Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

# Navigating and Querying Answer Sets: How Hard Is It Really and Why?

**Dominik Rusovac**[1], **Markus Hecher**[2], **Martin Gebser**[3], **Sarah Alice Gaggl**[1], **Johannes K. Fichte**[4]
[1]TU Dresden
[2]Massachusetts Institute of Technology
[3]University of Klagenfurt
[4]Linköping University
firstname.lastname@tu-dresden.de, hecher@mit.edu, martin.gebser@aau.at, johannes.fichte@liu.se

**Abstract**

Answer set programming is a popular declarative paradigm with countless applications for modeling and solving combinatorial problems. We can view a program as a knowledge database compactly representing conditions for solutions. Often we are interested in reasoning about solutions of filtering answer sets. At the heart of these questions is brave and cautious reasoning. For browsing answer sets, we combine both as restricting atoms of answer sets is only meaningful for atoms called *facets* that belong to some (brave) but not to all answer sets (cautious). Surprisingly, the precise computational complexity of facet problems remained widely open so far. In this paper, we study the complexity of answer set facets. We establish tight results for reasoning with facets, deciding upper and lower bounds as well as the exact number of facets, and comparing facets. Facet reasoning seems to be a natural problem formalism, residing in complexity families $\Sigma^{\mathrm{P}}$, $\Pi^{\mathrm{P}}$, $\mathrm{D}^{\mathrm{P}}$, and $\Theta^{\mathrm{P}}$, up to the third level. Moreover, our study considers quantitative importance questions on facets and generalizing from facets to conjunctions, disjunctions, and arbitrary queries. We complete our results by an experimental evaluation.

## 1 Introduction

Answer set programming (ASP) is a popular framework for declarative programming (Marek and Truszczyński 1999; Niemelä 1999; Brewka, Eiter, and Truszczyński 2011) allowing for encoding a problem by means of rules and constraints that form a logic program. Solutions to the program are so-called answer sets. Countless problems in AI and reasoning can be modeled within ASP and solved using tools such as `clingo` (Gebser et al. 2014), `WASP` (Alviano et al. 2015), or `DLV` (Alviano et al. 2017). While the qualitative and quantitative reasoning problems for ASP are of high worst-case complexity (Eiter and Gottlob 1995; Truszczyński 2011; Fichte et al. 2017; Hecher 2022), advances in highly efficient solvers and encoding techniques (Gebser et al. 2012) encompass numerous applications, such as configuration or planning (Soininen and Niemelä 1999; Gebser, Kaminski, and Schaub 2011).

When using answer set programs to compactly represent knowledge, for example with configuration problems (Soininen and Niemelä 1999), we may easily have a vast number of solutions. Different perspectives to reason with answer sets have been investigated over the last years, such as reasoning with incomplete information (Shen and Eiter 2016), introducing preferences (Brewka et al. 2023), accessing optimal (Brewka, Niemela, and Truszczynski 2003) or diverging answer sets (Böhl and Gaggl 2022; Böhl, Gaggl, and Rusovac 2023), and filtering answer sets (Fichte, Gaggl, and Rusovac 2022).

One of the most natural concepts for filtering answer sets is to restrict the presence or absence of atoms, which is only meaningful for atoms called *facets* that belong to some answer set (brave atoms) but not to all answer sets (cautious atoms). Example 1 provides a brief intuition on that concept.

**Example 1.** *Consider the following program*

$$P_1 = \{p \leftarrow {\sim}q;\ q \leftarrow {\sim}p;\ r \vee s \leftarrow q;\ t \leftarrow \}.$$

*The answer sets of $P_1$ are $\mathrm{AS}(P_1) = \{\{p,t\}, \{q,r,t\}, \{q,s,t\}\}$. Atom $t$ occurs in all answer sets and is not a facet. Filtering the answer sets such that every set contains $t$ yields the same answer sets and excluding $t$ results in no answer set. However, the atoms $p$, $q$, $r$, and $s$ are facets. Filtering for $p$ yields $\{p,t\}$, whereas excluding $p$ yields $\{q,r,t\}$ and $\{q,s,t\}$. Clearly, we can also query the answer sets using more complex questions.*

Example 1 illustrates the fundamental nature of facets for filtering answer sets, originally introduced by Alrabbaa, Rudolph, and Schweizer (2018) and studied recently for comparing sequences of facets (Fichte, Gaggl, and Rusovac 2022). Qualitative and quantitative reasoning based on facets is particularly useful to understand answer sets and uncertainty.

The complexity of facets, which refers in combinatorics to one dimension less than the structure itself, is a fundamental problem in computer science and mathematics (Papadimitriou and Yannakakis 1982). In the propositional satisfiability setting (SAT-UNSAT, Unique SAT), facets are mostly understood (Papadimitriou and Yannakakis 1982). However, the complexity of problems that involve facets in ASP as well as qualitative and quantitative reasoning remained widely open so far. When is the computation of facets hard and what are its sources of hardness? Can we expect efficient algorithms? How much harder is counting? We study such questions and provide a complete complexity landscape for key fragments of ASP. The problem asking whether a program has more facets than a second program already reaches the third level of the polynomial hierarchy. For fragments that account to propositional formulas, we reach up to the second level. Additionally, we may ask what happens if we

| Problem | Given | Task | Disj | Tight/Normal | Reference |
|---|---|---|---|---|---|
| AspFacetReason | $P, a \in \mathsf{at}(P)$ | $a \in \mathcal{F}(P)$ | $\Sigma_2^{\mathrm{P}}$-c | NP-c | Theorem 4 |
| Exact-k-Facets | $P, k \in \mathbb{N}_0$ | $\|\mathcal{F}(P)\| = k$ | $\mathrm{D}_2^{\mathrm{P}}$-c | $\mathrm{D}_1^{\mathrm{P}}$-c | Theorem 7 |
| AtLeast-k-Facets | $P, k \in \mathbb{N}_0$ | $\|\mathcal{F}(P)\| \geq k$ | $\Sigma_2^{\mathrm{P}}$-c | NP-c | Corollary 8 |
| AtMost-k-Facets | $P, k \in \mathbb{N}_0$ | $\|\mathcal{F}(P)\| \leq k$ | $\Pi_2^{\mathrm{P}}$-c | coNP-c | Corollary 9 |
| FacetNumCompare | $P_1, P_2$ | $\|\mathcal{F}(P_1)\| > \|\mathcal{F}(P_2)\|$ | $\Theta_3^{\mathrm{P}}$-c | $\Theta_2^{\mathrm{P}}$-c | Theorem 10 |

Table 1: Survey of the complexity results. We list problems in rows and key fragments in columns. Observe that facet reasoning on more restricted fragments like Horn or Stratified does not provide meaningful insights. Note also that this table immediately yields interesting consequences for facet reasoning over *Boolean formulas (SAT)*, as there is a strong relation to normal (tight) programs.

go beyond facets and query answer sets by conjunctive or disjunctive queries, or ASP programs.

**Contributions.** Our main contributions are as follows.

1. We systematically analyze answer set facets and establish complexity results for various qualitative and quantitative problems involving facets, outlined in Table 1. Interestingly, this renders facet reasoning a central problem modeling suite, as complexity spans over canonical classes up to the third level of the polynomial hierarchy.
2. We introduce fundamental concepts for formulating generic queries to ASP solution spaces, thereby generalizing facets. We provide unified queries on answer set programs and establish their computational complexity. Surprisingly, facet reasoning appears quite robust: Main reasoning questions can be enhanced by more elaborate queries without significantly changing their complexity.
3. We present techniques to incorporate our framework and concepts into existing systems. Indeed, we suggest implementations of facet reasoning that directly build upon the prominent answer set solver `clingo`, but could easily be incorporated into other systems. We then conclude our work by an initial empirical study, whose results are promising. Given its central role, we expect facet reasoning to be of interest also for other problem formalisms in KR and AI (e.g., quantified Boolean formulas).

### 1.1 Related Works

Concepts on facets indirectly also apply to debugging answer sets (Oetsch, Pührer, and Tompits 2018; Dodaro et al. 2019; De Vos et al. 2012; Schekotihin 2015; Gebser et al. 2008), where one is interested in understanding or correcting answer set programs. Facets are related to explanations (Fandinno and Schulz 2019; Alviano et al. 2023b; Eiter and Geibinger 2023; Eiter, Geibinger, and Oetsch 2023), which aim for understanding why a literal is in an answer set. Notions of more precise reasoning in ASP have been studied in the past (Fichte, Hecher, and Nadeem 2022). Beyond enumeration, there are attempts to count answer sets exactly (Fichte et al. 2024a; Kabir, Chakraborty, and Meel 2024; Eiter, Hecher, and Kiesel 2024) or approximately (Fichte et al. 2024a). Furthermore, we see a relation of facets to epistemic logic programs (ELP) (Shen and Eiter 2016; Gelfond 1991), which extend answer set programs by allowing modal operators meaning provably true or possible. Thereby, consequences from incomplete information about all or one answer set can be stated in a program itself. Solutions to an ELP can be seen as consequences over multiple collections of answer sets, known as world views. However, facets rather complement the epistemic view as they ask for atoms that are neither provably true nor false but still possible. In propositional satisfiability, similar concepts to facets exist where so-called assumptions form fundamental basics for iterative solving (Eén and Sörensson 2003). Faber and Woltran (2011) present program rewritings (manifold programs) for post-processing consequences and apply this to ideal extensions in abstract argumentation and epistemic programs. Janhunen et al. (2009) introduce concepts of splitting answer set programs based on modularity aspects. Uncertainty is directly related to probabilistic answer set programming and related reasoning questions (Bellodi et al. 2020; Azzolini and Riguzzi 2023b; Azzolini and Riguzzi 2023a).

## 2 Preliminaries

We assume that the reader is familiar with basics in propositional logic, ASP, and computational complexity. Below, we summarize notations.

