Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Counting Complexity of ASP

Max Bannach[1] , **Johannes K. Fichte**[2] , **Johanna Groven**[2] , **Markus Hecher**[3]
[1]European Space Agency, The Netherlands
[2]Linköping University, Sweden
[3]University of Potsdam, Germany & CNRS, CRIL, UMR8188, France
max.bannach@esa.int, firstname.lastname@liu.se, hecher@cril.fr

Abstract

Answer Set Programming (ASP) is a mature and widely used framework for modeling and solving problems in AI, knowledge representation and reasoning, and combinatorial search. Counting answer sets is of growing importance for analyzing search spaces, navigating ASP programs, and enabling probabilistic reasoning. While Truszczyński established a complete hierarchy for the computational complexity of ASP decision and reasoning problems (skeptical and credulous), a corresponding systematic treatment of counting problems has been missing so far. We close this gap by providing an almost complete characterisation of the counting complexity landscape for ASP. A remaining gap arises, caused by the minimality of disjunctions in rule heads for guessing. To address this issue, we first show that Krom programs do not admit a polynomial-time approximation scheme (FPRAS) under standard complexity-theoretic assumptions. Then, we replace disjunctions by choice rules and introduce a controlled fragment in which choices are allowed and every rule is simultaneously Horn and Krom (Choice-Horn-Krom). However, we prove that counting answer sets of an arbitrary ASP program can already be done by counting answer sets of two Choice-Horn-Krom programs. This result demonstrates the expressive power of ASP and yields a conceptually simpler alternative to Valiant's classical reduction from #SAT to #Krom-SAT, which a very well-known result in propositional logic.

1 Introduction

Answer Set Programming (ASP) (Brewka, Eiter, and Truszczyński 2011; Gebser et al. 2012; Janhunen and Niemelä 2016) is a mature and widely used framework for modeling and solving problems in AI with plenty of application areas, including industrial (Nabeshima et al. 2026; Alviano and Reiners 2025; Altay-Kern et al. 2025) and academic applications (Hahn et al. 2024; Baumeister et al. 2024). The computational complexity of decision and optimization problems in ASP is well understood ranging from polynomial-time solvable (Marek and Truszczyński 1991; Gelfond and Lifschitz 1988) to the second level of the polynomial hierarchy (Eiter and Gottlob 1995) for propositional programs. Truszczyński (2011) established a comprehensive trichotomy for decision and reasoning problems (skeptical and credulous) with detailed characterization of subclasses of programs. In the ASP community, these results are widely known as an equivalent of Schaefer's classes in propositional satisfiability (SAT), constraint satisfaction (CSP) (Schaefer 1978; Chen 2006), or Post's lattice in universal algebra (Post 2016; Lau 2006).

While decisions, reasoning, and optimization in ASP have been well studied, counting the number of solutions is becoming increasingly more important due to numerous applications (Darwiche 2020; Zhang et al. 2024; Zak et al. 2025). In logic programming, counting enables probabilistic reasoning without enumerating (Azzolini and Riguzzi 2025; Bogaerts and Van den Broeck 2015; Fierens et al. 2015), and navigating and understanding large search spaces (Fichte, Gaggl, and Rusovac 2022; Kabir and Meel 2023). Surprisingly, the complexity landscape of ASP counting is quite open. While we have results for Horn (Dowling and Gallier 1984), stratified (Gelfond and Lifschitz 1988), tight (Clark 1977), and disjunctive programs (Fichte et al. 2017), contrary to what one might expect, the existing trichotomy results do not carry over to counting.

In this paper, we address this gap by studying the computational complexity of ASP in the well-known trichotomy framework by Truszczyński. It turns out that ASP counting is much harder to comprehend than propositional satisfiability (#SAT). The minimality of disjunctions in rule heads significantly complicates any complexity analysis. To address this issue, we introduce a controlled fragment where we replace disjunctions with choice rules and allow that every rule is simultaneously Horn and Krom (Choice-Horn-Krom). In fact, the class of Horn together with choices is very natural as it comes very close to the concept of inductive definitions. In practice, reachability, type inference, constraint propagation, database rules fit very well with this type of programs. Our study on Choice-Horn-Krom provides a significant step to closing this gap. Moreover, we obtain new valuable insights into the complexity class #P itself. On the one hand, we can show that #Choice-Horn-Krom does not admit a polynomial-time approximation scheme (FPRAS). On the other hand, we establish that #ASP can be reduced to counting answer sets of two Choice-Horn-Krom programs.

