Theoretical Computer Science, 189(1-2): 129–177, 1998

Abduction from Logic Programs: Semantics and Complexity*

Thomas Eiter† **Georg Gottlob** **Nicola Leone**‡

Christian Doppler Lab for Expert Systems,
Information Systems Department, TU Vienna
A-1040 Wien, Paniglgasse 16, Austria
(eiter|gottlob|leone)@dbai.tuwien.ac.at

Abstract

Abduction– from observations and a theory, find using hypotheses an explanation for the observations – gained increasing interest during the last years. This form of reasoning has wide applicability in different areas of computer science; in particular, it has been recognized as an important principle of common-sense reasoning.

In this paper, we define a general abduction model for logic programming, where the inference operator (i.e., the semantics to be applied on programs), can be specified by the user. Advanced forms of logic programming have been proposed as valuable tools for knowledge representation and reasoning. We show that logic programming semantics can be more meaningful for abductive reasoning than classical inference by providing examples from the area of knowledge representation and reasoning. The main part of the paper is devoted to an extensive study of the computational complexity of the principal problems in abductive reasoning, which are: Given an instance of an abduction problem (1) does the problem have solution (i.e., an explanation); (2) does a given hypothesis belong to some explanation; and (3) does a given hypothesis belong to all explanations.

This problems are analyzed for different underlying logic programming semantics, namely, the well-founded semantics, the stable model semantics and the minimal model semantics, paying attention to normal and disjunctive logic programs for the case of propositional as well as function-free first order programs. The main results are that the above abductive reasoning tasks on propositional logic programs populate the classes at the lower end of the polynomial hierarchy up to Σ_4^P, and provide complete problems for a number of classes over the first four levels of the hierarchy. Similar results are obtained in the first-order case. This proves abduction from logic programs as a rich source of problems of varying complexity.

---

*A preliminary extended abstract of this paper appeared in the Proceedings of the Third International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '95), W. Marek, A. Nerode, and M. Truszczyński, editors, pp. 1–14, LNCS 982, 1995.

†Part of this author's work has been carried out while visiting Stanford University.

‡This author was partially supported by the *Istituto per la Sistemistica e l'Informatica* of the Italian National Research Council (*ISI-CNR*)

Contents

1 Introduction

Logic programming is well-known as a powerful approach to declarative programming. Advanced semantics for negation and the capability of disjunction make logic programming yet more appealing for expressing problems in a variety of domains [4, 45].

The use of logic for representing knowledge was suggested by McCarthy in 1959 [51], and several researchers from various fields of computer science have then shown merits of logic for this purpose. However, the inadequacy of classical logic for representing commonsense knowledge is nowadays well understood, as its monotonicity (adding new information will never invalidate old conclusions) contrasts to the non-monotonic nature of commonsense reasoning. Thus, a number of logical formalisms, aimed at modeling commonsense reasoning, has been devised, among them modal logics [53, 55], default logic [63], and circumscription [52] (cf. also [50, 46]).

Logic programming with negation [44] and/or disjunction [45] shares with those formalisms the property of being non-monotonic. Its formalism is simple and natural, and its semantics has rigorous mathematical foundations; moreover, despite its simplicity, logic programming can express a wide range of problems even of high computational complexity [21, 66]. Thus, the use of logic programming as a valuable tool for knowledge representation and common-sense reasoning has been recently advocated [26, 45, 4].

So far, research on logic programming was mainly concerned with aspects of deduction from logic programs, in particular inference of (possibly negated) facts that are implicit by the meaning of the program. Besides deduction, abduction is another important kind of reasoning (though not completely disconnected), which has been first studied in depth by Peirce [57]. Given the observation of some facts, abduction aims at concluding the presence of other facts, from which, together with an underlying theory, the observed facts can be explained, i.e., deductively derived. Thus, roughly speaking, abduction amounts to an inverse of modus ponens.

For example, medical diagnosis is a typical abductive reasoning process: From the symptoms and the medical knowledge, a diagnosis about a possible disease is abduced. Notice that this form of reasoning is not sound (a diagnosis may turn out false), and that in general several abductive explanations (i.e., diagnoses) for observations may be possible.

During the last years, there has been increasing interest in abduction in different areas of computer science. It has been recognized that abduction is an important principle of common-sense reasoning, and that abduction has fruitful applications in a number of areas such diverse as model-based diagnosis [60], speech recognition [28], maintenance of database views [36], and vision [12].

In the past, most research on abduction concerned abduction from classical logic theories. However, we argue that the use of logic programming to perform abductive reasoning can be more appropriate in several applicative domains.

Example 1 For instance, consider the following scenario. Assume that it is Saturday and is known that Joe goes fishing on Saturdays if it's not raining. This may be represented by the following theory T:

$$go\_fishing \leftarrow is\_saturday \wedge \neg rains \ ; \qquad is\_saturday \leftarrow$$

Now you observe that Joe is not out for fishing. Intuitively, from this observation we conclude that it rains (i.e, we abduce $rains$), for otherwise Joe would be out for fishing. Nevertheless, under classical inference, the fact $rains$ is not an explanation of $\neg go\_fishing$, as $T \cup \{rains\} \not\models \neg go\_fishing$ (neither can one find any explanation). On the contrary, if we adopt the semantics of logic programming, then, according with the intuition, we obtain that $rains$ is an explanation of $\neg go\_fishing$, as it is entailed by $T \cup \{rains\}$. ∎

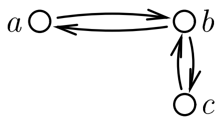
Notice that in the above example, the same result would be obtained under abduction using classical inference if we considered the Clark completion [13] of the theory; this approach has been investigated for acyclic theories in [14]. As will be shown next, there are simple and relevant examples where abduction yields different results when using logic programming compared to classical inference from the Clark completion.

Example 2 Consider the logic program below, which describes possible movements of a robot in a certain area. Initially, the robot is at position a; he can reach a certain position Y, if he either is at Y or moves straight from a reachable position X to it, provided that the straight way is viable.

$$\begin{array}{ll} r_1: & at(a) \leftarrow \\ r_2: & reach(Y) \leftarrow at(X) \\ r_3: & reach(Y) \leftarrow viable\_straight(X,Y) \wedge reach(X) \wedge move(X,Y) \end{array}$$

Combined with facts about which straight moves are possible exist and which moves are planned, the program tells us which positions b can potentially be reached by the robot and which not; the reachable are those for which $reach(b)$ is true. We stress that this knowledge can not be represented by applying the Clark completion on the theory, nor by any classical first order theory. This follow from the well-known fact that the transitive closure of a graph is not first-order definable [1].

Applying abduction on this program, one can plan the movements of the robot. Suppose the robot is requested to move from a to b, and passing through c is forbidden. Let $LP = \{r_1, r_2, r_3\} \cup B$, where B is a set of facts with predicate $viable\_straight$ which encode the viable straight ways. Then, each minimal set A of moves that entails $reach(b), \neg reach(c)$, i.e., $reach(b)$ and $\neg reach(c)$ are deducible from $LP \cup A$, represents a path for the robot from a to b that does not pass c. In other words, each minimal abductive explanation of the manifestations $reach(b), \neg reach(c)$ is a possible such path for the robot. (The natural condition of minimality on explanations is discussed below.) However, abduction from this program under classical inference –even from its completion– does not yield the same result. This is verified e.g. by the scenario in which the viable straight ways constitute the following graph:



Then, under logic programming semantics, there is the unique minimal abductive explanation $\{ move(a, b) \}$. This corresponds exactly to the only possible path, which directly leads from a to b. However, no abductive explanation (minimal or not) exists under classical inference, as $\neg reach(c)$ can not be deduced, even from the completion of the program. ∎

Further scenarios highlighting the difference between abduction under logic programming and classical inference from a program (resp. its completion) are described in Example 3 below and in Example 10 (Section 6).

In the context of logic programming, abduction has been first proposed by Kakas and Mancarella [37] and, during the recent years, common interest in the subject has been growing rapidly [14, 41, 37, 34, 17, 16, 68], also for the observation that, compared to deduction, this kind of reasoning has some advantages for dealing with incomplete information [16, 4]. A suggestive application of abduction from logic programs to data dictionary design is presented in [59].

The logic programs from above enjoy the property of being stratifiable [2], and thus has a generally accepted clear semantics. However, the declarative semantics of a logic program with negation is in general not immediately clear; different proposals for such a semantics exist. Thus, abductive reasoning on logic programs is affected by the particular semantics assigned to the program. Two dominant proposals for the semantics of logic programs with negation are the *well-founded* semantics [77] and the *stable model* semantics [25].

The well-founded semantics, differently from the previous proposals [2, 61], assigns a meaning to *every* logic program. The meaning of a program is a unique and always existing (possibly partial) model, which is called *well-founded model*. Intuitively, this semantics follows a "pessimistic" approach declaring false only atoms whose truth can be surely not derived from the program [18]. Inference of a literal from a general logic program under the well-founded semantics is polynomial; hence, this semantics can express only polynomial time database queries [77].

The stable model semantics is closely related to classical non-monotonic logics. Indeed, the semantics of a program is defined by the set of its stable models, where a model is stable if it "regenerates itself" (like the stable expansions of an autoepistemic theory [55]). Since a program has in general several stable models, two inference modalities arise: *brave* inference (a literal is true if it is true in some stable model), and *cautious* inference (a literal is true if it is true in every stable model).

Even if stable model semantics increases the expressivity of well-founded semantics allowing to express NP-hard problems [66], several common situations require the possibility of explicitly encoding disjunctive information, cf. [45]. This observation lead to extend logic programs by permitting disjunction in the consequents (*disjunctive logic programs*). A number of different proposals for the semantics of disjunctive logic programs exist to date [45], among them the minimal model semantics [54], the (disjunctive) stable model semantics [62, 26], and the well-founded semantics [64, 3]. Besides the minimal model semantics (which has been designed for negation-free programs), the stable model semantics is the most commonly acknowledged approach; such semantics allow to express also problems whose complexity lies at the second level of the polynomial hierarchy [21].

Thus, making abduction from logic programs, one should agree on which semantics has to be considered (e.g., well-founded or stable model semantics); moreover, if a "multiple model" (e.g. stable model) semantics is adopted, one should also specify whether the cautious or brave version is chosen.

In this paper, we define a general abduction model for logic programming, where the inference operator (i.e., the semantics to be applied on the program) is not fixed *a priori* but can be specified by the user. In our opinion, this endows our model with a great flexibility, as the appropriate semantics can be chosen for each application domain. For instance, abduction with brave inference seems more appropriate for diagnosis, while the cautious inference operator appears well suited for planning (see Section 3).

Roughly speaking, we describe abduction from logic programs as follows: Given a logic program LP formalizing a particular application domain, a set M of literals describing some manifestations, a set H of atoms containing possible individual hypotheses, and an inference operator $\models$ defining the semantics of the program, find an *explanation* (or *solution*) for M, i.e., a suitable set $S \subseteq H$ such that $LP \cup S \models M$.

For instance, the scenario in Example 1 (Joe's Saturday business) can be represented as an abductive problem where $LP = \{\ go\_fishing \leftarrow is\_saturday \wedge \neg rains,\ is\_saturday \leftarrow\ \}$, $M = \{\neg go\_fishing\}$, $H = \{rains\}$, and the inference operator is $\models_{wf}$ (entailment under well-founded semantics). It is immediately recognized that $S = \{rains\}$ is a solution. Let us consider another example.

Example 3 Imagine an electrical circuit consisting of a simple stove with two hot-plates wired in parallel and a control light, which is on if at least one of the plates is on operation. Each plate has a fuse, and it is known that one of them can not stand much current and will melt if the current gets high, but it is not known which one. Consider the following program P:

$$
\begin{aligned}
melted\_fuse_1 \vee melted\_fuse_2 &\leftarrow high\_current \\
light\_off &\leftarrow melted\_fuse_1 \wedge melted\_fuse_2 \\
light\_off &\leftarrow power\_failure \\
light\_off &\leftarrow broken\_bulb \\
burns\_plate_1 &\leftarrow \neg melted\_fuse_1 \wedge \neg power\_failure \\
burns\_plate_2 &\leftarrow \neg melted\_fuse_2 \wedge \neg power\_failure
\end{aligned}
$$

The first rule states that on high current, a fuse will melt. The second through fourth rule describe situations under which the control light is off, namely, if both fuses are melted, if the power fails, and if the bulb is broken. The last two rules state that a hot-plate burns if there is no power failure and the fuse is not melted.[1]

For example, given the observation $light\_off$, under the stable model semantics we can abduce $power\_failure$ as an explanation under both brave and cautious inference operators, i.e., $S = \{power\_failure\}$ is a solution for both the abductive problems $\mathcal{P} = \langle H, M, P, \models^b_{st}\rangle$ and $\mathcal{P}' = \langle H, M, P, \models^c_{st}\rangle$ where $H = \{power\_failure,\ broken\_bulb,\ high\_current\}$, $M = \{light\_off\}$, and $\models^b_{st}$ and $\models^c_{st}$ denotes the brave and cautious inference operator, respectively.

[1] To keep the example simple, we refrain from modeling switches for the hot-plates.

Indeed, $P \cup \{power\_failure\}$ has the single stable model $M_1 = \{power\_failure,\ light\_off\}$. Similarly, also *broken_bulb* is an explanation of *light_off*, while *high_current* is not.

Given the additional observation $\neg burns\_plate_1$ (i.e., considering the manifestation $M' = \{light\_off,\ \neg burns\_plate_1\}$), we still have *power_failure* as an explanation, but no longer *broken_bulb*. Under brave inference, however, we get that $S' = \{broken\_bulb,\ high\_current\}$ is an explanation (i.e., S' is a solution for $\langle H, M', P, \models^b_{st}\rangle$); this is reasonable in some sense. Notice that under classical inference, *power_failure* is not an explanation, since $P \cup \{power\_failure\} \not\models \neg burns\_plate_1$; however, it is an explanation on the completion of P.[2]

Finally, assume the observations *light_off*, $burns\_plate_1$, and $burns\_plate_2$ are made. The only explanation for that is *bulb_broken* (under cautious as well as brave inference), which is intuitive. Under classical inference, no explanation exists, even from the completed program (notice that abducible atoms should not be completed [14]). ∎

Usually, different abductive solutions may be regarded not as equally appealing. Following Occam's principle of parsimony, one is often willing to prefer as simplest solutions as possible. In particular, solutions should be nonredundant, i.e., an acceptable solution must not contain another solution properly. The property of subset-minimality is the most commonly applied criterion for acceptable abductive solutions, cf. [58, 74, 9, 41]. For example, for the observation *light_off* in the above example, $\{bulb\_broken,\ power\_failure\}$ would be a solution (under both variants of inference). The simpler solution *bulb_broken* (or *power_failure* as well) is clearly preferable. Besides subset-minimality, smallest size of solutions is a commonly applied criterion for acceptable solutions, cf. [58]. We will deal with both criteria.

In the context of abduction, three main decision problems arise: (1) to determine whether an explanation for the given manifestations *exists* at all; (2) to determine whether an individual hypothesis $h \in H$ is *relevant*, i.e., whether it is part of at least one acceptable explanation; (3) to determine whether an individual hypothesis is *necessary*, i.e., whether it occurs in all acceptable explanations. For instance, in the circuit example we have: (1) there exists a solution for the manifestation *light_off*, (2) the atoms *bulb_broken* and *power_failure* are relevant for some solution, (3) none of them (as well as any other hypothesis) is necessary. The complexity of the three decision problems above will be carefully analyzed in the paper.

The contribution of the present paper to abductive logic programming is twofold. First, we formally define a general abductive model for logic programs (informally presented above), in which also some previously proposed abductive framework can be easily modeled. Second, we precisely determine the computational complexity of abductive problems under: (1) the well-founded semantics for normal logic programs [77], (2) the stable model semantics for normal logic programs [25], (3) the stable model semantics for disjunctive logic programs [62], and (4) the minimal model semantics for disjunctive logic programs [54]. For each semantics we shall analyze the complexity of relevance, necessity and existence of abductive solutions, by considering also the case where solutions must satisfy some minimality criterion.

[2] Although the completion has been defined in [13] only for nondisjunctive logic programs, it is clear how to complete this particular program in the spirit of [13].

This constitutes the main part of the present paper, and provides many new and interesting results:

1. Even if stable-brave deduction is harder than well-founded deduction (the former is NP-complete while the latter is polynomial), the complexity of stable-brave abduction is essentially the same as of well-founded abduction (the various abductive problems under these semantics have a complexity ranging from P to Σ_2^P).

2. While the complexities of stable-brave and stable-cautious deduction lie at the same level in the polynomial hierarchy, the complexity of stable-cautious abduction is located one level higher than the one of stable-brave abduction, and thus most likely strictly harder.

3. Allowing disjunction in the rule heads further increases the complexity of the abductive problems. As for normal logic programs, cautious reasoning reveals to be harder than brave reasoning. Moreover, abduction under minimal model and stable model semantics have the same computational complexity.

4. The complexities of the various abductive reasoning tasks under the considered semantics range over the first four levels of the polynomial hierarchy. Most complexity classes from there, including the classes Σ_k^P, Π_k^P, $\mathrm{D_k^P}$, and $\Delta_k^P[O(\log n)]$ have abductive reasoning tasks complete for them. Thus, our analysis exhibits practical problems complete even for complexity classes for which few complete problems are known to date (e.g., the complexity class Σ_4^P).

The complexity results above refer to abductive problems on propositional logic programs. For function-free first order logic programs, the data-complexity of abduction (i.e., the program except for ground facts is fixed) coincides with the complexity of abduction from propositional logic programs. (In the nonfixed case, the complexity is expected to be exponentially higher.)

We first present our model for abduction from propositional logic programs and determine its complexity; after that, we provide a generalization of this model to allow for abduction from function-free first order logic programs and address the resulting complexity. Notice that, extending our work, complexity results for abduction from logic programs in the presence of functions appear in [47].

Our results impact on other work on logic programming as follows. For example, from the complexity classification, we know whether an efficient (i.e., polynomial time) transformation from an abduction problem to another reasoning problem whose complexity is known is possible or not. In particular, it tells us whether abduction can be efficiently transformed to deduction in logic programming, and thus whether a search for an efficient embedding of abductive logic programming into deductive logic programming is reasonable. Moreover, by the level of the hierarchy, we know how many sources of complexity an abduction problem bears, and which kind of backtracking algorithm for solving it will be suitable. (See also Section 8.)

The sequel of the paper is organized as follows. Section 2 contains some preliminaries on both logic programming and complexity theory. Section 3 formally defines our abduction model in the propositional case. The complexity of the various abductive problems for normal logic programs is determined in Section 4. Section 5 deals with the extension of the abduction model by allowing for disjunctive logic programs. Section 6 generalizes the abductive model to the case of function-free first-order logic programs and discusses the data-complexity. Section 7 overviews some related works. Finally, Section 8 gives our conclusions and summarizes the complexity results determined in the paper.

2 Preliminaries and Notation

2.1 Logic Programming

The section recalls the basic concepts of (propositional) logic programming. The syntax of logic programs is given first; then stable and well-founded semantics [25, 77] are provided. Stratified programs [2] and their properties are finally presented.