**Computational Complexity.** We follow standard terminology in computational complexity (Papadimitriou 1994) and the Polynomial Hierarchy (PH) (Stockmeyer and Meyer 1973; Stockmeyer 1976; Wrathall 1976). In particular, $\Delta_0^{\mathrm{P}} \coloneqq \Pi_0^{\mathrm{P}} \coloneqq \Sigma_0^{\mathrm{P}} \coloneqq \mathrm{P}$ and $\Delta_i^{\mathrm{P}} \coloneqq P^{\Sigma_{i-1}^{\mathrm{P}}}$, $\Sigma_i^{\mathrm{P}} \coloneqq \mathrm{NP}^{\Sigma_i^{\mathrm{P}}}$, and $\Pi_i^{\mathrm{P}} \coloneqq \mathrm{coNP}^{\Sigma_i^{\mathrm{P}}}$ for $i > 0$ where $C^D$ is the class $C$ of decision problems augmented by an oracle for some complete problem in class $D$. Recall that $\mathrm{PH} \coloneqq \bigcup_{i \in \mathbb{N}} \Delta_i^{\mathrm{P}}$ (Stockmeyer 1976). Interestingly, there are also complexity classes between $\Sigma_{\text{i-1}}^{\mathrm{P}}/\Pi_{\text{i-1}}^{\mathrm{P}}$ and $\Delta_{\mathrm{i}}^{\mathrm{P}}$. The class $\Delta_{\mathrm{i}}^{\mathrm{P}[\log(n)]}$, or $\Theta_{\mathrm{i}}^{\mathrm{P}}$ for short, permits only $\mathcal{O}(\log(n))$ many $\Sigma_{\text{i-1}}^{\mathrm{P}}$-oracle calls for every instance of size $n$ (Lukasiewicz and Malizia 2017). In fact, $\Theta_0^{\mathrm{P}} = \Theta_1^{\mathrm{P}} = \mathrm{P}$, and $\Sigma_i^{\mathrm{P}} \cup \Pi_i^{\mathrm{P}} \subseteq \Theta_{i+1}^{\mathrm{P}} \subseteq \Delta_{i+1}^{\mathrm{P}} \subseteq \Sigma_{i+1}^{\mathrm{P}} \cap \Pi_{i+1}^{\mathrm{P}}$ for all $i > 0$. The complexity class $\mathrm{D}_k^{\mathrm{P}}$ is defined as $\mathrm{D}_k^{\mathrm{P}} \coloneqq \{L_1 \cap L_2 \mid L_1 \in \Sigma_k^{\mathrm{P}}, L_2 \in \Pi_k^{\mathrm{P}}\}$, $\mathrm{D}^{\mathrm{P}} = \mathrm{D}_1^{\mathrm{P}}$ (Lohrey and Rosowski 2023), and $\mathrm{D}_k^{\mathrm{P}}$ is located below $\Theta_k^{\mathrm{P}}$.

**(Quantified) Boolean Formulas.** We define *propositional formulas* in the usual way; *literals* are variables or their negations. For a propositional formula $F$, we denote by $\mathsf{var}(F)$ the set of variables of $F$. Logical operators $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$ are used in the usual meaning. A *term* is a conjunction of literals, and a *clause* is a disjunction of literals. $F$ is in *conjunctive normal form (CNF)* if $F$ is a conjunction of clauses, and $F$ is in *disjunctive normal form (DNF)* if $F$ is a disjunction of terms. In both cases, we identify $F$ by the set of its clauses or terms, respectively. We assume that a propositional formula is in CNF, unless stated otherwise. Let $l \geq 0$ be an integer. A *quantified Boolean formula (QBF)* $Q$ is of the form

$$Q_1 V_1 . Q_2 V_2 . \cdots Q_l V_l . F,$$

where $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq l$, $Q_j \neq Q_{j+1}$ for $1 \leq j \leq l-1$, and the $V_i$ are disjoint, non-empty sets of propositional variables with $\bigcup_{i=1}^{l} V_i = \mathsf{var}(F)$ for a propositional formula $F$. Given a subset $X \subseteq \mathsf{var}(F)$, an assignment is a mapping $\tau : X \to \{0,1\}$. The truth evaluation $F_\tau$ of a propositional formula $F$ is defined in the standard way. An assignment $\tau$ satisfies $F$ if it evaluates to true, for short $F_\tau = 1$. We say that $F$ is satisfiable if there is some assignment that satisfies $F$. For a set $M \subseteq \mathsf{var}(F)$, by $\tau(M)$ we refer to its corresponding truth assignment, i.e., $\tau(M) = \{\mathsf{var}(F) \cap M \mapsto 1\} \cup \{\mathsf{var}(F) \setminus M \mapsto 0\}$, and use $M \models F$ as shorthand for $F_{\tau(M)} = 1$. For a given QBF $Q$ and an assignment $\alpha : X \to \{0,1\}$, $Q_\alpha$ is the QBF obtained from $Q$, where variables $x \in X$ are removed from preceding quantifiers accordingly. A QBF $Q$ with $Q_1 = \exists$ *evaluates to true* if there is an assignment $\alpha : V_1 \to \{0,1\}$ such that $Q_\alpha$ evaluates to true. If $Q_1 = \forall$, then $Q$ evaluates to true if, for every assignment $\alpha : V_1 \to \{0,1\}$, $Q_\alpha$ evaluates to true. $\text{QSAT}_l$ ($\text{QUNSAT}_l$) refers to the *problem of deciding satisfiability (unsatisfiability)* for a given QBF $Q$ of quantifier depth $l$. If $Q_1 = \exists$, the problem $\text{QSAT}_l$ is $\Sigma_l^{\mathrm{P}}$-complete, and the problem $\text{QUNSAT}_l$ is $\Pi_l^{\mathrm{P}}$-complete (Kleine Büning and Lettmann 1999; Papadimitriou 1994; Stockmeyer and Meyer 1973).

**Answer Set Programming (ASP).** For a comprehensive introduction, we refer to standard texts (Janhunen and Niemelä 2016; Calimeri et al. 2020; Gebser et al. 2012). We restrict ourselves to propositional programs. Let $l, m, n$ be non-negative integers such that $l \leq m \leq n$, and $a_1, \ldots, a_n$ distinct propositional atoms. A *disjunctive rule* $r$ is of the form

$$a_1 \vee \cdots \vee a_l \leftarrow a_{l+1}, \ldots, a_m, {\sim} a_{m+1}, \ldots, {\sim} a_n,$$

which, intuitively, means that at least one atom of $a_1, \ldots, a_l$ must be true if all atoms $a_{l+1}, \ldots, a_m$ are true and there is no evidence that any atom of $a_{m+1}, \ldots, a_n$ is true. By $H_r \coloneqq \{a_1, \ldots, a_l\}$, $B_r^+ \coloneqq \{a_{l+1}, \ldots, a_m\}$, and $B_r^- \coloneqq \{a_{m+1}, \ldots, a_n\}$, we denote the *head*, *positive* or *negative body* atoms of $r$, respectively. We say that $r$ is *normal* if $|H(r)| \leq 1$, *positive* if $B(r)^- = \emptyset$, and an *integrity constraint* if $H(r) = \emptyset$. Usually, if $B_r^+ \cup B_r^- = \emptyset$, we simply write $H_r$ instead of $H_r \leftarrow$. A *program* $P$ is a set of rules, where $\mathsf{at}(P) \coloneqq \bigcup_{r \in P} (H_r \cup B_r^+ \cup B_r^-)$ denotes its atoms. Moreover, a program $P$ has a certain property if all its rules have the property. The *dependency digraph* $\mathcal{D}_P$ is the digraph defined on the set $\bigcup_{r \in P} (H_r \cup B_r^+)$ of atoms, where for every rule $r \in P$, two atoms $b \in B_r^+$ and $a \in H_r$ are joined by an edge $(b, a)$. If $\mathcal{D}_P$ has no directed cycle, $P$ is called *tight* (Fages 1994). By **Normal**, **Disj**, and **Tight**, we denote the class of all normal, disjunctive, or tight programs, respectively. An interpretation $M \subseteq \mathsf{at}(P)$ *satisfies* a rule $r$ if $(H_r \cup B_r^-) \cap M \neq \emptyset$ or $B_r^+ \nsubseteq M$, and $M$ is a *model* of $P$ if $M$ satisfies every rule $r \in P$. The *(GL) reduct* of $P$ with respect to $M$ is defined as $P^M \coloneqq \{H_r \leftarrow B_r^+ \mid B_r^- \cap M = \emptyset\}$. Then, $M$ is an *answer set* of $P$ if $M$ is a model of $P$ such that no interpretation $N \subsetneq M$ is a model of $P^M$ (Gelfond and Lifschitz 1988). We let the set $\mathrm{AS}(P)$ consist of all answer sets of $P$. The program $P$ is *consistent* if $\mathrm{AS}(P) \neq \emptyset$, and *inconsistent* otherwise.

**Qualitative Reasoning.** The consistency problem asks to decide whether a program $P$ is consistent, which is $\Sigma_2^{\mathrm{P}}$-complete (Eiter and Gottlob 1995). If the input is restricted to normal programs, the complexity drops to NP-completeness (Bidoít and Froidevaux 1991; Marek and Truszczyński 1991). We define the *brave* consequences by

$$\mathcal{BC}(P) \coloneqq \bigcup_{M \in \mathrm{AS}(P)} M$$

and *cautious* consequences by

$$\mathcal{CC}(P) \coloneqq \bigcap_{M \in \mathrm{AS}(P)} M.$$

For an atom $a \in \mathsf{at}(P)$, deciding whether $a \in \mathcal{BC}(P)$ is $\Sigma_2^{\mathrm{P}}$-complete, and whether $a \in \mathcal{CC}(P)$ is $\Pi_2^{\mathrm{P}}$-complete (Eiter and Gottlob 1995).

**Search Space, Solution Space, and Assumptions.** By *search space* of a program $P$, we mean the power set $2^{\mathsf{at}(P)}$ over the atoms occurring in $P$, which encapsulates all possible interpretations. Answer set navigation provides concepts and notions to select answer sets of $P$ within the *solution space* $2^{\mathrm{AS}(P)}$, gathering subsets of the answer sets of $P$ (Fichte, Gaggl, and Rusovac 2022). An *assumption* is a literal $\ell$ used for selecting the answer sets of $P$ such that $\ell$ holds. Therefore, the integrity constraint $\mathrm{ic}(\ell) \coloneqq \{\leftarrow {\sim}\ell\}$, where ${\sim}\neg a$ stands for $a$, allows us to incorporate the assumption $\ell$ into $P[\ell] \coloneqq P \cup \mathrm{ic}(\ell)$. For a set $L$ of literals, let ${\sim}L \coloneqq \{{\sim}\ell \mid \ell \in L\}$ and $\neg L \coloneqq \{\neg\ell \mid \ell \in L\}$, assuming that $\neg\neg a = a$. By $L^+$ and $L^-$, we denote the sets of variables $a$ occurring as positive literals $\ell = a$ or negative literals $\ell = \neg a$ in $L$, respectively.