Contributions Table 1 presents our main contributions. We emphasise on the following results.

1. We establish a comprehensive overview for answer set counting in the framework of Truszczyński.

Class C	Name	Cons	#AS	Result
$\{[1,\infty,0],[0,\infty,\infty]\}$	Horn	P	FP	[A]
$\{[\infty,\infty,0],[0,0,0]\}$	NegIntFree	P	no-FPRAS†	Lem 4.3
$\{[\infty,1,0],[0,1,0]\}$	DHorn	P	open	
$\{[i,0,k],[0,j,k] \mid i+k\leq 2, j+k\leq 1\}$		NP-c	no-FPRAS†	Lem 4.3
$\{[i,j,k] \mid i+j+k\leq 2\}$	Krom	NP-c	no-FPRAS†	Lem 4.3
$\{[2,0,0],[0,2,0]\}$ or $\{[2,0,0],[1,1,0]\}$	Choice-Horn-Krom	P	#P-h	Th 6+8
$\{[1,\infty,\infty],[0,\infty,\infty]\}$	Normal	NP-c	#P-c	Lem 4.1, Obs 5
$\{[\infty,1,\infty],[0,\infty,\infty]\}$	DNormal	NP-c	#P-c	Lem 4.1, Obs 5
$\{[\infty,\infty,0],[0,\infty,0]\}$	NegFree	NP-c	#P-c	Lem 4.1, Obs 5
otherwise	Disj	Σ_2^{P}-c	$\#\cdot$coNP-c	Lem 4.4, [B]
	Strat	P	FP	[C]
	Tight	NP-c	#P-c	[D]

Table 1: C refers to the class of programs or the class that consists of $\Delta_P \preceq \Delta(C)$. †: unless RP = NP. [A]: (Dowling and Gallier 1984); [B]: (Fichte et al. 2017); [C]: (Gelfond and Lifschitz 1988); [D]: (Clark 1977). Results for Cons can be found in (Truszczyński 2011) and [C], [D].

2. We show that there is no polynomial-time approximation schema (FPRAS) for #Krom-ASP unless RP = NP, where RP consists of the decision problems solvable in randomized polynomial time with one-sided error.
3. We provide a reduction from an arbitrary ASP program to two Choice-Horn-Krom programs. We establish that counting the number of answer sets of both programs and subtracting the counts is sufficient for obtaining the count of the initial program.

Related Works Approximate and exact answer set counting solvers (Kabir et al. 2022; Kabir, Chakraborty, and Meel 2024; Kabir, Chakraborty, and Meel 2025) as well as algebraic counting solvers (Eiter, Hecher, and Kiesel 2024) are available. There are also translations (Janhunen 2006; Bomanson and Janhunen 2013) from ASP to SAT and its extensions (Gebser, Janhunen, and Rintanen 2014; Bomanson et al. 2016). Systematic answer set enumeration techniques have been implemented into state-of-the-art solvers (Gebser, Kaufmann, and Schaub 2009). Complexity results and practical considerations into ASP counting when including specific structure of the input (treewidth) have been considered (Jakl et al. 2008; Jakl, Pichler, and Woltran 2009; Jakl et al. 2008; Fichte et al. 2017). The counting complexity for circumscription, which is closely related to ASP, has been studied (Durand and Hermann 2008) and a trichotomy result been established (Durand, Hermann, and Nordh 2012). Coste-Marquis and Marquis (1999) study the complexity of closed world reasoning and circumscription focusing on classes of formulas such as Blake formulas, DNFs, disjunctions of Horn formulas, and disjunctions of renamable Horn formulas. Cadoli and Lenzerini (1994) study the decision complexity of closed world reasoning and circumscription, and in the context of Horn, DualHorn, and Krom formulas and related classes. A reduction from SAT to MAX-KROM-SAT was known since 1976 (Garey, Johnson, and Stockmeyer 1976). This reduction was improved by Trevisan et al. (2000) and is an active field of research. Recently, there has been progress on counting solvers for restricted fragments of #SAT (Dubray, Schaus, and Nijssen 2023). A similar counting reductions as in Section 4 were recently shown for #2SAT (Bannach et al. 2025; Bannach et al. 2026).

