.g., the language $\Gamma = \{R_=, R_{\neq}\}$, that is, the language containing the binary equality and difference relations. Then an instance of $\textsc{Abd}(\Gamma)$ is a tuple (V, H, M, KB), where KB is a conjunction of equality and difference constraints over V, H is a set of literals over V, and M is a propositional formula over V. Similarly, an instance of $\textsc{Abd}(\mathcal{C}_{Horn})$ is a tuple (V, H, M, KB), where KB is a Horn CNF and H and M are as before.

Observe that we can assume without loss of generality that KB is satisfiable in an instance. Indeed, if KB is unsatisfiable, then $KB \wedge \bigwedge E$ cannot be satisfiable (for any E), and thus there can be no explanation. Nevertheless, we do not enforce this assumption since satisfiable KBs cannot be distinguished from unsatisfiable ones efficiently in general.

The notion of explanation is illustrated in the following example.

Example 10 (continued) *Consider again $KB = (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2)$ of Example 1. The tuple*

$$P = (V = \{x_1, x_2, x_3, x_4\}, H = \{x_3, \neg x_4, x_4\}, M = x_2, KB)$$

is an instance of $\textsc{Abd}(\mathcal{C}_{CNF})$. It has exactly three explanations, namely $E_1 = \{x_3, \neg x_4\}$, $E_2 = \{x_3, x_4\}$, and $E_3 = \{x_3\}$.

On the contrary, $\{x_1\}$ is not an explanation, because it is not a subset of H, $\{x_4\}$ is not because it does not entail M together with KB, and finally $\{x_1, x_3\}$ is not because it is not consistent with KB (and it is not a subset of H).

In the literature it is common to impose a preference relation on the different explanations in order to concentrate on the most interesting/preferred explanations. One common preference criterion is subset minimality, i.e., an explanation E is said to be subset minimal ($\subset$-minimal) if there is no other explanation E' such that $E' \subset E$. We want to emphasize that in this paper we do not impose any preference relation on explanations. Note that in the example above $E_3 \subset E_1, E_2$, which does not matter in our setting but would make E_3 the only preferred explanation with respect to the subset minimality criterion.

3.2 Abduction under restrictions

We now define the abduction problem under restrictions on manifestations and hypotheses. We will use the following notation for restrictions on manifestations:

- PosLits denotes the class of all positive literals,
- NegLits denotes the class of all negative literals, and
- Lits denotes the class of all literals.
- Similarly, PosClauses, NegClauses and Clauses denote classes of clauses,
- PosTerms, NegTerms and Terms denote classes of terms, and
- PosCNFs, NegCNFs and CNFs denote classes of CNFs.

Problem 11 (abduction under restrictions) *Let Γ be a constraint language, and let $\mathcal{M}$ be a class of propositional formulas. An instance P of one of the decision problems* P-Abd(Γ,$\mathcal{M}$), N-Abd(Γ,$\mathcal{M}$), V-Abd(Γ,$\mathcal{M}$), *or* L-Abd(Γ,$\mathcal{M}$) *is an instance (V, H, M, KB) of the problem* Abd(Γ) *such that $M \in \mathcal{M}$ holds and*

- *$H \subseteq V$, that is, every $h \in H$ is positive, for P-Abd,*
- *$H \subseteq N(V)$ (every $h \in H$ is negative) for N-Abd,*
- *$H = V_H \cup N(V_H)$ for some $V_H \subseteq V$ (H is closed under complement) for V-Abd,*
- *$H \subseteq Lits(V)$ (H is unrestricted) for L-Abd.*

For all four problems, the question is whether there is an explanation for P.

Our motivation for studying various restrictions on the hypotheses and manifestations is to understand how these restrictions affect the complexity of the problem. We note that as far as no restriction is imposed to the knowledge base, generic polynomial-time reductions exist between all these restrictions. For instance, moving from a term M to a (fresh) variable m as the manifestation can be done up to adding $M \leftrightarrow m$, or even $M \rightarrow m$, to the knowledge base. Similarly, a negative hypothesis of the form $\neg h \in H$ can be removed by replacing $\neg h$ with a fresh variable h' in H, and adding $\neg h \leftrightarrow h'$ to KB.

This is the motivation for studying only restricted cases, such as positive hypotheses and single-variable manifestations, in generic heuristic approaches to solving abduction problems (see, e.g., Marquis' survey [37]). However, when the knowledge base is restricted to a particular language, as is the case here, such reductions in general fail to preserve the language. This is why we study various restrictions on H and M, and investigate their impact on the complexity of the problem. Note that in the particular case of Horn knowledge bases and single-literal manifestations, Eiter and Makino were similarly interested in the impact of the polarity imposed to the manifestation [23]. In our terms, among other results they compared the complexity of L-ABD($\mathcal{C}_{Horn}$,POSLITS) to that of L-ABD($\mathcal{C}_{Horn}$,NEGLITS).

Further restrictions on the problem, such as assuming that the manifestation is satisfiable or that every hypothesis occurs in the knowledge base, are of interest for practical purposes. Nevertheless, apart from very restricted cases they do not affect the complexity of the problem, so we only discuss them briefly in Section 14.

To conclude this section, observe that we do not consider DNF manifestations, that is, disjunctions of terms as manifestations. The reason for that is that already deciding $KB \models M$, where M is a DNF formula, is coNP-complete even if KB is empty (tautological), since then it is equivalent to deciding whether M is tautological. Consequently, there is no hope in finding interesting (tractable) classes of knowledge bases for abduction with such manifestations.

4 Related work and overview of results

In this section we first survey relevant literature on the complexity of abduction, and then give an overview of our results. We finally discuss how our results may be used in several contexts.

4.1 Related work

The earliest work about assumption-based propositional abduction is Reiter and de Kleer's, in the ATMS framework [45]. As far as this paper is concerned, the main result there is a characterization of explanations by means of prime implicates, reported in Section 6.2.

Subsequently, in the early nineties, several results were given concerning the computational complexity of abduction. Among them, Bylander *et al.* [7] consider *set-covering abduction*. That is, they assume that each hypothesis comes with the set of atomic manifestations which it can explain, and the question is to find a set of hypotheses which explains all manifestations. Under various assumptions on the interaction between hypotheses (incompatibility, independence, etc.), the authors investigate the frontier between tractable and intractable such problems. Since this objective is similar to ours, we give a detailed comparison of our work with theirs in Section 15.2. The complexity of abduction has also been studied for other representations of the knowledge base, for example when the knowledge base is represented by ordered binary decision diagrams [30].

The first complexity results in the logic-based setting were given by Selman and Levesque [48] for abduction problems with propositional Horn knowledge bases. Then Eiter and Gottlob performed a more systematic complexity analysis [20]. The main results, which we will use for our classification purposes, are the following ones.

Proposition 12 ([48]) P-ABD($\mathcal{C}_{Horn}$,POSLITS) *is* NP*-complete. Hardness holds even if the knowledge base is also restricted to be acyclic Horn.*

Proposition 13 ([20]) *The general problem* ABD($\mathcal{C}_{CNF}$) *is* Σ_2^{P}*-complete. Hardness holds even if all hypotheses are positive and manifestations are restricted to positive terms, that is,* P-ABD($\mathcal{C}_{CNF}$,POSTERMS) *is also* Σ_2^{P}*-complete.*

Proposition 14 ([20]) P-ABD($\mathcal{C}_{1v-Horn}$,POSTERMS) *is in* P.

Due to the intractability results of Propositions 12 and 13, several authors investigated polynomial classes of the abduction problem. Eshghi [24] gives a rather technical class based on acyclic Horn formulas (see also del Val's discussion of this result [16]); however, this class is not captured by our framework. Zanuttini [51] also gives several polynomial classes using the notion of projection, which is presented in Section 6.3. He also gives new proofs for several folklore polynomial classes (discussed by, e.g., Marquis [37]) and for classes of DNF formulas (which are not captured by our framework). The results which are relevant here are the following ones.

Proposition 15 ([51]) V-ABD($\mathcal{E}_{aff}$,CLAUSES) *is in* P.

Proposition 16 (see [37,51]) V-ABD($\mathcal{C}_{pos,=}$,CLAUSES), V-ABD($\mathcal{C}_{neg,=}$,CLAUSES), *and* V-ABD($\mathcal{C}_{bij}$,CLAUSES) *are in* P.

Finally, two classification results have recently been given in Schaefer's framework, concerning some restrictions which we study here. We however wish to emphasize that these results were given for finite constraint languages only, whereas we are interested here in both finite constraint languages and infinite (clausal) languages.

Theorem 17 (L-Abd [40]) *Let* Γ *be a finite constraint language. Then* L-ABD(Γ,TERMS) *is in* P *if* Γ *is in* ID_1*. Otherwise, it is* NP*-complete if* Γ *is in* IE_2*,* IV_2*,* ID_2*, or* IL_2*. Otherwise, it is* Σ_2^{P}*-complete.*

Theorem 18 (V-Abd [12]) *Let* Γ *be a finite constraint language. Then* V-ABD(Γ,POSLITS) *is in* P *if* Γ *is in* $IL_2, ID_2, IS_{10}, IS_{02}, IS_{00}^k$ *(for some* k*), or* IE_1*. Otherwise, it is* NP*-complete if* Γ *is in* IE_2 *or* IV_2*. Otherwise, it is* Σ_2^{P}*-complete.*

V-ABD(Γ,LITS) *is in* P *if* Γ *is in* $IL_2, ID_2, IS_{12}, IS_{02}, IS_{10}^k$*, or* IS_{00}^k *(for some* k*). Otherwise, it is* NP*-complete if* Γ *is in* IE_2 *or* IV_2*. Otherwise, it is* Σ_2^{P}*-complete.*

As concerns Theorem 18, observe in particular that there is no finite language which is in IS_{00} or IS_{01} but not in IS_{00}^k or IS_{01}^k for any $k \in \mathbb{N}$. In fact, infinite constraint languages Γ such that $\langle\Gamma\rangle = IS_{01}$ yield NP-complete problems (see our Proposition 63).

4.2 Overview of results

In this paper, building on the aforementioned results, we perform a systematic study of the computational complexity of propositional abduction. This study allows us to give the complexity of problems P-ABD(Γ,$\mathcal{M}$), N-ABD(Γ,$\mathcal{M}$), V-ABD(Γ,$\mathcal{M}$), and L-ABD(Γ,$\mathcal{M}$) when

- $\mathcal{M}$ is any of POSLITS, NEGLITS, LITS, and similarly for clauses, terms, and CNFs instead of literals,
- Γ is any constraint language or any clausal or equational language.

To that aim, we reuse the results given in the literature for specific languages. In particular, we reuse the classifications in [40] and [12]. Observe in particular that the latter concerns some maximally easy problems for us, in the sense that no other problem which we study can be reduced to them in polynomial time in a generic manner. So it proves very helpful for deriving hardness results. Dually, the former classification proves helpful for deriving membership results, as well as hardness results for CNF manifestations.

Nevertheless, these classifications leave large “gaps”. For instance, if Γ is a finite constraint language such that $\langle\Gamma\rangle = IL_2$ (unrestricted affine constraints), V-ABD(Γ,POSLITS) is in P [12] while L-ABD(Γ,TERMS) is NP-complete [40]. Thus these classifications tell nothing about the tractability frontier between both restrictions.

In this paper, for filling several such gaps we strengthen some results given in the literature. Nevertheless, we also give a number of brand new results.

- We study the complexity of P-ABD and N-ABD for 0-valid and 1-valid languages. We exhibit new trivial and new Σ_2^{P}-hard problems.

- We study the complexity of P-ABD and N-ABD for complementive languages, and identify one minimal coNP-hard and two minimal Σ_2^{P}-hard problems.
- We complete the literature on abduction with Horn and dual Horn knowledge bases (mainly Selman and Levesque's [48] and Eiter and Gottlob's [20] results). Our main new result is that P-ABD($\mathcal{C}_{dHorn}$,CNFS) is in P.
- We complete the literature on abduction with bijunctive and IHS-B knowledge bases. Abduction over bijunctive and IHS-B constraint languages are particularly interesting since the borderline between tractability and NP-hardness is mainly situated among problems over such constraint languages. We identify two new polynomial problems, namely L-ABD($\mathcal{C}_{impl}$,POSCNFS) and L-ABD($\mathcal{C}_{neg,=}$,POSCNFS), and many new hardness results with a unified reduction from satisfiability problems.
- Similarly, we complete the literature for affine knowledge bases. We give two new tractability results, namely for L-ABD($\mathcal{E}_{aff}$,CLAUSES) and V-ABD($\mathcal{E}_{aff}$,TERMS), and several new hardness results using original reductions from satisfiability problems.

We also wish to emphasize that as far as we know, CNF manifestations had never been studied before.

These results allow us to complete the picture on the complexity of propositional abduction, and in particular to identify the tractability frontier (between P and NP-hard). This frontier, as we discuss in Section 15, can be characterized by very simple conditions. Moreover, it parallels the frontier identified by Bylander *et al.* for set-covering abduction [7].

4.3 Applications

As already evoked, our results are essentially interesting from a complexity-theoretic and from an AI point of view. On the complexity-theoretic side, we give the complexity of abduction for various, fine-grained restrictions, and it turns out that abduction is always either in P, NP-complete, coNP-complete, or Σ_2^{P}-complete. This is interesting since Ladner's result states that if P $\neq$ NP, then there exist problems in NP that are neither in P nor NP-complete [33]. Such problems are said to be of intermediate complexity.

One way to interpret our results is that the infinite class of abduction problems that we study do not contain such problems of intermediate complexity. For further discussions on the complexity-theoretic topic of finding large subclasses of NP which do not contain problems of intermediate complexity, we refer the reader to Feder and Vardi's seminal paper [25].

Finally, since the complexity of abduction spans four classes (P, NP, coNP, and $\Sigma_2^{\rm P}$) and because abduction is a central problem in nonmonotonic reasoning (see, e.g., [5,37]), our results may serve as starting points for deriving complexity results for other problems, by using reductions to or from abduction.

From the AI point of view, our results may help the designers of knowledge-based agents or expert systems to choose the appropriate knowledge representation language. Indeed, depending on the application, and especially on the constraints on resolution of abduction problems, it might be necessary to ensure that such problems will be solved efficiently, or it may be acceptable that they are NP-hard[2]. Moreover, depending on the application, the sets of hypotheses and the manifestations may be restricted to particular classes. Then, using our results and the characteristics and requirements of the application, the designer of a knowledge-based system can choose the appropriate knowledge representation language. In particular, she might choose the most expressive one while respecting the tractability constraints.

We acknowledge that our restrictions on knowledge bases cannot capture all possible propositional knowledge representation languages. For instance, they cannot capture the class of Horn-renamable CNF formulas, or that of acyclic Horn formulas, since they impose global restrictions on formulas.

Nevertheless, classes defined by local properties are very important for knowledge representation. Indeed, they are stable under conjunction, that is, if KB_1 and KB_2 are in one of these classes, then so is $KB_1 \wedge KB_2$. This makes them suitable for merging mutually consistent theories without losing computational properties of each (simply conjunct both) and is important for knowledge approximation purposes, since such classes define a unique least upper bound [47]. Moreover, they have been given complete pictures of complexity for various reasoning tasks (e.g., inference under circumscription [39] and several tasks in default reasoning [9]), which allows to choose a knowledge representation language under constraints stemming from several tasks. Last but not least, the relational clone generated by any constraint or clausal/equational language can be recognized in polynomial time [11]. Thus an agent may make decisions depending on its current knowledge base. It can, for instance, decide not to try to find an exact solution to a planning problem through abduction because its current knowledge base is not tractable for it, and adopt another strategy, such as approximate planning.

[2] Note that even if the problem at hand is NP-hard, or even $\Sigma_2^{\rm P}$-hard, most instances may be solved efficiently in practice. This may be done, for instance, using state-of-the-art solvers for Quantified Boolean formulas, which is the approach in [19].

5 Reductions between abduction problems

We have defined restrictions on the abduction problem along three dimensions: 12 different types of manifestations, 4 different types of hypotheses, and an infinite number of restrictions over knowledge bases. Since our goal is to give a complexity classification of the problem for each combination of restrictions, this section explains how to restrict to a finite and limited number of cases.

5.1 Reductions between restrictions on manifestations and hypotheses

The following reductions are obvious.

Lemma 19 *Let* Γ *be a constraint language and let* $\mathcal{M}$ *and* $\mathcal{M}'$ *be classes of propositional formulas such that* $\mathcal{M} \subseteq \mathcal{M}'$*. Then* L-Abd($\Gamma$,$\mathcal{M}$) $\leq_{\mathrm{P}}$L-Abd(Γ,$\mathcal{M}'$)*. The same result holds for* P-Abd*,* N-Abd*, or* V-Abd *instead of* L-Abd*.*

Lemma 20 *Let* Γ *be a constraint language, and let* $\mathcal{M}$ *be a class of propositional formulas. Then* P-Abd(Γ,$\mathcal{M}$)*,* N-Abd(Γ,$\mathcal{M}$)*,* V-Abd(Γ,$\mathcal{M}$)*,* $\leq_{\mathrm{P}}$L-Abd(Γ,$\mathcal{M}$)*.*

5.2 Reductions between restrictions on knowledge bases

In this section we show how Post's lattice can be used to reduce the number of restrictions on knowledge bases that need to be considered when classifying the complexity of the abduction problem. This approach via Post's lattice is crucial for obtaining our complexity classifications. We conclude this section by demonstrating how the approach can be used to classify the complexity of P-Abd(Γ,Terms) for every constraint language Γ and every clausal or equational language $\mathcal{C}$.