Let a set V of propositional variables be given. A *positive literal* (*atom*) is an element $a \in V$; a *negative literal* is the negation $\neg a$ of an atom. A *program clause* (or *rule*) r is

$$a \leftarrow b_1 \wedge \cdots \wedge b_k \wedge \neg b_{k+1} \wedge \cdots \wedge \neg b_m, \qquad m \geq 0$$

where $a, b_1, \cdots, b_k$ are positive literals and $\neg b_{k+1}, \cdots, \neg b_m$ are negative literals. The atom a is called the *head* of r, while the conjunction $b_1 \wedge \cdots \wedge b_k \wedge \neg b_{k+1} \wedge \cdots \wedge \neg b_m$ is its *body*. A (propositional) *logic program* is a finite set LP of program clauses; it is *positive* (or $\neg$*-free*), if no negative literal occurs in it. As usual, it is tacitly assumed that V is, unless specified otherwise, the set of atoms occurring in LP.

A (total) *interpretation* is a subset of V. A positive literal a (resp. a negative literal $\neg a$) is true with respect to an interpretation I if $a \in I$ (resp. $a \notin I$); otherwise it is false. A rule is satisfied (or is true) w.r.t. I if its head is true or its body is false w.r.t. I. An interpretation I is a model for a logic program LP if it satisfies all rules in LP.

2.1.1 Stable model semantics

Given a logic program LP and an interpretation I, the *Gelfond-Lifschitz transform* of LP with respect to I is the logic program LP^I consisting of all rules $a \leftarrow b_1 \wedge \cdots \wedge b_k$ such that

$$(1)\ a \leftarrow b_1 \wedge \cdots \wedge b_k \wedge \neg b_{k+1} \wedge \cdots \wedge \neg b_m \in LP \quad \text{and} \quad (2)\ b_i \notin I, \text{ for all } k < i \leq m.$$

Notice that $\neg$ does not occur in LP^I, i.e., it is a Horn program. Each Horn program P has a least model (i.e., a model included in every model), denoted by $lm(P)$.
An interpretation I is called a *stable model* of LP [25] iff $I = lm(LP^I)$. The collection of all stable models of LP is denoted by $\mathrm{STM}(LP)$ (i.e., $\mathrm{STM}(LP) = \{I \mid I = lm(LP^I)\}$).

Example 4 Consider the following program LP:

$$a \leftarrow \neg b \qquad b \leftarrow \neg a \qquad c \leftarrow a \qquad c \leftarrow b$$

The stable models of LP are $M_1 = \{a, c\}$ and $M_2 = \{b, c\}$. Indeed, by definition of Gelfond-Lifschitz transform, $LP^{M_1} = \{\ a \leftarrow,\ c \leftarrow a,\ c \leftarrow b\ \}$ and $LP^{M_2} = \{\ b \leftarrow,\ c \leftarrow a,\ c \leftarrow b\ \}$; thus, it is immediately recognized that $lm(LP^{M_1}) = M_1$ and $lm(LP^{M_2}) = M_2$. ∎

The example shows that a logic program may have several stable models (there can also be none). The modalities of brave and cautious reasoning are used to handle this.

Brave reasoning (or *credulous reasoning*) infers that a literal Q is true in LP (denoted $LP \models^b_{st} Q$) iff Q is true with respect to M for some $M \in \mathrm{STM}(LP)$.

Cautious reasoning (or *skeptical reasoning*) infers that a literal Q is true in LP (denoted $LP \models^c_{st} Q$) iff (1) Q is true with respect to M for all $M \in \mathrm{STM}(LP)$ and (2) $\mathrm{STM}(LP) \neq \emptyset$.[3]

The inference relations $\models^b_{st}$ and $\models^c_{st}$ extend to sets of literals as usual.

Example 5 For the program LP of Example 4, a, b and c are brave inferences ($LP \models^b_{st} \{a, b, c\}$); the only cautious inference is c ($LP \models^c_{st} c$). ∎

2.1.2 Well-founded semantics

A *partial (3-valued) interpretation* I is a consistent set of literals over V. Each atom a is considered to be *true* in I if $a \in I$, *false* in I if $\neg a \in I$, and *undefined* in I otherwise.

Let LP be a program and I be a (partial) interpretation. A set N of atoms is an *unfounded set* of LP w.r.t. I if each atom a in N satisfies the following condition: For each rule $r \in LP$ whose head is a at least one of the following holds: (1) some literal in the body of r is false w.r.t. I, or (2) some positive literal of the body of r belongs to N.
Unfounded sets are closed under unions; the union of all unfounded sets w.r.t. I is called the *Greatest Unfounded Set* for LP with respect to I and denoted by $GUS_{LP}(I)$.

The well-founded model of LP is defined as the least fixed point of an operator which infers truth of atoms by classical immediate consequence and falsity using the Greatest Unfounded Set. Formally, define for program LP and interpretation I the following transformations:

1. $T_{LP}(I) = \{Q \mid \exists r \in LP \ s.t. \ Q \ is \ the \ head \ of \ r \ and \ the \ body \ of \ r \ is \ true \ w.r.t. \ I\}$
2. $W_{LP}(I) = T_{LP}(I) \cup \{\neg y \mid y \in GUS_{LP}(I)\}$

Consider the sequence $W_0 = \emptyset$, $W_n = W_{LP}(W_{n-1})$. The sequence finitely converges to a limit W_λ. W_λ is called the *well-founded model* of LP [77], which is denoted by $WF(LP)$. Observe that, since the positive inferences of W_{LP} are done by using the immediate consequence operator T_{LP}, the well-founded model is *supported*, i.e., a positive literal Q is in $WF(LP)$ only if there is a rule with head Q whose body is true w.r.t. $WF(LP)$. The well-founded model has the following important property.

Proposition 1 [77] *Given a logic program LP, the well-founded model $WF(LP)$ can be computed in polynomial time.*

[3] Note that (2), i.e., consistency of LP under stable semantics, is usually not requested in the definition of cautious consequence; we denote this version of cautious consequence by $\models^{cls}_{st}$. As shown in the next section, however, consistency is needed for abduction.

Example 6 Consider the following program LP:

$$p \leftarrow q \wedge \neg r \qquad q \leftarrow \neg p \qquad s \leftarrow r$$

We have ($W_0 = \emptyset$):

$$T_{LP}(W_0) = \emptyset; \quad GUS_{LP}(W_0) = \{r, s\}; \quad W_1 = W_{LP}(W_0) = \{\neg r, \neg s\};$$
$$T_{LP}(W_1) = \emptyset; \quad GUS_{LP}(W_1) = \{r, s\}; \quad W_2 = W_{LP}(W_1) = \{\neg r, \neg s\};$$

and hence $W_\lambda = W_2 = \{\neg r, \neg s\}$ is the well-founded model of LP. Thus, r and s are false in the well-founded semantics, while p and q are undefined. Indeed, while it is intuitive that r and s should be false, it is unclear which value should be assigned to p and q. On this program the well-founded semantics behaves better than the stable one which says nothing about the truth valuation of the literals as the program has no stable model. On the contrary, on program LP of Example 4 the well-founded model is empty and, differently from the stable semantics, does not entail truth of c; however, several researchers have claimed that c should be entailed from LP [5, 3]. ∎

The well-founded semantics regards $WF(LP)$ as the intended meaning of the program LP. Accordingly, it infers that a literal Q is true in LP, denoted $LP \models_{wf} Q$, iff it is true in the well-founded model (i.e., $Q \in WF(LP)$). This extends to inference of a set of literals as usual.

2.1.3 Stratified programs

Although stable and well-founded semantics are different in general, they coincide on the important class of stratified logic programs [2].

A logic program LP is *stratified* iff it is possible to partition the set of its atoms into strata $\langle S_1, \ldots, S_r \rangle$, such that for every rule $a \leftarrow b_1 \wedge \cdots \wedge b_k \wedge \neg c_1 \wedge \cdots \wedge \neg c_m$ in LP the following holds, where $\mathrm{Strat}(x) = i$ iff $x \in S_i$:

$$\mathrm{Strat}(b_j) \leq \mathrm{Strat}(a), \text{ for all } 1 \leq j \leq k, \quad \text{and} \quad \mathrm{Strat}(c_l) < \mathrm{Strat}(a), \text{ for all } 1 \leq l \leq m.$$

Proposition 2 *[77] Let LP be a stratified logic program. Then, the well-founded model of LP is total, and it coincides with the unique stable model of LP.*

As a consequence, $\models_{wf}$, $\models^b_{st}$, and $\models^c_{st}$ are all equivalent on stratified programs.

2.2 Complexity Theory

For NP-completeness and complexity theory, cf. [56]. The classes Σ^P_k, Π^P_k and Δ^P_k of the Polynomial Hierarchy (PH) (cf. [75]) are defined as follows:

$$\Delta^P_0 = \Sigma^P_0 = \Pi^P_0 = \mathrm{P} \quad \text{and for all } k \geq 1, \quad \Delta^P_k = \mathrm{P}^{\Sigma^P_{k-1}}, \ \Sigma^P_k = \mathrm{NP}^{\Sigma^P_{k-1}}, \ \Pi^P_k = \text{co-}\Sigma^P_k.$$

In particular, NP $= \Sigma^P_1$, co-NP $= \Pi^P_1$, and $\Delta^P_2 = \mathrm{P}^{\mathrm{NP}}$. Here P^C and NP^C denote the classes of problems that are solvable in polynomial time on a deterministic (resp. nondeterministic)

Turing machine with an oracle for any problem π in the class C. The oracle replies to a query in unit time, and thus, roughly speaking, models a call to a subroutine for π that is evaluated in unit time. If C has complete problems, then instances of any problem π' in C can be solved in polynomial time using an oracle for any C-complete problem π, by transforming them into instances of π; we refer to this by stating that an oracle for C is used. Notice that all classes C considered here have complete problems.

The classes Δ_k^P, $k \geq 2$, have been refined by the class $\Delta_k^P[O(\log n)]$, in which the number of calls to the oracle is in each computation bounded by $O(\log n)$, where n is the size of the input. The class $\mathrm{D}_\mathrm{k}^\mathrm{P}$, which is defined as the class of problems that consist of the conjunction of two (independent) problems from Σ_k^P and Π_k^P, respectively, is considered to be further restricted in computational power. Notice that for all $k \geq 1$,

$$\Sigma_k^P \subseteq \mathrm{D}_\mathrm{k}^\mathrm{P} \subseteq \Delta_{k+1}^P[O(\log n)] \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \subseteq \mathrm{PSPACE};$$

each inclusion is widely conjectured to be strict. Note that, by the rightmost inclusion, all these classes contain only problems that are solvable in polynomial space. They allow, however, a finer grained distinction between NP-hard problems that are in PSPACE.

The above complexity classes have complete problems under polynomial-time transformations involving quantified Boolean formulas (QBFs). A QBF is an expression of the form

$$Q_1 X_1 Q_2 X_2 \cdots Q_k X_k\, E, \quad k \geq 1, \tag{1}$$

where E is a Boolean expression whose atoms are from pairwise disjoint nonempty sets of variables $X_1, \ldots, X_k$, and the Q_i's are alternating quantifiers from $\{\exists, \forall\}$, for all $i = 1, \ldots, k$. If $Q_1 = \exists$ the we say the QBF is k-existential, otherwise it is k-universal. Validity of QBFs is defined in the obvious way by recursion to variable-free Boolean expressions. We denote by $\mathrm{QBF}_{k,\exists}$ (resp., $\mathrm{QBF}_{k,\forall}$) the set of all valid k-existential (resp., k-universal) QBFs (1).

Given a k-existential QBF Φ (resp. a k-universal QBF Ψ), deciding whether $\Phi \in \mathrm{QBF}_{k,\exists}$ (resp. $\Psi \in \mathrm{QBF}_{k,\forall}$), is a classical Σ_k^P-complete (resp. Π_k^P-complete) problem; deciding the conjunction $(\Phi \in \mathrm{QBF}_{k,\exists}) \wedge (\Psi \in \mathrm{QBF}_{k,\forall})$ is complete for $\mathrm{D}_\mathrm{k}^\mathrm{P}$. Deciding, given an arbitrary number of QBFs of form (1), whether the number of valid QBFs among them is odd is complete for $\Delta_{k+1}^P[O(\log n)]$. The problems remain as hard under the following restrictions: (1) E in (1) is in conjunctive normal form and each clause contains three literals (3CNF) if $Q_k = \exists$, and (2) E is in disjunctive normal form and each monom contains three literals (3DNF) if $Q_k = \forall$ [76].

3 Abduction Model

In this section, we describe our formal model for abduction from logic programs and state the main decisional reasoning tasks for abductive reasoning. The abduction model is defined here for propositional programs; the generalization to logic programs with variables is given in Section 6.

3.1 Logic Programming Abduction Problems

Definition 1 *Let V be a set of propositional atoms. A logic programming abduction problem (LPAP) $\mathcal{P}$ over V consists of a tuple $\langle H, M, LP, \models \rangle$, where $H \subseteq V$ is a finite set of hypotheses, $M \subseteq V \cup \{\neg v \mid v \in V\}$ is a finite set of manifestations, LP is a propositional logic program on V, and $\models$ is an inference operator.* ∎

For simplicity, we suppose throughout the rest of the paper that V is implicitly defined as the set of propositional atoms appearing in $\mathcal{P}$, which we denote by $V_{\mathcal{P}}$, and omit the phrase "over V".

Remark: This model does not include specification of constraints on the logic program. However, the most used form of constraint $\leftarrow a_1 \wedge \ldots \wedge a_n$ which means that $a_1, \ldots, a_n$ may not be true simultaneously, can be easily simulated in our model, by adding a rule $abs \leftarrow a_1 \wedge \ldots \wedge a_n$ to the logic program, where abs is a new propositional atom, and adding $\neg abs$ to the manifestations.

Definition 2 *Let $\mathcal{P} = \langle H, M, LP, \models \rangle$ be a LPAP, and let $S \subseteq H$. Then, S is a solution (or explanation) to $\mathcal{P}$ iff $LP \cup S \models M$.*

The set of all solutions to $\mathcal{P}$ is denoted by $Sol(\mathcal{P})$. ∎

In this paper we consider abductive problems for the inference operators $\models_{wf}$, $\models_{st}^{b}$, and $\models_{st}^{c}$. Note that, in the definition of cautious abduction, we have not used the classical skeptical operator (a literal is true if it is true in all stable models), but we have adopted the inference operator $\models_{st}^{c}$ which requires additionally that the program has (at least) one stable model. The reason for that is the need to guarantee that $LP \cup S$ is "consistent"; otherwise, any set of atoms S making the logic program inconsistent would be an explanation of all possible manifestations.

Example 7 Consider the *LPAP* $\mathcal{P} = \langle \{d, b\}, \{c\}, LP, \models_{st}^{cls} \rangle$, where $LP = \{a \leftarrow b \wedge \neg a\ , c \leftarrow d\ \}$ and $\models_{st}^{cls}$ is classical stable cautious inference. One reasonably expects that the observation of c abduces the atom d. Nevertheless, since $LP \cup \{b\}$ has no stable model and, as a consequence, $LP \cup \{b\} \models_{st}^{cls} c$, the set $\{b\}$ is a solution to $\mathcal{P}$. On the contrary, the problem does not arise with the inference operator $\models_{st}^{c}$. Indeed, if we consider the *LPAP* $\mathcal{P}' = \langle \{d, b\}, \{c\}, LP, \models_{st}^{c} \rangle$, then $\{b\}$ is not a solution to $\mathcal{P}'$, as this operator requires the existence of a stable model (ensuring the consistency of the program). ∎

The explicit introduction of the inference operator in the definition of abduction problem endows our abduction model with a great flexibility, since the appropriate semantics can be adopted on each application domain. This is particularly important in logic programming, as no semantics is generally accepted as the best one for *all* logic programs; rather, it is acknowledged that the right semantics must be chosen on the particular domain. It is interesting to briefly discuss on the applicability on different domains of brave and cautious reasoning, respectively. We argue that brave inference is well suited for diagnosis, while cautious inference is adequate for planning. In fact, consider a system represented by a logic program LP with stable model semantics. If the system is solicited by some input, represented by adding a set

A of atoms to LP, then each stable model of $LP \cup A$ is a possible evolution of the system, i.e., each stable model represents a possible reaction of the system to A.

Diagnosis consists, loosely speaking, in deriving from an observed system state (characterized by the truth of a set F of facts), a suitable input A which caused this evolution. Now, since *each* stable model of $LP \cup A$ is a possible evolution of the system with input A, we can assert that A is a possible input that caused F if $LP \cup A \models^b_{st} F$. Thus, diagnostic problems can be naturally represented by abductive problems with brave inferences.

Suppose now that we want that the system is in a certain state (described by a set F of facts), and we have to determine the "right" input that enforces this state of the system (planning). In this case it is not sufficient to choose an input A such that F is true in some possible evolution of the system; rather, we look for an input A such that F is true in *all* possible evolutions, as we want be sure that the system reacts in that particular way. In other words, we look for A such that $LP \cup A \models^c_{st} F$. Hence, planning activities can be represented by abductive problems with cautious inferences.

The next section shows that abduction under cautious inference has most likely a higher complexity than abduction under brave modality; thus, from the above point of view, planning is most likely harder than diagnosis.

The following properties of a hypothesis in a *LPAP* $\mathcal{P}$ are of natural interest with respect to computing abductive solutions.

Definition 3 *Let $\mathcal{P} = \langle H, M, LP, \models \rangle$ be a LPAP and $h \in H$. Then, h is relevant for $\mathcal{P}$ iff $h \in S$ for some $S \in Sol(\mathcal{P})$, and h is necessary for $\mathcal{P}$ iff $h \in S$ for every $S \in Sol(\mathcal{P})$.*

We refer to the property opposite to necessity also as dispensability (cf. [33]). Now, the main decisional problems in abductive reasoning are as follows: Given a *LPAP* $\mathcal{P} = \langle H, M, LP, \models \rangle$,

1. does there exist a solution for $\mathcal{P}$? (*Consistency*)

2. is a given hypothesis $h \in H$ relevant for $\mathcal{P}$, i.e., does h contribute to some solution of $\mathcal{P}$? (*Relevance*)

3. is a given hypothesis $h \in H$ necessary for $\mathcal{P}$, i.e., is h contained in all solutions of $\mathcal{P}$? (*Necessity*)

It is common in abductive reasoning to prune the set of all solutions and to focus, guided by some principle of solution preference, on a set of preferred solutions. Two important notions of preferred solutions are irredundant solutions, i.e., solutions which do not contain any other solution properly (cf. [58, 74, 41]), and solutions of smallest cardinality (cf. [58]). We refer to the former solutions of a *LPAP* P as $\subseteq$-solutions and to the latter as $\leq$-solutions, and denote their collection by $Sol_{\subseteq}(\mathcal{P})$ and $Sol_{\leq}(\mathcal{P})$, respectively.

The properties of relevance and necessity of a hypothesis for a *LPAP* $\mathcal{P}$ with respect to $Sol_{\subseteq}(\mathcal{P})$ (resp. $Sol_{\leq}(\mathcal{P})$) rather than $Sol(\mathcal{P})$, are referred to as $\subseteq$-relevance and $\subseteq$-necessity (resp. $\leq$-relevance and $\leq$-necessity).

The complexity of the problems consistency, relevance, and necessity under inference operators $\models_{wf}$, $\models^b_{st}$, and $\models^c_{st}$, combined with $\subseteq$-solutions and $\leq$-solutions, is studied in Section 4.