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), and counting complexity (Toda and Watanabe 1992; Hemaspaandra and Vollmer 1995; Durand, Hermann, and Kolaitis 2005). In particular, $\Delta_0^{\mathrm{P}} := \Pi_0^{\mathrm{P}} := \Sigma_0^{\mathrm{P}} := \mathrm{P}$, $\Delta_i^{\mathrm{P}} := P^{\Sigma_{i-1}^{\mathrm{P}}}$, $\Sigma_i^{\mathrm{P}} := \mathrm{NP}^{\Sigma_i^{\mathrm{P}}}$, and $\Pi_i^{\mathrm{P}} := \mathrm{coNP}^{\Sigma_i^{\mathrm{P}}}$ for $i > 0$ where C^D is the class C of decision problems augmented by an oracle for some complete problem in class D. We use of complexity classes preceded with the sharp-dot operator '#·'. A *witness* function is a function $w\colon \Sigma^* \to \mathcal{P}^{<\omega}(\Gamma^*)$, where Σ and Γ are alphabets, mapping to a finite subset of Γ^*. Such functions associate with the counting problem "given $x \in \Sigma^*$, find $|w(x)|$". If C is a decision complexity class then $\#\cdot C$ is the class of all counting problems whose witness function w satisfies (1.) $\exists$ polynomial p such that for all $y \in w(x)$, we have that $|y| \leqslant p(|x|)$, and (2.) the decision problem "given x and y, is $y \in w(x)$?" is in C. A *parsimonious* reduction between two counting problems $\#A, \#B$ preserves the cardinality between the corresponding witness sets and is computable in polynomial time. A *subtractive* reduction between two counting problems $\#A$ and $\#B$ is composed of two functions f, g between the instances of A and B such that $B(f(x)) \subseteq B(g(x))$ and $|A(x)| = |B(g(x))| - |B(f(x))|$, where A and B are respective witness functions. Let M be a deterministic Turing machine (DTM) or non-deterministic Turing Machine (NTM) such

that for all inputs x, every computation path on x is accepting or rejecting and is bounded in length by a function $t(x)$. For input x, we define $\mathrm{acc}_M(x), \mathrm{rej}_M(x)$ as the number of *accepting and rejecting computation paths* of x on M respectively. Note that $\mathrm{acc}_M(x) + \mathrm{rej}_M(x)$ is the number of all computation paths of x. We define **FP** as the class of all function problems that are polynomial-time reducible to $\mathrm{acc}_M(x)$ for some poly-time DTM M, #P as the class of all counting problems that are polynomial-time reducible to $\mathrm{acc}_M(x)$ for some poly-time NTM M. A randomized Turing Machine M (RTM) takes additionally to its normal input $x \in \{0,1\}^*$ a (random) bit string $r \in \{0,1\}^{q(x)}$, where $q(x)$ is a bound on the computation steps of M on x. The probability $\Pr[M \text{ accepts } x]$ of M accepting x is defined as the fraction of $r \in \{0,1\}^{q(x)}$ on which $(x,r) \in L(M)$ over the total number of random bit strings. Let **RP** be the class of computation problems L such that there exists a poly-time RTM M such that for $x \in L$, we have that $\Pr[M \text{ accepts } x] \geq 0.5$ and for $x \notin L$, we have $\Pr[M \text{ accepts } x] = 0$. A fully *randomized approximation scheme (**FPRAS**)* for a function $f : \{0,1\}^* \to \mathbb{R}_{\geq 0}$ is a randomized algorithm $A(x, \varepsilon, \delta)$ such that for every $x \in \{0,1\}^*, \varepsilon, \delta > 0$, $A(x, \varepsilon, \delta)$ outputs y such that $\Pr[(1-\varepsilon)f(x) \leq y \leq (1+\varepsilon)f(x)] \geq 1 - \delta$ in time polynomial in $|x|, \varepsilon^{-1}, \log(\delta^{-1})$, i.e., A computes at least an ε approximation of $f(x)$ with probability at least $1 - \delta$.