The key for the approach via Post's lattice is Lemma 22, which states that Abd(Γ') $\leq_{\mathrm{P}}$Abd(Γ) whenever Γ' is a finite and $\Gamma' \subseteq \langle\Gamma\rangle$ (for all restrictions over hypotheses and manifestations considered here). Hence, when studying the complexity of the abduction problem for *finite* constraint languages Γ, it is enough to consider one generating constraint language per relational clone. In particular, if Γ and Γ' are two finite constraint languages and $\langle\Gamma\rangle = \langle\Gamma'\rangle$, then Abd($\Gamma$) and Abd($\Gamma'$) are polynomial-time equivalent to each other.

We first need the following lemma, which allows to get rid of equality relations in knowledge bases.

Lemma 21 *Let* Γ *be a constraint language, and let* $\mathcal{M}$ *be any class of manifestations considered in this paper. Then* L-ABD($\Gamma \cup \{\mathrm{R}_=\}$,$\mathcal{M}$) $\leq_{\mathrm{P}}$L-ABD(Γ,$\mathcal{M}$). *The same holds for* V-ABD, P-ABD, *or* N-ABD *instead of* L-ABD.

PROOF. Let (V, H, M, KB) be an instance of L-ABD($\Gamma \cup \{\mathrm{R}_=\}$,$\mathcal{M}$). Build a knowledge base over Γ from KB in the following manner. For all constraints $\mathrm{R}_=(x_i, x_j)$ in KB replace every occurrence of x_j with x_i in KB, H, M, remove the constraint from KB, and remove x_j from V. Perform this identification iteratively, until KB does not contain any equality constraint any more. Clearly, this transformation can be performed in polynomial time and preserves the existence of an explanation. Finally, it is easily seen that it preserves any restriction on hypotheses and manifestations in the statement. □

We can now give the central lemma of our study. The proof checks that additional variables introduced while replacing a constraint in Γ' with its equivalent expression over Γ do not affect the existence of a solution.

Lemma 22 *Let* Γ *be a constraint language, and let* $\Gamma' \subseteq \langle\Gamma\rangle$ *be a finite constraint language (or a finite clausal or equational language). Let* $\mathcal{M}$ *be a class of propositional formulas. Then* L-ABD(Γ',$\mathcal{M}$) $\leq_{\mathrm{P}}$L-ABD(Γ,$\mathcal{M}$). *The same holds for* V-ABD, P-ABD, *or* N-ABD *instead of* L-ABD, *and for a class of clauses or equations* $\mathcal{C}$ *instead of a constraint language* Γ.

PROOF. Let $P' = (V', H', M', KB')$ be an instance of L-ABD(Γ',$\mathcal{M}$). Write $KB' = \bigwedge_{i \in I} R'_i(x'_{i,1}, \ldots, x'_{i,k_i})$ and for all $i \in I$, $V'_i = \{x'_{i,1}, \ldots, x'_{i,k_i}\}$. By the definition of a relational clone we know that for all $i \in I$, the constraint $R'_i(x'_{i,1}, \ldots, x'_{i,k_i})$ of KB' is logically equivalent to some formula $\exists V_i KB_i$ where $V_i \cap V'_i = \emptyset$ and KB_i is a $\Gamma \cup \{\mathrm{R}_=\}$-formula over $V_i \cup V'_i$. Importantly, for all $i \in I$ we assume $V_i \cap V' = \emptyset$ and for all $i, i' \in I$ with $i \neq i'$, we assume $V_i \cap V_{i'} = \emptyset$, i.e., all existentially quantified variables are fresh with respect to V' and different from each other. This is without loss of generality since the names of these variables are unconstrained.

We define an instance $P = (V, H, M, KB)$ of ABD($\Gamma \cup \{\mathrm{R}_=\}$) by

- $V = V' \cup \bigcup_{i \in I} V_i$,
- $H = H'$,
- $M = M'$, and
- $KB = \bigwedge_{i \in I} KB_i$.

In other words, we simply replace every constraint in KB' with its expression over $\Gamma \cup \{\mathrm{R}_=\}$, and we forget existential quantification. Clearly, this can be performed in linear time since languages are fixed and Γ' is finite, and thus all expressions of relations in Γ' can be stored in a lookup table once and for all.

We now claim that P and P' have exactly the same explanations. First, let E' be an explanation for P'. Then $KB' \wedge \bigwedge E'$ is satisfiable, thus the formula $\bigwedge_{i \in I} \exists V_i KB_i \wedge \bigwedge E'$ is satisfiable. Since all existentially quantified variables are fresh with respect to V' and different from each other, it follows that $KB \wedge \bigwedge E'$ is satisfiable. Now assume that $KB \wedge \bigwedge E'$ does not entail M. Then $KB \wedge \bigwedge E' \wedge \neg M$ is satisfiable. By the assumption on fresh variables again, it follows that $\bigwedge_{i \in I} \exists V_i KB_i \wedge \bigwedge E' \wedge \neg M$ is satisfiable, i.e., that $KB' \wedge \bigwedge E' \wedge \neg M$ is satisfiable. Since $M = M'$, this contradicts the fact that E' is an explanation for P'.

Thus every explanation E' for P' is an explanation for P, and the proof is similar for showing the converse. We conclude by invoking Lemma 21 for getting rid of equality constraints. $\square$

It is important to note that the reduction in the proof above preserves all the restrictions on hypotheses and manifestations considered in this paper. For example, if we reduce from a V-Abd(Γ',PosLits) instance, then the resulting instance will be a V-Abd(Γ,PosLits) instance. Thus, using Lemma 22, we will be able to give the complexity of all the restrictions on the Abd problem for any finite constraint language (Section 13) by considering the complexity of only one language per relational clone.

We now demonstrate the use of Post's lattice for classifying the complexity of Abd(Γ) problems by giving the complete classification of P-Abd(Γ,Terms) for every constraint language Γ and class of clauses/equations $\mathcal{C}$. The reasoning is similar for the other combinations of restrictions on hypotheses and manifestations.

Gathering together results for positive hypotheses and terms as manifestations from subsequent sections in the paper, and applying the (obvious) reductions in Section 5.1, we get the minimal set of results in Table 2. These results are minimal in the sense that they are irredundant with respect to reductions of the form P-Abd(Γ,Terms) $\leq_{\mathrm{P}}$P-Abd(Γ',Terms) as soon as $\Gamma \subseteq \langle \Gamma' \rangle$. In this table, for each clausal language $\mathcal{C}$, we give inside parentheses the corresponding relational clone, i.e., the relational clone ICl such that $ICl = \langle \mathcal{C} \rangle$.

The complete picture of complexity can now be obtained as follows.

First, as explained in Section 13.2, a upper bound for the complexity of a clausal or equational language $\mathcal{C}$ corresponding to a relational clone ICl carries over to every language Γ such that $\langle \Gamma \rangle \subseteq ICl$. Now, since all lower bounds

R. Clone/class	Complexity	Result
$\mathcal{C}_{CNF}$ (BR)	in Σ_2^{P}	Proposition 41
II_0	Σ_2^{P}-hard	Proposition 46
IN_2	Σ_2^{P}-hard	Proposition 47
$\mathcal{C}_{1v}$ (II_1)	in coNP	Proposition 44
IN	coNP-hard	Proposition 48
$\mathcal{C}_{Horn}$ (IE_2), $\mathcal{E}_{aff}$ (IL_2), $\mathcal{C}_{bij}$ (ID_2)	in NP	Proposition 35
IS_1^2	NP-hard	Proposition 62 (dual)
IL_0	NP-hard	Proposition 69
IL_3	NP-hard	Proposition 70
$\mathcal{C}_{1v-Horn}$ (IE_1), $\mathcal{E}_{1v-aff}$ (IL_1)	in P	Proposition 44
$\mathcal{C}_{dHorn}$ (IV_2)	in P	Proposition 52
$\mathcal{E}_{aff/2}$ (ID_1)	in P	Theorem 17

Table 2
Minimal set of results for P-ABD(Γ,TERMS)

are given for finite languages, the results on Figure 2 follow from the results in Table 2, where

- every result reported from the table is given by a bolded circle, and
- every other result follows from these together with reductions of the form P-ABD(Γ,TERMS) $\leq_{\mathrm{P}}$P-ABD(Γ′,TERMS) as soon as $\Gamma \subseteq \langle\Gamma'\rangle$ (Lemma 22).

Consequently, we have the complexity for every constraint language. For more details, we refer the reader to Section 13.2.

5.3 Exploiting the symmetry between 0 *and* 1

We now show how to exploit the symmetry between 0 and 1 (or positive and negative) in order to reduce the number of cases that need to be considered. We will use the notion of duality.

Definition 23 (dual) *The* dual *of a clause $C = (\ell_1 \vee \cdots \vee \ell_k)$ is the clause $C^d = (\overline{\ell_1} \vee \cdots \vee \overline{\ell_k})$. The dual of a class of clauses $\mathcal{C}$ is the class $\mathcal{C}^d = \{C^d \mid C \in \mathcal{C}\}$. The dual of an equation $Eqn = (x_1 \oplus \cdots \oplus x_n = a)$ is $Eqn^d = Eqn$ if n is even, and $Eqn^d = (x_1 \oplus \cdots \oplus x_n = a \oplus 1)$ otherwise. The dual of an n-ary Boolean relation R is $R^d = \{(\mu_1 \oplus 1, \ldots, \mu_n \oplus 1) \mid (\mu_1, \ldots, \mu_n) \in R\}$. The dual of a constraint language Γ is $\Gamma^d = \{R^d \mid R \in \Gamma\}$.*

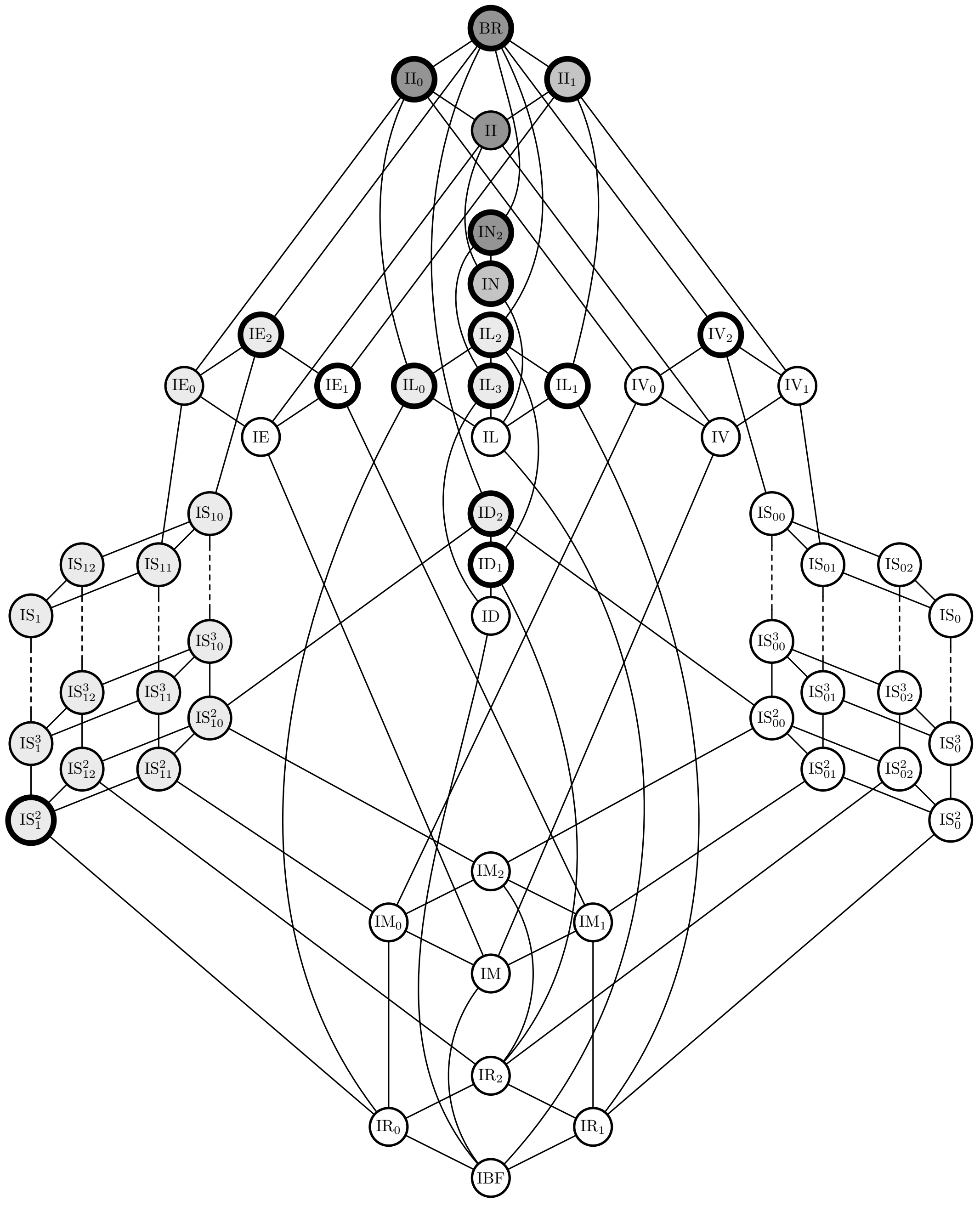

Fig. 2. Results for P-ABD(Γ,TERMS), white is in P, light grey is NP-complete, mid grey is coNP-complete, dark grey is Σ_2^{P}-complete.

Observe that in the graphical representation of Post's lattice of Boolean relational clones (Figure 1), the dual of a relational clone is simply its mirror image with respect to the vertical line through the center of the lattice [4]. Moreover, it is easy to see that if a relation R is represented by a conjunction of clauses (or equations) $C_1 \wedge \cdots \wedge C_k$, then R^d is represented by $C_1^d \wedge \cdots \wedge C_k^d$.

Example 24 (continued) *The dual of clause* $C = (x_1 \vee x_2 \vee \neg x_3)$ *is* $C^d = (\neg x_1 \vee \neg x_2 \vee x_3)$. *The dual of equation* $Eqn = (x_1 \oplus x_2 = 1)$ *is* Eqn *itself, and that of* $Eqn' = (x_1 \oplus x_2 \oplus x_3 = 0)$ *is* $(Eqn')^d = (x_1 \oplus x_2 \oplus x_3 = 1)$. *The dual of relation* $R = \{000, 010, 111\}$ *is* $R^d = \{111, 101, 000\}$. *As for languages, we have, e.g.,* $\mathcal{C}^d_{Horn} = \mathcal{C}_{dHorn}$, $\mathcal{E}^d_{aff/2} = \mathcal{E}_{aff/2}$ *and* $(IS^k_{00})^d = IS^k_{10}$.

These definitions allow us to state the following easy equivalences between problems which are in some sense symmetric to each other. The intuition is simply that switching the polarity of all literals in all components of the instance and replacing all relations (clauses/equations) by their duals preserve the existence of explanations.

Lemma 25 *Let* Γ *be a constraint language or a class of clauses/equations. Then the following equivalences hold:*

- P-Abd(Γ,PosLits) $\equiv_{\mathrm{P}}$N-Abd(Γ^d,NegLits),
- N-Abd(Γ,PosLits) $\equiv_{\mathrm{P}}$P-Abd(Γ^d,NegLits),
- V-Abd(Γ,PosLits) $\equiv_{\mathrm{P}}$V-Abd(Γ^d,NegLits), *and*
- L-Abd(Γ,PosLits) $\equiv_{\mathrm{P}}$L-Abd(Γ^d,NegLits).

They also hold when the manifestations are restricted to positive or negative clauses, terms, or CNFs instead of literals.

Consequently, in the rest of the paper we will only consider positive and unrestricted manifestations. The complexity for negative manifestations can be derived using Lemma 25.

Similarly, the lack of positive/negative polarity for manifestations and/or hypotheses allows to reduce the number of cases to consider. The following lemma is straightforward.

Lemma 26 *Let* Γ *be a constraint language or a class of clauses or equations, and let* $\mathcal{M}$ *be any of* Lits, Clauses, Terms, *or* CNFs. *Then,*

- N-Abd(Γ,$\mathcal{M}$) $\equiv_{\mathrm{P}}$P-Abd(Γ^d,$\mathcal{M}$),
- L-Abd(Γ,$\mathcal{M}$) $\equiv_{\mathrm{P}}$L-Abd(Γ^d,$\mathcal{M}$), *and*
- V-Abd(Γ,$\mathcal{M}$)*)* $\equiv_{\mathrm{P}}$V-Abd(Γ^d,$\mathcal{M}$).

5.4 Other reductions

The following lemma allows to impose polarities to hypotheses and manifestations when the language contains the disequality relation $\mathrm{R}_{\neq} = \{01, 10\}$.

Lemma 27 *Let* Γ *be a constraint language such that* $\mathrm{R}_{\neq} \in \langle\Gamma\rangle$, *and let* $\mathcal{M}$ *be* *either* PosLits *or* NegLits*. Then* L-Abd(Γ,Lits) $\leq_{\mathrm{P}}$P-Abd(Γ,$\mathcal{M}$)*,* N-Abd(Γ,$\mathcal{M}$)*. The same result holds for clauses, terms, or CNFs instead of literals.*

PROOF. Consider an instance (V, H, M, KB) of L-Abd(Γ,Terms). The desired instance of P-Abd(Γ,PosTerms) is simply obtained by introducing a fresh variable x' for any negative literal $\neg x$ in H or M, replacing $\neg x$ with x' and adding the constraint $\mathrm{R}_{\neq}(x, x')$ to KB. Finally, Lemma 22 allows to assume $\mathrm{R}_{\neq} \in \langle\Gamma\rangle$ instead of the stronger $\mathrm{R}_{\neq} \in \Gamma$. The reasoning is similar for the other restrictions. □

The two following lemmata concern conjunctive manifestations. The proof of the first one follows firstly from the definition of the abduction problem.