3.2 Generic Transformations Between Related Problems

Prior to the analysis of abduction problems under particular semantics, we note some general reducibilities among abduction problems and deductive inference problems. One the one hand, the reducibilities show us how different reasoning tasks are related; in fact, the relationship is close in some cases. On the other hand, they help to simplify the proofs of several complexity results. Moreover, we point out relationships between abduction using arbitrary and $\subseteq$-solutions.

3.2.1 Solution verification

It is clear that entailment of a conjunction of literals and solution verification are equivalent. We state this for later reference.

Proposition 3 (Equivalence of solution verification and entailment) *Deciding, given a LPAP $\mathcal{P} = \langle H, M, LP, \models\rangle$ and a set $S \subseteq H$, whether $S \in Sol(\mathcal{P})$ is w.r.t. polynomial-time transformation equivalent to the entailment problem $LP1 \models \gamma$, where $LP1$ is a logic program and γ is a conjunction of literals.*

3.2.2 Consistency

Deductive entailment is a subtask inherent to abduction. This problem can be reduced to consistency by means of a simple generic transformation, which gives us the following result.

Proposition 4 (Entailment reduces to consistency) *Entailment of a conjunction γ of literals from a logic program LP, $LP \models \gamma$, reduces in polynomial time to consistency under $\models$.*

Proof. Suppose we want to decide whether $LP \models \gamma$, where $\gamma = q_1 \wedge \cdots \wedge q_k$ and the q_i are literals. Then we consider the *LPAP* $\mathcal{P} = \langle\emptyset, \{q_1, \ldots, q_k\}, LP, \models\rangle$. The only possible solution to $\mathcal{P}$ is $\emptyset$. Obviously, $\emptyset \in Sol(\mathcal{P})$ if an only if $LP \models \gamma$. ∎

Thus, the complexity of entailment is a –naturally expected– lower complexity bound for consistency. As will be seen, it will be complemented in some cases by a matching upper bound.

Notice that, as an immediate corollary to Propositions 3 and 4, solution verification easily reduces to consistency in polynomial time.

3.2.3 Relevance

It turns out that the complexities of consistency and relevance always coincide. In fact, this can be established by means of generic reductions between the problems.

Proposition 5 (Equivalence of consistency and relevance) *Problems consistency and relevance are equivalent w.r.t. polynomial-time transformation.*

Proof. *Consistency can be reduced to relevance.* Let $\mathcal{P} = \langle H, M, LP, \models\rangle$ be an *LPAP*. Suppose we want to decide whether $Sol(\mathcal{P}) \neq \emptyset$. Then we consider $\mathcal{P}' = \langle H \cup \{q\}, M, LP, \models\rangle$ where q is a fresh atom. Now, $S \in Sol(\mathcal{P})$ if and only if $S \cup \{q\} \in Sol(\mathcal{P}')$, for every $S \subseteq H$. Hence, $\mathcal{P}$ is consistent if and only if q is relevant for $\mathcal{P}'$.

Relevance can be reduced to consistency. Let $\mathcal{P} = \langle H, M, LP, \models\rangle$ be an *LPAP* and $h \in H$. Suppose we want to decide whether there is some $S \in Sol(\mathcal{P})$ such that $h \in S$. Then we consider $\mathcal{P}' = \langle H, M, LP \cup \{h\}, \models\rangle$. Now, h is relevant for $\mathcal{P}$ if and only if $\mathcal{P}'$ is consistent. To see this, suppose first that h is relevant for $\mathcal{P}$. Hence, there exists an $S \subseteq H$ such that $h \in S$ and $LP \cup S \models M$. Clearly, $LP \cup \{h\} \cup S \models M$, and therefore $S \in Sol(\mathcal{P}')$. Thus, $\mathcal{P}'$ is consistent. Conversely, suppose that $\mathcal{P}'$ is consistent, i.e., there is some $S \in Sol(\mathcal{P}')$. Since $LP \cup \{h\} \cup S \models M$, we have $S \cup \{h\} \in Sol(\mathcal{P})$, which means that h is relevant for $\mathcal{P}$. ∎

This relationship allows to derive complexity results for relevance immediately from complexity results for consistency.

3.2.4 Necessity

Necessity is related to consistency similarly as relevance. In fact, necessity turns out to be equivalent to inconsistency, the complement of consistency.

Proposition 6 (Equivalence of inconsistency and necessity) *Problems inconsistency and relevance are equivalent w.r.t. polynomial-time transformation.*

Proof. *Inconsistency can be reduced to necessity* Let $\mathcal{P} = \langle H, M, LP, \models\rangle$ be an *LPAP*. Suppose we want to decide whether $Sol(\mathcal{P}) = \emptyset$. Then we consider as above $\mathcal{P}' = \langle H \cup \{q\}, M, LP, \models\rangle$ where q is a fresh atom. Now, q is necessary, i.e., q occurs in every solution $S \in Sol(\mathcal{P})$ if and only if $Sol(\mathcal{P}) = \emptyset$, i.e., $\mathcal{P}$ is inconsistent.

Necessity can be reduced to inconsistency. Let $\mathcal{P} = \langle H, M, LP, \models\rangle$ be an *LPAP* and $h \in H$. Suppose we want to decide whether $h \in S$ for all $S \in Sol(\mathcal{P})$. Then we consider $\mathcal{P}' = \langle H \setminus \{h\}, M, LP, \models\rangle$. Now, for every inference relation $\models$ from $\models_{wf}$, $\models_{st}^{b}$, and $\models_{st}^{c}$, we clearly have that $\mathcal{P}'$ is inconsistent if and only if h is necessary for $\mathcal{P}$. ∎

Note that the two reductions obviously hold for every "reasonable" inference relation; thus, inconsistency and necessity are equivalent in any reasonable inference setting.

3.2.5 Relevance and necessity under $\subseteq$-solutions

Using minimal solutions (i.e., $\subseteq$-solutions) instead of arbitrary solutions has different effects to relevance and necessity. For the former problem, we note the following proposition.

Proposition 7 *Suppose that consistency for a class of LPAPs is in the complexity class C. Then, $\subseteq$-relevance is in* NP^{C}.

Proof. That a hypothesis $h \in H$ is $\subseteq$-relevant for $\mathcal{P} = \langle H, M, LP, \models\rangle$ can be nondeterministically proved as follows. Guess a minimal solution S so that $h \in S$. (That is, guess $S' \subseteq H$ and consider $S = S' \cup \{h\}$.) The guess is verified by testing (1) $S \in \mathit{Sol}(\mathcal{P})$ and (2) there is no $S' \subset S$ such that $S' \in \mathit{Sol}(\mathcal{P})$. Since solution verification reduces efficiently to consistency (cf. Propositions 3 and 4), it follows from the hypothesis that (1) can be decided in polynomial time with a call to an oracle for consistency. Test (2) can be decided with checking whether for each $h' \in S$, the *LPAP* $\langle S \setminus \{h'\}, M, LP, \models\rangle$ is inconsistent. By hypothesis, each such test is polynomial with an oracle for consistency. Therefore, (1) and (2) can be decided in polynomial time with an oracle for consistency. It follows that $\subseteq$-relevance is in NP^C. ∎

There is a simple relationship between necessity on arbitrary and $\subseteq$-solutions, respectively.

Proposition 8 *Let $\mathcal{P} = \langle H, M, LP, \models\rangle$ be a LPAP and let $h \in H$. Then, h is $\subseteq$-necessary for $\mathcal{P}$ iff h is necessary for $\mathcal{P}$.*

From this observation, we do not deal with $\subseteq$-necessity explicitly in our analysis.

4 Complexity Results

In this section, we determine the complexity of the main reasoning tasks on the abduction model from the previous section. In particular, we study the complexity of abduction under well-founded, stable cautious, and stable brave semantics in the specified order. For each semantics each abductive problem is first analyzed in the case of no preference criterion and then under both the subset-minimality and the smallest-cardinality preference principle.

Several hardness results will be derived by reductions of validity testing for QBFs. The reduction will often use the rewriting of a boolean expression E in CNF or DNF into a set of propositional programming rules. Before starting the complexity analysis, we fix some helpful notation.

For a Boolean formula E in CNF (resp. DNF), we denote by $\overline{E}$ the DNF (resp. CNF) expression obtained by applying De Morgan's rule to $\neg E$ and cancelling all double negations. For instance, given $E = (x_1 \vee \neg x_2) \wedge x_3$, $\overline{E} = (\neg x_1 \wedge x_2) \vee \neg x_3$. Note that $\overline{\overline{E}} = E$. Furthermore, given a Boolean expression E in CNF, we define $transf(E)$ as follows:

$$\begin{aligned} transf(E) = \{ \quad & abs \leftarrow b_1 \wedge \cdots \wedge b_k \wedge \neg a_1 \wedge \cdots \wedge \neg a_l \;\mid \\ & \qquad a_1 \vee \cdots \vee a_l \vee \neg b_1 \vee \cdots \vee \neg b_k \in E \;\; (k, l \geq 0) \; \} \end{aligned}$$

Moreover, given a set Y of propositional variables, we shall denote by Y' the set $\{y' \mid y \in Y\}$.

Let X be the set of propositional atoms appearing in E. Each truth assignment I to the atoms in X can be naturally seen as the subset $X_I \subseteq X$ of the atoms true w.r.t. I (i.e., $X_I = \{x \in X \mid I(x) = \mathit{true}\}$). Observe that for each $X1 \subseteq X$, the program $transf(E) \cup X1$ is stratified; thus, by Proposition 2, it has a total well-founded model which coincides with its unique stable model.

In $transf(E)$, abs (standing for "absurd") represents falsity: given a truth assignment I for the atoms in a CNF expression E, a clause

$$a_1 \vee \cdots \vee a_l \vee \neg b_1 \vee \cdots \vee \neg b_k \in E$$

is satisfied by I iff abs is not derivable from the corresponding rule

$$abs \leftarrow b_1 \wedge \cdots \wedge b_k \wedge \neg a_1 \wedge \cdots \wedge \neg a_l \ \in transf(E) \cup X_I$$

(under well-founded or stable semantics). Thus, I is a model of E if and only if abs is not deducible from $X_I \cup transf(E)$. (Note that since the program is stratified, deducibility under well-founded, stable brave, and stable cautious semantics coincide).

For instance, consider the simple CNF expression $E = z_1 \vee z_2 \vee \neg z_3$. Then,

$$transf(E) = \{abs \leftarrow z_3 \wedge \neg z_1 \wedge \neg z_2 \ \}$$

It is immediately recognized that a truth assignment I to the atoms from $Z = \{z_1, z_2, z_3\}$ satisfies E if and only if $transf(E) \cup Z_I \models_{wf} \neg abs$ (equivalently, $transf(E) \cup Z_I \models^b_{st} \neg abs$, or $transf(E) \cup Z_I \models^c_{st} \neg abs$). Thus, the above stated correspondence can be profitably used to check the validity of QBFs. For instance, the formula $\exists Z E$ is valid (i.e., it is in $\mathrm{QBF}_{1,\exists}$) iff there exists $Z1 \subseteq Z$ such that $transf(E) \cup Z1 \models_{wf} \neg abs$ (equivalently, $transf(E) \cup Z1 \models^b_{st} \neg abs$, or $transf(E) \cup Z1 \models^c_{st} \neg abs$).

4.1 Well-Founded Semantics

4.1.1 No preference

From Proposition 1 it is immediate that checking whether a set S is a solution to a *LPAP* $\mathcal{P} = \langle H, M, LP, \models_{wf} \rangle$ is possible in polynomial time. On the other hand, this problem contains inference of an atom from a negation-free logic program (i.e., a Horn program) as a subproblem, which is well-known to be logspace-hard for P (cf. [78, 56]). Thus, by Proposition 3, we conclude

Proposition 9 *Let $\mathcal{P} = \langle H, M, LP, \models_{wf} \rangle$ be a LPAP. Deciding if $S \subseteq H$ fulfills $S \in Sol(\mathcal{P})$ is logspace-complete for* P*.*

While this is a positive result, checking the consistency of a *LPAP* is most likely intractable.

Theorem 1 *To decide if $Sol(\mathcal{P}) \neq \emptyset$ for a given LPAP $\mathcal{P} = \langle H, M, LP \models_{wf} \rangle$ is* NP*-complete.* NP*-hardness holds even if LP is a stratified logic program.*

Proof. NP-*Membership.* A guess $S \subseteq V_{\mathcal{P}}$ can be verified in polynomial (Proposition 9).
NP-*Hardness.* We transform deciding $\exists X E \in \mathrm{QBF}_{1,\exists}$, where E is in 3CNF, into checking $Sol(\mathcal{P}) \neq \emptyset$ for a *LPAP* $\mathcal{P}$.

Consider the *LPAP* $\mathcal{P} = \langle X, \{\neg abs\}, LP, \models_{wf} \rangle$, where $LP = transf(E)$.

Recall that, for each $X1 \subseteq X$, $LP \cup X1$ is stratified; hence, by Proposition 2 it has a total well-founded model. As a consequence, given $X1 \subseteq X$, each $x_i \in X$ is either true or false w.r.t. $WF(LP \cup X1)$. Now, consider the set of logic programs $\mathcal{L} = \{LP \cup X1 \mid X1 \subseteq X\}$. Because of the specific form of LP, there is a 1-1 correspondence between the possible truth assignments to the atoms in X, and the set of well-founded models of the programs in $\mathcal{L}$ (the image of a truth assignment I is $WF(LP \cup X_I)$). Furthermore, it is easy to see that E is satisfied by an assignment I if and only if $WF(LP \cup X_I) \models \neg abs$ (recall that the well-founded model is supported). Hence, there exists an assignment I satisfying E iff there exists $X1 \subseteq X$ such that $WF(LP \cup X1) \models \neg abs$. In other words, $\exists X E$ is valid iff $Sol(\mathcal{P}) \neq \emptyset$. Checking if $\Phi = \exists X E \in \mathrm{QBF}_{1,\exists}$ is NP-hard, thus deciding $Sol(\mathcal{P}) \neq \emptyset$ is NP-hard as well. Moreover, since LP is stratified, we have also shown that NP-hardness holds even if the logic program LP from $\mathcal{P}$ is stratified. ∎

As shown in Section 3.2, problem relevance has the same complexity as consistency.

Theorem 2 *Deciding if a given hypothesis is relevant for a given LPAP $\mathcal{P} = \langle H, M, LP, \models_{wf}\rangle$ is* NP*-complete.* NP*-hardness holds even if LP is a stratified logic program.*

Proof. The result for the general case is immediate from Proposition 5 and Theorem 1. Hardness for stratified logic program holds as consistency is already NP-hard in that case, and the reduction from consistency to relevance in the proof of Proposition 5 maps a stratified logic program to a stratified logic program. ∎

Finally, we state the result for necessity, which has complementary complexity of consistency.

Theorem 3 *Deciding if a given hypothesis is necessary for a given LPAP $\mathcal{P} = \langle H, M, LP, \models_{wf}\rangle$ is* co-NP*-complete.* co-NP*-hardness holds even if LP is a stratified logic program.*

4.1.2 $\subseteq$-preference with $\models_{wf}$

Now we determine the complexity of the main reasoning tasks if acceptable solutions must be irredundant, i.e., minimal with respect to ordinary set inclusion. Intuitively, this requirement makes solution verification harder, since, given a candidate S for a solution, current methods will have to inspect an exponential number of subsets in the worst case to prove that no solution properly included in S exists.

Theorem 4 *Deciding if a given hypothesis is $\subseteq$-relevant for a LPAP $\mathcal{P} = \langle H, M, LP, \models_{wf}\rangle$ is Σ_2^P-complete. Σ_2^P-hardness holds even if LP is a stratified logic program.*

Proof. Σ_2^P*-Membership.* By Proposition 7, it follows that $\subseteq$-relevance is in $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^P$.
Σ_2^P*-Hardness.* We transform deciding that a QBF $\Phi = \exists X \forall Y E$ is in $\mathrm{QBF}_{2,\exists}$, where E is in 3DNF, into deciding $\subseteq$-relevance.

Let $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$, and consider the following *LPAP* $\mathcal{P} = \langle H, Q \cup \{p\}, LP, \models_{wf}\rangle$, where $H = X \cup X' \cup Y \cup \{w\}$; the logic program LP is as follows:

$$\begin{aligned} LP \;\; = \;\; & transf(\overline{E}) \cup \{\, p \leftarrow \neg abs \;,\; p \leftarrow y_1 \wedge \cdots \wedge y_n \wedge w \,\} \cup \\ & \{p \leftarrow x_i \wedge x'_i \;,\; q_i \leftarrow x_i \;,\; q_i \leftarrow x'_i \mid i = 1 \ldots, m\} \end{aligned}$$

Observe that for each $S \subseteq H$, $LP \cup S$ is stratified.

Since each q_i can be explained only by x_i or x'_i, it is easily seen that each solution of $\mathcal{P}$ must contain x_i or x'_i for $i = 1, \dots, m$. On the other hand, if a set $S \subseteq H$ with the latter property contains both x_i and x'_i for some i, or it contains $y_1, \dots, y_n$ and w, then S is a solution to $\mathcal{P}$.

We claim that Φ is valid iff w is $\subseteq$-relevant for $\mathcal{P}$.

$\Rightarrow$. Let $X1 \subseteq X$ such that $\forall Y E$ is valid on $X1$ (that is, $\forall Y E$ is a logical consequence of $X1 \cup \{ \neg x'_i \mid x_i \in X \setminus X1 \}$). As observed above, the set $S_{X1} = X1 \cup (X - X1)' \cup Y \cup \{w\}$ is a solution to $\mathcal{P}$. Moreover, S_{X1} is a $\subseteq$-solution of $\mathcal{P}$. To show this, assume that $S \subset S_{X1}$ is a smaller solution for $\mathcal{P}$. It is easily seen that without loss of generality, $w \notin S$ (for, the only rule in $LP \cup S$ in which w occurs can not be applied); moreover, S coincides on $X \cup X'$ with S_{X1}. Since $LP \cup S \models_{wf} p$, some rule must derive p. Clearly, the only one which can do so is $p \leftarrow \neg abs$. Hence, $LP \cup S \models_{wf} \neg abs$, which means that the body of each rule in $transf(\overline{E})$ is false. Since the well-founded model M of $LP \cup S$ is total (as the program is stratified), it thus follows that on $X1$ and the valuation of the atoms in Y given by M, formula E is not true. Thus, $\forall Y E$ is not valid on $X1$, which is a contradiction. Hence, an S as hypothesized does not exist, and S_{X1} is a $\subseteq$-solution of $\mathcal{P}$. Consequently, w is $\subseteq$-relevant.

$\Leftarrow$. Let S be a $\subseteq$-solution to $\mathcal{P}$ such that $w \in S$. Then, S contains clearly exactly one of x_i and x'_i, for each $i = 1, \dots, m$, and $Y \subseteq S$. Since for each $Y1 \subseteq Y$, the set $S_{Y1} = (S \cap (X \cup X') \cup Y1$ is not a solution, we have that $LP \cup S_{Y1} \models \neg p$, and hence $LP \cup S_{Y1} \models abs$. Thus, on the set $X1 = X \cap S$, the formula $\forall Y E$ is valid, and thus Φ is valid. This proves the claim.

The result follows. ∎

4.1.3 $\leq$-preference with $\models_{wf}$

Next, we consider the abductive reasoning tasks based on well-founded inference where the solutions of smallest size are acceptable. A straightforward approach for solving the reasoning problems is to calculate the size s of the smallest solutions, and then to proceed as in case there is no preference, but admitting only solutions of the size s. While this procedure is simple, the following complexity results suggest that there is hardly a substantially more efficient algorithm.