**ASP Facets.** *Facets* restrict assumptions to (literals over) atoms of a program $P$ that are meaningful, i.e., atoms belonging to some but not to all answer sets (Alrabbaa, Rudolph, and Schweizer 2018). Thus, we let

$$\mathcal{F}(P) \coloneqq \mathcal{BC}(P) \setminus \mathcal{CC}(P).$$

The literature (Fichte, Gaggl, and Rusovac 2022) sometimes distinguishes *excluding* and *including* facets, depending on whether the facets should be part of answer sets or not. For computational aspects, it suffices to focus on including facets, which we do in the following, unless stated otherwise.

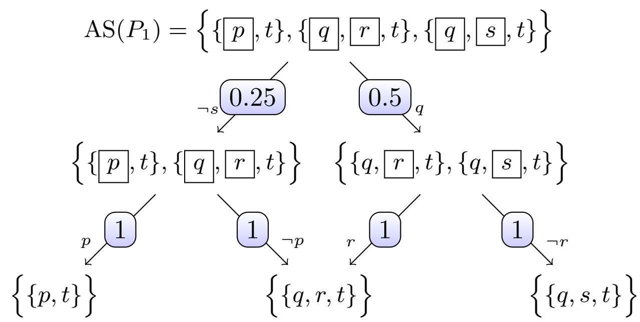


Figure 1: Parts of the search space of program $P_1$ in between the set of all answer sets $\mathrm{AS}(P_1)$ and singletons. Boxed atoms are facets in the respective sub-space. Edges are labeled with the activated facet (next to rectangle) and its significance (in rectangle).

## 3 Complexity of ASP Facets

Before we systematically analyze computational problems that arise with ASP facets, we note that the number of facets directly measures the *amount of uncertainty* in possible answer sets. A facet $\ell \in \{a, \neg a\}$ can be seen as an *uncertain event* $a$, since $a$ can either be included in or be excluded from answer sets. When we assume the truth of a facet (assumption), we reduce uncertainty among the answer sets. This leads to the notion of *significance* (Böhl, Gaggl, and Rusovac 2023), defined as follows for a program $P$ and a literal $\ell$:

$$\mathbb{S}[P, \ell] := \frac{|\mathcal{F}(P)| - |\mathcal{F}(P[\ell])|}{|\mathcal{F}(P)|}.$$

**Example 2** (Technical Example)**.** *Reconsider program $P_1$ from Example 1, its answer sets $\mathrm{AS}(P_1) = \{\{p,t\}, \{q,r,t\}, \{q,s,t\}\}$, and facets $p$, $q$, $r$, and $s$. Figure 1 illustrates the effects of gradually assuming facets being excluded or included from answer sets in form of a decision tree. More precisely, on the first level, we only illustrate $s$ being excluded or $q$ included. We observe that $\mathbb{S}[P_1, \neg s] = 0.25$ and $\mathbb{S}[P_1, q] = 0.5$, meaning that $q$ is twice as significant to the answer sets as $\neg s$. Furthermore, if we investigate $p$ or $q$, we have that $\mathbb{S}[P_1, p] = \mathbb{S}[P_1, \neg q] = 1$ and $\mathbb{S}[P_1, \neg p] = \mathbb{S}[P_1, q] = 0.5$. Thus, $p$ as well as $q$ have high significance among the facets.*

**Example 3** (Uncertainty Application)**.** *Today, learning is a core topic in AI. Reliable interpretability and explainability of learnt systems are under intense investigation, for example, to discover logical rules and enable predictions (Barbiero et al. 2023). Consider a program $P_{\mathrm{sum}}$ that learns the sum $S$ of adding two digits $A$ and $B$ (Manhaeve et al. 2018), i.e., $S = A + B$, using rules of the form $\mathtt{prediction\_sum}(S) \leftarrow \mathtt{digitL}(A) \wedge \mathtt{digitR}(B)$. Recall that $S = A + B$ implies $A = S - B$ and $B = S - A$, which we can use to visualize potential errors by employing significance. Consequently, if we assume that $S$ together with either input digit $A$ or $B$ are fixed, no uncertainty should occur. We can check this by determining whether $\mathbb{S}[P_{\mathrm{sum}}[\mathtt{prediction\_sum}(S)], \mathtt{digitL}(B)] = 1$ and $\mathbb{S}[P_{\mathrm{sum}}[\mathtt{prediction\_sum}(S)], \mathtt{digitR}(B)] = 1$. Figure 2 illustrates significance for values of $S$ and $A$ or $B$. Note that $\neg\mathtt{digitL}(X)$ means that $A \neq X$. Hence, excluding a certain input digit does not yield certain results as significance is different from $1$. Now, assume that the training process resulted in $\mathtt{prediction\_sum}(8)$ for $\mathtt{digitL}(3)$ and $\mathtt{digitR}(6)$. Then, both facets for $\mathtt{prediction\_sum}(8)$ have a significance of $0.842$, which is different from $1$ and thus indicates an error.*

Now, we are ready to start with a natural reasoning problem, which we define from the notions above. AspFacetReason asks, given a program $P$ and an atom $a \in \mathsf{at}(P)$, to decide whether $a \in \mathcal{F}(P)$. We start with a lower and upper bound on the AspFacetReason problem.

**Theorem 4.** *Let $P$ be a program and $a \in \mathsf{at}(P)$. The problem* AspFacetReason *is*

1. *for disjunctive programs $\Sigma_2^{\mathrm{P}}$-complete,*
2. *for tight programs* NP*-complete, and*
3. *for normal programs* NP*-complete.*

Before we establish our theorem, we require the following two lemmas and introduce auxiliary definitions. For a set $M \subseteq \mathsf{at}(P)$, we define $M' := \{a' \mid a \in M\}$ and $\mathrm{cp}(P) := \{H_r' \leftarrow B_r^{+\prime}, {\sim}B_r^{-\prime} \mid r \in P\}$. In other words, when constructing $\mathrm{cp}(P)$, we simply replace each atom in $P$ by another fresh atom.

First, we establish that we can detect facets by employing the consistency problem. Therefore, we ask whether a program $P$ under the assumption of an atom $a \in \mathsf{at}(P)$ along with the opposite assumption $\neg a'$ on a copy $\mathrm{cp}(P)$ result in a new program that is consistent. In this way, we ensure that the respective atom $a$ belongs to some but not to all answer sets of $P$, which is precisely the condition for a brave but not cautious consequence.

**Lemma 5.** *Let $P$ be a program and $a \in \mathsf{at}(P)$. Then, $a \in \mathcal{F}(P)$ if and only if the program $P[a] \cup \mathrm{cp}(P[\neg a])$ is consistent.*

*Proof.* First, we observe that $\{b_1, \ldots, b_\ell\} \in \mathrm{AS}(P)$ if and only if $\{b_1', \ldots, b_\ell'\} \in \mathrm{AS}(\mathrm{cp}(P))$ holds by construction, since $\mathrm{cp}(P)$ contains fresh auxiliary atoms that are in one-to-one correspondence with $\mathsf{at}(P)$. Hence, we have that $\mathrm{AS}(P \cup \mathrm{cp}(P)) = \mathrm{AS}(P) \times \mathrm{AS}(\mathrm{cp}(P))$.
**(⇒):** Assume that $a \in \mathcal{F}(P)$. Then, there are answer sets $M_1, M_2 \in \mathrm{AS}(P)$ such that $a \in M_1$ and $a \notin M_2$. Since $M_2' \in \mathrm{AS}(\mathrm{cp}(P))$, we have that $M_1 \cup M_2' \in \mathrm{AS}(P \cup \mathrm{cp}(P))$. Along with the fact that $M_1 \cup M_2'$ is a model of $\mathrm{ic}(a) \cup \mathrm{ic}(\neg a')$, we conclude that $M_1 \cup M_2' \in \mathrm{AS}(P[a] \cup \mathrm{cp}(P[\neg a]))$. Thus, the program $P[a] \cup \mathrm{cp}(P[\neg a])$ is consistent, which establishes the only-if direction.
**(⇐):** Assume that $P[a] \cup \mathrm{cp}(P[\neg a])$ is consistent. Then, there is an answer set $M \in \mathrm{AS}(P[a] \cup \mathrm{cp}(P[\neg a]))$. By construction, we have that $M \in \mathrm{AS}(P \cup \mathrm{cp}(P))$, $a \in M$, and $a' \notin M$. For $M_1 := M \cap \mathsf{at}(P)$ and $M_2' := M \cap \mathsf{at}(\mathrm{cp}(P))$, this yields $M_1, M_2 \in \mathrm{AS}(P)$ such that $a \in M_1$ and $a \notin M_2$. We conclude that $a \in \mathcal{F}(P)$, which establishes the if direction. □

Next, we show that we can employ a simple trick to connect the consistency problem and reasoning for a facet. There-