Lemma 28 *Let $P = (V, H, M, KB)$ be an instance of an abduction problem, where $M = (\varphi_1 \wedge \cdots \wedge \varphi_p)$ is a conjunction of formulas, and let $E \subseteq H$. Then E is an explanation for P if and only if for all $i \in \{1, \ldots, p\}$ it is an explanation for (V, H, φ_i, KB).*

Lemma 29 *Let $P = (V, H, M, KB)$ be an instance of an abduction problem where M is a term. Let ℓ be a literal formed upon a fresh variable $m \notin V$, and write C for the clause $(\ell \vee \bigvee_{\ell' \in M} \overline{\ell'})$. Then P has an explanation if and only if $P' = (V \cup \{m\}, H, \ell, KB \wedge C)$ has one.*

PROOF. Assume first that P' has an explanation E'. We first show that $KB \wedge \bigwedge E'$ entails $\bigwedge_{\ell' \in M} \ell'$. Assume to the contrary that there is a model μ of $KB \wedge \bigwedge E' \wedge (\bigvee_{\ell' \in M} \overline{\ell'})$. Then since m occurs only in C and $\bigvee_{\ell' \in M} \overline{\ell'} \subset C$, extending μ by $\mu \models \overline{\ell}$ yields a model of $KB \wedge C \wedge \bigwedge E' \wedge \overline{\ell}$, which contradicts the fact that E' is an explanation for P'. Thus $KB \wedge \bigwedge E'$ entails $\bigwedge_{\ell' \in M} \ell'$, that is, M. Now since $KB \wedge C \wedge \bigwedge E'$ is satisfiable, *a fortiori* $KB \wedge \bigwedge E'$ is satisfiable. Finally, E' is an explanation for P.

Conversely, assume that P has an explanation E. Then by definition of an explanation, there is a model μ of $KB \wedge \bigwedge E$; thus μ' defined to agree with μ over V and to satisfy ℓ is a model of $KB \wedge C \wedge \bigwedge E$. Now we also have $KB \wedge \bigwedge E \models M$. It follows that $KB \wedge C \wedge \bigwedge E \models \ell$, and finally, E is an explanation for P'. □

6 Tools and methods for studying abduction

In this section we present the main generic methods used in the literature for studying abduction problems from a computational point of view.

6.1 Complexity of satisfiability and deduction

By definition, solving an abduction problem involves solving a satisfiability and a deduction problem. We therefore recall some definitions and well-known complexity results about these two problems.

Problem 30 (Sat(Γ)) *Let Γ be a constraint language. An instance P of* SAT(Γ) *is a knowledge base KB over Γ, and the question is whether there exists at least one model of KB.*

Problem 31 (Deduction(Γ,$\mathcal{M}$)) *Let Γ be a constraint language and let $\mathcal{M}$ be a class of formulas. An instance P of* DEDUCTION(Γ,$\mathcal{M}$) *is a tuple (KB, Q), where*

- *KB is a knowledge base over Γ, and*
- *Q is a formula in $\mathcal{M}$.*

The question is whether KB entails Q.

Schaefer classified the complexity of SAT(Γ) for all possible finite constraint languages Γ [46]. Together with well-known results about the infinite languages studied here (see in particular [17] and [1]) we have the following theorem.

Theorem 32 (complexity of Sat) *Let Γ be a finite constraint language. Then* SAT(Γ) *is in* P *if $\Gamma \subseteq II_0$, $\Gamma \subseteq II_1$, $\Gamma \subseteq IE_2$, $\Gamma \subseteq IV_2$, $\Gamma \subseteq ID_2$, or $\Gamma \subseteq IL_2$. Otherwise, it is* NP*-complete. Moreover,* SAT($\mathcal{C}$) *is in* P *when $\mathcal{C}$ is one of $\mathcal{C}_{Horn}$, $\mathcal{C}_{dHorn}$, $\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{0v}$, and $\mathcal{C}_{1v}$.*

From the well-known facts that KB entails Q if and only if $KB \land \neg Q$ is unsatisfiable, and that KB entails $Q_1 \land Q_2$ if and only if it entails Q_1 and it entails Q_2, we have the following two corollaries of Theorem 32, which we will use in the paper.

Theorem 33 (upper bounds for Deduction) *Let $\mathcal{C}$ be a class of clauses or equations. Then* DEDUCTION($\mathcal{C}$,CNFS) *is in* coNP. *Moreover,* DEDUCTION($\mathcal{C}$,CNFS) *is in* P *when $\mathcal{C}$ is one of $\mathcal{C}_{Horn}$, $\mathcal{C}_{dHorn}$, $\mathcal{C}_{bij}$, and $\mathcal{E}_{aff}$.*

Theorem 34 (lower bound for Deduction) *Let Γ be a constraint language with $\langle\Gamma\rangle = II_1$. Then* DEDUCTION($\Gamma$,POSLITS) *is* coNP*-hard.*

PROOF. It follows from Post's lattice and Proposition 8 that $\langle II_1 \cup \{\mathrm{F}\}\rangle = BR$. Thus from Theorem 32 it follows that $\textsc{Sat}(II_1 \cup \{\mathrm{F}\})$ is NP-complete. Let $KB_{II_1 \cup \{\mathrm{F}\}} = KB_{II_1} \wedge \bigwedge_{i \in I} \mathrm{F}(x_i)$ be a knowledge base over $II_1 \cup \{\mathrm{F}\}$, where KB_{II_1} is a knowledge base over II_1 and I is nonempty (clearly, the problem remains NP-complete even with this assumption). Now let $i_0 \in I$, and let KB be the knowledge base obtained from KB_{II_1} by replacing every occurrence of x_i for some $i \in I \setminus \{i_0\}$ with x_{i_0}. By construction, KB is a knowledge base over II_1, and $KB \wedge \mathrm{F}(x_{i_0})$ is satisfiable if and only if $KB_{II_1 \cup \{\mathrm{F}\}}$ is. But $KB \wedge \mathrm{F}(x_{i_0})$ is unsatisfiable if and only if KB entails x_{i_0}, which concludes the proof. $\square$

Schaefer languages are precisely those maximal languages (for inclusion) for which deduction of clauses is tractable, i.e., $\mathcal{C}_{Horn}$, $\mathcal{C}_{dHorn}$, $\mathcal{C}_{bij}$ and $\mathcal{E}_{aff}$. From the results above we immediately get the following result.

Proposition 35 *For any type of restriction on hypotheses and manifestations considered in this paper and for all Schaefer languages $\mathcal{C}$,* $\textsc{Abd}(\mathcal{C})$ *is in* NP.

6.2 Prime implicates

The notion of a *prime implicate* is widely used for studying various computational problems in propositional logic, especially for problems in nonmonotonic reasoning. The relevance of this notion to abduction has been first pointed out by Reiter and de Kleer [45]. Marquis [37] gives a survey of the various notions of prime implicates, their use for nonmonotonic reasoning (including abduction), and methods for computing them.

Definition 36 (prime implicate) *Let KB be a propositional formula or a conjunction of constraints. A clause C is said to be a* prime implicate *of KB if KB entails C but no proper subclause of it.*

The following characterization of explanations, first shown by Reiter and de Kleer in the ATMS setting [45], will be of great use to us.

Lemma 37 ([45]) *Let $P = (V, H, M, KB)$ be an instance of an abduction problem, where $M = (m_1 \vee \cdots \vee m_p)$ is a nonempty clause, and let $E \subseteq H$. Then E is an explanation for P if and only if there is a prime implicate of KB of the form $(\overline{\ell_1} \vee \cdots \vee \overline{\ell_r} \vee m_{j_1} \vee \cdots \vee m_{j_s})$ with $\{\ell_1, \dots, \ell_r\} \subseteq E$, $\{j_1, \dots, j_s\} \subseteq \{1, \dots, p\}$ and $\{j_1, \dots, j_s\} \neq \emptyset$ (with possibly $\ell_i = m_j$ for some i, j).*

Finally, recall from Quine's result [44] that all the prime implicates of a CNF KB can be generated by *resolution*, i.e., by repeatedly adding $C_1 \vee C_2$ to KB

if there are a variable x and two clauses of the forms $(x \vee C_1)$ and $(\neg x \vee C_2)$ in KB, and removing clauses which are tautological or include others.

6.3 Projection

We will also use the notion of *projection* as a tool for studying the abduction problem. This notion is similar to the well-known notion of *elimination of middle terms*, or *existential abstraction*, and its use for abduction has been proposed in [51]. As for the complexity of computing projection in propositional logic, we refer the reader to [34].

Intuitively, projecting onto a set of variables V' amounts to existentially quantifying every other variable.

Definition 38 (projection of assignments) *Let μ be an assignment to a set of variables V, and let $V' \subseteq V$. The* projection of μ onto V', *denoted by $\mu_{|V'}$, is the assignment to V' which agrees with μ.*

Definition 39 (projection of formulas) *Let KB be a propositional formula or a conjunction of constraints, and let $V' \subseteq Vars(KB)$. A* projection of KB onto V' *is any knowledge base KB' with $Vars(KB') \subseteq V'$ and whose set of models over V' is $\{\mu_{|V'} \mid \mu \models KB\}$.*

Importantly, the projection of a formula is unique only up to logical equivalence. We will mainly use the following result. Its proof follows from Lemma 37 when the manifestation is a literal or clause, since it is well-known that a projection of a knowledge base KB onto a set of variables preserves the prime implicates of KB over this set [34, Proposition 16]. When the manifestation is a term of CNF, the proof follows from the literal or clause case together with Lemma 28.

Lemma 40 *Let $P = (V, H, M, KB)$ be an instance of any abduction problem. Let V' be any set of variables with $Vars(H) \cup Vars(M) \subseteq V' \subseteq V$, and let KB' be a projection of KB onto V'. Then the explanations for (V', H, M, KB') are exactly the explanations for P.*

When computable efficiently, projection used as above allows to circumvent the difficulty of what Selman and Levesque call the *support selection task* [48]. They argue that this task lies at the core of the computational difficulty of abduction, as witnessed by their study of the Horn case. We come back to this issue in our discussion (Section 15).

One case when a projection of a knowledge base can be computed efficiently is when this knowledge base has a polynomial number of prime implicates, all of which can be enumerated efficiently. This is so, e.g., for bijunctive knowledge bases, which allows us to derive most results in Section 11. However, this is not the only case, as the affine case shows (Section 12).

7 General case

In this section we only strengthen Eiter and Gottlob's statements [20] a little and adapt their proofs to our framework.

Proposition 41 (adapted from [20]) L-ABD($\mathcal{C}_{CNF}$,CNFS) *is in* Σ_2^{P}.

PROOF. Guess an explanation $E \subseteq H$ and check that $KB \wedge \bigwedge E$ is satisfiable. This verification is in NP by Theorem 32. Now, check that $KB \wedge \bigwedge E \models M$. This verification is in coNP by Theorem 33. Hence, the problem is in $\mathrm{NP}^{\mathrm{NP} \cup \mathrm{coNP}} = \Sigma_2^{\mathrm{P}}$. □

Proposition 42 (adapted from [20]) *Let* Γ *be a constraint language satisfying* $\langle\Gamma\rangle = BR$. *Then* P-ABD($\Gamma$,POSLITS) *and* N-ABD($\Gamma$,POSLITS) *are* Σ_2^{P}*-hard.*

PROOF. Eiter and Gottlob [20] show that P-ABD($\mathcal{C}_{CNF}$,POSLITS) is Σ_2^{P}-complete. Let Γ_3 be the constraint language containing all ternary relations that are the set of models of exactly one clause. It is well-known that every CNF is logically equivalent to a 3CNF formula with existentially quantified auxiliary variables. Reasoning as for Lemma 22, we get that P-ABD($\mathcal{C}_{CNF}$,POS-LITS) $\equiv_{\mathrm{P}}$P-ABD(Γ_3,POSLITS), and thus P-ABD(Γ_3,POSLITS) is Σ_2^{P}-complete. Now, since $\langle\Gamma_3\rangle = BR$, we get that P-ABD($\Gamma$,POSLITS) is Σ_2^{P}-complete.

The claim for N-ABD(Γ,POSLITS) follows since any positive hypothesis h can be changed to a negative one $\neg h'$, where h' is a fresh variable, up to adding $(h \vee h') \wedge (\neg h \vee \neg h')$, i.e., $h \leftrightarrow \neg h'$, to KB. □

Note that, in particular, the hardness result in the preceding proposition holds for *any* finite constraint language Γ such that $\langle\Gamma\rangle = BR$. We want to emphasize that, unless explicitly stated otherwise, all the hardness results in the paper for constraint languages Γ hold for any finite constraint language Γ' such that $\langle\Gamma'\rangle = \langle\Gamma\rangle$. This is important for us since we can only use Lemma 22 to derive new hardness results if the original (hard) abduction problem is defined over a finite constraint language.

8 0-valid and 1-valid languages

The "easy" abduction problems which we exhibit in this section are of a particular type. Indeed, for them the search space can be reduced to only one candidate explanation. The reasoning is similar to that for the definite Horn case (see, e.g., [20, Corollary 5.4]).

Lemma 43 *Let $P = (V, H, M, KB)$ be an instance of any abduction problem. If $KB \wedge \bigwedge H$ is satisfiable, then P has an explanation if and only if $KB \wedge \bigwedge H \models M$.*

PROOF. Obviously, if $KB \wedge \bigwedge H$ is satisfiable and $KB \wedge \bigwedge H \models M$, then $E = H$ is an explanation. Conversely, assume $KB \wedge \bigwedge H \not\models M$. Then there is a model μ of $KB \wedge \bigwedge H$ such that $\mu \not\models M$; then, for any $E \subseteq H$, $\mu \models KB \wedge \bigwedge E$ and $\mu \not\models M$, hence $KB \wedge \bigwedge E \not\models M$ and hence, E is not an explanation. □

As a direct consequence of Lemma 43, Theorems 32 and 33, we have the following results. The algorithm simply consists of deciding whether $KB \wedge \bigwedge H$ entails M.

Proposition 44 (1-valid) P-ABD($\mathcal{C}_{1v}$,CNFS) *is in* coNP*, and* P-ABD($\mathcal{C}_{1v-Horn}$,CNFS), P-ABD($\mathcal{E}_{1v-aff}$,CNFS) *are in* P.

We now give two new results, which give some upper and lower bounds, respectively, for the complexity of P-ABD and N-ABD.

Proposition 45 *The problem* N-ABD($\mathcal{C}_{0v}$,POSCNFS) *is trivial, in the sense that an instance (V, H, M, KB) has an explanation if and only if M is empty.*

PROOF. If M is empty, then it is tautological, thus KB entails M. It follows that there is an explanation if and only if KB is satisfiable, which is necessarily the case since it is 0-valid.

Now if M is nonempty, let μ_0 be the assignment of 0 to every variable in V. Since KB is 0-valid and H is a set of negative literals, we have $\mu_0 \models KB \wedge \bigwedge E$ for any $E \subseteq H$. But since M is positive but not tautological, we also have $\mu_0 \not\models M$. It follows that for all $E \subseteq H$ we have $KB \wedge \bigwedge E \not\models M$ and hence, no $E \subseteq H$ can be an explanation. □

Proposition 46 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = II_0$. Then* P-ABD($\Gamma$,POSLITS) *is Σ_2^{P}-hard. Similarly, if Γ is a constraint language satisfying $\langle\Gamma\rangle = II_1$, then* N-ABD($\Gamma$,POSLITS) *is Σ_2^{P}-hard.*

PROOF. Let Γ be a constraint language such that $\langle\Gamma\rangle = II_0$. By Post's lattice and Proposition 8, $\langle\Gamma \cup \{\mathrm{T}\}\rangle = BR$ and thus, by Proposition 42 P-ABD($\Gamma \cup \{\mathrm{T}\}$,POSLITS) is Σ_2^{P}-complete. We give a reduction of this latter problem to P-ABD(Γ,POSLITS).

To this aim, let $P' = (V', H', m', KB')$ be an instance of P-ABD($\Gamma\cup\{\mathrm{T}\}$,POSLITS). Write $KB' = KB_\Gamma \wedge \bigwedge_{x\in V_\mathrm{T}} \mathrm{T}(x)$, where KB_Γ is a conjunction of constraints over Γ and V_T is a set of variables. We assume $V_\mathrm{T} \neq \emptyset$ without loss of generality. Then we define KB to be any conjunction of constraints (possibly with existentially quantified auxiliary variables) over Γ and logically equivalent to $KB_\Gamma \wedge \bigwedge_{x\in V_\mathrm{T}}(\neg m' \vee x)$; such a formula exists because $\langle\Gamma\rangle = II_0$ is the set of all 0-valid relations and $(\neg m' \vee x)$ is 0-valid. We also define $H = H' \cup V_\mathrm{T}$, and $P = (V', H, m', KB)$. Then, clearly P has an explanation if and only if P' has one, since KB is logically equivalent to $KB_\Gamma \wedge (m' \rightarrow \bigwedge_{x\in V_\mathrm{T}} \mathrm{T}(x))$ (the reasoning is similar to that in [12, Lemma 19]). □

9 Complementive languages

We give two new lower bounds.

Proposition 47 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IN_2$. Then* P-ABD($\Gamma$,POSLITS) *and* N-ABD($\Gamma$,POSLITS) *are Σ_2^{P}-hard.*

PROOF. We know from Theorem 18 that V-ABD(Γ,POSLITS) is Σ_2^{P}-complete. Now V-ABD(Γ,POSLITS) $\leq_{\mathrm{P}}$L-ABD(Γ,LITS) (Lemmata 19 and 20), and since $\mathrm{R}_{\neq} \in IN_2$, we have L-ABD($\Gamma$,LITS) $\leq_{\mathrm{P}}$P-ABD(Γ,POSLITS), N-ABD(Γ,POSLITS) (Lemma 27), which concludes the proof. □

Proposition 48 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IN$. Then* P-ABD($\Gamma$,POSLITS) *is* coNP*-hard.*

PROOF. We know from Post's lattice and Proposition 8 that $\langle\Gamma \cup \{\mathrm{T}\}\rangle = II_1$. Thus it follows from Theorem 34 that DEDUCTION($\Gamma \cup \{\mathrm{T}\}$,POSLITS) is coNP-hard. We give a reduction of this latter problem to P-ABD(Γ,POSLITS).