Theorem 5 *Deciding if a given hypothesis is $\leq$-relevant for a LPAP $\mathcal{P} = \langle H, M, LP, \models_{wf} \rangle$ is $\Delta_2^P[O(\log n)]$-complete. $\Delta_2^P[O(\log n)]$-hardness holds even if LP is stratified.*

Proof. $\Delta_2^P[O(\log n)]$-*Membership.* Using an NP oracle, $\leq$-relevance of $h \in H$ can be decided as follows: (1) compute the size s of a wf-$\leq$-solution S by binary search on $[0, \dots, n]$ using the oracle. (Query the oracle whether there exists an $S \in Sol(\mathcal{P})$ such that $|S| \leq k$, starting with $k = n/2$; this problem is in NP, cf. proof of Theorem 1.) (2) Check whether there exists an $S \in Sol(\mathcal{P})$ such that $|S| = s$ and $h \in S$ using the oracle (again, this problem is in NP). Step (1) takes $O(\log |H|)$ oracle calls, and step (2) a single oracle call. Hence, membership in $\Delta_2^P[O(\log n)]$ follows.

$\Delta_2^P[O(\log n)]$-*Hardness.* We choose a simple reduction of the corresponding abduction problem under cautious classical inference ($\models_{cl}^c$), i.e., inference from all models where a model must exist. As follows immediately from [19, Theorem 5.9], deciding whether $h \in H$ is relevant for the $\leq$-solutions of an abduction problem $\langle H, M, LP, \models_{cl}^{c}\rangle$, where LP consists of definite Horn clauses (i.e. rules of the form $a \leftarrow b_1 \wedge \cdots \wedge b_n$, where $a, b_1, \ldots, b_n$ are atoms), is $\Delta_2^P[O(\log n)]$-hard.

Notice that for each $S \subseteq H$, $LP \cup S$ is negation-free; therefore, the unique minimal model $N_{LP \cup S}$ of $LP \cup S$, is the well-founded model of $LP \cup S$. On the other hand, since $LP \cup S$ is definite Horn, $LP \cup S \models_{cl}^{c} M$ iff $N_{LP \cup S} \models M$, i.e., $LP \cup S \models_{wf} M$. It follows that S is a solution to $\langle H, M, LP, \models_{cl}^{c}\rangle$ iff it is a solution of $\langle H, M, LP, \models_{wf}\rangle$. Consequently, deciding if h is $\leq$-relevant for $\langle H, M, LP, \models_{wf}\rangle$ is $\Delta_2^P[O(\log n)]$-hard. ∎

For deciding necessity of a hypothesis, we obtain by a similar proof the same complexity result. Notice that the classes $\Delta_k^P[O(\log n)]$ are closed under complement.

Theorem 6 *Deciding if a given hypothesis is $\leq$-necessary for a LPAP $\mathcal{P} = \langle H, M, LP, \models_{wf}\rangle$ is $\Delta_2^P[O(\log n)]$-complete. $\Delta_2^P[O(\log n)]$-hardness holds even if LP is a stratified logic program.*

We conclude this section with a remark on how the derived results can be sharpened. We focused on stratified programs for the hardness results; with proper modifications of the constructions, the same lower bounds can be derived also for positive programs, with the exception of $\subseteq$-relevance.

Theorem 7 *On the class of LPAPs $\mathcal{P} = \langle H, M, LP, \models_{wf}\rangle$ where LP is a positive logic program, the problems (1) consistency and relevance are* NP*-hard; (2) necessity is* co-NP*-hard; and (3) $\leq$-relevance and $\leq$-necessity are $\Delta_2^P[O(\log n)]$-hard.*

Proof. The program in the proof of Theorem 5 is already negation-free; this proves (3). To show (1) and (2), modify the program LP in the proof of Theorem 1 as follows: replace each occurrence of a negative literal, $\neg z$, in the logic program LP by z'; add rules $w_z \leftarrow z'$, $w_z \leftarrow z$, $f \leftarrow z \wedge z'$ to the program; add both z and z' to the hypotheses; and, add $w_z, \neg f$ to the manifestations. It is easy to see that the solutions of the resulting *LPAP* $\mathcal{P}'$, each of which contains exactly one of z and z', correspond 1-1 to the solutions of $\mathcal{P}$. Since the reductions from (in)consistency to necessity and relevance in Section 3.2 preserve positive programs, (1) and (2) follow. ∎

For positive programs, $\subseteq$-relevance is easier than for arbitrary stratified programs, since checking whether a solution S is $\subseteq$-minimal is polynomial. This is immediate from Proposition 9 and the following fact, which is not hard to see.

Proposition 10 *Let $\mathcal{P} = \langle H, M, LP, \models_{wf}\rangle$ where LP is positive, and let $S \in Sol(\mathcal{P})$. Then, no $S' \subsetneq S$ exists such that $S' \in Sol(\mathcal{P})$ iff for every $h \in S$, $S \setminus \{h\} \notin Sol(\mathcal{P})$.*

Thus, the following is derived.

Theorem 8 *Deciding if a given hypothesis is $\subseteq$-relevant for a LPAP $\mathcal{P} = \langle H, M, LP, \models_{wf}\rangle$ where LP is positive, is* NP*-complete.*

Proof. Membership in NP is shown along the line of proof of Proposition 7. Problem (1) there, solution verification, is polynomial by Proposition 9, and problem (2), no $S' \subset S$ exists such that $S' \in Sol(\mathcal{P})$, is also polynomial by Propositions 9 and 10. It follows that $\subseteq$-relevance is in NP.

On the other hand, since consistency is by Theorem 7 NP-hard for positive programs, clearly $\subseteq$-relevance is NP-hard for positive programs. The result follows. ∎

4.2 Stable Cautious Reasoning

The computational complexity of the abductive problems under stable cautious reasoning will be shown to be higher than the computational complexity of the abductive problems under well-founded inference. Our proofs of the hardness results involve in general more complex translations; in particular, since stratified logic programs have a unique stable model, we will have to use unstratified programs.

Given a set $Z = \{z_1, \ldots, z_n\}$ of propositional atoms we define $clos(Z)$ to be the following set of rules:

$$clos(Z) = \{z_i \leftarrow \neg z_i' \ ,\ z_i' \leftarrow \neg z_i \mid i = 1, \ldots, n\}$$

Let E be a Boolean CNF expression whose atoms include Z, and consider $LP(E, Z) = transf(E) \cup clos(Z)$. Note that $LP(E, Z)$ is not stratified.

Intuitively, z_i' represents the complement of z_i: z_i is true in a stable model M of LP iff z_i' is false in M, and conversely. The rules in $clos(Z)$ "give support" to the atoms in Z.

For instance, given the expression $E = z_1 \vee z_2 \vee \neg z_3 \leftarrow$ from the beginning of the section, the logic program $LP(E, Z) = clos(\{z_1, z_2, z_3\}) \cup transf(\{\, z_1 \vee z_2 \vee \neg z_3 \,\})$ is:

$$\begin{array}{ll} z_i \leftarrow \neg z_i' & (1 \leq i \leq 3) \\ z_i' \leftarrow \neg z_i & (1 \leq i \leq 3) \\ abs \leftarrow z_3 \wedge \neg z_1 \wedge \neg z_2 & \end{array}$$

It is immediately recognized that there is a 1-1 correspondence between the interpretations of E and the stable models of LP, where the models of E correspond to the stable models of LP in which abs is false. The latter are: $M_1 = \{z_3, z_1, z_2\}$, $M_2 = \{z_3, z_1', z_2\}$, $M_3 = \{z_3, z_1, z_2'\}$, $M_4 = \{z_3', z_1, z_2\}$, $M_5 = \{z_3', z_1', z_2'\}$, $M_6 = \{z_3', z_1', z_2\}$, $M_7 = \{z_3', z_1, z_2'\}$. Such a correspondence can be profitably used to check the validity of QBFs. For instance, the formula $\exists z_1 \exists z_2 \exists z_3 E$ is valid (i.e., it is in $\mathrm{QBF}_{1,\exists}$) iff $LP \models^b_{st} \neg abs$. The formula $\forall Z E$ is valid (i.e., it is in $\mathrm{QBF}_{1,\forall}$ iff $LP(E, Z) \models^c_{st} \neg abs$.

4.2.1 No preference

Now, let us consider solution verification for LPAP s that use inference $\models^c_{st}$. In the light of the results from [66], it is clear that this problem is most likely not polynomial.

Proposition 11 *Let $\mathcal{P} = \langle H, M, LP, \models^c_{st} \rangle$ be a LPAP. Deciding if $S \subseteq H$ fulfills $S \in Sol(\mathcal{P})$ is $\mathrm{D}^{\mathrm{P}}_1$-complete.*

Proof. By Proposition 3, solution verification is polynomial-time equivalent to inference of a conjunction of literals. As shown in [66], given a propositional logic program LP and atom q, deciding whether $LP \models^c_{st} q$ is $\mathrm{D}^{\mathrm{P}}_1$-complete. Clearly, $\mathrm{D}^{\mathrm{P}}_1$ is also an upper bound for cautious inference of a conjunction of literals (apart from a simple computational argument, this follows easily from the fact that $\mathrm{D}^{\mathrm{P}}_1$ is closed under conjunction). The result follows. ∎

Next, we consider consistency checking for *LPAP*s. Due to the higher complexity of solution verification, the problem resides for $\models^c_{st}$ one level higher up in the polynomial hierarchy compared to $\models_{wf}$.

Theorem 9 *To decide if Sol*$(\mathcal{P}) \neq \emptyset$ *for a given LPAP* $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ *is* Σ^P_2*-complete.*

Proof. Σ^P_2*-Membership.* We guess $S \subseteq H$. Then, by virtue of Proposition 11, we can verify that $S \in$ *Sol*$(\mathcal{P})$ in polynomial time with two calls to an oracle for NP. Hence, the problem is in Σ^P_2.
Σ^P_2*-Hardness.* We give a transformation from deciding $\Phi \in \mathrm{QBF}_{2,\exists}$, where $\Phi = \exists X \forall Y E$, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, and E is in 3DNF.

Consider the *LPAP* $\mathcal{P} = \langle X, \{abs\}, LP, \models^c_{st}\rangle$, where $LP = transf(\overline{E}) \cup clos(Y)$.

For each $X1 \subseteq X$, there is a 1-1 correspondence between the stable models of $LP \cup X1$ and the possible truth assignments for $y_1, \cdots, y_m$ (the truth value of each x_i in the stable models is determined by membership in $X1$). A truth assignment to Y satisfies E iff *abs* is true in the corresponding stable model. Thus, the formula $\forall Y E$ is valid dependent on $X1$, iff for each $N \in \mathrm{STM}(LP \cup X1)$, $N \models abs$, i.e., $LP \cup X1 \models^c_{st} abs$. On the other hand, Φ is valid iff there exists an $X1 \subseteq X$ such that $\forall Y E$ is valid dependent on $X1$. Therefore, Φ is valid iff there exists an $X1 \subseteq X$ such that $LP \cup X1 \models^c_{st} abs$. In other words, Φ is valid iff there exists $X1 \in$ *Sol*$(\mathcal{P})$, i.e., *Sol*$(\mathcal{P}) \neq \emptyset$ if and only if Φ is valid.

Whence, to decide if *Sol*$(\mathcal{P}) \neq \emptyset$ for a given *LPAP* $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ is Σ^P_2-hard. ∎

For deciding relevance, we obtain by Proposition 5 the same complexity result, and for deciding necessity, by Proposition 6 a complementary complexity result.

Theorem 10 *Deciding if a given hypothesis is relevant for a LPAP* $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ *is* Σ^P_2*-complete.*

Theorem 11 *Deciding if a given hypothesis is necessary for a given LPAP* $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ *is* Π^P_2*-complete.*

4.2.2 $\subseteq$-preference with $\models^c_{st}$

Now let us determine the effect of restricting solutions to those minimal under inclusion.

As in the case of well-founded inference, we face for the relevance test an increase in complexity, which is intuitive from the more difficult problem of solution verification. In fact, we encounter a problem complete for a class of the third level of the polynomial hierarchy,

Theorem 12 *Deciding if a given hypothesis is* $\subseteq$*-relevant for a LPAP* $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ *is* Σ^P_3*-complete.*

Proof. Σ_3^P-*Membership.* By Theorem 9, consistency is in Σ_2^P. Hence, by Proposition 7, $\subseteq$-relevance is in $\mathrm{NP}^{\Sigma_2^P} = \Sigma_3^P$.
Σ_3^P-*Hardness.* We describe a transformation from deciding if a QBF $\Phi = \exists X \forall Y \exists Z E$, where E is in 3CNF, belongs to $\mathrm{QBF}_{3,\exists}$.

Let $X = \{x_1, \ldots, x_{n_x}\}$, $Y = \{y_1, \ldots, y_{n_y}\}$, and $Z = \{z_1, \ldots, z_{n_z}\}$. Take atoms $X' = \{x'_1, \ldots, x'_{n_x}\}$, $Q = \{q_1, \ldots, q_{n_x}\}$ and w, and define a logic program LP as follows:

$$\begin{aligned} LP \;=\; & \mathit{transf}(E) \cup \mathit{clos}(Z) \cup \{\mathit{abs} \leftarrow y_1 \wedge \cdots \wedge y_{n_y} \wedge w\ \} \cup \\ & \{\mathit{abs} \leftarrow x_i \wedge x'_i\ ,\ q_i \leftarrow x_i\ ,\ q_i \leftarrow x'_i \mid i = 1 \ldots, n_x\}. \end{aligned}$$

Consider the *LPAP* $\mathcal{P} = \langle H, Q \cup \{abs\}, LP, \models_{st}^c \rangle$, where $H = X \cup X' \cup Y \cup \{w\}$.

Since each q_i can be explained only by x_i or x'_i, it is easily seen that each solution of $\mathcal{P}$ must contain x_i or x'_i for $i = 1, \ldots, n_x$. On the other hand, each subset $S \subseteq H$ with the latter property that contains $\{w, y_1, \ldots, y_{n_y}\}$ or $\{x_i, x'_i\}$, for some i, is a solution to $\mathcal{P}$. However, a minimal solution contains either $\{w, y_1, \ldots, y_{n_y}\}$ or $\{x_i, x'_i\}$ for some i, but it can not contain both.

We claim that Φ is valid iff w is $\subseteq$-relevant for $\mathcal{P}$.

Prior to the proof of the claim, suppose for the moment that $n_x = 0$. Then $\mathcal{P}$ would be of the form $\mathcal{P}' = \langle Y \cup \{w\}, \{abs\}, LP', \models_{st}^c \rangle$, where $LP' = \mathit{transf}(E) \cup \mathit{clos}(Z) \cup \{\, \mathit{abs} \leftarrow y_1 \wedge \cdots \wedge y_{n_y} \wedge w\ \}$ is the logic program for $\exists Y \forall Z \overline{E}$ according to the transformation in the proof of Theorem 9, augmented by the rule $\mathit{abs} \leftarrow y_1 \wedge \cdots \wedge y_{n_y} \wedge w$. Clearly, $Y \cup \{w\}$ is a solution to $\mathcal{P}'$. Since $\langle Y, \{abs\}, \mathit{transf}(E) \cup \mathit{clos}(Z), \models_{st}^c \rangle$ has, as shown in the proof of Theorem 9, a solution iff $\exists Y \forall Z \overline{E}$ is valid, it follows that $Y \cup \{w\}$ is the only solution of $\mathcal{P}$ iff $\exists Y \forall Z \overline{E}$ is false.

Utilizing these observations, we now proof the claim.
($\Rightarrow$). Let $X1 \subseteq X$ such that $\forall Y \exists Z E$ is valid on $X1$, i.e., $\exists Y \forall Z \overline{E}$ is false. By the above consideration, $S = X1 \cup (X - X1)' \cup Y \cup \{w\}$ is a solution to $\mathcal{P}$. Moreover, it holds that $Y \cup \{w\}$ is the only solution of the *LPAP* $\langle Y \cup \{w\}, \{abs\}, LP' \cup X1 \cup (X - X1)', \models_{st}^c \rangle$. Consequently, S is a $\subseteq$-minimal solution of $\mathcal{P}$, which means that w is $\subseteq$-relevant.
($\Leftarrow$). Let $S \in \mathit{Sol}_{\subseteq}(\mathcal{P})$ such that $w \in S$, and let $X1 = X \cap S$. Clearly, $Y \subseteq S$. Thus, it follows that each subset $Y1 \subseteq Y$ is not a solution of the *LPAP* $\langle Y \cup \{w\}, \{abs\}, LP' \cup X1, \models_{st}^c \rangle$. Thus, $\exists Y \forall Z \overline{E}$ is false on $X1$, i.e., $\forall Y \exists Z E$ is valid on $X1$. This means that Φ is valid. The claim is proved.

Clearly, $\mathcal{P}$ is constructible in polynomial time from Φ. Thus, the result follows. ∎

4.2.3 $\leq$-preference with $\models_{st}^c$

In the setting with inference $\models_{wf}$, we have seen that the straightforward algorithm to decide relevance and necessity, respectively, is by computing the size s of a smallest solution and proceed then as under no preference, but admitting only solutions of size s. The following results show that with respect to worst best complexity, there is most likely no significantly more efficient algorithm.

Theorem 13 *Deciding if a given hypothesis is $\leq$-relevant for a LPAP $\mathcal{P} = \langle H, M, LP, \models_{st}^b \rangle$ is $\Delta_3^P[O(\log n)]$-complete.*

Proof. $\Delta_3^P[O(\log n)]$-*Membership.* The proof is similar to the membership part for Theorem 5, where the NP oracle for querying solutions is replaced by a Σ_2^P oracle.
$\Delta_3^P[O(\log n)]$-*Hardness.* We show hardness by a reduction from the corresponding abduction problem under classical semantics. Theorem 4.6 in [19] states that deciding, given a set of propositional clauses T, sets H, M of propositional atoms and an atom $h \in H$, whether h belongs to any smallest (w.r.t. size) subset $S \subseteq H$ such that $T \cup S \models_P M$ and $T \cup S$ is consistent (where $\models_P$ is classical propositional consequence), is $\Delta_3^P[O(\log n)]$-hard. If we view T as a CNF expression and assume that abs is a logical constant for falsity, this means that, in our terminology, deciding whether a hypothesis $h \in H$ is $\leq$-relevant for an *LPAP* $\mathcal{P} = \langle H, M, transf(T), \models_P \rangle$, is $\Delta_3^P[O(\log n)]$-hard, where $transf(\cdot)$ is the transformation from the beginning of Section 4.

Now let $X = V_{\mathcal{P}} \setminus \{abs\}$ be the atoms occurring in T and define

$$LP = transf(T) \cup \{x \leftarrow \neg x' \ , \ x' \leftarrow \neg x \mid x \in X\} \cup \{b \leftarrow abs \wedge \neg b \ \}$$

where $X \cap X' = \emptyset$ and b is a fresh atom. Then, the following is easily checked: For each $S \subseteq H$, if N is a stable model of $LP \cup S$, then $N \cap V_{\mathcal{P}}$ is a model of $T \cup S$; note that in each stable model of $LP \cup S$, both abs and b must be false. Conversely, if N is a model of $T \cup S$, then $N \cup (X - N)'$ is a stable model of LP. Thus, it follows that S is a solution of $\mathcal{P}' = \langle H, M, LP, \models_{st}^c \rangle$ iff S is a solution of $\mathcal{P}$. Consequently, h is $\leq$-relevant for $\mathcal{P}$ iff it is $\leq$-relevant for $\mathcal{P}'$. It follows that $\leq$-relevance under $\models_{st}^b$ is $\Delta_3^P[O(\log n)]$-hard. ∎

As in the case of inference $\models_{wf}$, we obtain the same result for necessity. (Recall that $\Delta_3^P[O(\log n)]$ is closed under complement.)