(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 $\mathrm{var}(F)$ the set of variables of F. Logical operators $\wedge$, $\vee$, and $\neg$, 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 its set of clauses or terms, respectively. We require a restricted version of quantified Boolean formulas, namely, 2-QBF in the form $Q = \exists V_1 . \forall V_2 . F$ where V_1 and V_2 are disjoint sets of propositional variables and F is a propositional formula in disjunctive normal form (DNF) that contains only the variables in $V_1 \cup V_2$. The truth (evaluation) of QBFs is defined in the standard way, where Q evaluates to true if there exists an assignment $\alpha : V_1 \to \{0,1\}$ such that $\forall V_2 . F[\alpha]$ evaluates to true. The evaluation problem #2-QBF outputs the number of total assignments α for which $\forall V_2 . F[\alpha]$ evaluates to true.

Answer Set Programming (ASP) For a comprehensive introduction, we refer to standard texts (Janhunen and Niemelä 2016; Gebser et al. 2012). We restrict ourselves to propositional programs and will also consider choice rules. Let ℓ, m, n be non-negative integers such that $\ell \leq m \leq n$, and $a_1, \ldots, a_n$ distinct propositional atoms. We let programs consist of *rules*, in particular, *integrity rules* (constraints), *choice rules* (guesses). If the program may contain a choice rule, we make that explicit otherwise we assume no choice rules. A *disjunctive rule* r is of the form $a_1 \vee \cdots \vee a_l \leftarrow a_{\ell+1}, \ldots, a_m, {\sim}a_{m+1}, \ldots, {\sim}a_n$, which, intuitively, means that at least one atom of $a_1, \ldots, a_\ell$ must be true if all atoms $a_{\ell+1}, \ldots, a_m$ are true and there is no evidence that any atom of $a_{m+1}, \ldots, a_n$ is true. A *(restricted) choice rule* is an expression of the form, $\{a_1\}$, intuitively, a_1 is allowed to be choosen freely (as in propositional satisfiability), i.e., true or false. By $H_r := \{a_1, \ldots, a_l\}$, $B_r^+ := \{a_{l+1}, \ldots, a_m\}$, and $B_r^- := \{a_{m+1}, \ldots, a_n\}$, we denote the *head*, *positive* or *negative body* atoms of r, respectively. A *program* Π is a set of rules, where $\mathrm{at}(\Pi) := \bigcup_{r \in \Pi}(H_r \cup B_r^+ \cup B_r^-)$ denotes its atoms. We call a rule r with both $H_r \neq \emptyset$ and body $B_r^+ \cup B_r^- \neq \emptyset$ *non-simple*. An interpretation $M \subseteq \mathrm{at}(\Pi)$ *satisfies* a disjunctive rule r if $(H_r \cup B_r^-) \cap M \neq \emptyset$ or $B_r^+ \not\subseteq M$ and always satisfies a choice rule. M is a *model* of Π if M satisfies every rule $r \in \Pi$. The *reduct* r^M (i) of a restricted choice rule r is the set $\{a \mid a \in H_r \cap M\}$ of rules (the rule $a \leftarrow$ or no rule) and (ii) of a disjunctive rule r is the singleton $\{H_r \leftarrow B_r^+ \mid B_r^- \cap M = \emptyset\}$, and $\Pi^M := \{r' \mid r' \in r^M, r \in \Pi\}$ is called *GL reduct* of Π with respect to M. Then, M is an *answer set* of Π if M is a model of Π such that no interpretation $N \subsetneq M$ is a model of Π^M (Gelfond and Lifschitz 1988; Gelfond and Lifschitz 1991). We let the set $\mathrm{AS}(\Pi)$ consist of all answer sets of Π. The problem Cons(Π) asks to decide whether $\mathrm{AS}(\Pi) \neq \emptyset$ and *#AS*(Π) asks to output $|\mathrm{AS}(\Pi)|$.