Let (V, KB_T, q) be an instance of DEDUCTION($\Gamma \cup \{\mathrm{T}\}$,POSLITS), where $q \in V$, and write $KB_\mathrm{T} = KB \wedge \bigwedge_{x\in V_\mathrm{T}} \mathrm{T}(x)$, where KB is a knowledge base over Γ. Now define an instance P of P-ABD(Γ,POSLITS) by $P = (V, H = V_\mathrm{T}, q, KB)$. Since KB is 1-valid we have that $KB \wedge \bigwedge H$ is satisfiable, and thus, by Lemma 43, P has an explanation if and only if $KB \wedge \bigwedge H$ entails q, i.e., if and only if KB_T entails q. □

10 Horn and dual Horn languages

In this section we build over a number of results given in the literature for Horn knowledge bases, mainly by Selman and Levesque [48], Eiter, Gottlob, and Makino [20,23], and Khardon and Roth [32].

Lemma 49 (see [32] and [50, Lemma 1]) *Let $P = (V, H, M, KB)$ be an instance of* $\textsc{Abd}(\mathcal{C}_{Horn})$*, where* $M \in \textsc{PosLits} \cup \textsc{PosClauses} \cup \textsc{PosTerms} \cup \textsc{PosCNFs}$*. Then P has an explanation if and only if it has a positive one.*

Corollary 50 *Let Γ be a Horn language, and let $\mathcal{M}$ be* $\textsc{PosLits}$*,* $\textsc{PosClauses}$*,* $\textsc{PosTerms}$*, or* $\textsc{PosCNFs}$*. Then* $\text{L-}\textsc{Abd}(\Gamma,\mathcal{M}) \equiv_{\mathrm{P}} \text{V-}\textsc{Abd}(\Gamma,\mathcal{M}) \equiv_{\mathrm{P}} \text{P-}\textsc{Abd}(\Gamma,\mathcal{M})$.

Proposition 51 (adapted from [20, Corollary 5.4]) $\text{L-}\textsc{Abd}(\mathcal{C}_{1v-Horn},\textsc{PosCNFs})$ *is in* P.

PROOF. Let $P = (V, H, M, KB)$ be an instance. From Lemma 49 it follows that P has an explanation if and only if $P' = (V, H \cap V, M, KB)$ has one. Now KB is 1-valid, thus $KB \wedge \bigwedge(H \cap V)$ is satisfiable. Thus, by Lemma 43 P' has an explanation if and only if $KB \wedge \bigwedge(H \cap V)$ entails M, which can be decided in polynomial time since KB is Horn (Theorem 33). □

The following result is new and gives quite a broad class of tractable abduction problems. Observe that by duality, it also shows that it is tractable to decide whether a CNF has a negative explanation with respect to a Horn knowledge base.

Proposition 52 $\text{P-}\textsc{Abd}(\mathcal{C}_{dHorn},\textsc{CNFs})$ *is in* P.

PROOF. Let $P = (V, H, M, KB)$ be an instance. We assume without loss of generality that KB is satisfiable (see the end of Section 3.1). Let H' be the set of all (positive) literals $h \in H$ such that $KB \wedge h$ is satisfiable; H' can be computed efficiently by testing every $h \in H$ since KB is dual Horn. Then P has an explanation if and only if the instance (V, H', M, KB) has one, since every candidate explanation containing h for some literal $h \in H \setminus H'$ would be inconsistent with KB. Now the set of models of a dual Horn knowledge base is closed under componentwise logical or (this is dual to the well-known closure of Horn theories under logical and, see, e.g., [46]). Hence, since $KB \wedge h$ is satisfiable for every $h \in H'$, $KB \wedge \bigwedge H'$ is satisfiable. We now conclude from Lemma 43 that P has an explanation if and only if $KB \wedge \bigwedge H'$ entails M, which can be decided in polynomial time since KB is dual Horn (Theorem 33). □

We finally give two hardness results. The proof of the first one follows directly from [12] (V-Abd case, as reported in Theorem 18) together with Corollary 50.

Proposition 53 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IE_0$. Then* P-Abd($\Gamma$,PosLits) *is* NP*-hard.*

Proposition 54 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IE_0$. Then* P-Abd($\Gamma$,NegLits) *and* V-Abd($\Gamma$,NegLits) *are* NP*-hard.*

PROOF. By Proposition 53, P-Abd(Γ,PosLits) is NP-hard. Now the clause $(\neg x \vee \neg y)$ is in IE_0, thus Lemma 29 gives a reduction from P-Abd(Γ,PosLits) to P-Abd(Γ,NegLits) with choosing a negative literal for ℓ. The proof is similar for V-Abd, using Theorem 18 for hardness of V-Abd(Γ,PosLits). $\square$

11 Bijunctive and IHS-B languages

Bijunctive and IHS-B restrictions share an important property, summarized in the next lemma.

Lemma 55 *Let $\mathcal{C}$ be any of $\mathcal{C}_{bij}$, $\mathcal{C}_{impl}$, $\mathcal{C}_{IHSB+/k}$, or $\mathcal{C}_{IHSB-/k}$ for some k. Then every prime implicate of a knowledge base KB over $\mathcal{C}$ is in $\mathcal{C}$, and the set of all these prime implicates can be computed in time polynomial in the size of KB. In particular, there are only a polynomial number of them.*

PROOF. It is easily seen that all the prime implicates are in $\mathcal{C}$. Indeed, all of them can be generated by resolution, and as is easily seen from the forms of the clauses, resolution preserves each class in the statement. Thus, starting from a formula over $\mathcal{C}$, only clauses in $\mathcal{C}$ can be generated.

Clearly, the number of prime implicates of a theory over $\mathcal{C}_{bij}$, $\mathcal{C}_{impl}$, $\mathcal{C}_{IHSB+/k}$, or $\mathcal{C}_{IHSB-/k}$ is polynomial in the size of the theory, since the size of clauses over $\mathcal{C}$ is bounded by 2 or k in all cases. All can be generated in polynomial time since one can simply generate all clauses of size 2 (resp. k) and for each one, test whether it is entailed by KB and none of its proper subclauses is, in polynomial time in all cases (Theorem 33). $\square$

Remark 56 *The fact that the language is fixed, and thus that k is fixed for languages $\mathcal{C}_{IHSB-/k}$ and $\mathcal{C}_{IHSB+/k}$, is crucial in the proof of Lemma 55. Indeed, the statement does not hold for infinite languages $\mathcal{C}_{IHSB-}$ and $\mathcal{C}_{IHSB+}$.*

We first give easy consequences of Lemmata 37 and 55, which generalize folklore results based on prime implicate generation (see in particular Marquis' survey [37]. The algorithm consists of generating the prime implicates of KB over $Vars(H) \cup Vars(M)$ until one as in Lemma 37 is found or all have been tested.

Proposition 57 *Let* $\mathcal{C}$ *be any of* $\mathcal{C}_{bij}$, $\mathcal{C}_{neg,=}$, *or* $\mathcal{C}_{IHSB-/k}$ *for some* $k \in \mathbb{N}$. *Then* L-ABD($\mathcal{C}$,CLAUSES) *is in* P.

PROOF. The only case not handled by Lemma 55 is $\mathcal{C}_{neg,=}$, because of the equality relation. We show that if a knowledge base KB over $\mathcal{C}_{neg,=}$ contains no equality constraint, then it has a polynomial number of prime implicates, all of which can be generated efficiently. We then conclude with Lemma 21.

Indeed, since clauses in KB are either unit positive or negative, it can be seen that once resolution has been applied to each pair of clauses consisting of a positive and a negative one, it cannot be applied any more. Since there can be at most one positive clause per variable, all prime implicates can be generated in polynomial time. □

Proposition 58 L-ABD($\mathcal{C}_{IHSB-}$,POSCLAUSES) *is in* P.

PROOF. By a reasoning similar to that in the proof of Lemma 55 we have that every prime implicate of a IHS-$B-$ theory is IHS-$B-$. Now since the only IHS-$B-$ clauses which contain at least one positive literal are unary and implicative ones, from Lemma 37 we get that the only minimal candidate explanations of an instance of L-ABD($\mathcal{C}_{IHSB-}$,POSCLAUSES) are the empty one and those restricted to only one (positive) literal, all of which can be tested efficiently. □

We now give two new tractability results, for which some more work is needed because of conjunctive manifestations.

Proposition 59 L-ABD($\mathcal{C}_{impl}$,POSCNFS) *is in* P.

PROOF. First observe that since KB is bijunctive, one can decide in polynomial time whether $\emptyset$ is an explanation, by deciding whether KB is satisfiable and entails M. Thus we assume hereafter that $\emptyset$ is not an explanation.

Let $P = (V, H, M, KB)$ be an instance of the problem, and assume first that M consists of a single (positive) clause C. In this case, since the only prime implicates of a knowledge base over $\mathcal{C}_{impl}$ are in $\mathcal{C}_{impl}$ (Lemma 55), we conclude that a set $E \subseteq H$ is an explanation for P if and only if $KB \wedge \bigwedge E$ is satisfiable and there is a literal $\ell \in E$ such that $KB \wedge (\ell)$ entails C (or, as a subcase, $E = \emptyset$ is an explanation); moreover, such an ℓ has to be a positive literal. Thus the set E_C of all such ℓ's can be computed in polynomial time by testing the $|H|$ candidates.

Now consider the case when M is a CNF of the form $(C_1 \wedge \cdots \wedge C_p)$. Then by Lemma 28 and the reasoning above, a set $E \subseteq H$ is an explanation for P if and only if $KB \wedge \bigwedge E$ is satisfiable and for all $i \in \{1, \ldots, p\}$, $\emptyset$ is an explanation for C_i or there is some $h_i \in E$ such that $h_i \in E_{C_i}$. As is easily seen, this is true if and only if the formula $KB \wedge \bigwedge_{i=1, KB \not\models C_i}^{p} (\bigvee_{h \in E_{C_i}} h)$ is satisfiable. Since for all $i \in \{1, \ldots, p\}$, E_{C_i} contains only positive literals, this formula is IHSB+ and thus, it can be decided in polynomial time whether it is satisfiable (Theorem 32, since $\mathcal{C}_{IHSB+} \subseteq \mathcal{C}_{dHorn}$). □

Proposition 60 L-Abd($\mathcal{C}_{neg,=}$,PosCNFs) *is in* P.

PROOF. Let (V, H, M, KB) be an instance. We first invoke Lemma 21 for assuming without loss of generality that KB contains no equality constraint. Now, reasoning as in Lemma 57 we get that the only prime implicates of KB which contain at least one positive literal are unit clauses. Thus, by Lemma 37 there is an explanation for a positive clause C of M if and only if $\emptyset$ or $\{m\}$, for some m in C, is an explanation. It then follows from Lemma 28 that there is an explanation for M if and only if H contains at least one variable in each clause of M which is not entailed by KB alone, which can be decided efficiently. □

11.2 Bijunctive and IHS-B languages: Lower bounds

We will mainly use the following lemma, which provides a class of reductions from satisfiability problems to abduction problems. The intuition behind the reduction is that we reduce the test for satisfiability of a set of clauses in a formula to a test for explainability of the satisfaction of these clauses, where satisfaction of a clause is explainable by any of the literals in this clause. The clauses which are not transformed by the reduction serve as constraints over the possible explanations. Importantly, this is exactly the intuition behind our characterization of tractable vs. NP-complete abduction problems, as we shall see in Section 15.

Lemma 61 *Let $\varphi = \bigwedge_{i \in I} C_i$ be a CNF formula, where for all i, $C_i = \bigvee_{j \in J_i} \ell_{i,j}$ and every $\ell_{i,j}$ is a literal. Let $I' \subseteq I$ be a set of indices. For all $i \in I'$ let x_i be a new variable ($x_i \notin Vars(\varphi)$), and let s_i be a literal formed upon x_i (intuitively, "clause C_i is satisfied"). Finally, define*

- $KB = \bigwedge_{i \in I \setminus I'} C_i \wedge \bigwedge_{i \in I'} \bigwedge_{j \in J_i} (\overline{\ell_{i,j}} \vee s_i)$,
- $V = Vars(\varphi) \cup \{x_i \mid i \in I\}$,
- $H = \{\ell_{i,j} \mid i \in I, j \in J_i\} \cup \{\overline{\ell_{i,j}} \mid i \in I, j \in J_i\}$, *and*
- $M = \bigwedge_{i \in I} s_i$.

Then φ is satisfiable if and only if the abduction problem $P = (V, H, M, KB)$ has an explanation. The same result holds with $H = \{\ell_{i,j} \mid i \in I, j \in J_i\}$.

PROOF. Assume first that φ has a model μ, and define E to be the set of all literals in H which are satisfied by μ. Then since μ satisfies C_i for all $i \in I$, it satisfies C_i for all $i \in I \setminus I'$; now define the assignment μ' to V to agree with μ over $Vars(\varphi)$ and to satisfy every s_i. Then μ' satisfies C_i for every $i \in I \setminus I'$ and $(\overline{\ell_{i,j}} \vee s_i)$ for every $i \in I, j \in J_i$, thus it satisfies KB. Moreover, clearly μ' satisfies $\bigwedge E$. Finally, $KB \wedge \bigwedge E$ is satisfiable.

We now show that $KB \wedge \bigwedge E$ entails M. Assume to the contrary that $KB \wedge \bigwedge E \wedge (\bigvee_{i \in I} \overline{s_i})$ is satisfiable. Then there is an $i \in I$ such that $KB \wedge \bigwedge E \wedge \overline{s_i}$ is satisfiable. Write μ' for one of its models. Then μ' satisfies $\bigwedge E$, thus it agrees with μ over $Vars(H)$; then since μ satisfies φ, μ' satisfies $\ell_{i,j}$ for at least one $j \in J_i$. Thus μ' satisfies $\ell_{i,j} \wedge \overline{s_i}$, which contradicts the fact that it satisfies KB. Finally, $KB \wedge \bigwedge E$ entails M, and E is an explanation for P.

Conversely, assume that P has an explanation E. Then there is a model μ of $KB \wedge \bigwedge E$, and we show that $\mu_{|Vars(\varphi)}$ is a model of φ. First, for all $i \in I \setminus I'$, μ satisfies C_i since C_i is in KB. Now assume, towards a contradiction, that there is an $i \in I'$ such that for all $j \in J_i$, μ satisfies $\overline{\ell_{i,j}}$. Define the assignment μ' to agree with μ over $V \setminus \{x_i\}$ and to satisfy $\overline{s_i}$. Then μ' satisfies $KB \wedge \bigwedge E$ but does not satisfy M, which contradicts the fact that E is an explanation for P. It follows that for all $i \in I'$, μ satisfies $\ell_{i,j}$ for at least one $j \in J_i$, and thus it satisfies C_i. Finally, μ satisfies φ, as desired. $\Box$

Based on this general reduction, we are able to give new hardness results for several abduction problems with bijunctive and IHS-B knowledge bases.

Proposition 62 *Let Γ be a constraint language. Then,*

- *if $\langle\Gamma\rangle = IS_{11}^2$,* V-ABD($\Gamma$,POSTERMS) *and* P-ABD($\Gamma$,POSTERMS) *are* NP-*hard;*
- *if $\langle\Gamma\rangle = IM$,* V-ABD($\Gamma$,TERMS) *is* NP-*hard;*

- *if* $\langle\Gamma\rangle = IS_0^2$, V-Abd($\Gamma$,PosTerms) *and* N-Abd($\Gamma$,PosTerms) *are* NP-*hard.*

PROOF. We first prove the case $\langle\Gamma\rangle = IS_{11}^2$. From Theorem 32 it follows that Sat($\{(x_1 \vee x_2 \vee x_3), (\neg x_1 \vee \neg x_2)\}$) is NP-complete. Now Lemma 61 gives a reduction from this problem to V-Abd(Γ,PosTerms) or to P-Abd(Γ,PosTerms). Indeed, let $\varphi = \bigwedge_{i \in I_p}(x_{i,1} \vee x_{i,2} \vee x_{i,3}) \wedge \bigwedge_{i \in I_n}(\neg x_{i,1} \vee \neg x_{i,2})$, where I_p, I_n are two disjoint sets of indices. Then by Lemma 61 we have the desired reduction by choosing $I' = I_p$ and for all $i \in I'$, $s_i = x_i$.

The proof is similar for cases $\langle\Gamma\rangle = IM$ and $\langle\Gamma\rangle = IS_0^2$, up to considering respectively,

- Sat($\{(x_1 \vee x_2 \vee x_3), (\neg x_1 \vee \neg x_2)\}$), $I' = I_p \cup I_n$, for all $i \in I_p$, $s_i = x_i$, and for all $i \in I_n$, $s_i = \neg x_i$, and
- Sat($\{(x_1 \vee x_2), (\neg x_1 \vee \neg x_2 \vee \neg x_3)\}$), $I' = I_n$, and for all $i \in I'$, $s_i = x_i$. □

The next proposition is a special case. Indeed, observe that Γ is necessarily infinite, since any finite language included in IS_{11} must have bounded width and thus, be included in IS_{11}^k for some $k \in \mathbb{N}$ (yielding $\langle\Gamma\rangle \subseteq IS_{11}^k \subsetneq IS_{11}$). This special case is discussed in Section 13.2.

Proposition 63 *Let Γ be an (infinite) language satisfying $\langle\Gamma\rangle = IS_{11}$. Then* V-Abd($\Gamma$,NegLits) *and* P-Abd($\Gamma$,NegLits) *are* NP-*hard.*

PROOF. Proposition 62 shows that V-Abd(Γ,PosTerms) is NP-hard if $\langle\Gamma\rangle = IS_{11}^2$. Now Lemma 29 gives a reduction from this problem to V-Abd(Γ',NegLits), where $\langle\Gamma'\rangle = IS_{11}$, by choosing a negative literal for ℓ. The proof is similar for P-Abd(Γ,PosTerms). □

12 Affine languages

The main tool which we will use with affine formulas is projection. This will be done through the following lemma.