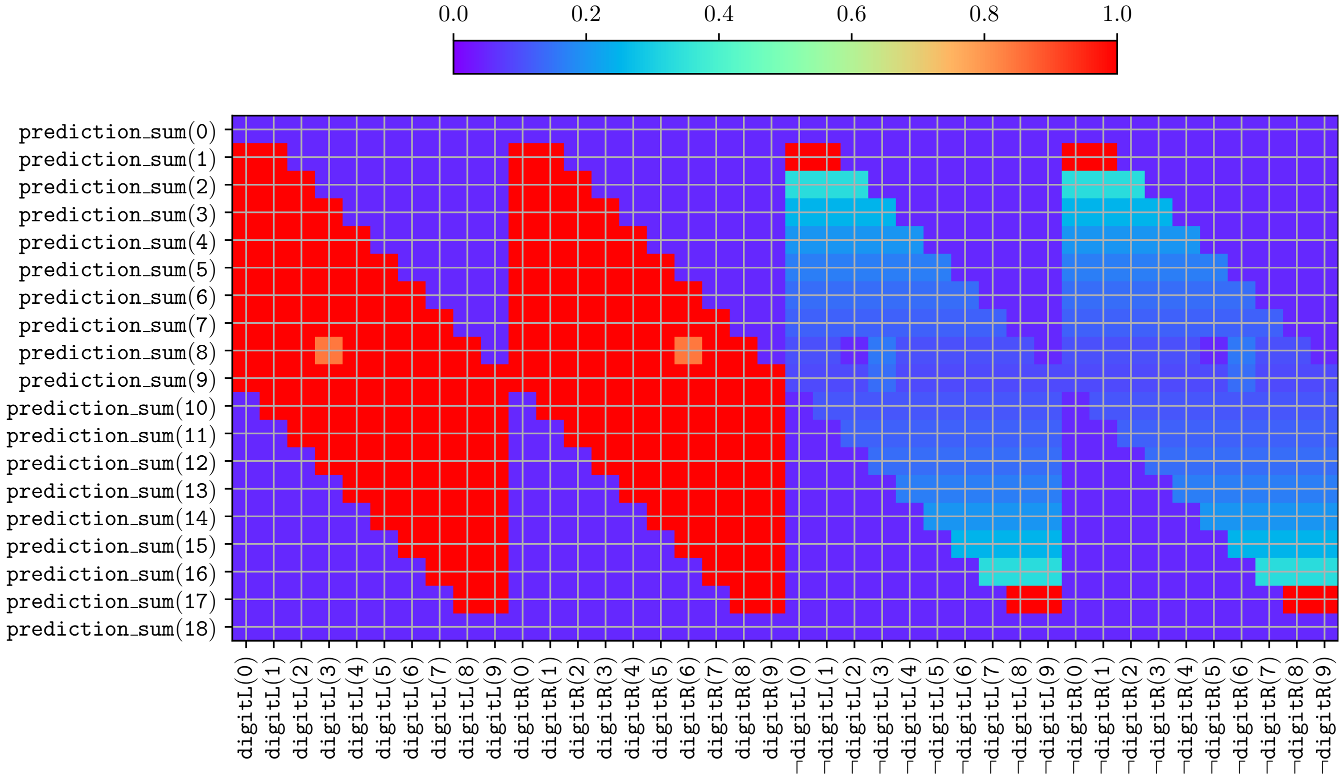

Figure 2: Heatmap illustrating significance. The color of a cell indicates the significance $\mathbb{S}[P_{\text{sum}}[\texttt{prediction\_sum}(X)], \ell]$, where $X \in \{0, 1, \ldots, 18\}$ and $\ell \in \{\texttt{digitL}(0), \ldots, \texttt{digitR}(0), \ldots, \neg\texttt{digitL}(0), \ldots, \neg\texttt{digitR}(0), \ldots\}$. The significance is taken to be 0 if $\ell$ is not a facet of $P_{\text{sum}}[\texttt{prediction\_sum}(X)]$.

fore, we generate an answer set that contains some fresh atom and another one that does not contain this atom.

**Lemma 6.** *Let $P$ be a program, $b$ and $b'$ fresh atoms, and $P' = P \cup \{b \leftarrow \sim b';\ b' \leftarrow \sim b\}$. Then, the program $P$ is consistent if and only if $b \in \mathcal{F}(P')$.*

*Proof.* **($\Rightarrow$):** Assume that the program $P$ is consistent. Then, there is an answer set $M \in \text{AS}(P)$. By construction, $M \cup \{b\}$ and $M \cup \{b'\}$ are answer sets of $P'$, where $b \notin M$. We conclude that $b \in \mathcal{F}(P')$, which establishes the only-if direction.
**($\Leftarrow$):** Assume that $b \in \mathcal{F}(P')$. Then, $b \in \mathcal{BC}(P')$ yields the existence of an answer set $M \in \text{AS}(P')$ such that $b \in M$. By construction, we have that $M \setminus \{b\}$ is an answer set of $P$. Thus, the program $P$ is consistent, which establishes the if direction. □

Now we are ready to establish the proof of Theorem 4.

*Proof of Theorem 4. ("Membership"):* By Lemma 5, we can decide $a \in \mathcal{F}(P)$ by checking whether the program $P_a \coloneqq P[a] \cup \text{cp}(P[\neg a])$ is consistent. We observe that $P_a$ is disjunctive, tight, or normal if and only if the same property holds for $P$. By well-known results for the consistency problem (Truszczyński 2011), we conclude that AspFacetReason $\in \Sigma_2^{\text{P}}$ for disjunctive programs $P$, and AspFacetReason $\in$ NP if $P$ is tight or normal.

*("Hardness"):* The program $P'$ from Lemma 6 is disjunctive, tight, or normal if and only if the same property holds for $P$. Since the consistency problem is $\Sigma_2^{\text{P}}$-hard for disjunctive programs $P$, and NP-hard if $P$ is tight or normal (Truszczyński 2011), we conclude the corresponding hardness results for deciding whether $b \in \mathcal{F}(P')$. □

### 3.1 Exact Number of Facets

Next, we turn our attention to the complexity of counting facets, where the number of facets is bound by $0 \leq |\mathcal{F}(P)| \leq |\texttt{at}(P)|$ for a program $P$. Before we study the function problem, we investigate a parameterized version by taking a bound $k$ on the number of facets as input. In detail, the problem Exact-k-Facets asks, given a program $P$ and an integer $k$, to decide whether $|\mathcal{F}(P)| = k$. The next statement establishes upper and lower bounds.

**Theorem 7.** *Let $P$ be a program and integer $k \in \mathbb{N}_0$. The problem* Exact-k-Facets *is*

1. *for disjunctive programs $\text{D}_2^{\text{P}}$-complete,*
2. *for tight programs $\text{D}^{\text{P}}$-complete, and*
3. *for normal programs $\text{D}^{\text{P}}$-complete.*

*Proof.* We reduce to/from $\text{D}^{\text{P}}$- (Papadimitriou and Yannakakis 1982) and $\text{D}_2^{\text{P}}$-complete (Shen and Eiter 2016; Lohrey and Rosowski 2023) problems.

*("Membership"):*

**Tight/Normal.** We reuse the idea of the construction from Lemma 5, while replacing the assumptions and integrity constraints by rules that allow for an atom to not be a facet. Replicating this construction for all atoms $a \in \mathsf{at}(P)$, a sequential at least $k$ counter permits checking whether at least $k$ atoms are facets. In more detail, we make use of the following shorthands for atoms $a \in \mathsf{at}(P)$:

$$P_a := P_a^{\text{bc}} \cup P_a^{\text{cc}} \cup \{a^f \leftarrow a_a^{\text{bc}}, {\sim} a_a^{\text{cc}}\}$$

where $P_a^{\text{bc}}$ and $P_a^{\text{cc}}$ construct different copies of $P$ that replace each atom $b \in \mathsf{at}(P)$ by a fresh atom $b_a^{\text{bc}}$ or $b_a^{\text{cc}}$, respectively. The role of $P_a^{\text{bc}}$ is to indicate $a \in \mathcal{BC}(P)$ by the truth of $a_a^{\text{bc}}$, and likewise $P_a^{\text{cc}}$ witnesses $a \notin \mathcal{CC}(P)$ if $a_a^{\text{cc}}$ is false. Hence, the atom $a^f$ can be true only if $a \in \mathcal{F}(P)$. Then, we rely on existing works in propositional satisfiability, cf. (Tseytin 1983; Sheridan 2004), to define a sequential at least $k$ counter with respect to $\mathsf{at}(P) = \{a_1, \ldots, a_n\}$ by the following program:

$$\begin{aligned} P_k := \{&s_{i,1} \leftarrow a_i^f \mid 1 \leq k \leq i \leq n\} \cup \\ \{&s_{i,j} \leftarrow a_i^f, s_{i+1,j-1} \mid k-j < i < n, 1 < j \leq k\} \cup \\ \{&s_{i,j} \leftarrow s_{i+1,j} \mid k-j < i < n, 1 \leq j \leq k\} \cup \\ \{&\leftarrow {\sim} s_{1,k} \mid 1 \leq k\} \end{aligned}$$

The program $P_k$ yields $s_{1,k}$ as true if and only if at least $k \geq 1$ of the atoms $\{a_1^f, \ldots, a_n^f\}$ hold, which in turn means that $|\mathcal{F}(P)| \geq k$. Now consider the programs:

$$P_{\text{cons}}^{\text{fc}} := P_k \cup \textstyle\bigcup_{a \in \mathsf{at}(P)} P_a$$
$$P_{\text{incons}}^{\text{fc}} := P_{k+1} \cup \textstyle\bigcup_{a \in \mathsf{at}(P)} P_a.$$

Since $P_k$ and $P_{k+1}$ check for at least $k$ or $k+1$ facets, respectively, we require consistency for $P_{\text{cons}}^{\text{fc}}$, and inconsistency for $P_{\text{incons}}^{\text{fc}}$. For tight as well as normal programs, these problems are in NP or coNP (Truszczyński 2011), respectively, which establishes $\text{D}^{\text{P}}$-membership.

**Disjunctive.** The same construction also works for disjunctive programs, while increasing the complexity by one level.

*("Hardness"):*

**Tight/Normal.** We reduce from SAT-UNSAT to asking for exactly $k$ facets. Therefore, let $F_{\text{sat}}$ and $F_{\text{unsat}}$ be propositional formulas in CNF given as sets $\{L_1^x, \ldots, L_{m_x}^x\}$ of clauses for $x \in \{\text{sat}, \text{unsat}\}$, where each clause $L_i^x$ is represented by the set of its literals. Without loss of generality, we assume that $L_i^x \neq \emptyset$ and $L_i^x \cap \neg L_i^x = \emptyset$, i.e., the clauses are neither inconsistent nor tautological. For $x \in \{\text{sat}, \text{unsat}\}$, we associate the formula $F_x$ with the following program:

$$\begin{aligned} P_x := \{&b_a^x \leftarrow {\sim} b_{\neg a}^x;\ b_{\neg a}^x \leftarrow {\sim} b_a^x \mid a \in \mathsf{var}(F_x)\} \cup \\ \{&s_i^x \leftarrow b_\ell^x \mid 1 \leq i \leq m_x, \ell \in L_i^x\} \cup \\ \{&s^x \leftarrow c^x, s_1^x, \ldots, s_{m_x}^x\}. \end{aligned}$$

Then, we construct the program $P$ from $P_{\text{sat}}$ and $P_{\text{unsat}}$:

$$P := P_{\text{sat}} \cup P_{\text{unsat}} \cup \{c^{\text{sat}} \leftarrow {\sim} c^{\text{unsat}};\ c^{\text{unsat}} \leftarrow {\sim} c^{\text{sat}};\ \leftarrow c^{\text{sat}}, {\sim} s^{\text{sat}}\}.$$

Next, we show that the following condition holds: $|\mathcal{F}(P)| = 3 + 2 \cdot (|\mathsf{var}(F_{\text{sat}})| + |\mathsf{var}(F_{\text{unsat}})|) + |F_{\text{sat}}| + |F_{\text{unsat}}|$ if and only if $F_{\text{sat}}$ is satisfiable and $F_{\text{unsat}}$ is unsatisfiable.