ASP Subclasses We say that a rule r is *normal* if $|H_r| \leq 1$, *positive* if $B_r^- = \emptyset$, and an *integrity constraint* if $H_r = \emptyset$. Moreover, a program Π has a certain property if all its rules have the property. The *dependency graph* D_Π of Π is the directed graph on the atoms $\bigcup_{r \in \Pi} H_r \cup B_r^+$, where for every rule $r \in \Pi$, two atoms $a \in B_r^+$ and $b \in H_r$ are joined by an edge (a, b). We say Π is *tight* if D_Π is acyclic (Fages 1994). A program Π is called head-cycle free if its incidence graph does not contain any directed cycles (Ben-Eliyahu and Dechter 1994). By Disj, Normal, DNormal, NegFree, Horn, DHorn, NegIntFree, and Tight, we denote the class of all disjunctive, normal, negation-free, dual-normal, Horn, dual-Horn, or tight programs, respectively, all without choice rules. Truszczyński went beyond those classes and defined detailed restrictions to analyze the computational complexity and establish a Trichotomy (Truszczyński 2011). The (arity) restrictions are given by specifying upper bounds on the arity of the rules. Therefore, let $U \in \{0, 1, 2, \ldots\} \cup \{\infty\}$ where ∞ represents unbounded arity. Now, let $i, m, n \in U$. Then, $\alpha := [i, m, n]$ defines $|H_r| \leq i$, $\left|B_r^+\right| \leq m$, and $\left|B_r^-\right| \leq n$. Moreover, we let $\alpha_1 := i$, $\alpha_2 := m$, and $\alpha_3 := n$. Using these definitions, we can define a class of programs by stating sets of restricted arities, e.g., $\{[1, \infty, \infty], [0, \infty, \infty]\}$ specifies Normal. We let $\mathcal{A} = \{[k, m, n] \mid k, m, n \in U\}$ consist of all possible arities of a disjunctive program including representations for unbounded arities. A total order $\preceq$ on $\mathcal{A}$ allows for comparing classes of programs defined on the restrictions of arities. Formally, for two restrictions $\alpha, \beta \in \mathcal{A}$, we have $\alpha \preceq \beta$ if and only if (i) $\alpha_1 = 0$, then $\beta_1 = 0$, and (ii) $\alpha_i \leq \beta_i$, for $i \in \{1, 2, 3\}$ where $\leq$ corresponds to the normal total on the natural numbers extended by $a \leq \infty$ for all $a \in \mathbb{N}_0 \cup \{\infty\}$. Let $\Delta \subseteq \mathcal{A}$, then we define $C(\Delta)$ to be the class of all *finite* programs P that satisfy the following condition: for every rule $r \in P$ there is $\alpha \in \Delta$ such that $\alpha_r \preceq \alpha$, where α_r denotes the arity of r. The decision complexity of subclasses is well-studied with a full

characterization based on the arity characteristics.

Proposition 1 (Truszczyński, 2011)**.** *Let* $\Pi \in C(\Delta)$ *be a program. Then,* $\textsc{Cons}(\Pi) \in \mathrm{P}$ *if* $\Delta \preceq \{[1,\infty,0],[0,\infty,\infty]\}$ *or* $\Delta \preceq \{[\infty,1,0],[0,1,0]\}$ *or* $\Delta \preceq \{[i,j.0] \mid i+j \leq 2\}$*; and otherwise* $\textsc{Cons}(\Pi) \in$ NP*-c, if* $\Delta \preceq \{[1,\infty,\infty],[0,\infty,\infty]\}$ *or* $\Delta \preceq \{[\infty,1,\infty],[0,\infty,\infty]\}$ *or* $\Delta \preceq \{[\infty,\infty,0],[0,\infty,0]\}$*; and otherwise* $\textsc{Cons}(\Pi) \in \Sigma_2^{\mathrm{P}}$*-c.*