Lemma 64 ([51]) *Let KB be an affine formula, and let $V' \subseteq Vars(KB)$. Then there is a projection of KB onto V' which is affine, and such a projection can be computed in polynomial time.*

Interestingly, contrary to the case of bijunctive and bounded IHS-B knowledge bases, tractability of projection here is not a consequence of a polynomial number of prime implicates. Indeed, even in the case of a single linear equation of the form ($x_1 \oplus \cdots \oplus x_n = 0$), an affine formula may have an exponential number of prime implicates (2^{n-1} in the example, namely all clauses over exactly $x_1, \ldots, x_n$ with an odd number of negative literals).

12.1 Affine languages: Upper bounds

We first restate Nordh and Zanuttini's result [40] about affine formulas of width 2. Indeed, they state it for finite languages, but their proof obviously holds for the corresponding infinite language as well.

Proposition 65 (adapted from [40]) L-ABD($\mathcal{E}_{\mathit{aff}/2}$,TERMS) *is in* P.

We now give new tractability results. So as to use projection consistently with its definition, observe that in the affine case we can assume $\mathit{Vars}(H) \cup \mathit{Vars}(M) \subseteq \mathit{Vars}(KB)$ without loss of generality. Indeed, for any variable $x \in (\mathit{Vars}(H) \cup \mathit{Vars}(M)) \setminus \mathit{Vars}(KB)$, a fresh variable new_x can be introduced and $x \oplus \mathit{new}_x = 0$ added to KB without changing the set of explanations.

Proposition 66 L-ABD($\mathcal{E}_{\mathit{aff}}$,CLAUSES) *is in* P.

PROOF. Consider an instance (V, H, M, KB). Note that since $(x \neq y) \equiv (x \oplus y = 1) \in \mathcal{E}_{\mathit{aff}}$ we can use Lemma 27 to reduce the instance to an equivalent one (V', H', M', KB') where M' is a positive clause and H' is a set of negative literals. Now, in order to eliminate all variables that are neither in H' nor in M', project KB' onto $V'' = \mathit{Vars}(H') \cup \mathit{Vars}(M')$, getting a formula KB'' over the set of variables V''.

Now since $(x = y) \equiv (x \oplus y = 0) \in \mathcal{E}_{\mathit{aff}}$ we can assume $\mathit{Vars}(H') \cap \mathit{Vars}(M') = \emptyset$ without loss of generality, since any variable x in the intersection could be duplicated into x_H and x_M up to adding $(x_H = x_M)$ to KB''.

We now have an instance $P = (V'', H', M', KB'')$ where KB'' is a set of linear equations, H' is the set of negative literals $\{\neg x \mid x \in V'' \setminus \mathit{Vars}(M')\}$, and M' is a positive clause. We can then use exactly the same reduction as in [40, Proposition 11] to show that P has an explanation if and only if the negative term $\neg M'$ does not follow from KB'' when circumscribing all variables. Since this problem is in P if $\neg M'$ is a single literal [18, Theorem 7], and the case of a term is easily seen to be polynomial-time reducible to it, we have the result. □

The next proposition will use the notion of a *full* explanation. Given an instance $P = (V, H, M, KB)$ of any abduction problem, an explanation E for P is said to be *full* if $Vars(E) = Vars(H)$. What we will use is the fact that an instance of V-ABD, for any restriction on the manifestation, has an explanation if and only if it has a full one. Indeed, given a nonfull explanation E, since $KB \wedge \bigwedge E$ has at least one model μ, it is easily seen that the set E' defined to be the set of all literals over H assigned true by μ is a full explanation for P.

Proposition 67 V-ABD($\mathcal{E}_{aff}$,TERMS) *is in* P.

PROOF. Let $P = (V, H, M, KB)$ be an instance, and write V_H for $Vars(H)$. We first consider the case of a positive literal as a manifestation, i.e., $M = m$ for some $m \in V$ (the case of a negative literal is dual). Assume $m \notin H$, which is without loss of generality since otherwise P has an explanation if and only if $KB \wedge m$ is satisfiable, which can be decided efficiently. Write $KB_{H \cup \{m\}}$ for an affine projection of KB onto $V_H \cup \{m\}$. By Lemma 64 such a knowledge base can be computed in polynomial time. Now define $P_{H \cup \{m\}} = (V_H \cup \{m\}, H, m, KB_{H \cup \{m\}})$. By Lemma 40, $P_{H \cup \{m\}}$ has an explanation if and only if P has one. Moreover, obviously, if m does not occur in $KB_{H \cup \{m\}}$ then $P_{H \cup \{m\}}$ has no explanation. Otherwise, let $Eqn_m = (m \oplus \bigoplus_X x = a)$ be an equation of $KB_{H \cup \{m\}}$ containing m.

Now let $KB_{H,m=1}$ be an affine projection of $KB_{H \cup \{m\}} \wedge m$ onto V_H. We claim that the full explanations of $P_{H \cup \{m\}}$ are in bijection with the models of $KB_{H,m=1}$.

Indeed, let E be a full explanation for $P_{H \cup \{m\}}$. Then by definition of an explanation, there is a model μ of $KB_{H \cup \{m\}} \wedge \bigwedge E \wedge m$; thus $\mu_{|V_H}$ satisfies $KB_{H,m=1} \wedge \bigwedge E$, and thus $KB_{H,m=1}$. Conversely, let μ be a model of $KB_{H,m=1}$, and write E for the set of all literals over V_H and satisfied by μ; we show that E is a full explanation of $P_{H \cup \{m\}}$. First, $KB_{H,m=1} \wedge \bigwedge E$ is satisfiable, thus $KB_{H \cup \{m\}} \wedge m \wedge \bigwedge E$ is satisfiable, and thus $KB_{H \cup \{m\}} \wedge \bigwedge E$ is satisfiable. We are thus left with proving $KB_{H \cup \{m\}} \wedge \bigwedge E \models m$. Assume towards a contradiction that there is a model μ' of $KB_{H \cup \{m\}} \wedge \bigwedge E \wedge \neg m$. In particular, μ' satisfies $\neg m$ and agrees with μ over $Vars(E) = V_H$. Moreover, since μ is a model of $KB_{H,m=1}$, it satisfies the equation $\bigoplus_X x = a \oplus 1$ (i.e., Eqn_m with $m = 1$); it follows that μ' satisfies $\bigoplus_X x = a \oplus 1$; since it also satisfies $\neg m$, it satisfies $m \oplus \bigoplus_X x = a \oplus 1$. Thus it does not satisfy equation Eqn_m, which contradicts the fact that it satisfies $KB_{H \cup \{m\}}$.

We conclude that in the case of a single literal m as the manifestation, the full explanations of $P_{H \cup \{m\}}$ are exactly the models of $KB_{H,m=1}$. It follows from Lemma 28 that the full explanations in the case of a manifestation $M = m_1 \wedge \cdots \wedge m_k$ are exactly the models of $KB_{H,m_1=1} \wedge \cdots \wedge KB_{H,m_k=1}$. Since every $KB_{H,m_i=1}$ is an affine formula and can be computed in polynomial time, we finally get a polynomial algorithm. $\square$

12.2 Affine languages: Lower bounds

We finally give hardness proofs for affine languages, all of which are new. To that aim, we use original reductions from satisfiability problems, inspired by the clausal case (Lemma 61).

Proposition 68 *Let* Γ *be a constraint language satisfying* $\langle\Gamma\rangle = IL$. *Then* L-ABD($\Gamma$,POSTERMS) *is* NP-*hard.*

PROOF. We give a reduction from SAT($\{(x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0), (x_1), (\neg x_1 \vee \neg x_2)\}$), which is NP-complete by Theorem 32. Let $\varphi = \varphi_0 \wedge \bigwedge_{i\in I}(\neg x_{i,1} \vee \neg x_{i_2}) \wedge \bigwedge_{k\in K}(x_k = 1)$, where φ_0 contains only equations of the form $(x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0)$. For all $i \in I$ let $s_i \notin Vars(\varphi)$ be a fresh variable (intuitively meaning that clause i is satisfied), and for all $i \in I$ let $p_{i,1}, n_{i,1}, p_{i,2}, n_{i,2} \notin Vars(\varphi)$ be four fresh variables ("p" stands for "positive" and "n" for "negative"). Write H_p for the set of all $p_{i,j}$'s and H_n for that of all $n_{i,j}$'s. We define an instance $P = (V, H, M, KB)$ of L-ABD(Γ,POSTERMS) by

- $V = Vars(\varphi) \cup \{s_i \mid i \in I\} \cup H_p \cup H_n$,
- $H = N(Vars(\varphi)) \cup \{x_k \mid k \in K\} \cup H_p \cup N(H_n)$,
- $M = \{s_i \mid i \in I\} \cup \{x_k \mid k \in K\}$, and
- $KB = \varphi_0 \wedge \bigwedge_{i\in I}(x_{i,1} \oplus p_{i,1} \oplus n_{i,1} \oplus s_i = 0) \wedge \bigwedge_{i\in I}(x_{i,2} \oplus p_{i,2} \oplus n_{i,2} \oplus s_i = 0)$.

The intuition is that the new equations play the role of "implications" $\neg x_{i,1} \to s_i$ and $\neg x_{i,2} \to s_i$.

We claim that P has an explanation if and only if φ is satisfiable. First, if μ is a model of φ, then clearly $\{\neg x \mid x \in Vars(\varphi) \text{ and } \mu(x) = 0\} \cup \{x_k \mid k \in K\} \cup \{p_{i,j}, \neg n_{i,j} \mid i \in I \text{ and } \mu(x_{i,j}) = 0\}$ is an explanation for P. Conversely, assume E is an explanation for P. Then since for all $i \in I$, $s_i, p_{i,1}, p_{i,2}, n_{i,1}, n_{i,2}$ only occur in equations $(x_{i,j} \oplus p_{i,j} \oplus n_{i,j} \oplus s_i = 0)$, E has to contain $p_{i,j}$ or $\neg p_{i,j}$ and $n_{i,j}$ or $\neg n_{i,j}$ for at least one $j \in \{1,2\}$. Indeed, otherwise flipping the values of s_i and some $p_{i,j}$'s or $n_{i,j}$'s in a model of $KB \wedge \bigwedge E \wedge \bigwedge M$ would yield a model of $KB \wedge \bigwedge E \wedge \neg s_i$, contradicting the fact that E is an explanation for P. Thus for all $i \in I$ there is a $j \in \{1,2\}$ such that E contains $p_{i,j}$ and $\neg n_{i,j}$, since $\neg p_{i,j}, n_{i,j} \notin H$. Now by definition of an explanation, there is a model of $KB \wedge \bigwedge E \wedge \bigwedge M$, and it follows from the reasoning above that this model satisfies $\neg x_{i,1}$ or $\neg x_{i,2}$ for all $i \in I$, since otherwise one of the equations $(x_{i,j} \oplus p_{i,j} \oplus n_{i,j} \oplus s_i = 0)$ would not be satisfied. Finally, it satisfies x_k for all

$k \in K$ because these literals are in M. Thus, defining μ to be the projection of this model onto $Vars(\varphi)$, we get that μ is a model of φ, as desired. $\square$

Proposition 69 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IL_1$. Then* N-ABD($\Gamma$,POSTERMS) *is* NP*-hard. Similarly, if Γ is a constraint language satisfying $\langle\Gamma\rangle = IL_0$, then* P-ABD($\Gamma$,POSTERMS) *is* NP*-hard.*

PROOF. As regards IL_1, the proof is similar to that of Proposition 68, except that we start from SAT($\{(x_1 \oplus x_2 \oplus x_3 = 1), (\neg x_1 \vee \neg x_2)\}$), we introduce only one fresh variable $n_{i,j}$ per occurrence of variable in a negative clause, we build equation $(x_{i,j} \oplus n_{i,j} \oplus s_i = 1)$ instead of $(x_{i,j} \oplus p_{i,j} \oplus n_{i,j} \oplus s_i = 0)$, and we only add $\neg n_{i,j}$ to H.

The proof is dual for IL_0, starting from SAT($\{(x_1 \oplus x_2 \oplus x_3 = 0), (x_1 \vee x_2)\}$) and building $(x_{i,j} \oplus p_{i,j} \oplus p'_{i,j} \oplus s_i = 0)$ with $p_{i,j}, p'_{i,j} \in H$. $\square$

Proposition 70 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IL_3$. Then* P-ABD($\Gamma$,POSTERMS) *and* N-ABD($\Gamma$,POSTERMS) *are* NP*-hard.*

PROOF. Since $IL \subset IL_3$, by Proposition 68 L-ABD(Γ,POSTERMS) is NP-hard. Now we know L-ABD(Γ,POSTERMS)$\leq_{\mathrm{P}}$L-ABD(Γ,TERMS) (Lemma 19). Finally, since $\mathrm{R}_{\neq}(x, y)$ is the set of models of $x \oplus y = 1$, $\mathrm{R}_{\neq}$ is in IL_3. Thus by Lemma 27 we know that L-ABD(Γ,TERMS) $\leq_{\mathrm{P}}$P-ABD(Γ,POSTERMS), N-ABD(Γ,POSTERMS), which concludes the proof. $\square$

Proposition 71 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = ID$. Then* V-ABD($\Gamma$,POSCNFS), P-ABD($\Gamma$,POSCNFS), *and* N-ABD($\Gamma$,POSCNFS) *are* NP*-hard.*

PROOF. We consider the case of V-ABD. The other cases follow from it using Lemmata 20 and 27.

We give a reduction from the satisfiability problem for CNF formulas. Let $\varphi = \bigwedge_{i \in I} C_i$ be a CNF formula. For every variable $x \in Vars(\varphi)$ let $p_x, n_x \notin Vars(\varphi)$ be two fresh variables ("p" stands for "positive" and "n" for "negative"). Let φ' be the CNF formula obtained from φ by replacing every positive occurrence x of a variable in a clause with the positive literal p_x, and every negative occurrence $\neg x$ of a variable with the positive literal n_x. By construction, φ' is a positive CNF.

We define an instance P of V-ABD(Γ,POSCNFS) as follows:

- $KB = \bigwedge_{x \in Vars(\varphi)} (p_x \neq n_x)$,

- $V = \{p_x \mid x \in \mathit{Vars}(\varphi)\} \cup \{n_x \mid x \in \mathit{Vars}(\varphi)\}$,
- $M = \varphi'$, and
- $H = \{p_x \mid x \in \mathit{Vars}(\varphi)\} \cup \{\neg p_x \mid x \in \mathit{Vars}(\varphi)\}$.

Then it is easily seen that if φ is satisfiable with a model μ, then E defined to be $\{p_x \mid \mu(x) = 1\} \cup \{\neg p_x \mid \mu(x) = 0\}$ is an explanation for P. Conversely, if E is an explanation for P, then it is easily seen that any assignment to $\mathit{Vars}(\varphi)$ satisfying $\mu(x) = 1$ (resp. $\mu(x) = 0$) for all $p_x \in E$ (resp. $\neg p_x \in E$) is a model of φ, which concludes the proof. $\square$

Proposition 72 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IL$. Then* V-ABD($\Gamma$,POSCNFS) *is* NP-*hard.*

PROOF. We give a reduction from the satisfiability problem for CNF formulas, similar to the one in the proof of Proposition 71. Let $\varphi = \bigwedge_{i\in I} C_i$ be a CNF formula. For every variable $x \in \mathit{Vars}(\varphi)$ let $p_x, n_x \notin \mathit{Vars}(\varphi)$ be two fresh variables ("p" stands for "positive" and "n" for "negative"). Let φ' be the CNF formula obtained from φ by replacing every positive occurrence x of a variable in a clause with the positive literal p_x, and every negative occurrence $\neg x$ of a variable with the positive literal n_x. By construction, φ' is a positive CNF. Moreover, let $x_1, \ldots, x_4 \notin \mathit{Vars}(\varphi)$.

We define an instance P of V-ABD(Γ,POSCNFS) as follows:

- $\mathit{KB} = (x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0) \wedge \bigwedge_{x\in \mathit{Vars}(\varphi)}(p_x \oplus n_x \oplus x_3 \oplus x_4 = 0)$,
- $V = \{p_x \mid x \in \mathit{Vars}(\varphi)\} \cup \{n_x \mid x \in \mathit{Vars}(\varphi)\} \cup \{x_1, x_2, x_3, x_4\}$,
- $M = x_1 \wedge (x_3 \vee x_4) \wedge \varphi'$, and
- $H = \{p_x, \neg p_x \mid x \in \mathit{Vars}(\varphi)\} \cup \{x_1, \neg x_1, x_2, \neg x_2\}$.

The idea is that the manifestations $x_1 \wedge (x_3 \vee x_4)$ together with the equation $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$ in KB force every model μ of $\mathit{KB} \wedge \bigwedge E$ (for any explanation E) to satisfy $\mu(x_3) \neq \mu(x_4)$. Consequently, the equations $(p_x \oplus n_x \oplus x_3 \oplus x_4 = 0)$ force $\mu(p_x) \neq \mu(n_x)$ for any model μ of $\mathit{KB} \wedge \bigwedge E$ (and any explanation E).