By construction, any answer set $M \in \text{AS}(P)$ contains either $b_a^x$ or $b_{\neg a}^x$ for each $a \in \mathsf{var}(F_x)$ and $x \in \{\text{sat}, \text{unsat}\}$. Intuitively, we represent the possible truth assignments for $F_{\text{sat}}$ and $F_{\text{unsat}}$ by the combinations of $b_a^x$ or $b_{\neg a}^x$, respectively, and make sure that all of them are "generated" into answer sets including $c^{\text{unsat}}$. Moreover, we have that $s_i^x \in M$ if and only if $b_\ell^x \in M$ for some literal $\ell \in L_i^x$. Intuitively, an atom $s_i^x$ represents that the truth assignment selected via $b_\ell^x$ atoms satisfies the clause $L_i^x \in F_x$.

Now, by $c^x \in M$, we establish that $s^x \in M$ if and only if the assignment $\{a \mapsto 1 \mid b_a^x \in M\} \cup \{a \mapsto 0 \mid b_{\neg a}^x \in M\}$ satisfies $F_x$. Hence, there is some answer set $M \in \text{AS}(P)$ for which $s^x \in M$ if and only if $F_x$ is satisfiable. Finally, the integrity constraint $\leftarrow c^{\text{sat}}, {\sim} s^{\text{sat}}$ ensures that there is no answer set $M \in \text{AS}(P)$ such that $\{c^{\text{sat}}, s^{\text{sat}}\} \subseteq M$ if and only if $F_{\text{sat}}$ is unsatisfiable.

In consequence, we conclude for the set of facets: If $F_{\text{sat}}$ is unsatisfiable, $\mathcal{F}(P) \subseteq \{s^{\text{unsat}}\} \cup \bigcup_{x \in \{\text{sat},\text{unsat}\}} (\{b_a^x, b_{\neg a}^x \mid a \in \mathsf{var}(F_x)\} \cup \{s_i^x \mid 1 \leq i \leq m_x\})$. If $F_{\text{sat}}$ is satisfiable, $\{c^{\text{unsat}}, c^{\text{sat}}, s^{\text{sat}}\} \cup \bigcup_{x \in \{\text{sat},\text{unsat}\}} (\{b_a^x, b_{\neg a}^x \mid a \in \mathsf{var}(F_x)\} \cup \{s_i^x \mid 1 \leq i \leq m_x\}) \subseteq \mathcal{F}(P)$. Moreover, $s^{\text{unsat}} \in \mathcal{F}(P)$ if and only if $F_{\text{unsat}}$ is satisfiable. In turn, the claim holds that $|\mathcal{F}(P)| = 3 + 2 \cdot (|\mathsf{var}(F_{\text{sat}})| + |\mathsf{var}(F_{\text{unsat}})|) + |F_{\text{sat}}| + |F_{\text{unsat}}|$ if and only if $F_{\text{sat}}$ is satisfiable and $F_{\text{unsat}}$ is unsatisfiable, which establishes the reduction and thus hardness.

**Disjunctive.** We reduce from 2-QBF SAT-UNSAT (valid-invalid) to asking for exactly $k$ facets, employing a well-known reduction of the $\text{QSAT}_2$ problem to disjunctive programs (Eiter and Gottlob 1995). Therefore, let $Q_{\text{sat}}$ and $Q_{\text{unsat}}$ be QBFs of the form $\exists V_1^x . \forall V_2^x . F_x$ for $x \in \{\text{sat}, \text{unsat}\}$, where we represent propositional formulas $F_x$ in DNF as sets $\{L_1^x, \ldots, L_{m_x}^x\}$ of terms $L_i^x$ given by the set of their literals. Then, we construct a program $P'$, reusing the construction of program $P$, from the following programs $P_x$ for $x \in \{\text{sat}, \text{unsat}\}$:

$$\begin{aligned} P_x := \{&b_a^x \leftarrow {\sim} b_{\neg a}^x;\ b_{\neg a}^x \leftarrow {\sim} b_a^x \mid a \in V_1^x\} \cup \\ \{&b_a^x \leftarrow s^x;\ b_{\neg a}^x \leftarrow s^x;\ b_a^x \vee b_{\neg a}^x \mid a \in V_2^x\} \cup \\ \{&s^x \leftarrow c^x, b_{\ell_1}^x, \ldots, b_{\ell_l}^x \mid 1 \leq i \leq m_x, \\ &\qquad L_i^x = \{\ell_1, \ldots, \ell_l\}\}. \end{aligned}$$

As above, any answer set $M \in \text{AS}(P')$ contains either $b_a^x$ or $b_{\neg a}^x$ for $x \in \{\text{sat}, \text{unsat}\}$ and each variable $a \in V_1^x$. For each variable $a \in V_2^x$, we have that $\{b_a^x, b_{\neg a}^x\} \cap M \neq \emptyset$ due to the disjunctive rule $b_a^x \vee b_{\neg a}^x$. Provided that $c^x \in M$, the rules $b_a^x \leftarrow s^x$ and $b_{\neg a}^x \leftarrow s^x$ establish that $\{s^x\} \cup \{b_a^x, b_{\neg a}^x \mid a \in V_2^x\} \subseteq M$ if and only if $M_1^x \cup M_2^x \models F_x$ for $M_1^x := \{a \in V_1^x \mid b_a^x \in M\}$ and every $M_2^x \subseteq V_2^x$. Hence, some answer set $M \in \text{AS}(P')$ with $s^x \in M$ exists if and only if $\exists V_1^x . \forall V_2^x . F_x$ is satisfiable. Consequently, similar reasoning as in the **Tight/Normal** case yields that $\mathcal{F}(P') = \{c^{\text{unsat}}, c^{\text{sat}}, s^{\text{sat}}\} \cup \bigcup_{x \in \{\text{sat},\text{unsat}\}} \{b_a^x, b_{\neg a}^x \mid a \in V_1^x \cup V_2^x\}$ and $|\mathcal{F}(P')| = 3 + 2 \cdot (|V_1^{\text{sat}} \cup V_2^{\text{sat}}| + |V_1^{\text{unsat}} \cup V_2^{\text{unsat}}|)$ if and only if $Q_{\text{sat}}$ is satisfiable and $Q_{\text{unsat}}$ is unsatisfiable. □

**Corollary 8.** *Let $P$ be a program and integer $k \in \mathbb{N}_0$. The problem* AtLeast-k-Facets*, which asks whether $|\mathcal{F}(P)| \geq k$, is*

1. *for disjunctive programs $\Sigma_2^{\mathrm{P}}$-complete,*
2. *for tight programs* NP*-complete, and*
3. *for normal programs* NP*-complete.*

*Proof.* Reconsidering the membership part of the proof of Theorem 7, we require the consistency check for $P^{\text{fc}}_{\text{cons}}$ only. For the hardness part(s), we consider programs $P \coloneqq P_{\text{sat}} \cup \{c^{\text{sat}} \leftarrow {\sim}c^{\text{unsat}};\ c^{\text{unsat}} \leftarrow {\sim}c^{\text{sat}}\}$ along with $k \coloneqq 3 + 2 \cdot |\mathsf{var}(F_{\text{sat}})| + |F_{\text{sat}}|$ or $k \coloneqq 3 + 2 \cdot |V_1 \cup V_2|$, respectively, in order to decide whether a propositional formula $F_{\text{sat}}$ in CNF or a QBF of the form $\exists V_1.\forall V_2.F_{\text{sat}}$ is satisfiable. □

**Corollary 9.** *Let $P$ be a program and integer $k \in \mathbb{N}_0$. The problem* AtMost-k-Facets*, which asks whether $|\mathcal{F}(P)| \leq k$, is*

1. *for disjunctive programs $\Pi_2^{\mathrm{P}}$-complete,*
2. *for tight programs* coNP*-complete, and*
3. *for normal programs* coNP*-complete.*

*Proof.* Dual to the proof of Corollary 8, membership requires the inconsistency check for $P^{\text{fc}}_{\text{incons}}$ only. Moreover, programs $P \coloneqq P_{\text{unsat}} \cup \{c^{\text{sat}} \leftarrow {\sim}c^{\text{unsat}};\ c^{\text{unsat}} \leftarrow {\sim}c^{\text{sat}}\}$ and either $k \coloneqq 2 + 2 \cdot |\mathsf{var}(F_{\text{unsat}})| + |F_{\text{unsat}}|$ or $k \coloneqq 2 + 2 \cdot |V_1 \cup V_2|$ express deciding unsatisfiability for a propositional formula $F_{\text{unsat}}$ in CNF or a QBF of the form $\exists V_1.\forall V_2.F_{\text{unsat}}$. □

Next, we establish complexity results comparing the number of facets of two given programs. Our results rely on the reductions as established in the proof of Theorem 7. The FacetNumCompare problem asks, given two programs $P_1$ and $P_2$, to decide whether $|\mathcal{F}(P_1)| > |\mathcal{F}(P_2)|$.