Theorem 14 *Deciding if a given hypothesis is $\leq$-necessary for a LPAP $\mathcal{P} = \langle H, M, LP, \models_{st}^c \rangle$ is $\Delta_3^P[O(\log n)]$-complete.*

4.3 Stable Brave Reasoning

In this section we analyze the complexity of the main abductive reasoning tasks on LPAP s that use inference $\models_{st}^b$. Interestingly, the complexity coincides with the one for inference $\models_{wf}$, excepting the problem of solution verification; in contrast to abduction, deduction under the brave variant of the stable model semantics is harder than well-founded deduction.

All hardness results but one will be shown as consequences of the theorems stated in Section 4.1 on well-founded semantics, by using that: (1) well-founded and stable deduction coincide on stratified programs, and (2) all hardness results for the well-founded semantics have been proven for stratified programs.

From (1), we obtain the following proposition on the equivalence of abduction using well-founded semantics and abduction using stable brave semantics.

Proposition 12 *Let $\mathcal{P} = \langle H, M, LP, \models_{wf} \rangle$ and $\mathcal{P}' = \langle H, M, LP, \models_{st}^b \rangle$ be two LPAPs, where LP is a stratified logic program. Then, $S \in Sol(\mathcal{P}')$ if and only if $S \in Sol(\mathcal{P})$.*

Proof. Being LP stratified, for each $S \subseteq H$, $LP \cup S$ is stratified as well, since the rules in S can not violate the stratification condition (see Section 2.1.3) as they have an empty body. As a consequence, by Proposition 2 $LP \cup S$ has a unique stable model, which coincides with the total well-founded model. Therefore, $LP \cup S \models_{st}^{b} M$ iff $LP \cup S \models_{wf} M$. Whence, $S \in \mathit{Sol}(\mathcal{P}')$ if and only if $S \in \mathit{Sol}(\mathcal{P})$. ∎

This relation, together with the results from Section 4.1, allows us to conclude the following.

Lemma 1 *For each of the considered abductive reasoning tasks, the hardness result for LPAPs that use well-founded reasoning (cf. Section 4.1) provides a lower bound for the LPAPs that use brave stable reasoning.*

For solution verification, we get higher complexity for stable brave reasoning, which is inherited from deciding stable model existence.

Proposition 13 *Let $\mathcal{P} = \langle H, M, LP, \models_{st}^{b} \rangle$ be a LPAP and $S \subseteq H$. Given $\mathcal{P}$ and S, deciding if $S \in \mathit{Sol}(\mathcal{P})$ is* NP*-complete.*

Proof. By Proposition 3, solution verification is polynomial-time equivalent to brave stable inference of a conjunction of literals from a logic program. From the results in [6, 49, 48, 72], we have that deciding whether $LP \models_{st}^{b} q$ where LP is a logic program and q a literal, is NP-complete. Clearly, NP is an upper bound for brave stable inference of a conjunction of literals. This proves the result. ∎

We observe that solution verification under inference $\models_{st}^{b}$ (cf. proof above) and $\models_{wf}$ differs merely by the fact that in the former case, a (stable) model N proving the solution property is guessed, while in the latter case this model (the well-founded model) is computed in polynomial time. Thus, each of the nondeterministic algorithms in the membership parts of the proofs of the complexity results for abductive reasoning tasks under $\models_{wf}$ can be easily modified to a similar algorithm for $\models_{st}^{b}$, by adding to the guess of a solution S a guess for N. Since the stable model property of N can be decided in polynomial time, we obtain nondeterministic algorithms proving the same membership results for each reasoning task under $\models_{st}^{b}$ as under $\models_{wf}$. Together with Lemma 1 and results in [49, 72], we thus have the following results.

Theorem 15 *Let $\mathcal{P} = \langle H, M, LP, \models_{st}^{b} \rangle$ be a given LPAP. Then,*

(1) Deciding if $\mathit{Sol}(\mathcal{P}) \neq \emptyset$ is NP*-complete (this follows from results of [49, 72]);*

(2) Deciding if a given hypothesis is relevant for $\mathcal{P}$ is NP*-complete;*

(3) Deciding if a given hypothesis is necessary for $\mathcal{P}$ is co-NP*-complete;*

(4) Deciding if a given hypothesis is $\subseteq$-relevant for $\mathcal{P}$ is Σ_2^P-complete;

(5) Deciding if a given hypothesis is $\subseteq$-necessary for $\mathcal{P}$ is co-NP*-complete;*

(6) Deciding if a given hypothesis is $\leq$-relevant for $\mathcal{P}$ is $\Delta_2^P[O(\log n)]$-complete; and

(7) Deciding if a given hypothesis is $\leq$-necessary for $\mathcal{P}$ is $\Delta_2^P[O(\log n)]$-complete.

Proof. We provide here only membership proofs for (1), (4), and (6). For the other clauses, the claim follows from the results above. The same applies to the hardness parts. In particular, NP-hardness in part (1) follows from Proposition 4.

(1) NP-*Membership.* We can guess $S \subseteq V_{\mathcal{P}}$ and $N \subseteq V_{\mathcal{P}}$, and verify in polynomial time that N is a stable model of $LP \cup S$ and $N \models M$.

(4) Σ_2^P-*Membership.* By Proposition 13 and part (1), solution verification and consistency are in NP. Hence, by Proposition 7, $\subseteq$-relevance is in $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^P$.

(6) $\Delta_2^P[O(\log n)]$-*Membership.* The proof is similar to the membership part of Theorem 5; we use the NP oracle to query existence of solutions S, restricted to size $|S| \leq k$ and $|S| = s$, respectively, under inference $\models_{st}^{b}$ instead of $\models_{wf}$; clearly, these queries are in NP (cf. part 1). ∎

5 Abduction from Disjunctive Logic Programs

5.1 Disjunctive Logic Programming

In this section we provide a short overview of minimal model and stable model semantics for (propositional) disjunctive logic programs [54, 62]. A comprehensive treatment of disjunctive logic programming is given in [45], to which the reader is referred for unexplained concepts.

A *disjunctive logic program* (DLP) is a logic program where disjunction is allowed in the head of the rules. Thus, a disjunctive logic program clause (simply clause or rule) is a formula

$$a_1 \vee \cdots \vee a_n \leftarrow b_1 \wedge \cdots \wedge b_k \wedge \neg b_{k+1} \wedge \cdots \wedge \neg b_m, \qquad n \geq 1,\ m \geq 0.$$

Total interpretations and models are defined as for (normal) logic programming (see Section 2.1), where satisfiability of the rules is extended in the obvious way.

The first semantics proposed for disjunctive logic programs has been the minimal model semantics, presented by Minker for positive (i.e., $\neg$-free) disjunctive logic programs [54].

A model M is *minimal* for a disjunctive logic program LP if no strict subset of M is a model for LP. The set of all minimal models for LP, denoted by $\mathrm{MM}(LP)$, has been adopted in [54] as the intended meaning of LP.

Example 8 Let $P = \{a \vee b\}$. It is easy to verify that $\{a\}$ and $\{b\}$ are the minimal models of P ($\mathrm{MM}(P) = \{\{a\}, \{b\}\}$).
$\{a, b\}$ is the only minimal model of the program $P = \{a \vee b,\ b \leftarrow a,\ a \leftarrow b\}$. ∎ ∎

Other semantics for disjunctive logic programs have been recently proposed, cf. [45]. A generally acknowledged proposal is the extension of the stable model semantics [25] to disjunctive logic programs [62].

Given a DLP LP and an interpretation I, the *Gelfond-Lifschitz transform* of LP with respect to I, LP^I, is the DLP consisting of all clauses $a_1 \vee \cdots \vee a_n \leftarrow b_1 \wedge \cdots \wedge b_k$ such that

(1) $a_1 \vee \cdots \vee a_n \leftarrow b_1 \wedge \cdots \wedge b_k \wedge \neg b_{k+1} \wedge \cdots \wedge \neg b_m \in LP$ and
(2) $b_i \notin I$, for all $k < i \leq m$.

An interpretation I is called a *(disjunctive) stable model* of LP iff $I \in \mathrm{MM}(LP^I)$ [62]. The set of all stable models of LP is denoted by $\mathrm{STM}(LP)$.

Example 9 Let

$$P = \quad a \vee b \leftarrow c \quad b \leftarrow \neg a \wedge \neg c \quad a \vee c \leftarrow \neg b.$$

Consider $I = \{b\}$. Then,

$$P^I = \quad a \vee b \leftarrow c \quad b \leftarrow .$$

Check that I is a minimal model of P^I; thus, I is a stable model of P. ∎

Notice that LP^I is $\neg$-free, and that LP^I coincides with LP if LP is $\neg$-free. Thus,

Proposition 14 *If LP is a positive ($\neg$-free) DLP, then* $\mathrm{STM}(LP) = \mathrm{MM}(LP)$*, i.e. minimal and stable model semantics of LP coincide.*

Both minimal model and stable model semantics provide in general a set of models as the intended meaning of a disjunctive logic program. Thus, brave and cautious reasoning are immediately extended to both semantics for DLPs. As for normal logic programs, we denote by $\models^b_{st}$ and $\models^c_{st}$ the inference operators of brave and cautious reasoning of stable semantics, respectively. Similarly, $\models^b_{mm}$ and $\models^c_{mm}$ will denote the corresponding operators for minimal model semantics.

The definition of *LPAP* is extended by allowing in $\langle H, M, LP, \models\rangle$ that LP is a disjunctive logic program and $\models$ an inference operator for DLP; we refer to this extended notion as *disjunctive LPAP*.

We conclude this section by recalling some results on the complexity of stable model semantics that will be used in the next section.

Proposition 15 ([21, 20]) *Let LP be a propositional disjunctive logic program. Then,*

(1) Given LP and a model M of LP, deciding whether $M \in \mathrm{STM}(LP)$ is co-NP*-complete;*

(2) Given LP, deciding whether it has a stable model is Σ^P_2-complete;

(3) Given LP and literals $q_1, \ldots, q_k$, deciding whether $LP \models^b_{st} q_1 \wedge \cdots \wedge q_k$ is Σ^P_2-complete (hard already for $k = 1$); and

(4) Given LP and literals $q_1, \ldots, q_k$, deciding whether $LP \models^{cls}_{st} q_1 \wedge \cdots \wedge q_k$ is Π^P_2-complete (hard already for $k = 1$).

5.2 Complexity Results

In the present section we study, for abduction from disjunctive logic programs, the complexity of the problems of *consistency*, *relevance*, and *necessity* under inference operators $\models^b_{st}$, $\models^c_{st}$, $\models^b_{mm}$, and $\models^c_{mm}$, combined with $\subseteq$-solutions.

Several hardness results will be proven by reductions from QBFs into abduction problems; the programs of the disjunctive *LPAP*s will be suitable adaptations and extensions of the disjunctive program reported next (which has been first described in [20]).

Let Φ be a formula of form $\exists X \forall Y E$, where E is a Boolean expression in 3DNF, i.e., $E = D_1 \vee \cdots \vee D_r$ and each $D_i = L_{i,1} \wedge L_{i,2} \wedge L_{i,3}$ is a conjunction of literals $L_{i,j}$; we allow here that atoms not in $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ may occur in E. Define the following DLP $LP(\Phi)$:

$$
\begin{array}{llll}
x_i \vee x_i' \leftarrow & & & \text{for each } i = 1, \ldots, n \\
y_j \vee y_j' \leftarrow & y_j \leftarrow w & y_j' \leftarrow w & \text{for each } j = 1, \ldots, m \\
w \leftarrow y_j \wedge y_j' & & & \text{for each } j = 1, \ldots, m \\
w \leftarrow \sigma(L_{k,1}) \wedge \sigma(L_{k,2}) \wedge \sigma(L_{k,3}) & & & \text{for each } k = 1, \ldots, r
\end{array}
$$

where σ maps literals to atoms as follows:

$$
\sigma(L) = \begin{cases} x_i' & \text{if } L = \neg x_i \text{ for some } i = 1, \ldots, n \\ y_j' & \text{if } L = \neg y_j \text{ for some } j = 1, \ldots, m \\ L & \text{otherwise} \end{cases}
$$

Intuitively, x_i' corresponds to $\neg x_i$ and y_j' corresponds to $\neg y_j$.

Lemma 2 *Let $\Phi = \exists X \forall Y E$ be a QBF and $LP(\Phi)$ the disjunctive logic program defined above. Then, (1) $\mathrm{STM}(LP(\Phi)) \neq \emptyset$, and (2) Φ is valid iff $LP(\Phi) \models^b_{st} w$.*

Proof. Since Φ is a QBF, all atoms in E are from $X \cup Y$, and hence $LP(\Phi)$ is a positive program. Therefore, (1) follows from Proposition 14. For part (2), we outline a condensed proof that emerges from the proofs of Theorems 3.3 and 3.5 of [20].

The following facts are easy to see:
Fact 1: For every model M of $LP(\Phi)$, it holds that $w \in M$ iff $Y \cup Y' \subseteq M$.
Fact 2: For every set $X1 \subseteq X$, the set $M_{X1} = X1 \cup (X \setminus X1)' \cup Y \cup Y' \cup \{w\}$ is a model of $LP(\Phi)$.
Fact 3: Every minimal model M of $LP(\Phi)$ satisfies $M = X1 \cup (X \setminus X1)' \cup A$ for some $X1 \subseteq X$ and $A \subseteq Y \cup Y' \cup \{w\}$.

Now if Φ is valid, there is a subset $X1 \subseteq X$ such that $\forall Y E$ is valid on $X1$. Hence, it follows that M_{X1} is a minimal model of $LP(\Phi)$, and thus $LP(\Phi) \models^b_{st} w$. Conversely, if $LP(\Phi) \models^b_{st} w$, then a minimal model M exists such that $w \in M$. By Facts 1–3, it is immediate that $M = M_{X1}$ for some $X1 \subseteq X$. As a consequence, $\forall Y E$ is valid on $X1$, and hence Φ is valid. ∎

Part (2) on brave reasoning and the fact that $LP(\Phi)$ is positive imply the following on cautious reasoning.

Corollary 1 *The QBF $\Phi = \exists X \forall Y E$ is not valid iff $LP(\Phi) \models^c_{st} \neg w$.*

Note that $LP(\Phi)$ is constructible from Φ in polynomial time. We also utilize, like in the case of nondisjunctive logic programs, generic reductions between abduction and entailment problems. It is immediate that all reductions and results in Section 3.2 hold also on disjunctive *LPAP*s, where nondisjunctive programs are replaced by disjunctive programs. In the sequel, when we refer to results from Section 3.2, we actually mean the generalization to disjunctive *LPAP*s.

5.2.1 Disjunctive stable cautious reasoning

First, we explore the complexity of solution verification. Since the complexity of inference from the stable models of a logic program increases by one level in the polynomial hierarchy if disjunctive rules are permitted, we obtain a corresponding increase in complexity for abductive solution verification.

Proposition 16 *Let $\mathcal{P} = \langle H, M, LP, \models^c_{st} \rangle$ be a disjunctive LPAP. Deciding if $S \subseteq H$ fulfills $S \in Sol(\mathcal{P})$ is $\mathrm{D}^{\mathrm{P}}_2$-complete.*

Proof. By Proposition 3, solution verification is polynomial-time equivalent to inference $LP1 \models^c_{st} \gamma$, where γ is a conjunction of literals. We show that deciding $LP1 \models^c_{st} \gamma$ is $\mathrm{D}^{\mathrm{P}}_2$-complete, which proves the result. The problem reduces to deciding whether (1) $LP1$ has a stable model and (2) $LP1 \models^{cls}_{st} \gamma$, which is by Proposition 15 in Σ^P_2 and Π^P_2, respectively. It follows that $LP1 \models^c_{st} \gamma$ is in $\mathrm{D}^{\mathrm{P}}_2$. On the other hand, from Proposition 15 and Corollary 1, it is immediate that, given disjunctive logic programs $LP1$ and $LP2$, where $LP2$ is positive, and a conjunction γ of literals, deciding whether simultaneously (1) $LP1$ has a stable model and (2) $LP2 \models^{cls}_{st} \gamma$ is $\mathrm{D}^{\mathrm{P}}_2$-hard. Suppose w.l.o.g. that $LP1$ has no variable in common with $LP2$ and γ, and let $LP = LP1 \cup LP2$. Then, $LP \models^c_{st} \gamma$ iff (1) and (2) are true. This proves $\mathrm{D}^{\mathrm{P}}_2$-hardness. ∎

Similarly, the complexity of consistency checking migrates one level up in the polynomial hierarchy.

Theorem 16 *To decide if $Sol(\mathcal{P}) \neq \emptyset$ for a given disjunctive LPAP $\mathcal{P} = \langle H, M, LP, \models^c_{st} \rangle$ is Σ^P_3-complete.*

Proof. Σ^P_3*-Membership.* We guess $S \subseteq H$, and verify that $S \in Sol(\mathcal{P})$. From Proposition 16, the check that $S \in Sol(\mathcal{P})$ can be done in polynomial time with two calls to an oracle for Σ^P_2. Hence, the problem is in Σ^P_3.
Σ^P_3*-Hardness.* We show that the complementary problem, i.e., deciding whether $Sol(\mathcal{P}) = \emptyset$, is hard for Π^P_3. We give a transformation from checking if a QBF $\Phi = \forall Z \exists X \forall Y E$ is in $\mathrm{QBF}_{3,\exists}$ into deciding whether $Sol(\mathcal{P}) = \emptyset$, where $Z = \{z_1, \ldots, z_o\}$, $X = \{x_1, \ldots, x_n\}$, and $Y = \{y_1, \ldots, y_m\}$. Without loss of generality, we assume that E is in 3DNF, $E = D_1 \vee \cdots \vee D_r$ and each $D_i = L_{i,1} \wedge L_{i,2} \wedge L_{i,3}$ is a conjunction of literals $L_{i,j}$ over atoms from $Z \cup X \cup Y$. Consider the disjunctive *LPAP* $\mathcal{P} = \langle Z, \{\neg w\}, LP_1, \models^c_{st} \rangle$, where LP_1 is the disjunctive logic

program $LP(\exists X\forall Y E)$. Note that the rules $w \leftarrow \sigma(L_{k,1}) \wedge \sigma(L_{k,2}) \wedge \sigma(L_{k,3})$ may contain also negative literals of the form $\neg z_i$.

We will prove that Φ is valid if and only if $Sol(\mathcal{P}) = \emptyset$.