It is easily seen that if φ is satisfiable with a model μ, then E defined to be $\{p_x \mid \mu(x) = 1\} \cup \{\neg p_x \mid \mu(x) = 0\} \cup \{x_1, \neg x_2\}$ is an explanation for P. Conversely, if E is an explanation for P, then $\mathit{KB} \wedge \bigwedge E$ is satisfiable and entails $x_1 \wedge (x_3 \vee x_4) \wedge \varphi'$. Now, assume that there is a model μ of $\mathit{KB} \wedge \bigwedge E$ such that $\mu(x_3) = \mu(x_4) = 1$. Note that x_3 and x_4 do not occur in H and that they occur together in every equation in KB. So, it is easy to see that there is also a model μ' of $\mathit{KB} \wedge \bigwedge E$ such that $\mu'(x_3) = \mu'(x_4) = 0$. This is a contradiction with the fact that E is an explanation, since μ' does not satisfy $(x_3 \vee x_4)$. Hence, any model μ of $\mathit{KB} \wedge \bigwedge E$ satisfy $\mu(x_3) \neq \mu(x_4)$ and consequently, $\mu(p_x) \neq \mu(n_x)$ for all $x \in \mathit{Vars}(\varphi)$. Hence, the assignment μ' to $\mathit{Vars}(\varphi)$ defined by $\mu'(x) = \mu(p_x)$ for all $x \in \mathit{Vars}(\varphi)$ is a model of φ. $\square$

For the next proposition, observe that even the empty language $\Gamma = \emptyset$ satisfies $\langle\Gamma\rangle = IBF$.

Proposition 73 *Let Γ be a constraint language satisfying $\langle\Gamma\rangle = IBF$. Then* V-ABD($\Gamma$,CNFS) *is* NP*-hard.*

PROOF. Recall than IBF only contains the equality relations. We give a reduction from the satisfiability problem for CNFs. Let φ be a CNF, we define the following instance P of V-ABD(Γ,CNFS):

- KB is the empty CNF (always satisfied),
- $V = Vars(\varphi)$,
- $M = \varphi$, and
- $H = Lits(Vars(\varphi))$.

Then it is easily seen that the (full) explanations of P correspond exactly to the models of φ. $\square$

13 Summary and complete classification

We are now in position to give a complete picture of the complexity of propositional abduction for the 48 restrictions over hypotheses and manifestations. By a "complete" picture, we mean that our results give the complexity of abduction for any constraint language and for any class of clauses or equations, as explained in Section 13.2.

13.1 Summary of results

The complete complexity picture of abduction in given in Table 3. In this table, for each restriction on hypotheses and manifestations and each complexity class, the minimal and maximal languages in this class (with respect to language inclusion) are listed. More precisely, the languages listed on the first line in each cell are the maximal languages in the complexity class, and the languages on the second line are the minimal languages hard for the class.

As an example, consider the cell for NP-complete L-ABD problems where the manifestations are expressed by POSCNFS. The first row in this cell is $\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$, and the second row is IS_0^2, IS_{11}^2, ID, IV, IL. This means that L-ABD(Γ,POSCNFS) is NP-complete for any Γ such that Γ is a subset of one of the languages listed in the first row, and one of the languages on the second row is a subset of $\langle\Gamma\rangle$. To further exemplify, consider the constraint language $\Gamma = \{R\}$, where $R = \{001, 010, 100, 111\}$ (i.e., the relation expressed by $x_1 \oplus x_2 \oplus x_3 = 1$). Then, $\Gamma \subseteq \mathcal{E}_{aff}$ and $IL \subseteq \langle\Gamma\rangle$. Hence, by Proposition 35, L-Abd(Γ,PosCNFs) is in NP, and by Proposition 72 (together with the obvious reduction from V-Abd to L-Abd) L-Abd(Γ,PosCNFs) is NP-hard.

The fact that all the results in Table 3 can be derived from the results reported in the paper (in the manner described above) has been checked by a computer program, which is available from the authors.

Also note that the results for negative restrictions on manifestations have been omitted from the table; to recover them, simply use Lemma 25.

Finally, we collapsed the rows concerning (positive) clauses and (positive) literals. Indeed, it turns out that the complexity is always the same for both types of manifestations. Section 15.4 gives an explanation for that fact.

13.2 On the completeness of the classification

In this section we motivate our claims that the classifications are complete in the sense that all constraint languages and classes of equations and clauses are covered.

We begin by noting that the upper bounds on the complexity of abduction problems in the paper, which are all given in terms of clausal and equational languages $\mathcal{C}$, also hold for any constraint language Γ such that $\Gamma \subseteq \langle\mathcal{C}\rangle$. If Γ is a finite constraint language, then this is obvious since we can use a simple lookup table to translate a Γ-formula into a CNF or system of equations over $\mathcal{C}$. For infinite constraint languages Γ the situation is slightly more involved. We make use of a result from [11], stating that we can transform (in polynomial time) any Γ-formula where the relations are given in extension into an equivalent CNF-formula/system of equations over *any* clausal/equational language $\mathcal{C}$ from Table 1 such that $\langle\Gamma\rangle \subseteq \langle\mathcal{C}\rangle$. Since all the upper bounds on the complexity of abduction in the paper are given for clausal and equational languages $\mathcal{C}$ from Table 1, we get that these upper bounds also apply to constraint languages Γ such that $\Gamma \subseteq \langle\mathcal{C}\rangle$.

All our hardness results, except for a few special cases discussed below, are proved for finite constraint languages Γ. Obviously, using Lemma 22, these hardness results implies hardness for any constraint language Γ' such that $\Gamma \subseteq \langle\Gamma'\rangle$. The exceptional constraint languages Γ for which we cannot prove hardness by first proving that some finite constraint language $\Gamma' \subseteq \langle\Gamma\rangle$ is hard and then applying Lemma 22, are:

- L-Abd(Γ,$\mathcal{M}$), where $\langle\Gamma\rangle \in \{IS_{01}, IS_{00}\}$, $\mathcal{M} \in \{$(Pos)Lits, (Pos)Clauses$\}$,

	$\mathcal{M}$	P	NP-C.	coNP-C.	Σ_2^P-C.
V-Abd	PosLits or PosClauses	$\mathcal{C}_{IHSB+/k}$, $\mathcal{C}_{pos,=}$, $\mathcal{C}_{IHSB-}$, $\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{dHorn}$, $\mathcal{C}_{Horn}$ IS_{01}, IV, IE_0		$\mathcal{C}_{CNF}$ IN
	Lits or Clauses	$\mathcal{C}_{IHSB+/k}$, $\mathcal{C}_{pos,=}$, $\mathcal{C}_{IHSB-/k}$, $\mathcal{C}_{neg,=}$, $\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$	$\mathcal{C}_{dHorn}$, $\mathcal{C}_{Horn}$ IS_{01}, IS_{11}, IV, IE		$\mathcal{C}_{CNF}$ IN
	PosTerms	$\mathcal{C}_{impl}$, $\mathcal{C}_{neg,=}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{C}_{Horn}$ IS_0^2, IS_{11}^2, IV		$\mathcal{C}_{CNF}$ IN
	Terms	$\mathcal{E}_{aff}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{C}_{Horn}$ IM, IS_0^2, IS_1^2		$\mathcal{C}_{CNF}$ IN
	PosCNFs	$\mathcal{C}_{impl}$, $\mathcal{C}_{neg,=}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IS_0^2, IS_{11}^2, ID, IV, IL		$\mathcal{C}_{CNF}$ IN
	CNFs		$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IBF		$\mathcal{C}_{CNF}$ IN
P-Abd	PosLits or PosClauses	$\mathcal{C}_{IHSB-}$, $\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{Horn}$ IE_0	$\mathcal{C}_{1v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_0
	Lits or Clauses	$\mathcal{C}_{IHSB-/k}$, $\mathcal{C}_{neg,=}$, $\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{Horn}$ IS_{11}	$\mathcal{C}_{1v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_0
	PosTerms	$\mathcal{C}_{neg,=}$, $\mathcal{E}_{aff/2}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{1v-aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IS_{11}^2, IL_3, IL_0	$\mathcal{C}_{1v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_0
	Terms	$\mathcal{E}_{aff/2}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{1v-aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IS_1^2, IL_3, IL_0	$\mathcal{C}_{1v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_0
	PosCNFs	$\mathcal{C}_{neg,=}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{1v-aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IS_{11}^2, ID, IL_0	$\mathcal{C}_{1v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_0
	CNFs	$\mathcal{C}_{dHorn}$, $\mathcal{E}_{1v-aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IS_1^2, ID, IL_0	$\mathcal{C}_{1v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_0
N-Abd	PosLits or PosClauses	$\mathcal{C}_{IHSB+/k}$, $\mathcal{C}_{pos,=}$, $\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$, $\mathcal{C}_{0v}$	$\mathcal{C}_{dHorn}$ IS_{01}		$\mathcal{C}_{CNF}$ IN_2, II_1
	Lits or Clauses	$\mathcal{C}_{IHSB+/k}$, $\mathcal{C}_{pos,=}$, $\mathcal{C}_{bij}$, $\mathcal{C}_{0v-dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$	$\mathcal{C}_{dHorn}$ IS_{01}	$\mathcal{C}_{0v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_1
	PosTerms	$\mathcal{E}_{aff/2}$, $\mathcal{C}_{Horn}$, $\mathcal{C}_{0v}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$ IS_0^2, IL_1, IL_3		$\mathcal{C}_{CNF}$ IN_2, II_1
	Terms	$\mathcal{E}_{aff/2}$, $\mathcal{C}_{0v-dHorn}$, $\mathcal{E}_{0v-aff}$, $\mathcal{C}_{Horn}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$ IS_0^2, IL_1, IL_3	$\mathcal{C}_{0v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_1
	PosCNFs	$\mathcal{C}_{Horn}$, $\mathcal{C}_{0v}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$ IS_0^2, ID, IL_1		$\mathcal{C}_{CNF}$ IN_2, II_1
	CNFs	$\mathcal{C}_{0v-dHorn}$, $\mathcal{E}_{0v-aff}$, $\mathcal{C}_{Horn}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$ IS_0^2, ID, IL_1	$\mathcal{C}_{0v}$ IN	$\mathcal{C}_{CNF}$ IN_2, II_1
L-Abd	PosLits or PosClauses	$\mathcal{C}_{IHSB+/k}$, $\mathcal{C}_{pos,=}$, $\mathcal{C}_{IHSB-}$, $\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{dHorn}$, $\mathcal{C}_{Horn}$ IS_{01}, IV, IE_0		$\mathcal{C}_{CNF}$ IN
	Lits or Clauses	$\mathcal{C}_{IHSB+/k}$, $\mathcal{C}_{pos,=}$, $\mathcal{C}_{IHSB-/k}$, $\mathcal{C}_{neg,=}$, $\mathcal{C}_{bij}$, $\mathcal{E}_{aff}$	$\mathcal{C}_{dHorn}$, $\mathcal{C}_{Horn}$ IS_{01}, IS_{11}, IV, IE		$\mathcal{C}_{CNF}$ IN
	PosTerms	$\mathcal{C}_{impl}$, $\mathcal{C}_{neg,=}$, $\mathcal{E}_{aff/2}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IS_0^2, IS_{11}^2, IV, IL		$\mathcal{C}_{CNF}$ IN
	Terms	$\mathcal{E}_{aff/2}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IM, IS_0^2, IS_1^2, IL		$\mathcal{C}_{CNF}$ IN
	PosCNFs	$\mathcal{C}_{impl}$, $\mathcal{C}_{neg,=}$, $\mathcal{C}_{1v-Horn}$	$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IS_0^2, IS_{11}^2, ID, IV, IL		$\mathcal{C}_{CNF}$ IN
	CNFs		$\mathcal{C}_{bij}$, $\mathcal{C}_{dHorn}$, $\mathcal{E}_{aff}$, $\mathcal{C}_{Horn}$ IBF		$\mathcal{C}_{CNF}$ IN

Table 3
Complexity of propositional abduction

- L-Abd(Γ,$\mathcal{M}$), where $\langle\Gamma\rangle \in \{IS_{01}, IS_{00}, IS_{11}, IS_{10}\}$, $\mathcal{M} \in \{$Lits, Clauses$\}$,
- N-Abd(Γ,$\mathcal{M}$), where $\langle\Gamma\rangle \in \{IS_{01}, IS_{00}\}$, $\mathcal{M} \in \{$(Pos)Lits, (Pos)Clauses$\}$,
- P-Abd(Γ,$\mathcal{M}$), where $\langle\Gamma\rangle \in \{IS_{11}, IS_{10}\}$, $\mathcal{M} \in \{$Lits, Clauses$\}$,
- V-Abd(Γ,$\mathcal{M}$), where $\langle\Gamma\rangle \in \{IS_{01}, IS_{00}\}$, $\mathcal{M} \in \{$(Pos)Lits, (Pos)Clauses$\}$, and
- V-Abd(Γ,$\mathcal{M}$), where $\langle\Gamma\rangle \in \{IS_{01}, IS_{00}, IS_{11}, IS_{10}\}$, $\mathcal{M} \in \{$Lits, Clauses$\}$.

These cases are exceptional since they are all NP-complete (by Proposition 63 and obvious reductions), but if Γ is replaced by any finite $\Gamma' \subseteq \Gamma$, then the problems are in P. This is because these languages include relations described by clauses of arbitrary length, while their finite subsets only allow to express clauses of bounded length. Indeed, they only contain binary clauses and clauses with only one polarity, such as $(\neg x_1 \vee \cdots \vee \neg x_k)$. So the only clauses with can be resolved against each other to infer a new clause are two binary clauses, or a “long” clause and a binary one, so in no case can a clause of length greater than k be inferred.

As a sidenote we want to remark that in the literature on computational problems over restricted constraint languages there are two notions of tractability: local and global. A problem over a constraint language Γ is said to be *locally tractable* if the problem is tractable over every finite subset of Γ, and *global tractability* coincides with the tractability notion used in this paper. Thus, as first noted by Creignou [10] (and as should be obvious from the discussion above), the notions of global and local tractability disagree for the abduction problem. This highlights a difference between the complexity of abduction and many other computational problems over constraint language restrictions, such as Sat(Γ), for which the notions of local and global tractability coincide.

Coming back to the completeness of the classification, it is a tedious but straightforward task (thanks to Post's classification) to check that, for all constraint languages Γ and restrictions on hypotheses and manifestations considered in the paper (which are not in one of the special cases already treated above), our results show that either

- the abduction problem over Γ is in P as a consequence of $\Gamma \subseteq \langle\mathcal{C}\rangle$ for some tractable clausal or equational language $\mathcal{C}$, or
- the abduction problem over Γ is NP-complete (coNP-complete, Σ_2^{P}-complete) as a consequence of $\Gamma' \subseteq \langle\Gamma\rangle$ for some finite NP-hard (coNP-hard, Σ_2^{P}-hard) constraint language Γ', and $\Gamma \subseteq \langle\mathcal{C}\rangle$ for some clausal or equational language $\mathcal{C}$ which is in NP (coNP, Σ_2^{P}).

Hence, we have a classification for the complexity of abduction over any constraint language Γ and any combination of restrictions on hypotheses and manifestations considered in the paper.

When it comes to clausal and equational languages $\mathcal{C}$, the reasoning is very similar. First note that all finite clausal/equational languages $\mathcal{C}$ are covered by our discussion about constraint languages above. This is because abduction over $\mathcal{C}$ has the same complexity as abduction over Γ when Γ and $\mathcal{C}$ are both finite and $\langle\Gamma\rangle = \mathcal{C}$. This is because finite languages allow us to use a simple lookup table to translate between the different representations in polynomial time. Hence, we can concentrate on infinite clausal/equational languages $\mathcal{C}$. Now, by Lemma 22 any lower bound on the complexity of abduction over a finite constraint language Γ carries over to any (infinite) clausal or equational language $\mathcal{C}$ such that $\Gamma \subseteq \langle\mathcal{C}\rangle$. Moreover, the infinite clausal languages $\mathcal{C}$ corresponding to the exceptional cases discussed above (where lower bounds cannot be proved by reductions from finite constraint languages) can all be proved to be NP-hard by Proposition 63. For infinite clausal or equational languages $\mathcal{C}$ consisting *only* of either clauses or equations (and not in the exceptional cases discussed above) the results in the paper show that either

- the abduction problem over $\mathcal{C}$ is in P as a consequence of $\mathcal{C} \subseteq \mathcal{C}'$ for some tractable clausal or equational language $\mathcal{C}'$, or
- the abduction problem over $\mathcal{C}$ is NP-complete (coNP-complete, Σ_2^{P}-complete) as a consequence of $\Gamma' \subseteq \langle\mathcal{C}\rangle$ for some finite NP-hard (coNP-hard, Σ_2^{P}-hard) constraint language Γ', and $\mathcal{C} \subseteq \mathcal{C}'$ for some clausal or equational language $\mathcal{C}'$ which is in NP (coNP, Σ_2^{P}).

The only remaining problem is upper bounds for infinite clausal and equational languages $\mathcal{C}$ expressed by both equations and clauses. For example, $\mathcal{C} = \mathcal{C}_{neg,=} \cup \{(x_1 \oplus x_2 \oplus x_3 = 1)\}$. Post's classification yields that all these infinite clausal/equational languages $\mathcal{C}$ satisfy $\langle\mathcal{C}\rangle \in \{\langle\mathcal{C}_{CNF}\rangle, \langle\mathcal{C}_{0v}\rangle, \langle\mathcal{C}_{1v}\rangle\}$. It is very easy to verify that the (trivial) upper bounds that hold for abduction problems over $\mathcal{C}_{CNF}$, $\mathcal{C}_{0v}$, and $\mathcal{C}_{1v}$ also hold for the corresponding abduction problems over $\mathcal{C}$ (see the proofs of Propositions 41, 44, and 45). Hence, our results cover any clausal and equational language.

As already mentioned, the authors have used a computer program to double-check the completeness of the classification.

14 Further restrictions

As evoked in Section 3.2, several restrictions on the problem were not considered until now but are worth investigating. It turns out that most of them do not affect the complexity of the problem.

14.1 Unsatisfiable and tautological manifestations

The first kind of restrictions is about the satisfiability of manifestations. For instance, in typical applications, abduction is the process of explaining a given observation of the world. As a consequence, one may assume that manifestations are always satisfiable.