**Theorem 10.** *Let $P_1$ and $P_2$ be programs. The problem* FacetNumCompare *is*

1. *for disjunctive programs $\Theta_3^{\mathrm{P}}$-complete,*
2. *for tight programs $\Theta_2^{\mathrm{P}}$-complete, and*
3. *for normal programs $\Theta_2^{\mathrm{P}}$-complete.*

*Proof. ("Membership"):* By binary search over $0 \leq k \leq |\mathsf{at}(P_1)|$, the number $k \coloneqq |\mathcal{F}(P_1)|$ of facets can be determined with at most $\mathcal{O}(\log(|\mathsf{at}(P_1)|))$ many oracle calls on programs $P^{\text{fc}}_{1_{\text{cons}}}$ and $P^{\text{fc}}_{1_{\text{incons}}}$ as in the proof of Theorem 7. Then, deciding whether $|\mathcal{F}(P_1)| > |\mathcal{F}(P_2)|$ amounts to checking $|\mathcal{F}(P_2)| \leq k - 1$, where Corollary 9 provides the complexity.

*("Hardness"):* We reduce from PARITY(QSat$_l$) for $1 \leq l \leq 2$, where QSat$_1$ matches SAT and PARITY(QSat$_l$) is $\Theta^{\mathrm{P}}_{l+1}$-complete (Eiter and Gottlob 1997; Wagner 1987). Therefore, let $I_1, \ldots, I_n$ be instances of QSat$_l$, ordered such that $I_i - 1$ is satisfiable if $I_i$ is satisfiable for $1 < i \leq n$. Then, the QSat$_l$ instances $I_1, \ldots, I_n$ satisfy PARITY(QSat$_l$) if $I_1, \ldots, I_i$ are satisfiable and $I_{i+1}, \ldots, I_n$ are unsatisfiable for an odd integer $1 \leq i \leq n$. For each QSat$_l$ instance $I_i$, let the program $P_i$ be constructed similar to $P_{\text{sat}}$ from the hardness part of the proof of Theorem 7, where we denote the $c^{\text{sat}}$ and $s^{\text{sat}}$ atoms by $c_i^{\text{sat}}$ or $s_i^{\text{sat}}$, respectively, and assume without loss of generality that $\mathsf{at}(P_i) \cap \mathsf{at}(P_j) = \emptyset$ for all $1 \leq i < j \leq n$. Now consider the programs:

$$P_{\text{odd}} \coloneqq \textstyle\bigcup_{i=1}^{n}(P_i \cup \{c_i^{\text{sat}} \leftarrow {\sim}c_i^{\text{unsat}};\ c_i^{\text{unsat}} \leftarrow {\sim}c_i^{\text{sat}}\})$$
$$P_{\text{even}} \coloneqq P_{\text{odd}} \cup \{\leftarrow s^{\text{sat}}_{2\cdot i-1}, {\sim}s^{\text{sat}}_{2\cdot i} \mid 1 \leq i \leq \lceil n/2 \rceil\}.$$

By construction, we have that

$$\mathcal{F}(P_{\text{odd}}) = \textstyle\bigcup_{i=1}^{n}(\{c_i^{\text{unsat}}\} \cup (\mathsf{at}(P_i) \setminus \{s_i^{\text{sat}}\}) \cup \{s_i^{\text{sat}} \mid I_i \text{ is satisfiable}\})$$

and $\mathcal{F}(P_{\text{even}}) \subseteq \mathcal{F}(P_{\text{odd}})$. The integrity constraints added by $P_{\text{even}}$ establish that $\mathcal{F}(P_{\text{even}}) = \mathcal{F}(P_{\text{odd}})$ if and only if $\max(\{0\} \cup \{1 \leq i \leq n \mid I_i \text{ is satisfiable}\})$ is even. We conclude that $|\mathcal{F}(P_{\text{odd}})| > |\mathcal{F}(P_{\text{even}})|$ holds if and only if the QSat$_l$ instances $I_1, \ldots, I_n$ satisfy PARITY(QSat$_l$). □

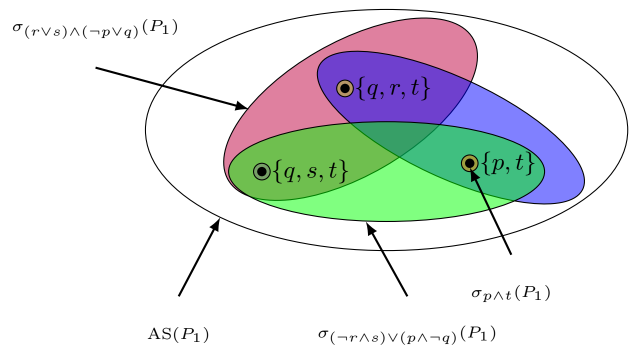


Figure 3: Euler diagram illustrating relationships between answer sets of program $P_1$ from Example 1 according to propositional queries. The outer circle represents the set $\mathrm{AS}(P_1)$ of all answer sets, while the colored regions indicate specific subsets.

# 4 Querying Solution Spaces

So far, we have associated facets with single assumptions only. However, Figure 1 already illustrates that constraining answer sets by several assumptions is interesting for systematically and gradually restricting the presence or absence of atoms. Applications related to declarative queries (Codd 1970) rely on such elaborate techniques to reason with data.

## 4.1 Selecting Answer Sets

To enable declarative querying, we consider propositional queries that allow us to select answer sets matching specific conditions. For a program $P$ and a propositional formula $F$, we denote the answer sets of $P$ that satisfy $F$ by $\sigma_F(P) \coloneqq \{M \in \mathrm{AS}(P) \mid M \models F\}$. Moreover, we say that $F$ is *simple* if $\mathsf{var}(F) \subseteq \mathsf{at}(P)$.

**Example 11.** *Figure 3 illustrates the selection of answer sets among those of program $P_1$ from Example 1. We find that*

$$\begin{aligned}\sigma_{(\neg r\wedge s)\vee(p\wedge\neg q)}(P_1) &= \mathrm{AS}(P_1[p]) \cup \mathrm{AS}(P_1[s]) \\ &= \mathrm{AS}(P_1[\neg q]) \cup \mathrm{AS}(P_1[\neg r]) \\ &= \sigma_{(p\vee s)\wedge(\neg q\vee\neg r)}(P_1) \\ &= \{\{p, t\}, \{q, s, t\}\}.\end{aligned}$$

Next, we consider basic selection operations on answer sets and provide techniques to express a propositional query itself in ASP. Thereby, we immediately settle computational aspects as the complexity does not increase. In the following, we let $L$ denote a set of literals.

### 4.2 Matching All Elements (Terms)

If we are interested in the answer sets that satisfy all literals in $L$, we can consider the conjunction over $L$ (also called term or cube in the propositional satisfiability setting). We easily observe that selection on a term coincides with taking its literals $L$ as assumptions.

**Observation 12.** *Let $P$ be a program and $L$ a set of literals. Then, $\sigma_{\bigwedge_{\ell \in L} \ell}(P) = \mathrm{AS}(P \cup \{ \leftarrow \sim\ell \mid \ell \in L\})$.*

*Proof.* By definition, $\sigma_{\bigwedge_{\ell \in L} \ell}(P) = \{M \in \mathrm{AS}(P) \mid M \models \bigwedge_{\ell \in L} \ell\}$. Since $\bigwedge_{\ell \in L} \ell$ holds if and only if every $\ell \in L$ evaluates to true, we have that $\sigma_{\bigwedge_{\ell \in L} \ell}(P) = \{M \in \mathrm{AS}(P) \mid L^+ \subseteq M, L^- \cap M = \emptyset\}$. We conclude that $\sigma_{\bigwedge_{\ell \in L} \ell}(P) = \mathrm{AS}(P \cup \{ \leftarrow \sim\ell \mid \ell \in L\})$. □

### 4.3 Matching at Least One Element (Clauses)

If we are interested in the answer sets that satisfy some (at least one) literal in $L$, we can simply consider the disjunction over $L$ and make the following observation.

**Observation 13.** *Let $P$ be a program and $L$ a set of literals. Then, $\sigma_{\bigvee_{\ell \in L} \ell}(P) = \mathrm{AS}(P \cup \{ \leftarrow \sim L\})$.*

*Proof.* By definition, $\sigma_{\bigvee_{\ell \in L} \ell}(P) = \{M \in \mathrm{AS}(P) \mid M \models \bigvee_{\ell \in L} \ell\}$. Since $\bigvee_{\ell \in L} \ell$ holds if and only if some $\ell \in L$ evaluates to true, we have that $\sigma_{\bigvee_{\ell \in L} \ell}(P) = \{M \in \mathrm{AS}(P) \mid (L^+ \cap M) \cup (L^- \setminus M) \neq \emptyset\}$. We conclude that $\sigma_{\bigvee_{\ell \in L} \ell}(P) = \mathrm{AS}(P \cup \{ \leftarrow \sim L\})$. □

### 4.4 Matching CNFs

When selecting answer sets that match a formula $F$ in CNF, we require the answer sets to satisfy at least one literal of each clause. Letting $F$ be given as set $\{L_1, \ldots, L_m\}$ of clauses, where each clause $L_i$ is represented by the set of its literals, we make the following observation.

**Observation 14.** *Let $P$ be a program and $F$ a simple formula in CNF. Then, $\sigma_F(P) = \mathrm{AS}(P \cup \{ \leftarrow \sim L_i \mid L_i \in F\})$.*

*Proof.* The integrity constraints $\{ \leftarrow \sim L_i \mid L_i \in F\}$ yield the intersection of answer sets in $\sigma_{\bigvee_{\ell \in L_i} \ell}(P)$ over all clauses $L_i \in F$, which establishes the observation. □

### 4.5 Matching DNFs

For a formula $F$ in DNF, we require the answer sets to satisfy all literals of at least one term. Representing $F$ as set $\{L_1, \ldots, L_m\}$ of terms $L_i$ given by the set of their literals, the selection on $F$ can be expressed as follows.