Let $Z1 \subseteq Z$. Each stable model M of $LP_1 \cup Z1$ contains all and only the z_i's occurring in $Z1$. Thus, as usual, we can identify $Z1$ with the truth assignment to Z which maps into *true* all and only the elements from $Z1$. From Corollary 1, it follows that $\Phi' = \exists X\forall Y E$ is false on $Z1$ if and only if $\neg w$ is true in every stable model of $LP_1 \cup Z1$ ($LP_1 \cup Z1 \models^c_{st} \neg w$). Since $\Phi = \forall Z \Phi'$, we have that Φ is valid iff there does not exist $Z1 \subseteq Z$ such that $LP_1 \cup Z1 \models^c_{st} \neg w$ (i.e., no $Z1$ is in $Sol(\mathcal{P})$). Thus, Φ is valid iff $Sol(\mathcal{P}) = \emptyset$. As deciding if such a Φ is valid is Π^P_3-hard, deciding if $Sol(\mathcal{P}) = \emptyset$ is Π^P_3-hard as well. As a consequence, deciding if $Sol(\mathcal{P}) \neq \emptyset$ is hard for the dual class Σ^P_3. ∎

As for normal logic programs, by Proposition 5 relevance has the same complexity as consistency checking, and by Proposition 6, necessity has complementary complexity.

Theorem 17 *Deciding if a given hypothesis is relevant for a disjunctive LPAP $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ is Σ^P_3-complete.*

Theorem 18 *Deciding if a given hypothesis is necessary for a given disjunctive LPAP $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ is Π^P_3-complete.*

Adopting minimality with respect to inclusion as the criterion for acceptability led to an increase of the complexity of relevance testing for normal logic programs by one level of the polynomial hierarchy. The same happens in the case of disjunctive logic programs, and we end up with a problem complete for a class of the fourth level of the polynomial hierarchy.

Theorem 19 *Deciding if a given hypothesis is $\subseteq$-relevant for a disjunctive LPAP $\mathcal{P} = \langle H, M, LP, \models^c_{st}\rangle$ is Σ^P_4-complete.*

Proof. Σ^P_4-*Membership.* By Theorem 16, consistency is in Σ^P_3. Hence, by Proposition 7, $\subseteq$-relevance is in $\mathrm{NP}^{\Sigma^P_3} = \Sigma^P_4$.
Σ^P_4-*Hardness.* We describe a transformation from checking if a QBF $\Phi = \exists T\forall Z\exists X\forall Y E$ is in $\mathrm{QBF}_{4,\exists}$ into deciding $\subseteq$-relevance, where $T = \{t_1, \ldots, t_l\}$, $Z = \{z_1, \ldots, z_o\}$, $X = \{x_1, \ldots, x_n\}$, and $Y = \{y_1, \ldots, y_m\}$. We assume that E is in 3DNF, i.e., $E = D_1 \vee \cdots \vee D_r$ and each $D_i = L_{i,1} \wedge L_{i,2} \wedge L_{i,3}$ is a conjunction of literals $L_{i,j}$ over atoms from $T \cup Z \cup X \cup Y$. Consider the following logic program LP_2:

$(g0)$	$q \leftarrow s \wedge z_1 \wedge \ldots \wedge z_o$		$q \leftarrow \neg w$	
$(g1)$	$abs \leftarrow t_i \wedge t'_i$	$q_i \leftarrow t$	$q_i \leftarrow t'_i$	for each $i = 1, \ldots, l$
$(g2)$	$x_i \vee x'_i \leftarrow$			for each $i = 1, \ldots, n$
$(g3)$	$y_j \vee y'_j \leftarrow$	$y_j \leftarrow w$	$y'_j \leftarrow w$	for each $j = 1, \ldots, m$
$(g4)$	$w \leftarrow y_j \wedge y'_j$			for each $j = 1, \ldots, m$
$(g5)$	$w \leftarrow \sigma(L_{k,1}) \wedge \sigma(L_{k,2}) \wedge \sigma(L_{k,3})$			for each $k = 1, \ldots, r$

where the rules $(g2)-(g5)$ constitute the program $LP(\exists X\forall Y E)$. Consider the disjunctive *LPAP*

$$\mathcal{P} = \langle Z \cup T \cup T' \cup \{s\}, \{q, \neg abs\} \cup Q, LP_2, \models^c_{st}\rangle,$$

where $T' = \{t'_1, \ldots, t'_l\}$ and $Q = \{q_1, \ldots, q_l\}$. We will prove that Φ is valid if and only if s is $\subseteq$-relevant for $\mathcal{P}$.
($\Rightarrow$). Observe first that each set $S \subseteq H$ containing $\{s\} \cup Z$, and such that, for each $1 \leq i \leq l$, exactly one element from $\{t_i, t'_i\}$ is in S, is a solution to $\mathcal{P}$. Moreover, if S and S' are two solutions to $\mathcal{P}$, then neither $S \cap (T \cup T') \subset S' \cap (T \cup T')$ nor $S' \cap (T \cup T') \subset S \cap (T \cup T')$ holds.
Since Φ is valid, there exists $T1 \subseteq T$ such that $\Phi' = \forall Z \exists X \forall Y E$ is valid on $T1$. Let $S = T1 \cup (T - T1)' \cup Z \cup \{s\}$. From the above observation, S is a solution to $\mathcal{P}$. Moreover, since Φ' is valid on $T1$, there exists no solution to the disjunctive *LPAP* $\mathcal{P}' = \langle Z, \neg w, LP_1, \models^c_{st}\rangle$, where LP_1 consists of the rules $(g2)-(g5)$ (clearly, the t_i atoms in the rules $(g5)$ are replaced by their truth values in $T1$); this is shown by a proof similar to the one of Theorem 16. Therefore, S is a $\subseteq$-minimal solution of $\mathcal{P}$, as having $T1 \cup (T - T1)'$ fixed, $\{s\} \cup Z$ is necessary to derive q, and, from the above observations, every other solution to $\mathcal{P}$ necessarily differs from S on $T \cup T'$.
($\Leftarrow$). Let $S \in \mathit{Sol}_{\subseteq}(\mathcal{P})$ such that $s \in S$, and let $T1 = T \cap S$. Z must be contained in S, otherwise $S - \{s\} \in \mathit{Sol}(\mathcal{P})$ which contradicts the minimality of S. Thus, it follows that each subset $Z1 \subseteq Z$, is not a solution of the disjunctive *LPAP* $\mathcal{P}' = \langle Z, \neg w, LP_1, \models^c_{st}\rangle$, where LP_1 consists of the rules $(g2)-(g5)$ (see above). Thus, from the proof of Theorem 16, $\Phi' = \forall Z \exists X \forall Y E$ is valid on $T1$. As a consequence, Φ is valid.

Since $\mathcal{P}$ is constructible in polynomial time from Φ, Σ^P_4-hardness of deciding $\subseteq$-relevance under inference $\models^c_{st}$ follows. ∎

5.2.2 Disjunctive stable brave reasoning

In the case of abduction from normal logic programs, we found that the complexity under the brave variant of the stable semantics is the same as under the well-founded semantics (with one exception), and we established the results by relationships between these semantics. In the case of disjunctive programs, we can not take profit of such results. However, in all cases disjunctive rules lead for the brave variant to the same increase in complexity as for the cautious variant, namely one level up in the polynomial hierarchy.

We start by establishing this for solution verification.

Proposition 17 *Let $\mathcal{P} = \langle H, M, LP, \models^b_{st}\rangle$ be a disjunctive LPAP and $S \subseteq H$. Given $\mathcal{P}$ and S, deciding if $S \in \mathit{Sol}(\mathcal{P})$ is Σ^P_2-complete.*

Proof. By Proposition 3, solution verification is polynomial-time equivalent to inference. Thus, the result immediately follows from Proposition 15.3. ∎

One can show that consistency checking is precisely as complex as solution verification analogously as done in the case of normal logic programs in Section 4.3.

Theorem 20 *To decide if Sol($\mathcal{P}$) $\neq \emptyset$ for a given disjunctive LPAP $\mathcal{P} = \langle H, M, LP, \models^b_{st} \rangle$ is Σ^P_2-complete.*

Proof. Σ^P_2-Membership: Guess $S \subseteq V_{\mathcal{P}}$ and $N \subseteq V_{\mathcal{P}}$, and verify that: (1) N is a stable model of $LP \cup S$, and, (2) $N \models M$. By virtue of Proposition 15, (1) is done by a single call to a co-NP oracle; (2) can be simply checked in polynomial time. Hence, the problem lies in Σ^P_2. Hardness for this class follows from Propositions 4 and 15. ∎

For necessity and relevance, we get as in the case of normal logic programs by Propositions 6 and 5 the same result as for (in)consistency checking.

Theorem 21 *Deciding if a given hypothesis is relevant for a given disjunctive LPAP $\mathcal{P} = \langle H, M, LP, \models^b_{st} \rangle$ is Σ^P_2-complete.*

Theorem 22 *Deciding if a given hypothesis is necessary for a given disjunctive LPAP$\mathcal{P}$= $\langle H, M, LP, \models^b_{st} \rangle$ is Π^P_2-complete.*

Finally, we consider the effect of restricting the set of solutions to those minimal with respect to inclusion. Again, we find a complexity increase by one level in the polynomial hierarchy compared to normal logic programs.

Theorem 23 *Deciding if a given hypothesis is $\subseteq$-relevant for a given disjunctive LPAP $\mathcal{P} = \langle H, M, LP, \models^b_{st} \rangle$ is Σ^P_3-complete.*

Proof. Σ^P_3-*Membership.* By Theorem 20 consistency is in Σ^P_2. Hence, by Proposition 7, $\subseteq$-relevance is in $\mathrm{NP}^{\Sigma^P_2} = \Sigma^P_3$.
Σ^P_3-*Hardness.* We demonstrate that the complementary problem, i.e., $\subseteq$-dispensability, is hard for Π^P_3. We provide a transformation from checking if a QBF $\Phi = \forall Z \exists X \forall Y E$ is in $\mathrm{QBF}_{3,\exists}$, where $Z = \{z_1, \dots, z_o\}$, $X = \{x_1, \dots, x_n\}$, and $Y = \{y_1, \dots, y_m\}$. As above, we assume that E is in 3DNF, $E = D_1 \vee \cdots \vee D_r$ and each $D_i = L_{i,1} \wedge L_{i,2} \wedge L_{i,3}$ is a conjunction of literals $L_{i,j}$ over atoms from $Z \cup X \cup Y$.
Consider the disjunctive *LPAP* $\mathcal{P} = \langle Z \cup \{w\}, \{\ w\}, LP_1, \models^b_{st} \rangle$, where LP_1 is the disjunctive logic program $LP(\exists X \forall Y E)$, which has been already used in the proof of Theorem 16.

We claim that Φ is valid if and only if w is $\subseteq$-dispensable for $\mathcal{P}$.

Given $Z1 \subseteq Z$, we can identify $Z1$ with the truth assignment to Z which maps into *true* all and only the elements from $Z1$. From Lemma 2, it follows that $\Phi' = \exists X \forall Y E$ is true on $Z1$ if and only if $LP_1 \cup Z1 \models^b_{st} w$. Since $\Phi = \forall Z \Phi'$, we have that Φ is valid iff $\forall Z1 \subseteq Z$, $LP_1 \cup Z1 \models^b_{st} w$. Therefore, Φ is valid iff w is $\subseteq$-dispensable for $\mathcal{P}$. The claim is proven.
Thus, $\subseteq$-dispensability under inference $\models^c_{st}$ is Π^P_3-hard, and, consequently, $\subseteq$-relevance under $\models^c_{st}$ is hard for the dual class Σ^P_3. ∎

5.2.3 Minimal model semantics

Our proofs of the results on disjunctive abduction problems using stable models allow to establish similar complexity results for abduction based on minimal models. We denote by $\models^b_{mm}$ and $\models^c_{mm}$ brave and cautious inference from the minimal models of a disjunctive logic program, respectively. Notice that any disjunctive logic program has some minimal model, and hence cautious inference as used here and classical cautious inference coincide.

One can easily check that the upper complexity bounds for disjunctive stable models carry over to minimal models as well. Moreover, since $\models^c_{mm}$ resorts to classical cautious minimal model inference, for deciding whether a set S is a solution for a disjunctive *LPAP* $\langle H, M, LP, \models^c_{mm}\rangle$, we get even a smaller bound of Π^P_2 compared to D^P_2 for the same problem using stable models.

To derive the matching hardness results in all cases but the mentioned exception, it is sufficient to modify the abduction problems in the proofs of the results on abduction using disjunctive stable models such that the programs are positive. (Recall that the stable models of a positive program coincide with its minimal models.)

In the proofs of Proposition 17 and Theorem 20, respectively, we can assume that the disjunctive programs are positive; thus, these constructions work also when minimal models are used. In all the other proofs, we modify the constructed programs by eliminating negative literals as follows.

(1) Elimination of literals $\neg z_i$: Replace each occurrence of $\neg z_i$ in a rule by z'_i; add rules $r_i \leftarrow z_i$ and $r_i \leftarrow z'_i$ to the program (r_i is a new atom); add z'_i to the set H, and add r_i to the set M.

(2) Elimination of $\neg w$: Replace the rule $q \leftarrow \neg w$ by $q \vee q' \leftarrow w$ (q' is a new atom); remove q from the set M and add $\neg q'$ instead.

The programs in the abduction problems resulting from these steps are positive. Informally, step (1) introduces an atom z'_i for the complement of z_i, and at least one of both must appear in each solution. If both z_i and z'_i appear in a solution, then some of them can be discarded and the resulting set is another solution of the abduction problem. Step (2) hinges on the fact that if q, which must be provable from the stable models of the program augmented by the solution, has to be derived in a stable model from the rule $q \leftarrow \neg w$, then w must be false in it, and thus, given the replacement clause, q' will be false in a stable model; if q is derived from another clause, then clearly q' will be false.

Formally, one can show that in each case the employed reasoning task has the same answer on the original problem and the modified problem. Thus, the hardness results for abduction using disjunctive stable models carry over to the minimal model semantics. Proving Π^P_2-hardness of solution verification for *LPAP*s with inference $\models^c_{st}$ is easy. Thus, we have the following result.

Theorem 24 *The complexity of each of the reasoning tasks from above (consistency, relevance, necessity etc) on disjunctive LPAPs using inference $\models^x_{mm}$, where $x \in \{b, c\}$, coincides with the complexity of the same reasoning task on disjunctive LPAPs using inference $\models^x_{st}$, with* *the exception of solution verification for $\models^c_{mm}$, which is Π^P_2-complete. Moreover, hardness holds in each case for the class of positive programs.*

Note that minimal model inference is, contrary to well-founded and stable model inference, independent of the syntactical form of program clauses. For example, the clauses $a \leftarrow \neg b$, $b \leftarrow \neg a$, and $a \vee b$ all have the same consequences under minimal model inference; this is obviously not true for well-founded and stable model inference. A consequence of syntax independence is the following result.

Theorem 25 *For the abductive reasoning tasks above (solution verification, consistency, relevance etc) on disjunctive LPAPs using $\models^x_{mm}$ inference, $x \in \{b, c\}$, hardness holds already on instances with normal (i.e., nondisjunctive) programs.*

Proof. Observe that atoms in the head of a disjunctive rule can be negated and moved to the body without a change in the classical models of the rule. Thus, under inference $\models^x_{mm}$, $x \in \{b, c\}$, each disjunctive program can be efficiently rewritten to an equivalent normal program. By the results above, this proves the theorem. ∎

6 Abduction from Function-free First Order Programs

In this section, we generalize the abduction model to the (function-free) first order case, i.e., logic programs with variables (also called *datalog* programs), and we discuss the complexity of this case.

We start by briefly recalling syntax and semantics of programs with variables.

A *function-free first order rule* is a formula

$$A \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_l, \quad 1 \leq n, 0 \leq m, l, \tag{2}$$

such that each A, B_i, and C_j is an atom $p(t_1, \ldots, t_a)$ of an a-ary predicate p, $a \geq 0$, where each term $t_1, \ldots t_a$, is either a variable or a constant; a *function-free logic program* is a finite set of such rules. Note that a function-free logic program resorts to a propositional program if all predicates in its rules are 0-ary.

The semantics of a function-free logic program LP is defined by first instantiating LP over the sets of constants that occur in LP and by then applying to the grounded program, which is regarded as a propositional program, the definitions given in the propositional case. Thus, the definitions of well-founded and stable model semantics given in Sections 2.1 and 5.1 are generalized to function-free logic programs.

Now, we generalize the concepts of logic programming abduction problem and solution.

Definition 4 *A function-free logic programming abduction problem (FLPAP) $\mathcal{P}$ consists of a tuple $\langle H, M, LP, \models \rangle$, where H is a finite set of predicates called hypotheses, M is a finite set of ground literals called manifestations, LP is a function-free logic program, and $\models$ is an inference operator.*

Let $\mathcal{P} = \langle H, M, LP, \models\rangle$ be a FLPAP, and let S be a subset of the ground instantiation of the predicates in H with the constants from $LP \cup M$. Then, S is a solution (or explanation) to $\mathcal{P}$ iff $LP \cup S \models M$.

The set of all solutions to $\mathcal{P}$ is denoted by $Sol(\mathcal{P})$. ∎

The properties of relevance and necessity of a ground atom $g(\mathbf{a})$, $g \in H$, for $\mathcal{P}$ are defined analogous to the propositional case, and the computational problems *relevance* and *necessity* adapted accordingly.

Example 10 Consider the following logic program LP, describing a game where each player is identified by its position a a player wins if he can move and its opponent has no moves – this program has been used by Kolaitis to prove that some interesting queries can not be expressed by stratified programs [40].

$$win(X) \leftarrow move(X, Y) \wedge \neg win(Y) \ .$$

The legal moves are represented by facts defining the predicate *move*. Let $\mathcal{P}$ be the *LPAP* $\langle H, M, LP, \models_{wf}\rangle$, where LP is the program above, the set of abducible predicates is $H = \{move, player\}$, and the set of manifestation is $M = \{player(a), player(b), player(c), win(a)\}$. The abduction problem $\mathcal{P}$ asks for configurations of the game (i.e., set of moves), where a, b, and c are the players and a is the winner; each abductive solution of $\mathcal{P}$ identifies such a configuration. For instance, $S_1 = \{move(c, a),\ move(a, b),\ player(a),\ player(b),\ player(c)\}$ is a solution, while $S_2 = S_1 \cup \{move(b, c)\}$ is not.

Notice that under classical inference, S_1 is not a solution, and does classical inference leads to different conclusions. However, S_1 is a solution if we consider the completion of LP. To highlight the difference between $\models_{wf}$ and classical inference from the completion, consider $S_3 = \{move(a, b),\ move(a, c),\ move(b, c),\ move(c, b)\}$. Then, S_3 is not a solution under $\models_{wf}$, but under classical inference from the completion of LP. ∎

Notice that in the above definition, the hypotheses are predicates rather than ground facts. One might argue that, in certain cases, it would be more appropriate to have explicitly a set of abducible ground facts from which a solution must be formed, e.g., if a certain ground fact should not appear in a solution. Technically, this more restricted form of abduction can be easily emulated in our framework. For each abducible ground fact $p(a)$, introduce a new 0-ary predicate (propositional letter) $h_{p(a)}$; put $h_{p(a)}$ into the abducible predicates H, and add to the program LP the rule $p(a) \leftarrow h_{p(a)}$. Then, for all considered logic programming semantics (as well as for any other reasonable semantics), the solutions of the obtained abduction problem (which consist of subsets of H), correspond in the obvious way to the solutions from the set of abducible ground facts.