We first give a straightforward result.

Proposition 74 *Let $P = (V, H, M, KB)$ be an instance of any abduction problem. If M is tautological, then P has an explanation if and only if KB is satisfiable. If M is unsatisfiable, then P has no explanation.*

Observe that one can efficiently recognize unsatisfiable and tautological literals (vacuously), clauses, or terms. Thus the complexity of abduction is not affected by the extra assumption that such manifestations are nontautological or satisfiable, since deciding whether the knowledge base is satisfiable is always at least as hard as abduction. As for CNFs, tautological ones can be efficiently recognized, but not unsatisfiable ones. Nevertheless, it turns out that the complexity is not affected either.

Proposition 75 L-ABD(Γ,CNFS) *is polynomial-time reducible to* L-ABD(Γ,CNFS) *where the manifestation is guaranteed to be satisfiable. The same result holds for* V-ABD*,* P-ABD*, or* N-ABD *instead of* L-ABD*.*

PROOF. Let (V, H, M, KB) be an instance of L-ABD(Γ,CNFS). Let *new* be a fresh variable ($new \notin V$), and let $V' = V \cup \{new\}$, $M' = M \vee new$. Clearly, from the CNF M a CNF M'' for $M \vee new$ can be computed efficiently by distributing $\vee new$ to each clause, and M'' is satisfiable. We claim that (V', H, M'', KB) has an explanation if and only if (V, H, M, KB) has one. Indeed, if $KB \wedge E$ is satisfiable and entails M, this is still true for M''. Conversely, if $KB \wedge E$ is satisfiable and entails $M \vee new$, then we have that $KB \wedge E \wedge \neg M \wedge \neg new$ is unsatisfiable. Since *new* does not occur at all in $KB \wedge E \wedge \neg M$ (and $\neg new$ alone is satisfiable), we have that $KB \wedge E \wedge \neg M$ alone is unsatisfiable, that is, $KB \wedge E \models M$. □

14.2 Variables in hypotheses and manifestations

Another interesting kind of restrictions is over the variables allowed to occur in the set of hypotheses and in manifestations. In particular, it is interesting to consider cases where variables which do not occur in the knowledge base occur in hypotheses and manifestations. This indeed allows to model situations where the knowledge base tells nothing about a part of the abduction problem at hand.

First of all, it turns out that allowing or not such extra variables in the set of *hypotheses* does not affect the complexity of abduction.

Proposition 76 *Let Γ be a constraint language, and let $P = (V, H, M, KB)$ be an instance of* $\textsc{Abd}(\Gamma)$. *Then for any explanation E for P there is an explanation E_1 for P such that $E_1 \subseteq E$ and $\mathit{Vars}(E_1) \subseteq \mathit{Vars}(KB) \cup \mathit{Vars}(M)$.*

PROOF. Let E be an explanation for P, and let $E_1 = E \cap \mathit{Lits}(\mathit{Vars}(KB) \cup \mathit{Vars}(M))$ and $E_2 = E \setminus E_1$. Then clearly $KB \wedge \bigwedge E_1$ is satisfiable, since so is $KB \wedge \bigwedge E$. Now by definition of an explanation we have $KB \wedge \bigwedge E_1 \wedge \bigwedge E_2 \models M$. Since the left-hand side is satisfiable and $\mathit{Vars}(E_2)$ is disjoint from $\mathit{Vars}(KB)$, $\mathit{Vars}(E_1)$, and $\mathit{Vars}(M)$, we get that $\bigwedge E_2$ is irrelevant to the entailment relation. Thus E_1 alone is an explanation. $\square$

As concerns variables occurring in manifestations, the situation is a bit more involved. The following lemma is relevant to the cases of literals, clauses, or terms, for which the complexity is not affected. Indeed, it shows that for such manifestations, one can consider independently the part of the manifestation which is over $\mathit{Vars}(KB)$ and the other part, with almost no computational overhead for the latter.

Proposition 77 *Let Γ be a constraint language, and let $P = (V, H, M, KB)$ be an instance of* $\textsc{Abd}(\Gamma)$. *If $M = M_1 \wedge M_2$ with $\mathit{Vars}(M_1) \subseteq \mathit{Vars}(KB)$ and $\mathit{Vars}(M_2) \cap \mathit{Vars}(KB) = \emptyset$, then for any explanation E for P there is a partition of E into $\{E_1, E_2\}$ such that E_1 is an explanation for (V, H, M_1, KB) and $\bigwedge E_2 \models M_2$ holds.*

Dually, if $M = M_1 \vee M_2$ with $\mathit{Vars}(M_1) \subseteq \mathit{Vars}(KB)$ and $\mathit{Vars}(M_2) \cap \mathit{Vars}(KB) = \emptyset$, then for any explanation E for P there is a partition of E into $\{E_1, E_2\}$ such that E_1 is an explanation for (V, H, M_1, KB) or $\bigwedge E_2 \models M_2$ holds.

PROOF. Consider $M = M_1 \wedge M_2$; the case $M = M_1 \vee M_2$ is similar. Let E be an explanation for P, and let $E_1 = E \cap \mathit{Lits}(\mathit{Vars}(KB))$ and $E_2 = E \setminus E_1$. Then by definition of an explanation we have $KB \wedge \bigwedge E_1 \wedge \bigwedge E_2 \models M_1$ and $KB \wedge \bigwedge E_1 \wedge \bigwedge E_2 \models M_2$. In the first entailment relation, since $\mathit{Vars}(M_1) \subseteq \mathit{Vars}(KB \wedge \bigwedge E_1)$ and $\mathit{Vars}(E_2) \cap \mathit{Vars}(KB \wedge \bigwedge E_1) = \emptyset$ hold, and the left-hand side is satisfiable, we get that E_2 is irrelevant, that is, $KB \wedge \bigwedge E_1$ alone entails M_1; since moreover $KB \wedge \bigwedge E_1 \wedge \bigwedge E_2$ is satisfiable, E_1 is an explanation for

(V, H, M_1, KB). Similarly, in the second entailment relation both KB and E_1 are irrelevant, thus $\bigwedge E_2 \models M_2$ holds. $\square$

We now turn to CNF manifestations. The following lemma shows that for almost all languages, we can assume $\mathit{Vars}(M) \subseteq \mathit{Vars}(KB)$ or not without affecting the complexity of abduction.

Proposition 78 *Let* Γ *be any constraint language with* $\Gamma \not\subseteq IR_2$ *or* $\mathrm{R}_= \in \Gamma$*. Then one can assume up to polynomial-time reductions that an instance* (V, H, M, KB) *of* $\textsc{Abd}(\Gamma)$ *satisfies* $\mathit{Vars}(M) \subseteq \mathit{Vars}(KB)$*.*

PROOF. If $\Gamma \not\subseteq IR_2$ or $\mathrm{R}_= \in \Gamma$, then Γ contains at least one relation R which is nonempty, n-ary and not equivalent to any term of length n. Then there is at least one place $i \in \{1, \ldots, n\}$ and two tuples $(\mu_1, \ldots, \mu_{i-1}, 0, \mu_{i+1}, \ldots, \mu_n)$, $(\mu'_1, \ldots, \mu'_{i-1}, 1, \mu'_{i+1}, \ldots, \mu'_n)$ in R. Then the assumption $\mathit{Vars}(M) \subseteq \mathit{Vars}(KB)$ can be enforced as follows. For each variable x in $\mathit{Vars}(M) \setminus \mathit{Vars}(KB)$ add $n-1$ fresh variables $new_{x,1}, \ldots, new_{x,n-1}$ to V and add the constraint $R(new_{x,1}, \ldots, new_{x,i-1}, x, new_{x,i}, \ldots$ to KB. Write KB' and V' for the resulting KB and V. By construction x is unconstrained by KB', and thus (V', H, M, KB') has an explanation if and only if (V, H, M, KB) has one. $\square$

The only special case is the following, for the class of CNF formulas containing only unit clauses (and no equality relations), when $\mathit{Vars}(M) \subseteq \mathit{Vars}(KB)$.

Proposition 79 L-$\textsc{Abd}(\mathcal{C}_{\mathit{unit},=} \backslash \{\mathrm{R}_=\}, \textsc{CNFs})$ *is in* P *if instances* (V, H, M, KB) *are required to satisfy* $\mathit{Vars}(M) \subseteq \mathit{Vars}(KB)$*. This holds even if nothing is known about the satisfiability of* M*.*

PROOF. Obvious since with the assumptions, KB is logically equivalent to a term which defines a complete assignment to $\mathit{Vars}(M)$. Thus we only have to decide whether this assignment satisfies M. $\square$

15 Discussion

We now explain in an intuitive manner what makes propositional abduction hard. We focus on Schaefer languages (Horn, dual Horn, bijunctive and affine), i.e., those languages for which deduction of CNFs is tractable, since they are the most interesting ones for reasoning and knowledge representation. Recall from Proposition 35 that abduction is in NP for all such languages. Thus we explain what makes it NP-complete. We also compare our observations with Bylander *et al.*'s about set-covering abduction [7].

We discuss the two most meaningful cases first, namely those of *clausal* Schaefer languages (Horn, dual Horn and bijunctive) with terms and literals as manifestations. The other cases are only briefly discussed afterwards.

15.1 Manifestations expressed by terms

Given restrictions on the abduction problem, we say that a literal is a *valid* hypothesis (resp. individual manifestation) if it can be part of H (resp. of M) for some instance (V, H, M, KB) of the problem. For instance, the valid individual manifestations for N-ABD($\mathcal{C}_{CNF}$,POSTERMS) are all positive literals.

Fact 80 *Let $\mathcal{C}$ be a clausal Schaefer language, and assume a restriction on hypotheses and a restriction of manifestations to* POSTERMS, NEGTERMS, *or* TERMS. *Then propositional abduction is* NP*-hard for $\mathcal{C}$ if and only if it is so with stronger restrictions or $\mathcal{C}$ can express both*

- *implication from hypotheses to individual manifestations, that is, $\ell_H \rightarrow \ell_M$ for any valid hypothesis ℓ_H and valid manifestation ℓ_M, and*
- *forbidden combinations of hypotheses, that is, $\overline{\ell_{H,1}} \vee \overline{\ell_{H,2}}$ for any two valid hypotheses $\ell_{H,1}, \ell_{H,2}$.*

For instance, P-ABD($\mathcal{C}$,POSTERMS) is NP-hard exactly when $\mathcal{C}$ can express both $(x_1 \rightarrow x_2)$ and $(\neg x_1 \vee \neg x_2)$ for any two variables x_1, x_2. P-ABD($\mathcal{C}$,TERMS) is NP-hard exactly when P-ABD($\mathcal{C}$,POSTERMS) or P-ABD($\mathcal{C}$,NEGTERMS) is NP-hard, or when $\mathcal{C}$ can express $(x_1 \rightarrow x_2)$, $(x_1 \rightarrow \neg x_2)$ (implication) and $(\neg x_1 \vee \neg x_2)$.

Importantly, observe from Fact 80 that if abduction is NP-hard for, e.g., unrestricted terms as manifestations, then so it is for positive or for negative terms with the same restriction on hypotheses. That is, arbitrary combinations of polarities in the manifestation do not add to the complexity of the problem. The same holds for hypotheses, that is, being able to explain with unrestricted hypotheses (or sets of hypotheses closed under complement) is not harder than being able to explain with only positive or with only negative hypotheses.

The validity of Fact 80 is proved by the generic reduction given by Lemma 61, applied to each precise problem. More intuitively, the idea is that in the NP-hard cases, each literal m to be explained in a term gives rise to several possible explanations, one for each implication $h \rightarrow m$ expressed by the knowledge base. Thus, each m gives rise to a disjunction of hypotheses, and the whole manifestation M gives rise to a conjunction of disjunctions (CNF) of hypotheses. Now if the knowledge base can forbid some combinations of hypotheses, we thus have a second CNF, which excludes some models of the first one as explanations. Since the clauses in the first CNF are unbounded, it is easily seen that this characterization of explanations captures a whole class of hard satisfiability problems.

As for the tractable cases, if no implication from hypotheses to individual manifestations can be expressed, then there cannot be any explanation, since an explanation for a term is an explanation for each of its literals (Lemma 28). Now if no conjunction of hypotheses can be forbidden, then the search space can be reduced to the conjunction of all hypotheses (Lemma 43).

Importantly, Fact 80 gives some intuition about the complexity of abduction for some classes of formulas which are not captured by Schaefer's framework of constraint languages. Consider for instance the class of *acyclic* Horn CNFs, that is, of Horn CNFs which do not contain any *cyclic* set of clauses of the form $\{(\cdots \vee \neg x_1 \vee \cdots \vee x_2), (\cdots \vee \neg x_2 \vee \cdots \vee x_3), \ldots (\cdots \vee \neg x_k \vee \cdots \vee x_1), \}$. Clearly, such a formula can contain an arbitrary number of clauses equivalent to $(h \rightarrow m)$ and $(\neg h_1 \vee \neg h_2)$ for variables h, m, h_1, h_2. Thus Fact 80 gives the intuition that explaining positive terms with positive hypotheses is NP-hard when the knowledge base is restricted to be an acyclic Horn CNF. This result has indeed been shown by Selman and Levesque [48]. Note however that we cannot formally state Fact 80 for such general classes of formulas, since the ability to express a given relation would not be properly defined.

Another important remark is that in our generic reduction from the satisfiability problem, all variables (or all positive, all negative literals) not occurring in the query can be hypotheses while the problems remains hard. We come back to this point when comparing to the case of manifestations expressed by single literals and in the conclusion.

15.2 A parallel with set-covering abduction

In their seminal paper [7], Bylander *et al.* study the complexity of set-covering abduction, and also identify a frontier between some polynomial and some NP-complete abduction problems.

In their framework, an abduction problem is given by a set of manifestations M, a set of hypotheses H (hypotheses and manifestations are atoms), and by a map e ("explains") from subsets of H to subsets of M. An explanation is a subset E of H such that $e(E) = M$ and E is minimal for inclusion. The only assumptions about e is that the size of its representation is polynomial in the size of H and M, and that it is tractable to compute $e(E)$ for $E \subseteq H$ as well as a kind of inverse of e (we refer to their paper for details).

Bylander *et al.* study the complexity of abduction under various further assumptions about e. Of direct relevance to our study are the following restrictions. The problem is said to be

- *independent* if $\forall E \subseteq H, e(E) = \bigcup_{h \in E} e(\{h\})$,
- *monotonic* if $\forall E, E' \subseteq H, E \subseteq E' \Rightarrow e(E) \subseteq e(E')$, and
- *an incompatibility problem* if there is a set I of pairs of elements of H (incompatible hypotheses) such that $\forall E \subseteq H, (\exists \{h, h'\} \in I, h, h' \in H) \Rightarrow e(E) = \emptyset$.

Observe that all independent problems are also monotonic. The notion of independence is adapted to incompatibility as follows. An incompatibility problem is said to be independent if $e(E) = \bigcup_{h \in E} e(\{h\})$ as soon as there is no $\{h, h'\} \in I$ such that $h, h' \in E$.

Proposition 81 ([7]) *The problem of deciding whether an explanation exists is in* P *for independent abduction problems and for monotonic abduction problems. It is* NP*-complete for independent incompatibility problems.*

Although the framework is different from ours, it is clear that the condition for NP-hardness in Proposition 81 is very close to our Fact 80.

Incompatible pairs of hypotheses of Bylander *et al.*'s framework clearly correspond in our framework to sets of hypotheses E which are not consistent with the knowledge base ($KB \wedge \bigwedge E$ is unsatisfiable)[3]. Now, since we study abduction in classical propositional logic, where the consequence relation $\models$ is monotonic, our abduction problems are monotonic. Thus, leaving details out, our abduction problems are incompatibility monotonic problems in the terms of [7], where e is represented in intension by the knowledge base KB. It is also easily seen that any independent incompatibility problem can be transformed into a problem in our framework.

With this correspondence in mind, it is clear that Fact 80 confirms Bylander *et al.*'s results. Indeed, the condition for NP-hardness in Proposition 81 states that abduction is hard when some combinations of hypotheses are forbidden (expressiveness of implication from hypotheses to manifestations is implicit in their framework), but becomes tractable without this assumption. Thus our observation for terms as manifestations can be seen as generalizing Bylander *et al.*'s to more complex interactions between hypotheses, and between hypotheses and manifestations.

[3] As observed in [7], considering incompatible pairs, triples, etc. instead of only pairs does not affect the complexity results.

The condition for manifestations expressed by literals is a bit more involved than that for terms, but it can intuitively be seen as the condition allowing to capture the complexity of conjunctive manifestations because it allows to express $(m_1 \wedge \cdots \wedge m_k \rightarrow m)$. Call *submanifestation* any literal which is neither a hypothesis nor the manifestation (intuitively, the m_i's in the previous implication). Given a polarity restriction (to positive, to negative, or to any literal), a submanifestation is said to be *valid* if it satisfies the restriction.

Fact 82 *Let $\mathcal{C}$ be a clausal Schaefer language, and assume a restriction on hypotheses and a restriction of manifestations to* PosLits*,* NegLits*, or* Lits*. Then propositional abduction is* NP*-hard for $\mathcal{C}$ if and only if it is so with stronger restrictions or there is a polarity restriction on submanifestations such that $\mathcal{C}$ can express*

- *implication from hypotheses to individual submanifestations, that is, $\ell_H \rightarrow \ell_S$ for any valid hypothesis ℓ_H and valid submanifestation ℓ_S,*
- *implication from arbitrary conjunctions of submanifestations to individual manifestations, that is, $(\ell_{S,1} \wedge \cdots \wedge \ell_{S,k} \rightarrow \ell_M)$ for any $k \in \mathbb{N}$ and any k valid submanifestations $\ell_{S,1}, \ldots, \ell_{S,k}$,*
- *forbidden combinations of hypotheses, that is, $\overline{\ell_{H,1}} \vee \overline{\ell_{H,2}}$ for any two valid hypotheses $\ell_{H,1}, \ell_{H,2}$.*

For instance, N-Abd($\mathcal{C}$,PosLits) is NP-hard when, among other cases, $\mathcal{C}$ can express implication from hypotheses to individual (negative) submanifestations $(\neg x_1 \rightarrow \neg x_2)$, implication from arbitrary conjunctions of submanifestations to manifestations $((\neg x_1 \wedge \cdots \wedge \neg x_n \rightarrow x_{n+1})$, for any n), and forbidden conjunctions of hypotheses $(x_1 \vee x_2)$.