**Observation 15.** *Let $P$ be a program and $F$ a simple formula in DNF. Then, $\sigma_F(P) = \{M \setminus \{a_1, \ldots, a_m\} \mid M \in \mathrm{AS}(P \cup \{a_i \leftarrow \sim\ell \mid 1 \leq i \leq m, \ell \in L_i\} \cup \{ \leftarrow a_1, \ldots, a_m\})\}$, where $a_1, \ldots, a_m$ are fresh atoms.*

*Proof.* Any $M \in \mathrm{AS}(P \cup \{a_i \leftarrow \sim\ell \mid 1 \leq i \leq m, \ell \in L_i\} \cup \{ \leftarrow a_1, \ldots, a_m\})$ excludes at least one atom $a_i \in \{a_1, \ldots, a_m\}$. Along with the fact that $a_i \notin M$ if and only if $\mathsf{var}(F) \cap M \models L_i$, the observation is immediate. □

**Example 16.** *Reconsider Example 11 and Figure 3. Mapping the CNF formula $(p \vee s) \wedge (\neg q \vee \neg r)$ to ASP yields the set $\{ \leftarrow \sim p, \sim s;\ \leftarrow q, r\}$ of integrity constraints. For the DNF formula $(\neg r \wedge s) \vee (p \wedge \neg q)$, we obtain the rules $\{a_1 \leftarrow r;\ a_1 \leftarrow \sim s;\ a_2 \leftarrow \sim p;\ a_2 \leftarrow q;\ \leftarrow a_1, a_2\}$.*

**Example 17** (Example 3 continued)**.** *Recall the predicted sum $8 = 3 + 6$ from Example 3. Querying for answer sets that match the CNF formula $\mathtt{prediction\_sum}(8) \wedge (\mathtt{digitL}(3) \vee \mathtt{digitR}(6))$ reveals this erroneous prediction in view of $\{\mathtt{digitL}(3), \mathtt{digitR}(6), \mathtt{prediction\_sum}(8), \mathtt{prediction\_sum}(9)\} \in \mathrm{AS}(P_{\mathrm{sum}} \cup \{ \leftarrow \sim\mathtt{prediction\_sum}(8);\ \leftarrow \sim\mathtt{digitL}(3), \sim\mathtt{digitR}(6)\})$. For a practical demonstration, such example queries can be explored in an interactive web application.*[1]

### 4.6 Non-Simple Formulas

For formulas $F$ containing variables that do not occur in a program $P$, we can also use the above constructions and add a "choice rule" $a \vee a'$ for each $a \in \mathsf{var}(F) \setminus \mathsf{at}(P)$. This does not increase the complexity, unless the number $|\mathsf{var}(F)|$ of variables is significantly larger than the original number $|\mathsf{at}(P)|$ of atoms.

## 5 Empirical Case Study

To demonstrate that reasoning upon facets is practically feasible, we conduct experiments on planning, argumentation, and configuration problem instances (Rusovac et al. 2024). We compare three approaches to count facets, all making use of the solver `clingo` (Gebser et al. 2014) version 5.6.2.[2] Our experiments are performed on an eight core Intel i7-10510U CPU 1.8 GHz machine with 16 GB of RAM, limiting the runtime on each instance to 60 seconds. Considering the explorative nature of our case study, we refrain from an extensive setup (Fichte et al. 2024b; Fichte et al. 2021).

**Solvers.** The `fasb` system[3] uses `clingo` to obtain brave and cautious consequences, and then the facets can be read off as their difference. While the total number of answer sets may be exponential, algorithms to compute brave or cautious consequences (Alviano et al. 2023a; Gebser, Kaufmann, and Schaub 2009) avoid an exhaustive enumeration and inspect a linear number of answer sets only. Moreover, the `model-guided` approach runs `clingo` with its default optimization strategy (Gebser et al. 2015) on a prototypical meta-encoding of our theoretical reduction (cf. Lemma 5), turning the input into a manifold program (Faber and Woltran 2011) such that an optimal answer set yields the facets. The `core-guided` approach works similar to

[1]https://drwadu.github.io/web-fasb.github.io/kr-example/
[2]https://github.com/potassco/clingo/releases/tag/v5.6.2
[3]https://github.com/drwadu/fasb/releases/tag/v0.2.0-beta

| solver | median[s] | mean[s] | solved |
|---|---|---|---|
| fasb | 0.6 | 11.2 | 92 |
| core-guided | 60.1 | 59.0 | 7 |
| model-guided | 60.1 | 60.0 | 2 |

Table 2: Overview of runtimes and numbers of solved instances out of 100 instances.

model-guided solving, yet running clingo with a core-guided optimization strategy (--opt-strategy=usc).

**Instances.** We consider instances that admit a large number of answer sets as well as many facets, making them interesting for selection queries or filtering among a large number of answer sets. Our benchmark set includes four smoke test planning instances as well as four claim-centric argumentation frameworks by Böhl, Gaggl, and Rusovac (2023), the latter relying on preferred extensions that necessitate disjunction, and four PC configuration instances (Fichte, Gaggl, and Rusovac 2022). Moreover, eight Linux package configuration instances stem from the 2012 MISC competition (Mancoosi Project 2019), where we omit optimization criteria on guessed packages forming the configuration. The instances are randomly selected from the available benchmark sets, and we additionally construct queries on each instance, resulting in a total number of 100 instances to run: the 20 original instances and variants that append randomly generated queries, corresponding to simple $\{1, 2\}$-CNFs and $\{1, 2\}$-DNFs that comprise three clauses or terms each.

**Expectations.** We expect that (E1) the theoretical reduction is limited in practice due to its size overhead, and that (E2) the core-guided approach comes nevertheless close to fasb due to the optimization strategy exploiting local structure.

**Observations.** Table 2 reveals that fasb outperforms both the model-guided and core-guided reduction approaches. The eight instances unsolved by fasb are disjunctive programs that express argumentation frameworks by Böhl, Gaggl, and Rusovac (2023). We also observe that the core-guided optimization strategy yields more solved instances than the model-guided approach, but its seven solved instances still remain far from fasb's performance.

**Summary.** Our results confirm expectation (E1), but do not match (E2). The algorithmic approach of fasb dominates reduction, regardless of the optimization strategy used to obtain the facets. We also note that computing facets for disjunctive programs is not only theoretically, but also practically, harder than for normal programs. Overall, our experiments demonstrate that both qualitative and quantitative reasoning upon the facets of an ASP program is feasible, utilizing the search algorithms readily supplied by state-of-the-art solvers.

# 6 Conclusion and Future Work

In this paper, we provide the first thorough study of the computational complexity of ASP facets. We systematically investigate qualitative and quantitative problems involving facets, establishing the complexity results outlined in in Table 1. Interestingly, this renders facet reasoning a central problem modeling suite, as complexity spans over canonical classes up to the third level of the polynomial hierarchy. In addition, we extend facet reasoning to queries on ASP solution spaces. We show that facet reasoning is quite robust, as more elaborate queries do not significantly increase the complexity. This underlines the significance of facets for analyzing large solution spaces. Finally, we conduct an empirical evaluation studying several approaches to compute facets. Our experiments demonstrate the feasibility of facet reasoning, where state-of-the-art solvers' algorithms for obtaining brave and cautious consequences show to be particularly efficient.

Given its central role for qualitative and quantitative problem settings, we expect facet reasoning to be of interest for various formalisms in KR and AI, e.g., quantified Boolean formulas, classical planning (Speck, Mattmüller, and Nebel 2020), abstract argumentation (Dachselt et al. 2022; Dewoprabowo et al. 2022; Fichte, Hecher, and Meier 2024), claim-centric argumentation (Fichte et al. 2023), description logics, epistemic logic programming (Eiter et al. 2024), constraint programming, and paraconsistent reasoning (Fichte, Hecher, and Meier 2021).

Our research opens up many promising directions for future work. It will be interesting to investigate and characterize practical applications that can be addressed by facet reasoning, while approximate or even exact solution counting techniques would be required otherwise. In the context of diversity of solutions (Ingmar et al. 2020), facets might be a promising tool to partition solution spaces at reasonable computational cost. Moreover, we plan to investigate the complexity of facets in the presence of preferences (Brewka et al. 2023). For reliability, also proofs of correctness might be of interest, which can be obtained by proof systems (Alviano et al. 2019; Fichte, Hecher, and Roland 2022).

## Acknowledgments

The authors are listed in reverse alphabetical order. Work has partially been carried out while Hecher and Fichte visited the Simons Institute at UC Berkeley. Our research is supported by the Austrian Academy of Sciences (ÖAW), DOC Fellowship; the Austrian Science Fund (FWF), grants P30168 and J4656; the BMBF, grant 01IS20056_NAVAS; the BMBF and DAAD, project 57616814 SECAI; ELLIIT funded by the Swedish government; the Society for Research Funding in Lower Austria (GFF), grant ExzF-0004; and the Vienna Science and Technology Fund (WWTF), grants ICT19-065 and ICT22-023. We are grateful to the anonymous reviewers for valuable comments that helped to improve this paper.

## References

Alrabbaa, C.; Rudolph, S.; and Schweizer, L. 2018. Faceted answer-set navigation. In *RuleML+RR'18*, 211–225. Springer.

Alviano, M.; Dodaro, C.; Leone, N.; and Ricca, F. 2015. Advances in WASP. In *LPNMR'15*, 40–54. Springer.

Alviano, M.; Calimeri, F.; Dodaro, C.; Fuscà, D.; Leone, N.; Perri, S.; Ricca, F.; Veltri, P.; and Zangari, J. 2017. The ASP system DLV2. In *LPNMR'17*, 215–221. Springer.

Alviano, M.; Dodaro, C.; Fichte, J. K.; Hecher, M.; Philipp, T.; and Rath, J. 2019. Inconsistency proofs for ASP: The ASP-DRUPE format. *Theory and Practice of Logic Programming* 19(5-6):891–907.

Alviano, M.; Dodaro, C.; Fiorentino, S.; Previti, A.; and Ricca, F. 2023a. ASP and subset minimality: Enumeration, cautious reasoning and MUSes. *Artificial Intelligence* 320:103931:1–25.

Alviano, M.; Trieu, L. L. T.; Son, T. C.; and Balduccini, M. 2023b. Advancements in xASP, an XAI system for answer set programming. In *ICCL'23*. CEUR Workshop Proceedings (CEUR-WS.org).

Azzolini, D., and Riguzzi, F. 2023a. Inference in probabilistic answer set programming under the credal semantics. In *AIxIA'23*, 367–380. Springer.

Azzolini, D., and Riguzzi, F. 2023b. Lifted inference for statistical statements in probabilistic answer set programming. *International Journal of Approximate Reasoning* 163:109040.