When we do abduction from a function-free program, two interesting issues emerge concerning the computational complexity. The first one is answering the questions for abduction problems restricted to instances where the rules with nonempty body are fixed (i.e., the hypotheses, the manifestations, and the elementary facts in the program may change and may be regarded as "data"). This restriction corresponds to abductive reasoning with a static background knowledge on the relationship between elementary facts. We refer to the complexity of this problem as the *data complexity* of abduction from function-free programs;

The second issue is answering the questions we described in Section 3 (relevance, necessity, etc.) for an arbitrary given abduction problem (i.e., also the background knowledge may change); we refer to this as *combined complexity*. The notions of data and combined complexity have been introduced by Vardi [78] in the context of query languages.

Concerning the data complexity, we observe the following.

Proposition 18 *For each type of considered reasoning problem on a FLPAP $\mathcal{P} = \langle H, M, LP, \models \rangle$, the complexity of the same type of reasoning problem on a propositional LPAP is an upper bound for data complexity.*

Proof. Let B denote those predicates which occur in some rule of LP with nonempty body. A reasoning problem on $\mathcal{P}$ can be efficiently transformed into an equivalent reasoning problem $\mathcal{P}'$ of the same type as follows.

Let $LP1$ be the instantiation of $LP \cup M$, and let for each ground atom $p(\mathbf{a})$ be $x_{p(\mathbf{a})}$ a new propositional atom. Now, proceed as follows: Remove from H all predicates that are not in B, remove from LP all rules with empty body, and remove from M all literals that involve predicates not in B; let $H1$, $M1$, and $LP1$ denote the results of these operations, respectively. Then, add for each literal $p(\mathbf{a})$ (resp. $\neg p(\mathbf{a})$) from M that has been removed to $M1$ the literal $x_{p(\mathbf{a})}$ (resp. $\neg x_{p(\mathbf{a})}$); if the rule $p(\mathbf{a}) \leftarrow$ is in LP_g, then add $x_{p(\mathbf{a})} \leftarrow$ to $LP1$, and if $p \in H$, then add $x_{p(\mathbf{a})}$ to $H1$. Finally, if the considered reasoning task involves a ground hypothesis $g(\mathbf{a})$ (as in relevance etc.) such that g occurs not in B, then replace $g(\mathbf{a})$ with $x_{g(\mathbf{a})}$ and add $x_{g(\mathbf{a})}$ to H_1, and add for each constant a in LP_g to $LP1$ the fact $u(a) \leftarrow$, where u is a fresh unary predicate. Let $H2$, $M2$, and $LP2$ be the result of these steps. Basically, we have replaced ground atoms that do not interfere with proper rules of LP by propositional atoms, and removed hypotheses and nonground facts that are not needed to answer the reasoning task. Notice that in $LP2 \cup M2$ the same constants occur as in $LP \cup M$.

It holds that the reasoning problem (with possibly modified ground hypothesis) on $\mathcal{P}' = \langle H2, M2, LP2, \models \rangle$ yields the same answer as the reasoning problem on $\mathcal{P}$.

For fixed rules with nonempty head in LP, the size of the instantiation of $LP2 \cup M2$ over its constants is polynomial, as is the size of the instantiation of $H2$ over these constants. Hence, it follows that the complexity of each type of reasoning problem on the first order *LPAP* $\mathcal{P}$ is bounded by the complexity of the same type of reasoning problem in the propositional case. ∎

On the other hand, we have that complexity in the propositional case also provides a matching lower bound for the data complexity. To establish this, we describe how hard instances of the problems in the propositional case can be reduced to data complexity in the first order case.

The key observation is that in the propositional case, all the problems from above remain as hard as in the general case if the program of the *LPAP* contains a rule *true* $\leftarrow$ and each other rule in the program is of the form

$$a \leftarrow b_1 \wedge \cdots \wedge b_i \wedge \neg b_{i+1} \wedge \cdots \wedge \neg b_3, \qquad \text{for some } i \in \{0, \ldots, 3\}; \tag{3}$$

we call such a *LPAP 3-bounded.* This can be established as each positive rule $a \leftarrow b_1 \wedge b_2 \wedge b_3 \wedge \cdots \wedge b_n$ can be equivalently replaced by two rules $a \leftarrow c \wedge b_3 \wedge \cdots \wedge b_n$ and $c \leftarrow b_1 \wedge b_2$, and *true* can be used to augment rule bodies with less than three literals; notice that in all hardness proofs above, negative literals occur only in rules with at most three literals in the body.

For each 3-bounded *LPAP* $\mathcal{P} = \langle H, M, LP, \models \rangle$ we define a *FLPAP* $\mathcal{P}_{fo} = \langle \{s\}, M', LP', \models \rangle$ as follows. The predicate $s(X)$ intuitively means that X belongs to the solution of $\mathcal{P}$. In the program LP', the constants are the propositional atoms $V_{\mathcal{P}}$ of $\mathcal{P}$. Each rule of the form (3) is encoded by the atom $r_i(a, b_1, b_2, b_3)$, where $i \in \{0, \dots, 3\}$ is the number of positive literals $b_1, \dots, b_i$ in the body. The rules of LP' are the following:

(1) $t(true) \leftarrow$; $\quad$ ($t(X)$ intuitively means that X is true)

(2) $t(X) \leftarrow s(X)$;

(3) $t(X_0) \leftarrow t(X_1) \wedge \cdots \wedge t(X_i) \wedge \neg t(X_{i+1}) \wedge \cdots \wedge \neg t(X_3) \ \wedge \ r_i(X_0, X_1, X_2, X_3)$,
for each $i = 0, \dots, 3$; and

(4) $r_i(a, b_1, \dots, b_3) \leftarrow$, such that $a \leftarrow b_1 \wedge \cdots \wedge b_i \wedge \neg b_{i+1} \wedge \cdots \wedge \neg b_3$ occurs in LP.

The set of manifestations is $M' = \{t(a) | a \in M\} \cup \{\neg t(a),\ \neg s(b) | \neg a \in M, b \in V_{\mathcal{P}} - H\}$.

Example 11 Consider the *LPAP* $= \langle \{b, d\}, \{a, \neg c\}, \{a \leftarrow b \wedge \neg d\ , \models_{wf} \rangle$. The 3-bounded *LPAP* $\mathcal{P} = \langle \{b, d\}, \{a, \neg c\}, \{true \leftarrow\ , a \leftarrow true \wedge b \wedge \neg d\ , \models_{wf}\} \rangle$ has the same solutions; in fact it has the unique solution $\{b\}$. The components LP' and M' in the *FLPAP* $\mathcal{P}_{fo}$ are as follows. Besides the rules (1)–(3), LP' contains only the fact $r_2(a, true, b, d) \leftarrow$. The set $V_{\mathcal{P}}$ of atoms in $\mathcal{P}$ is $\{a, b, c, d, true\}$; hence, $M' = \{t(a), \neg t(c), \neg s(a), \neg s(c), \neg s(true)\}$. The reader may verify that $\{s(b)\}$ is the unique solution to $\mathcal{P}_{fo}$. ∎

The following property of $\mathcal{P}_{fo}$ is not difficult to establish.

Lemma 3 *Let $\mathcal{P}$ be a 3-bounded LPAP with inference $\models_{wf}$, $\models^b_{st}$, or $\models^c_{st}$, and let $\mathcal{P}_{fo}$ be the FLPAP as defined above. Then, Sol($\mathcal{P}$) $= \{\{a_1, \dots, a_n\} | \{s(a_1), \dots, s(a_n)\} \in$ Sol($\mathcal{P}_{fo}$)$\}$.*

Notice that $\mathcal{P}_{fo}$ can be constructed from $\mathcal{P}$ in logarithmic space, and that the rules in LP' with nonempty body are the same for every $\mathcal{P}$. Thus, from Proposition 18 and Lemma 3 we obtain the following result.

Theorem 26 *For any reasoning task on any type of abduction problem considered above, the data complexity of the first-order case coincides with the complexity in the propositional case.*

We remark that the same complexity results can be shown for the restriction to instances where everything except the manifestations is kept fixed.

As with many logical query languages, the combined complexity is much higher than the data complexity. The exponential analogue of each complexity class –EXP for P, NEXP for NP, NEXP$^{\mathrm{NP}}$ for Σ_2^P etc– is an immediate upper bound, and is also in each case a matching lower bound. For the case with polynomial data complexity, this follows from folklore results about datalog. Note that the exponential analogue of $\Delta^P_{k+1}[O(\log n)]$ is $\mathrm{EXP}^{\Sigma^P_k}[Pol(n)]$, i.e., exponential time with a polynomial number of calls to an Σ^P_k oracle; this class coincides with $\mathrm{PSPACE}^{\Sigma^P_k}$, i.e., polynomial space with an oracle for Σ^P_k, if the oracle queries may be arbitrarily large [27]. A detailed analysis of the combined complexity of logic programming abduction is carried out in [27].

Disjunctive propositional logic programs can be generalized to function-free logic programs in the same way as normal logic programs; from above, it should be clear how syntax and semantics of such programs are defined. Moreover, it is straightforward to extend the definition of function-free logic program abduction problem to allow for disjunctive function-free programs; we omit the details.

As in the latter case, one can show that the results of the reasoning tasks for the disjunctive propositional case carry over to the data complexity of the disjunctive function-free case. The arguments for showing these to be upper bounds are similar, and the proof of the matching lower bounds is based on a proper modification of the above described translation of abduction from propositional programs into abduction from function-free first order programs. Similarly, the combined complexity of disjunctive abduction problems is exponentially higher than the data complexity.

7 Related Work

In this section we briefly overview some works on abductive logic programming.

The best known definition of abductive logic programming is due to Kakas and Mancarella. In [37], they define abduction from (normal) logic programs under the brave modality of the stable model semantics; integrity constraints can also be imposed on the solutions. In our framework, their proposal can be represented by *LPAP*s of the form $\langle H, M, LP, \models^b_{st}\rangle$, where LP is a logic program (integrity constraints can be simulated as shown in Section 3). Thus, the various abductive problems for [37] have exactly the computational complexity determined in Section 4.3.

Console, Theseider Dupré and Torasso in [14] analyze abductive problems $\langle H, M, LP, \models\rangle$ where LP is a hierarchical (i.e., non recursive) program and $\models$ is the entailment operator based on Clark's program completion.

The relation of abduction to negation as failure has been investigated by Eshghi and Kowalski [22], Inoue [29], and Iwayama and Satoh [32], respectively. Under some syntactic conditions, a 1-1 correspondence between the solutions of an abductive problem and the answer sets of an extended logic program obtained from it has been proven.

In a recent work [4], Baral and Gelfond define a notion of *entailment* for abductive logic programs. In some sense their approach is similar to *LPAP*s of the form $\langle H, M, LP, \models^c_{st}\rangle$, but they impose stronger conditions.

An important bunch of work on abductive logic programming concerns the definition of *abductive procedures*, which compute an abductive explanation of a given manifestation [22, 35, 32, 70, 15, 17, 7].

The seminal work in this area is due to Eshghi and Kowalski [22], who consider abduction based on stable model semantics and provide a proof procedure to compute an abductive solution. This procedure has been then extended by Kakas and Mancarella to manipulate arbitrary abducibles. Both [22] and [35] procedures guarantee correctness only for call-consistent logic programs. In [70], the Sato and Iwayama give an abductive procedure which is correct for general logic programs provided that consistency (i.e., the existence of an abductive solution) is guaranteed. Other abductive procedures [15, 17, 7] consider semantics different from stable models (like, e.g., the completion semantics [13]) on the logic program.

Our work presents some relationships to the papers on abductive procedures. For instance, the use of metalevel constraints, adopted in [22] (on the *demo* predicate) to eliminate negation from the logic program, could be avoided by utilizing a technique similar to that shown in Section 5.2.3.

Moreover, in our framework most abductive procedures can be seen as a method to check the existence of solutions (i.e., $Sol(\mathcal{P}) \neq \emptyset$) of a *LPAP* $\mathcal{P} = \langle H, M, LP, \models^b_{st} \rangle$, where LP is a normal (non-disjunctive) logic program. From Theorem 15, this problem is NP-complete. As a consequence, from one hand we derive that no abductive procedure may (in the worst case) terminate in polynomial time (unless P = NP). On the other hand, this result indicates that an "optimal" abductive procedure can be implemented by a "flat" backtracking algorithm. (For an extended discussion on backtracking algorithms, see Section 8.) Unfortunately, even if most abductive procedures are based on backtracking, they usually consist of two mutually recursive functions; it is thus difficult to state whether they comply the above mentioned upper complexity bound.

Critical overviews of several approaches to abductive logic programming have been given by Kakas, Kowalski and Toni [34] and by Inoue [30].

To our knowledge, complexity aspects of abductive logic programming have been less addressed than semantical aspects. In [19] Eiter and Gottlob present a detailed analysis of the complexity of abduction from classical propositional theories, which has also been discussed for restricted cases by Selman and Levesque [74], Bylander [8], and Friedrich *et al.* [24], who focussed on Horn and definite Horn theories and the $\subseteq$-minimality preference criterion. In this context, Eshghi has identified in [23] a subclass of Horn theories for which finding a $\subseteq$-minimal solution is tractable.

The results of [19] and this paper show that abduction from logic programs has complexity quite different than abduction from classical theories. Notice that in the abductive framework of [19], the inference operator used is $\models_P$, i.e., classical logical consequence; moreover, the manifestations are there restricted to propositional atoms. For a comparison with the results of this paper, we consider $\models^c_{st}$ inference, since it corresponds in nature to $\models_P$, and omit the other inference operators $\models_{wf}$, $\models^b_{st}$ considered here.

The results shows that in the general case (clausal theories resp. disjunctive logic programs), logic programming abduction is for each reasoning task harder (at least Σ^P_3/Π^P_3-hard) than classical abduction (which is Σ^P_2/Π^P_2-complete) and can get twice as hard (Σ^P_4-complete for $\subseteq$-relevance). In classical abduction, relevance and necessity have complementary complexity (Σ^P_2-complete resp. Π^P_2-complete), and $\subseteq$-minimality has no effect on the complexity. In logic programming abduction, however, $\subseteq$-minimality does affect relevance, and leads to an increase in the complexity by one level of the polynomial hierarchy.

In [19], also classical abduction from definite Horn theories has been studied, which correspond 1-1 to Horn programs. However, due to different possible sets of manifestations in the frameworks, the results from [19] do not carry over automatically to the framework considered here; in fact, several corresponding reasoning tasks have different complexity. In particular, consistency, relevance, necessity and $\subseteq$-necessity in the framework of [19] are polynomial, while they are NP-complete in the framework of this paper. Hence, the results of the present paper show as a byproduct that the slight generalization of the framework in [19] by allowing also negative literals as manifestations causes intractability for a number of tractable abductive reasoning tasks.

Abduction from normal logic programs and first-order programs has no corresponding counterpart in [19], and thus we can not compare respective results.

Sakama and Inoue have investigated the semantics and complexity of abductive logic programming in [31, 68]. They determined in [31] the complexity of the existence of abductive solutions on disjunctive programs for the particular case where the inference operator is $\models^{b}_{st}$ (stable brave), and showed that the problem is Σ^P_2-complete. The NP-completeness result for the same problem for normal logic programs is contributed in [68] to [29, 32].

In [68], the authors also consider abduction from logic programs under the *possible model* semantics [67, 69]. In particular, they show that existence of an abductive solution from a disjunctive logic program is NP-complete under this semantics. Thus, from the computational side, as with deduction [68, 20], the possible model semantics is more attractive than the disjunctive stable model semantics.

Even the effective computation of abductive explanation has been addressed in several works [17, 15, 22, 32, 38], where suitable extensions of the *SLDNF* procedure have been designed for abduction (these procedures return an abductive explanation of the given query).

Finally, the use of abductive logic programming for dealing with incomplete information has been discussed by Denecker and De Schreye [16] and by Baral and Gelfond [4].

For related complexity results in the area of logic programming and nonmonotonic-reasoning, cf. [11, 71, 10, 73]. The complexity of abduction in a set-covering based rather than logic-based framework has been analyzed in [9], where polynomial and NP-complete abductive problems are discussed.

Most recently, our work in this paper has been complemented by complexity results for abduction from logic programs in the presence of functions [47].

8 Conclusion

Abduction from logic programs has become a topic of growing interest recently. We have described in this paper a basic framework of abduction from logic programs, and we have performed a detailed study of the computational complexity of the main decisional problems that emerge in abductive reasoning (consistency, relevance of a hypothesis, and necessity of a hypothesis). The analysis considers the most acknowledged proposals for the semantics of logic programs, namely the well-founded semantics and the stable model semantics (in the usual variants of brave and cautious inference), and the stable as well as the minimal model semantics for disjunctive programs. Moreover, it took into account commonly applied criteria for reducing the space of all solutions by means of a preference ($\preceq$) to those which are minimal with respect to inclusion ($\subseteq$) and cardinality ($\leq$), respectively.

The complexity results for the case of propositional logic programs are summarized in Tables 1–6. They cover the lower end of the polynomial hierarchy, refined by classes $\Delta_k^P[O(\log n)]$, up to Σ_4^P. Each entry represents completeness for the respective class; in particular, P means logspace-completeness for P. (Notice that all polynomial time transformations in this paper are computable in logarithmic space; in fact, all results can be proven under logspace-reductions as well).

The results reflect the conceptual complexity of combining the different common semantics and solution preference criteria. They show that abduction from logic programming, a topic of increasing interest, provides a rich variety of complete problems for a number of slots within the polynomial hierarchy. This gives support to the original belief that the polynomial hierarchy would be useful for classifying computational problems, cf. [75].

Briefly, the following conclusions can be drawn. Abduction from logic programs has the same complexity for well-founded inference ($\models_{wf}$) and the brave variant of stable inference ($\models_{st}^b$), excepting the verification of a given solution, which is polynomial under the former and NP-complete under the latter. Notice that the hardness results for well-founded inference hold for stratified programs and for negation-free programs (i.e., Horn programs) as well, with the exception of deciding $\subseteq$-relevance, which is "only" NP-complete for Horn programs. Since the semantics of Horn programs is clear, this shows that, loosely speaking, the well-founded semantics does mostly not increase the inherent complexity of abduction from logic programs, which makes it attractive.

For the appropriate cautious variant of stable inference (cautious inference $\models_{st}^c$), the complexity is in each case precisely one level higher in the polynomial hierarchy than for $\models_{st}^b$, again excepting solution verification, which has marginally higher complexity; this is noticeable since usually, the brave and the cautious variant of a reasoning task in a nonmonotonic formalism are of dual complexity (typically Σ_2^P and Π_2^P). Furthermore, allowing for disjunctive rules leads to another increase by one level in the polynomial hierarchy. This is explained by the fact that testing the stable model property migrates from polynomial time to co-NP-completeness. The effect of the solution preference criteria on the complexity meets the intuitive expectations. Imposing subset-minimality ($\subseteq$) leads in general to an increase by a full level in the polynomial hierarchy, while minimum cardinality ($\leq$) leads merely to a minor increase, which can be interpreted as the need of computing the size of a smallest solution for solving the problem. It is worth mentioning that for abduction using classical semantics, imposing $\subseteq$ does not lead to a complexity increase, cf. [19].

All results for disjunctive logic programming abduction in Tables 4–6 remain valid for the restriction to positive disjunctive logic programs, where the stable semantics coincides with the minimal model semantics. The results for the case of propositional programs carry over to the data complexity in the case of function-free first order logic programs. The combined complexity, which was not dealt with here in detail, parallels the data complexity with the corresponding exponential analogues (EXP for P, NEXP for NP etc); a treatment thereof appears in [27].