Observe that in general, depending on the restriction on submanifestations, there may be several different expressiveness conditions for the same restrictions on hypotheses and manifestations. Nevertheless, as in the case of terms as manifestations, it can be seen from Fact 82 that allowing arbitrary combinations of polarities for hypotheses makes the problem no harder than allowing only positive or negative sets of hypotheses (the corresponding statement for manifestations is obvious).

The proof of our observation can be derived from Lemmata 61 and 29. Intuitively, the "intermediate layer" serves to generate conjunctions of (intermediate) hypotheses needed to explain the manifestation, through each implication $y_1 \wedge \cdots \wedge y_n \rightarrow m$ where the y_i's are submanifestations. Thus the manifestation can be explained by a disjunction of conjunctions of submanifestations, each of which can be seen as a term which has to be explained by hypotheses. Thus, provided arbitrarily long conjunctions of submanifestations can entail the manifestation, the complexity is the same as in the case of a term.

An important difference to manifestations expressed by terms is that in general, NP-hardness arises when some "intermediate" set of literals can be expressed, where these literals can neither be hypothesized nor observed. For instance, this explains why the complexity of explaining single literals from Horn CNFs falls from NP-complete to polynomial when all variables except those in the manifestation are hypotheses, as shown by Selman and Levesque [48]. This also confirms their intuition that selecting the right set of literals is the computational core in abduction. Nevertheless, this intuition is confirmed only as far as literals are considered as manifestations. Indeed, as we have seen this is not the case for manifestations expressed by terms.

Finally, like for terms our characterization gives intuition about the complexity of abduction for classes of formulas which are not captured by Schaefer's framework. Consider for instance the class of monotone CNF formulas, that is, formulas in which every variable occurs always negated or always unnegated. A variable which always occurs with the same polarity cannot be a submanifestation for an abduction problem like in Fact 82, since it should occur both on the right of implications coming "from the hypotheses layer"(and thus, unnegated) and on the left of implications "going to the manifestation layer" (and thus, negated). Thus if all variables are monotone, the intermediate layer has to be empty and thus, abduction should be tractable. We thus recover (the intuition about) a well-known result (see, e.g., [37, Section 4.2]).

15.4 Manifestations expressed by clauses or CNFs

We are now able to explain *a posteriori* why one can observe that the complexity of the abduction problem is always the same, may (positive) clauses or (positive) literals be used as manifestations.

To this aim, first observe that naturally, when the problem is hard for literals, then so it has to be for clauses. Now concerning tractable cases, we use the condition exhibited above for literals. Indeed, the condition states that abduction is tractable for literals if either

(1) the knowledge bases cannot express a "two-layers" set of implications, from hypotheses to literals, and from arbitrary combinations of the latter to the manifestation,
(2) or no conjunction of hypotheses can be forbidden.

It is easily seen that Condition 2 is the same, may literals or clauses (or terms) be considered as manifestations. Now regarding Condition 1, if it is true with literals, then it has to be true for clauses, since literals are a special case of clauses. We thus deduce tractability exactly as for literals.

Let us mention that, as noted for example by Eiter and Makino [23, Section 5.2], there is a rather generic polynomial-time reduction from clausal manifestations to single-literal manifestations. The idea is that $M = (\ell_1 \vee \cdots \vee \ell_k)$ can be replaced with a literal ℓ, over a fresh variable, up to adding constraints $\ell_i \rightarrow \ell$ $(i = 1, \ldots, k)$ to the knowledge base. This reduction covers many cases, but fails to explain, for instance, why P-Abd($\mathcal{C}_{1v-Horn}$,Clauses) has the same complexity as P-Abd($\mathcal{C}_{1v-Horn}$,Lits). Indeed, the added constraints would not preserve $\mathcal{C}_{1v-Horn}$ in general.

Similarly to the case of clauses vs. literals, we can observe that, as far as clausal, Schaefer language are concerned, the complexity of abduction with manifestations expressed by CNFs is the same as that with manifestations expressed by terms. The case of affine knowledge bases is different, with problems harder for CNFs than for terms. Nevertheless, we do not elaborate on that point here, since most of the hardness for CNF manifestations simply comes from deciding whether the manifestation alone is satisfiable.

15.5 Affine knowledge bases

The case of equational instead of clausal languages is a bit more involved, but follows the same reasoning. We only sketch the idea, since equational languages are less interesting than CNF ones for knowledge representation purposes.

The main difference to the clausal case is that implications cannot be directly expressed. Indeed, if, for instance, $h \rightarrow m$ is a prime implicate of an affine knowledge base, then so must $m \rightarrow h$ be. In other words, in this case, the knowledge base entails $h \leftrightarrow m$. It follows that if the knowledge base entails both $h_1 \rightarrow m$ and $h_2 \rightarrow m$, in fact it entails both $h_1 \leftrightarrow m$ and $h_2 \leftrightarrow m$, and finally, any explanation for m must also satisfy h_1 and h_2.

In general this makes the problem easier. The hard cases arise when implication can be simulated by imposing polarities on some hypotheses, just as is done, for instance, in the proof of Proposition 68. This also explains why the complexity for equational languages sometimes changes from V-Abd to L-Abd, contrary to clausal languages.

15.6 A note on NP*-complete and* coNP*-complete cases*

As can be seen from the results presented in the previous sections, it turns out that some restrictions on knowledge bases yield NP-complete problems, while others yield coNP-complete problems. This may seem strange at first sight, and deserves some explanation. In particular, we are not aware of any previous results in the literature where coNP-complete propositional abduction problems have been identified.

Recall that Σ_2^{P} is $\mathrm{NP}^{\mathrm{NP}}$ or, equivalently, $\mathrm{NP}^{\mathrm{NP}\cup\mathrm{coNP}}$.

Essentially there are two sources of complexity in the general, Σ_2^{P}-complete abduction problem: finding the right candidate explanation E (non-determinism, represented by the "basis" of the "exponential" $\mathrm{NP}^{\mathrm{NP}\cup\mathrm{coNP}}$), and checking that it is indeed a witness, i.e., that $KB \wedge \bigwedge E$ is satisfiable and $KB \wedge \bigwedge E \models M$ (represented by the "exponent").

That various restrictions on the hypotheses, manifestations, and knowledge bases in the abduction problem yield NP-complete problems is not very surprising. These NP-complete cases occur when both checking a candidate explanation is a polynomial problem, which is the case for Schaefer languages (Proposition 35), and the satisfiability problem for a set of clauses can be reduced to finding "the right candidate explanation", as illustrated by Lemma 61. The perhaps more surprising coNP-complete cases all occur when checking a candidate explanation is coNP-complete and only one candidate explanation needs to be considered, removing nondeterminism as a source of complexity and thus, in some sense, the "basis of the exponential". Observe that in this paper, when only one candidate explanation needs to be considered, this is always a result of $KB \wedge \bigwedge H$ being trivially satisfiable as is, for example, the case for P-Abd($\mathcal{C}_{1v}$,$\mathcal{M}$).

16 Conclusion and future work

We have presented a thorough study of the complexity of deciding whether there is an explanation for a propositional abduction problem, under various restrictions over hypotheses and manifestations. We have derived the complexity for every possible local restriction on the knowledge bases for the problem, that is, for every Boolean constraint language and for every clausal or equational language, under the assumptions that the constraints in the knowledge base are given in extension or implicitly as CNFs or systems of equations, respectively. In particular this covers all the classical classes of CNF formulas defined by local properties and commonly considered for knowledge representation purposes. Moreover, we have shown that the problem, when not tractable, is complete for one of NP, coNP, or Σ_2^{P}.

In order to obtain these results, we have used now well-known techniques for obtaining tractability (e.g., prime implicate generation and projection). We have nevertheless uncovered new tractable cases. But we have also exhibited new conditions for intractability, some of which turn out to be very weak, as already observed in [40].

Importantly, this study allows us to precisely explain what makes propositional abduction hard when the knowledge base is over a Schaefer language. Several points are very interesting, such as the need for variables which are neither hypothesized nor observed for making the task of explaining a literal hard, contrary to the case of a term. It also turns out that the conditions for intractability of abduction are so weak that almost no interesting class of problems can be tractable. In fact, only classes where there can be no explanation or there can be no conflicting hypotheses are tractable. This confirms the need for designing algorithms which are efficient in practice. Not surprisingly, our conditions for intractability confirm the results and observations by Bylander *et al.* in a different framework [7].

We have also argued that our explanation of the hardness of abduction gives hints for the complexity of problems which do not fit our framework. We demonstrated that argument for nonlocal (structural) propositional restrictions, but conjecture that it could also apply to completely different frameworks, such as abduction with nonclassical consequence relations or abduction in logic programming.

Propositional abduction turns out to be a very rich problem from a computational complexity perspective. By imposing various syntactic restrictions on the problem, no less than four complexity classes are covered. Thus our results can serve as a source of results allowing to derive the complexity of other problems. As a matter of fact, it is well-known that under various restrictions, several nonmonotonic reasoning problems reduce to each other. As an example, our classification for abduction with negative hypotheses and positive terms as manifestations gives the complexity of circumscriptive inference of negative clauses, as far as clausal languages restrict the knowledge base. Indeed, it is known (see [40, Proposition 11]) that a negative clause follows from a knowledge base by circumscription if and only if its negation cannot be explained with all negative variables as hypotheses (except those in the clause). Thus, the complement of circumscriptive inference of negative clauses is at most as hard as abduction of positive terms with negative hypotheses. Now, as we discuss in Section 15.1, our classification for Schaefer clausal languages is preserved by the assumption that all variables except those in the manifestation are abducible. Thus, provided that KB is in CNF and subject to a restriction over each of its clauses, we get that deciding whether a negative clause follows from a knowledge base KB under circumscription of all variables is

- polynomial-time solvable if KB is restricted to be Horn or 0-valid,

- otherwise, coNP-complete if KB is restricted to be in 2CNF or dual Horn,
- otherwise, Π_2^{P}-complete.

We believe that our results may be used to derive other results in a similar fashion.

We now wish to note that in the paper, we have only studied the decision problem associated to abduction, leaving the search problem out. That is, we have not considered the complexity of computing an explanation instead of only deciding whether there is at least one. However, obviously the search problem is hard as soon as the decision one is, and as is easily seen from our results, all of which are constructive, the converse is also true. Importantly, this is still true when searching for $\subset$-minimal explanations, because tractability of abduction entails tractability of satisfiability (considering an unsatisfiable manifestation) and deduction (considering an empty set of hypotheses), and this in turn is enough for minimizing an explanation with a greedy algorithm.

To conclude, it would be very interesting to extend this work into several directions. First of all, other problems related to abduction are of importance: for instance, deciding whether a hypothesis is relevant (part of at least one preferred explanation) or necessary (part of all of them) [20]; enumerating all explanations [23]; counting the number of (preferred) explanations [27]. Our work provides results which can serve as a basis for studying the complexity of these problems, in particular for deriving hard cases. Moreover, as we have done, it would be interesting to understand what exactly makes these problems easy or hard.

Another important direction is to go further than the propositional setting and Schaefer's framework. As mentioned above, we believe that our observations about the reasons for (in)tractability can explain results beyond our framework. In particular, it would be interesting to study the complexity of abduction with nonlocal restrictions over knowledge bases (like, e.g., Eshghi [24] and del Val [15] do) and with higher-cardinality domains.

References

[1] B. Aspvall, M. Plass, and R. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[2] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM SIGACT-Newsletter*, 34(4):38–52, 2003.

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

[4] E. Böhler, H. Schnoor, S. Reith, and H. Vollmer. Bases for Boolean co-clones. *Information Processing Letters*, 96(2):59–66, 2005.

[5] G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic reasoning: an overview.* CSLI, 1997.

[6] J. Buchler, editor. *Philosophical Writings of Peirce.* Dover Publications, New York, 1955.

[7] T. Bylander, D. Allemang, M. Tanner, and J. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49:25–60, 1991.

[8] M. Cadoli, F. Donini, P. Liberatore, and M. Schaerf. Preprocessing of intractable problems. *Information and Computation*, 176:89–120, 2002.

[9] P. Chapdelaine, M. Hermann, and I. Schnoor. Complexity of default logic on generalized conjunctive queries. In *Proc. 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, pages 58–70, 2007.

[10] N. Creignou. Personal communication, 2006.

[11] N. Creignou, P. Kolaitis, and B. Zanuttini. Preferred representations of Boolean relations. Technical Report 119, Electronic Colloquium on Computational Complexity (ECCC), 2005.

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

[13] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[14] A. del Val. A new method for consequence finding and compilation for restricted languages. In *Proc. 16th National Conference on Artificial Intelligence (AAAI'99)*, pages 259–264. AAAI Press, 1999.

[15] A. del Val. The complexity of restricted consequence finding and abduction. In *Proc. 17th National Conference on Artificial Intelligence (AAAI'00)*, pages 337–342. AAAI Press/MIT Press, 2000.

[16] A. del Val. On some tractable classes in deduction and abduction. *Artificial Intelligence*, 116(1-2):297–313, 2000.

[17] W. Dowling and J. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.

[18] A. Durand and M. Hermann. The inference problem for propositional circumscription of affine formulas is coNP-complete. In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'03)*, number 2607 in Lecture Notes in Computer Science, pages 451–462, 2003.

[19] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving advanced reasoning tasks using quantified boolean formulas. In *Proc. 17th National Conference on Artificial Intelligence (AAAI'00)*, pages 417–422. AAAI Press, 2000.

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

[21] T. Eiter, G. Gottlob, and N. Leone. Abduction from logic programs: Semantics and complexity. *Theoretical Computer Science*, 189:129–177, 1997.

[22] T. Eiter, G. Gottlob, and N. Leone. Semantics and complexity of abduction from default theories. *Artificial Intelligence*, 90:177–223, 1997.

[23] T. Eiter and K. Makino. On computing all abductive explanations from a propositional Horn theory. *Journal of the ACM*, 54(5), 2007.

[24] K. Eshghi. A tractable class of abduction problems. In *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 3–8. Morgan Kaufmann, 1993.

[25] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[26] D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27(1):95–100, 1968.

[27] M. Hermann and R. Pichler. Counting complexity of propositional abduction. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 417–422, 2007.

[28] A. Herzig, J. Lang, and P. Marquis. Planning as abduction. In *IJCAI'01 Workshop on Planning under Uncertainty*, 2001.

[29] J. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.

[30] T. Horiyama and T. Ibaraki. Reasoning with ordered binary decision diagrams. *Discrete Applied Mathematics*, 142(1–3):151–163, 2004.

[31] H. Kautz, M. Kearns, and B. Selman. Horn approximations of empirical data. *Artificial Intelligence*, 74:129–145, 1995.

[32] R. Khardon and D. Roth. Reasoning with models. *Artificial Intelligence*, 87:187–213, 1996.

[33] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[34] J. Lang, P. Liberatore, and P. Marquis. Propositional independence — formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.

[35] P. Liberatore and M. Schaerf. Compilability of propositional abduction. *ACM Transactions on Computational Logic*, 8(1), 2007.

[36] F. Lin and J.-H. You. Abduction in logic programming: A new definition and an abductive procedure based on rewriting. *Artificial Intelligence*, 140:175–205, 2002.

[37] P. Marquis. Consequence finding algorithms. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems (DRUMS)*, volume 5, pages 41–145. Kluwer Academic, 2000.

[38] C. Morgan. Hypothesis generation by machine. *Artificial Intelligence*, 2:179–187, 1971.

[39] G. Nordh. A trichotomy in the complexity of propositional circumscription. In *Proc. 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'04)*, pages 257–269, 2005.

[40] G. Nordh and B. Zanuttini. Propositional abduction is almost always hard. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 534–539, 2005.

[41] H. Pople. On the mechanization of abductive logic. In *Proc. 3rd International Joint Conference on Artificial Intelligence (IJCAI'73)*, pages 147–152, 1973.

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

[43] R. Pschel and L. Kaluznin. *Funktionen- und Relationenalgebren.* DVW, Berlin, 1979.

[44] W.V. Quine. On cores and prime implicants of truth functions. *American Mathematical Monthly*, 66:755–760, 1959.

[45] R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: preliminary report. In *Proc. 6th National Conference on Artificial Intelligence (AAAI'87)*, pages 183–188. AAAI Press/MIT Press, 1987.

[46] T. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory Of Computing (STOC'78)*, pages 216–226. ACM Press, 1978.

[47] B. Selman and H. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

[48] B. Selman and H. Levesque. Abductive and default reasoning: a computational core. In *Proc. 8th National Conference on Artificial Intelligence (AAAI'90)*, pages 343–348. AAAI Press, 1990.

[49] L. Simon and A. del Val. Efficient consequence finding. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 359–365. Morgan Kaufman, 2001.

[50] B. Zanuttini. Approximation of relations by propositional formulas: complexity and semantics. In *Proc. 5th Symposium on Abstraction, Reformulation and Approximation (SARA'02)*, number 2371 in Lecture Notes in Artificial Intelligence, pages 242–255. Springer Verlag, 2002.

[51] B. Zanuttini. New polynomial classes for logic-based abduction. *Journal of Artificial Intelligence Research*, 19:1–10, 2003.