Barbiero, P.; Ciravegna, G.; Giannini, F.; Zarlenga, M. E.; Magister, L. C.; Tonda, A.; Lio, P.; Precioso, F.; Jamnik, M.; and Marra, G. 2023. Interpretable neural-symbolic concept reasoning. In *ICML'23*, 1801–1825. PMLR.

Bellodi, E.; Alberti, M.; Riguzzi, F.; and Zese, R. 2020. Map inference for probabilistic logic programming. *Theory and Practice of Logic Programming* 20(5):641–655.

Bidoít, N., and Froidevaux, C. 1991. Negation by default and unstratifiable logic programs. *Theoretical Computer Science* 78(1):85–112.

Böhl, E., and Gaggl, S. A. 2022. Tunas - fishing for diverse answer sets: A multi-shot trade up strategy. In *LPNMR'22*, 89–102. Springer.

Böhl, E.; Gaggl, S. A.; and Rusovac, D. 2023. Representative answer sets: Collecting something of everything. In *ECAI'23*, 271–278. IOS Press.

Brewka, G.; Delgrande, J.; Romero, J.; and Schaub, T. 2023. A general framework for preferences in answer set programming. *Artificial Intelligence* 325:104023.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Brewka, G.; Niemela, I.; and Truszczynski, M. 2003. Answer set optimization. In *IJCAI'03*. IJCAI Organization.

Calimeri, F.; Faber, W.; Gebser, M.; Ianni, G.; Kaminski, R.; Krennwallner, T.; Leone, N.; Maratea, M.; Ricca, F.; and Schaub, T. 2020. ASP-Core-2 input language format. *Theory and Practice of Logic Programming* 20(2):294–309.

Codd, E. F. 1970. A relational model of data for large shared data banks. *Communications of the ACM* 13(6):377–387.

Dachselt, R.; Gaggl, S. A.; Krötzsch, M.; Méndez, J.; Rusovac, D.; and Yang, M. 2022. NEXAS: A visual tool for navigating and exploring argumentation solution spaces. In *COMMA'22*, 116–127. IOS Press.

De Vos, M.; Kisa, D. G.; Oetsch, J.; Pührer, J.; and Tompits, H. 2012. Annotating answer-set programs in Lana. *Theory and Practice of Logic Programming* 12(4-5):619–637.

Dewoprabowo, R.; Fichte, J. K.; Gorczyca, P. J.; and Hecher, M. 2022. A practical account into counting Dung's extensions by dynamic programming. In *LPNMR'22*, 387–400. Springer.

Dodaro, C.; Gasteiger, P.; Reale, K.; Ricca, F.; and Schekotihin, K. 2019. Debugging non-ground ASP programs: Technique and graphical tools. *Theory and Practice of Logic Programming* 19(2):290–316.

Eén, N., and Sörensson, N. 2003. An extensible SAT-solver. In *SAT'08*, 502–518. Springer.

Eiter, T., and Geibinger, T. 2023. Explaining answer-set programs with abstract constraint atoms. In *IJCAI'23*, 3193–3202. IJCAI Organization.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence* 15(3-4):289–323.

Eiter, T., and Gottlob, G. 1997. The complexity class $\Theta_2^{\mathrm{P}}$: Recent results and applications in AI and modal logic. In *FCT'97*, 1–18. Springer.

Eiter, T.; Fichte, J. K.; Hecher, M.; and Woltran, S. 2024. Epistemic logic programs: Non-ground and counting complexity. In *IJCAI'24*, 3333–3341. IJCAI Organization.

Eiter, T.; Geibinger, T.; and Oetsch, J. 2023. Contrastive explanations for answer-set programs. In *JELIA'23*, 73–89. Springer.

Eiter, T.; Hecher, M.; and Kiesel, R. 2024. aspmc: New frontiers of algebraic answer set counting. *Artificial Intelligence* 330:104109.

Faber, W., and Woltran, S. 2011. Manifold answer-set programs and their applications. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning – Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, 44–63. Springer.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science* 1(1):51–60.

Fandinno, J., and Schulz, C. 2019. Answering the "why" in answer set programming – a survey of explanation approaches. *Theory and Practice of Logic Programming* 19(2):114–203.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2017. Answer set solving with bounded treewidth revisited. In *LPNMR'17*, 132–145. Springer.

Fichte, J. K.; Hecher, M.; McCreesh, C.; and Shahab, A. 2021. Complications for computational experiments from modern processors. In *CP'21*, 25:1–25:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Fichte, J. K.; Hecher, M.; Mahmood, Y.; and Meier, A. 2023. Quantitative reasoning and structural complexity for claim-centric argumentation. In *IJCAI'23*, 3212–3220. IJCAI Organization.

Fichte, J. K.; Gaggl, S. A.; Hecher, M.; and Rusovac, D. 2024a. IASCAR: Incremental answer set counting by anytime refinement. *Theory and Practice of Logic Programming* 1–28.

Fichte, J. K.; Geibinger, T.; Hecher, M.; and Schlögel, M. 2024b. Parallel empirical evaluations: Resilience despite concurrency. In *AAAI'24*, 8004–8012. AAAI Press.

Fichte, J. K.; Gaggl, S. A.; and Rusovac, D. 2022. Rushing and strolling among answer sets – navigation made easy. In *AAAI'22*, 5651–5659. AAAI Press.

Fichte, J. K.; Hecher, M.; and Meier, A. 2021. Knowledge-base degrees of inconsistency: Complexity and counting. In *AAAI'21*, 6349–6357. AAAI Press.

Fichte, J. K.; Hecher, M.; and Meier, A. 2024. Counting complexity for reasoning in abstract argumentation. *Journal of Artificial Intelligence Research* 80.

Fichte, J. K.; Hecher, M.; and Nadeem, M. A. 2022. Plausibility reasoning via projected answer set counting – a hybrid approach. In *IJCAI'22*, 2620–2626. IJCAI Organization.

Fichte, J. K.; Hecher, M.; and Roland, V. 2022. Proofs for propositional model counting. In *SAT'22*, 30:1–30:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Gebser, M.; Pührer, J.; Schaub, T.; and Tompits, H. 2008. A meta-programming technique for debugging answer-set programs. In *AAAI'08*, 448–453. AAAI Press.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer set solving in practice*. Morgan & Claypool Publishers.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2014. Clingo = ASP + control: Preliminary report. *CoRR* abs/1405.3694.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Romero, J.; and Schaub, T. 2015. Progress in clasp series 3. In *LPNMR'15*, 368–383. Springer.

Gebser, M.; Kaminski, R.; and Schaub, T. 2011. aspcud: A linux package configuration tool based on answer set programming. In *LoCoCo'11*, 12–25. EPTCS.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2009. The conflict-driven answer set solver clasp: Progress report. In *LPNMR'09*, 509–514. Springer.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP'88*, 1070–1080. MIT Press.

Gelfond, M. 1991. Strong introspection. In *AAAI'91*, 386–391. AAAI Press.

Hecher, M. 2022. Treewidth-aware reductions of normal ASP to SAT – is normal ASP harder than SAT after all? *Artificial Intelligence* 304:103651.

Ingmar, L.; Garcia de la Banda, M.; Stuckey, P. J.; and Tack, G. 2020. Modelling diversity of solutions. In *AAAI'20*, 1528–1535. AAAI Press.

Janhunen, T., and Niemelä, I. 2016. The answer set programming paradigm. *AI Magazine* 37(3):13–24.

Janhunen, T.; Oikarinen, E.; Tompits, H.; and Woltran, S. 2009. Modularity aspects of disjunctive stable models. *Journal of Artificial Intelligence Research* 35:813–857.

Kabir, M.; Chakraborty, S.; and Meel, K. S. 2024. Exact ASP counting with compact encodings. In *AAAI'24*, 10571–10580. AAAI Press.

Kleine Büning, H., and Lettmann, T. 1999. *Propositional logic – deduction and algorithms*. Cambridge University Press.

Lohrey, M., and Rosowski, A. 2023. On the complexity of diameter and related problems in permutation groups. In *ICALP'23*, 134:1–134:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Lukasiewicz, T., and Malizia, E. 2017. A novel characterization of the complexity class $\Theta_k^{\mathrm{P}}$ based on counting and comparison. *Theoretical Computer Science* 694:21–33.

Mancoosi Project. 2019. Data from the Mancoosi solver competition and articles. Zenodo: https://zenodo.org/records/3556644.

Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2018. DeepProbLog: Neural probabilistic logic programming. In *NeurIPS'18*, 3753–3763. Curran Associates, Inc.

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

Marek, V. W., and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*, 375–398. Springer.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25(3-4):241–273.

Oetsch, J.; Pührer, J.; and Tompits, H. 2018. Stepwise debugging of answer-set programs. *Theory and Practice of Logic Programming* 18(1):30–80.

Papadimitriou, C. H., and Yannakakis, M. 1982. The complexity of facets (and some facets of complexity). In *STOC'82*, 255–260. ACM.

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

Rusovac, D.; Hecher, M.; Gebser, M.; Gaggl, S. A.; and Fichte, J. K. 2024. Navigating and querying answer sets: How hard is it really and why? (Empirical case study). Zenodo: https://doi.org/10.5281/zenodo.12737216.

Schekotihin, K. 2015. Interactive query-based debugging of ASP programs. In *AAAI'15*, 1597–1603. AAAI Press.

Shen, Y.-D., and Eiter, T. 2016. Evaluating epistemic negation in answer set programming. *Artificial Intelligence* 237:115–135.

Sheridan, D. 2004. The optimality of a fast CNF conversion and its use with SAT. In *SAT'04*, 329–334.

Soininen, T., and Niemelä, I. 1999. Developing a declarative rule language for applications in product configuration. In *PADL'99*, 305–319. Springer.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic top-$k$ planning. In *AAAI'20*, 9967–9974. AAAI Press.

Stockmeyer, L. J., and Meyer, A. R. 1973. Word problems requiring exponential time. In *STOC'73*, 1–9. ACM.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science* 3(1):1–22.

Truszczyński, M. 2011. Trichotomy and dichotomy results on the complexity of reasoning with disjunctive logic programs. *Theory and Practice of Logic Programming* 11(6):881–904.

Tseytin, G. S. 1983. On the complexity of derivation in propositional calculus. In *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, 466–483. Springer.

Wagner, K. W. 1987. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science* 51:53–80.

Wrathall, C. 1976. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science* 3(1):23–33.