LPAP $\mathcal{P} = \langle H, M, LP, \models \rangle$	Deciding $S \in \mathit{Sol}(\mathcal{P})$	Deciding $\mathit{Sol}(\mathcal{P}) \neq \emptyset$
well-founded inference ($\models_{wf}$)	P	NP [74]
stable brave inference ($\models^b_{st}$)	NP	NP [29, 32]
stable cautious inference ($\models^c_{st}$)	$\mathrm{D}^{\mathrm{P}}_1$	$\mathbf{\Sigma}^P_2$

Table 1: Complexity of solution verification and consistency checking

Logic Programming Abduction	Deciding whether $h \in H$ is $\preceq$-relevant for $\mathcal{P}$		
LPAP $\mathcal{P} = \langle H, M, LP, \models \rangle$	$=$ (*no preference*)	$\subseteq$	$\leq$
well-founded inference ($\models_{wf}$)	NP	$\mathbf{\Sigma}^P_2$	$\mathbf{\Delta}^P_2[O(\log n)]$
stable brave inference ($\models^b_{st}$)	NP	$\mathbf{\Sigma}^P_2$	$\mathbf{\Delta}^P_2[O(\log n)]$
stable cautious inference ($\models^c_{st}$)	$\mathbf{\Sigma}^P_2$	$\mathbf{\Sigma}^P_3$	$\mathbf{\Delta}^P_3[O(\log n)]$

Table 2: Complexity of deciding relevance

Logic Programming Abduction	Deciding whether $h \in H$ is $\preceq$-necessary for $\mathcal{P}$		
LPAP $\mathcal{P} = \langle H, M, LP, \models \rangle$	$=$ (*no preference*)	$\subseteq$	$\leq$
well-founded inference ($\models_{wf}$)	co-NP	co-NP	$\mathbf{\Delta}^P_2[O(\log n)]$
stable brave inference ($\models^b_{st}$)	co-NP	co-NP	$\mathbf{\Delta}^P_2[O(\log n)]$
stable cautious inference ($\models^c_{st}$)	$\mathbf{\Pi}^P_2$	$\mathbf{\Pi}^P_2$	$\mathbf{\Delta}^P_3[O(\log n)]$

Table 3: Complexity of deciding necessity

Disj. *LPAP* $\mathcal{P} = \langle H, M, LP, \models\rangle$	Deciding $S \in \mathit{Sol}(\mathcal{P})$	Deciding $\mathit{Sol}(\mathcal{P}) \neq \emptyset$
stable brave inference ($\models^b_{st}$)	$\mathbf{\Sigma}^P_2$	$\mathbf{\Sigma}^P_2$ [31]
stable cautious inference ($\models^c_{st}$)	D^P_2	$\mathbf{\Sigma}^P_3$

Table 4: Complexity of solution verification and consistency for disjunctive *LPAP*s

Disj. Logic Programming Abduction	Deciding whether $h \in H$ is $\preceq$-relevant for $\mathcal{P}$	
Disj. *LPAP* $\mathcal{P} = \langle H, M, LP, \models\rangle$	$=$ (*no preference*)	$\subseteq$
stable brave inference ($\models^b_{st}$)	$\mathbf{\Sigma}^P_2$	$\mathbf{\Sigma}^P_3$
stable cautious inference ($\models^c_{st}$)	$\mathbf{\Sigma}^P_3$	$\mathbf{\Sigma}^P_4$

Table 5: Complexity of deciding relevance for disjunctive *LPAP*s

Disj. Logic Programming Abduction	Deciding whether $h \in H$ is $\preceq$-necessary for $\mathcal{P}$	
Disj. *LPAP* $\mathcal{P} = \langle H, M, LP, \models\rangle$	$=$ (*no preference*)	$\subseteq$
stable brave inference ($\models^b_{st}$)	$\mathbf{\Pi}^P_2$	$\mathbf{\Pi}^P_2$
stable cautious inference ($\models^c_{st}$)	$\mathbf{\Pi}^P_3$	$\mathbf{\Pi}^P_3$

Table 6: Complexity of deciding necessity for disjunctive *LPAP*s

Our complexity results have the following impact on the design of algorithms. First of all, they provide a complete picture of efficient (polynomial time) intertranslatability between different reasoning tasks, and tell us which algorithms for specific abductive reasoning tasks can also be used to solve other abductive reasoning tasks, by applying an appropriate transformation of the input. Several efficient transformations are given in Secion 3.2.

Suppose now we want to implement an inference engine that is capable of performing a certain abductive reasoning task.

Consider a reasoning task at the first level of the polynomial hierarchy. If the problem is NP-complete, then we can implement a "flat" backtracking algorithm. For example, in case of stable model inference, Saccà's backtracking fixpoint procedure [65, 43] can be suitably adapted. Optimization techniques may be applied to cut down branches in the computation tree, but if the algorithm should work on all instances, it is unrealistic that we obtain a polynomial algorithm. Furthermore, from the general nature of NP-complete problems, we must expect to put some analytical and programming effort in order to obtain a feasible algorithm for a reasonably large class of instances. For the co-NP-complete reasoning tasks, we obtain may use backtracking algorithms that solve the complementary problem. Note that for the NP-complete reasoning tasks (but not for the co-NP-complete), randomized local search techniques [39, 42] could be applied that lead to success with high probability.

For $\Delta_2^P[O(\log n)]$-complete problems, a combination of binary search and flat backtracking may be used, following the scheme in the proof of Theorem 5. First, the size of a smallest solution is computed, and then only solution candidates of this size are considered. The backtracking part solves the oracle queries.

Next suppose we want to design an algorithm for a reasoning task at the second level, e.g. for a Σ_2^P-complete one. Then, double backtracking is possible, i.e., backtracking where at terminal nodes a co-NP-complete problem has to be solved (e.g., classical propositional inference); in case a library procedure is not available, the latter can be done by using flat backtracking for solving the complementary problem. Note, however, that unlike in case of an NP-complete reasoning task, an overall flat backtracking algorithm is not possible (unless $\Sigma_2^P = \mathrm{NP}$).

For problems at even higher levels, multiply nested backtracking is feasible; if the problem is complete for Σ_k^P, k-nested backtracking is possible, but not $(k-1)$-nested (unless the polynomial hierarchy collapses). Library procedures for subtasks (e.g., for $\models_{st}^b$, $\models_{st}^c$ inference) may be used and decrease the backtracking level accordingly. For the $\Delta_{k+1}^P[O(\log n)]$-complete problems a combination of binary search and k-nested backtracking is possible.

The level of the polynomial hierarchy at which an abductive reasoning problem resides should be kept in mind; it serves as a basic constraint of whether a proposed backtracking algorithm for that problem is correct or not.

In order to obtain polynomial time algorithms for abduction, the instances have to be restricted so that all sources of complexity are eliminated. The higher the level at which an abduction problem resides, the stronger is the restriction of the instances. It remains to identify polynomial classes of abduction problems. In particular, it would be interesting to see whether there are polynomial classes of abduction problems from the fourth level that are relevant in practice.

Acknowledgements The authors would like to thank the referees for their valuable and instructive comments that helped to improve the presentation of this paper. In particular, Section 3.2 was proposed by a referee; it greatly improves understanding of the relationships between the different abduction problems investigated, and simplified proofs.

References

[1] A. Aho and J. Ullman. Universality of Data Retrieval Languages. In *Proc. ACM Symposium on Principles of Programming Languages*, pp. 110–117, 1979.

[2] K. Apt and H. Blair. Arithmetic Classification of Perfect Models of Stratified Programs. In R. Kowalski and K. Bouwen, editors, *Proc. Fifth Joint International Conference and Symposium on Logic Programming (JICSLP '88)*, pp. 766–779. MIT Press, 1988.

[3] C. Baral, J. Lobo, and J. Minker. Generalized Disjunctive Well-Founded Semantics for Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 5:89–132, 1992.

[4] C. Baral and M. Gelfond. Logic Programming and Knowledge Representation *Journal of Logic Programming*, 19/20:73–148, 1994.

[5] C. Baral and V. S. Subrahmanian. Stable and Extension Class Theory for Logic Programs and Default Logics. *Journal of Automated Reasoning*, 8:345–366, 1992.

[6] N. Bidoit and C. Froidevaux. General Logic Databases and Programs: Default Semantics and Stratification. *Information and Computation*, 19:15–54, 1991.

[7] A. Brogi, E. Lamma, P. Mancarella, and P. Mello. An Abductive Framework for Extended Logic Programming. In *Proc. ICLP '95*, pp. 330–343, 1995.

[8] T. Bylander. The Monotonic Abduction Problem: A Functional Characterization on the Edge of Tractability. In *Proc. Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pp. 70–77, 1991.

[9] T. Bylander, D. Allemang, M. Tanner, and J. Josephson. The Computational Complexity of Abduction. *Artificial Intelligence*, 49:25–60, 1991.

[10] M. Cadoli and M. Lenzerini. The Complexity of Closed World Reasoning and Circumscription. *Journal of Computer and System Sciences*, 43:165–211, April 1994.

[11] M. Cadoli and M. Schaerf. A Survey of Complexity Results for Non-monotonic Logics. *Journal of Logic Programming*, 17:127–160, 1993.

[12] E. Charniak and P. McDermott. *Introduction to Artificial Intelligence.* Addison Wesley, Menlo Park, Ca, 1985.

[13] K.L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, Plenum Press, pp. 293–322, 1978.

[14] L. Console, D. Theseider Dupré, and P. Torasso. On the Relationship Between Abduction and Deduction. *Journal of Logic and Computation*, 1(5):661–690, 1991.

[15] M. Denecker and D. De Schreye. SLDNFA: An Abductive Procedure for Normal Abductive Logic Programs. In *Proc. JICSLP '92*, pp. 686–700. MIT Press, 1992.

[16] M. Denecker and D. De Schreye. Representing Incomplete Knowledge in Abductive Logic Programming. In *Proc. International Symposium on Logic Programming (ILPS '93)*, pp. 147–163. MIT Press, 1993.

[17] P. Dung. Negation as Hypotheses: An Abductive Foundation for Logic Programming. In *Proc. International Conference on Logic Programming (ICLP '91)*, pp. 3–17. MIT Press, 1991.

[18] P.M. Dung. On the Relation between Stable-and Well-founded Semantics of Logic Programs. *Theoretical Computer Science*, 105:7–25, 1992.

[19] T. Eiter and G. Gottlob. The Complexity of Logic-Based Abduction. *Journal of the ACM*, 42:3–42, 1995.

[20] T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 15(3/4): 289–323, 1995.

[21] T. Eiter and G. Gottlob and H. Mannila. Adding Disjunction to Datalog. *Proc. ACM Symposium on Principles of Database Systems (PODS '94)*, May 1994, pp. 267–278.

[22] K. Eshghi and R. Kowalski. Abduction Compared with Negation as Failure. In *Proc. ICLP '89*, pp. 234–254. MIT Press, 1989.

[23] K. Eshghi. A Tractable Class of Abduction Problems. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI '93)*, pp. 3–8, 1993.

[24] G. Friedrich, G. Gottlob, and W. Nejdl. Hypothesis Classification, Abductive Diagnosis, and Therapy. In *Proc. International Workshop on Expert Systems in Engineering*, pp. 69–78. LNCS/LNAI 462, 1990.

[25] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. Fifth Logic Programming Symposium*, pp. 1070–1080. MIT Press, 1988.

[26] M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

[27] G. Gottlob, N. Leone and H. Veith. Second-Order Logic and the Weak Exponential Hierarchies. Technical Report CD-TR 95/80, Christian Doppler Lab for Expert Systems, TU Wien, 1995. Extended abstract in: *Proc. Conference on Mathematical Foundations of Computer Science (MFCS '95)*, pp. 66–81, LNCS 969, 1995.

[28] J. R. Hobbs and M. E. Stickel. Interpretation as Abduction. In *Proc. 26th Annual Meeting of the Assoc. for Computational Linguistics*, pp. 95–103, 1988.

[29] K. Inoue. Extended Logic Programs with Default Assumptions. In *Proc. ICLP '91*, pp. 490–504. MIT Press, 1991.

[30] K. Inoue. Studies on Abductive and Nonmonotonic Reasoning. PhD thesis, Kyoto University, 1992.

[31] K. Inoue and C. Sakama. Transforming Abductive Logic Programs to Disjunctive Programs. In *Proc. ICLP '93*, pp. 335–353, 1993.

[32] N. Iwayama and K. Satoh. Computing Abduction using the TMS. In *Proc. ICLP '91*, pp. 505–518, 1991.

[33] J. Josephson, B. Chandrasekaran, J. J. W. Smith, and M. Tanner. A Mechanism for Forming Composite Explanatory Hypotheses. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17:445–454, 1987.

[34] A. Kakas and R. Kowalski and F. Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719–771, 1992.

[35] A. Kakas and P. Mancarella. On the Relation Between Truth Maintenance and Abduction. In *Proc. of PRICAI'90*, pp. 438–443, 1990.

[36] A. Kakas and P. Mancarella. Database Updates Through Abduction. In *Proc. Conference on Very Large Databases (VLDB '90)*, pp. 650–661, 1990.

[37] A. Kakas and P. Mancarella. Generalized Stable Models: a Semantics for Abduction. In *Proc. of ECAI-90*, pp. 385–391, 1990.

[38] A. Kakas and P. Mancarella. Stable Theories for Logic Programs. In *Proc. ILPS '91*, pp. 88–100, 1991.

[39] H. Kautz and B. Selman. An Empirical Study of Greedy Local Search for Satisfiability Testing. In *Proc. National Conference on Artificial Intelligence (AAAI '93)*, pp. 46–52. MIT Press, 1993.

[40] P.G. Kolaitis. The Expressive Power of Stratified Logics Programs. *Information and Computation 90*, 1991, pp. 50-66.

[41] K. Konolige. Abduction versus Closure in Causal Theories. *Artificial Intelligence*, 53:255–272, 1992.

[42] E. Koutsopias and Ch. Papadimitriou. On the Greedy Algorithm for Satisfiability. *Information Processing Letters*, 43:53–55, 1992.

[43] N. Leone, M. Romeo, P. Rullo, and D. Saccà, Effective Implementation of Negation in Database Logic Query Languages. In *LOGIDATA+: Deductive Database with Complex Objects*, LNCS 701, pp. 159–175, 1993.

[44] J. W., Lloyd. *Foundations of Logic Programming,* Springer-Verlag, 1987.

[45] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT Press, 1992.

[46] W. Lukasiewicz. *Non-Monotonic Reasoning.* Ellis Horwood Limited, Chichester, England, 1990.

[47] W. Marek, A. Nerode, and J. Remmel. On the Complexity of Abduction. In *Proc. IEEE Symposium on Logic in Computer Science (LICS '96)*, 1996. Forthcoming.

[48] W. Marek and M. Truszczyński. Autoepistemic Logic. *Journal of the ACM*, 38(3):588–619, 1991.

[49] W. Marek and M. Truszczyński. Computing Intersection of Autoepistemic Expansions. In A. Nerode, W. Marek, and V. Subrahmanian, editors, *Proc. First Int'l Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR '91)*, pp. 37–50, 1991. MIT Press.

[50] W. Marek and M. Truszczyński. *Nonmonotonic Logics.* Springer, 1993.

[51] J. McCarthy. Programs with Common Sense. In *Proc. of the Teddington Conference on the Mechanization of Thought Processes*, pp. 75–91, London, 1959. Her Majesty's Stationery Office.

[52] J. McCarthy. Circumscription – A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[53] D. McDermott and J. Doyle. Non-Monotonic Logic I. *Artificial Intelligence*, 13:41–72, 1980.

[54] J. Minker. On Indefinite Data Bases and the Closed World Assumption. In *Proc.* 6^{th} *Conference on Automated Deduction (CADE '82)*, pp. 292–308, LNCS 138, 1982.

[55] R. Moore. Semantical Considerations on Nonmonotonic Logics. *Artificial Intelligence*, 25:75–94, 1985.

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

[57] C. S. Peirce. Abduction and Induction. In J. Buchler, editor, *Philosophical Writings of Peirce*, chapter 11. Dover, New York, 1955.

[58] Y. Peng and J. Reggia. *Abductive Inference Models for Diagnostic Problem Solving.* Springer, 1990.

[59] F. Pirri, C. Pizzuti. Explaining Incompatibilities in Data Dictionary Design through Abduction. *Data and Knowledge Engineering*, North-Holland, 13:101–139,1994.

[60] D. Poole. Normality and Faults in Logic Based Diagnosis. In *Proc. IJCAI '89*, pp. 1304–1310, 1989.

[61] T. Przymusinski. On the Declarative and Procedural Semantics of Stratified Deductive Databases. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pp. 193–216. Morgan Kaufman, Washington DC, 1988.

[62] T. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing*, 9:401–424, 1991.

[63] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[64] K. Ross. The Well-Founded Semantics for Disjunctive Logic Programs. *Proc. First Int'l Conf. on Deductive and Object-Oriented Databases (DOOD-89)*, pp. 352–369. North-Holland, 1989.

[65] D. Saccá and C. Zaniolo. Stable Models and Nondeterminism in Logic Programs with Negation. In *Proc. ACM PODS '90*, pp. 305–218, 1990.

[66] D. Saccá. Deterministic and Nondeterministic Stable Model Semantics for Unbound DATALOG Queries, *Proc. 5th Int'l Conference on Database Theory (ICDT '95)*, LNCS 893, pp. 353–367, 1995.

[67] C. Sakama. Possible Model Semantics for Disjunctive Databases. In *Proc. First Int'l Conference on Deductive and Object-Oriented Systems (DOOD '89)*, pp. 337–351. North-Holland, 1989.

[68] C. Sakama and K. Inoue. On the Equivalence between Disjunctive and Abductive Logic Programs. In *Proc. ICLP '94*, pp. 88–100, 1994.

[69] C. Sakama and K. Inoue. An Alternative Approach to the Semantics of Disjunctive Logic Programs and Deductive Databases. In *Journal of Automated Reasoning*, 13:145–172, 1994.

[70] K. Satoh, N. Iwayama. A Query Evaluation Method for Abductive Logic Programming. In *Proc. JICSLP '92*, pp. 671–685. MIT Press, 1992.

[71] M. Schaerf. Logic Programming and Autoepistemic Logics: New Relations and Complexity Results. In *Proc. Italian AI Conference (IA*AI) '93*, LNCS 728, 1993.

[72] J. Schlipf. The Expressive Powers of Logic Programming Semantics. *Journal of Computer and System Sciences*, 51(1):64–86, 1995. Preliminary version in: *Proc. ACM PODS '90*, pp. 196–204.

[73] J. Schlipf. Complexity and Undecidability Results in Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 15(3/4), 1995.

[74] B. Selman and H. J. Levesque. Abductive and Default Reasoning: A Computational Core. In *Proc. AAAI '90*, pp. 343–348. MIT Press, 1990.

[75] L. Stockmeyer. Classifying the Computational Complexity of Problems. *Journal of Symbolic Logic*, 52(1):1–43, 1987.

[76] L. Stockmeyer and A. Meyer. Word Problems Requiring Exponential Time. In *Proc. 5th ACM Symposium on Theory of Computing (STOC '73)*, pp. 1–9, 1973.

[77] A. Van Gelder, K. Ross, and J. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.

[78] M. Vardi. Complexity of Relational Query Languages. In *Proc. 14th ACM STOC*, pp. 137–146, 1982.