For Horn, existing results (Dowling and Gallier 1984) directly translate to counting due to the possible number of answer sets. The best-known reduction for programs in Normal to SAT (Janhunen 2006) uses so-called level mappings and results in a subquadratic overhead, while preserving answer sets one-by-one (bijection). The general case for programs in Disj has been established $\# \cdot$ coNP-c by Fichte et al. (2017). In addition, there are results for classes that are based on acyclicity. For Strat, the answer sets are bounded and therefore existing decision results immediately translate (Gelfond and Lifschitz 1988). For Tight, the reductions to SAT preserve the number of answer sets (Clark 1977) and we can easily reduce SAT to a Tight program, resulting in #P-c (Valiant 1979b; Valiant 1979a). In detail, for a program Π, let the non-simple rules, whose head contain $a \in \mathrm{at}(\Pi)$ be $\mathfrak{H}(a) = \{r \in \Pi \mid r \text{ non-simple}, a \in H_r\}$. The completion $\mathfrak{C}(\Pi)$ is obtained from Π by adding the following sets of rules (Clark 1977) $\mathfrak{C}(\Pi) = \{r^h \leftarrow B_r^+, \neg B_r^- \mid h \in \mathrm{at}(\Pi), r \in \mathfrak{H}(h)\} \cup \{\bot \leftarrow h, \neg r_1^h, \ldots, \neg r_\ell^h \mid h \in \mathrm{at}(\Pi), \mathfrak{H}(h) = \{r_1, \ldots, r_\ell\}\} \cup \{\bot \leftarrow r^h, \neg a;\ \bot \leftarrow r^h, b \mid h \in \mathrm{at}(\Pi), r \in \mathfrak{H}(h), a \in B_r^+, b \in B_r^-\}$. Intuitively, for a tight program the completion contains all the necessary rules required to compute answer sets via rule satisfiability. We call program Π with $\mathfrak{C}(\Pi) = \Pi$ *completed*. Other existing reductions to SAT by loop formulas (Lin and Zhao 2003) might already result in an exponential overhead in the decision case. However, there are translation for head-cycle free programs, whose answer sets can be characterized just via rule satisfiability (Ben-Eliyahu and Dechter 1994).

3 ASP Counting Complexity

While counting complexity results for basic classes of ASP programs are known, there is no systematic approach to characterize the complexity based on restrictions of arities in the head, positive body, and negative body. We follow the structural characterization introduced by (Truszczyński 2011) for decision and reasoning problems and investigate the computational complexity of fragments. We present a simplified overview in Table 1. Before we present our results in detail, we make the following observation.

Observation 2. *Let* $\Pi \in C(\{[x,0,0]\} \cup S)$ *for some* $x \in \mathbb{N}$ *and* $S \subseteq \mathcal{A}$ *such that* Π *is head-cycle free. Then, for each* $k < x$*, there exists a program* $\Pi' \in C(\{[k,0,x-k]\} \cup S)$ *with* $AS(\Pi) = AS(\Pi')$ *such that* Π' *is computable in log-space and linear time from* Π *for every fixed* k*,* x*.*

Proof. In more detail, for head-cycle-free programs, we can employ the construction (Ben-Eliyahu and Dechter 1994, Theorem 4.17). Rules of the form $a_1 \vee \ldots \vee a_k$ can be replaced with k rules $\{a_1 \leftarrow {\sim}a_2, \ldots, {\sim}a_k\} \cup \{a_2 \leftarrow {\sim}a_1, {\sim}a_3, \ldots, {\sim}a_k\} \cup \ldots \cup \{a_k \leftarrow {\sim}a_1, \ldots, {\sim}a_{k-1}\}$, without affecting the set of answer sets. This transformation is clearly computable in log-space and linear time with respect to Π, as it only increases the number of rules by k. We observe that their proof holds for arbitrary splits of literals between the head and negative body. □

In more detail, we have the following Theorem.

Theorem 3. *Let* $\Pi \in C(\Delta)$ *be a program where* Δ *defines a structural restriction on* H_r*,* B_r^+*, and* B_r^-*.*

1. $\#AS(\Pi) \in \mathrm{P}$ *if* $\Delta_\Pi \preceq \{[1,\infty,0],[0,\infty,\infty]\}$*;*
2. $\#AS(\Pi) \in \#\mathrm{P}$ *if otherwise*
   - $\{[1,\infty,0],[0,\infty,\infty]\} \preceq \Delta_\Pi$*, or*
   - $\{[i,j,k] \mid i+j+k \leq 2\} \preceq \Delta_\Pi$*,*

   and
   - $\Delta_\Pi \preceq \{[1,\infty,\infty],[0,\infty,\infty]\}$*, or*
   - $\Delta_\Pi \preceq \{[\infty,1,\infty],[0,\infty,\infty]\}$*, or*
   - $\Delta_\Pi \preceq \{[\infty,\infty,0],[0,\infty,0]\}$*;*
3. $\#AS(\Pi) \in \# \cdot \mathrm{coNP}$*, otherwise.*

Before we can establish this theorem, we systematically investigate which cases yield hardness and establish the following lower bounds.

Lemma 4. *Let* $\Pi \in C(\Delta)$ *be a program. Then,* $\#AS(\Pi)$

1. *is* #P*-hard if* Δ *is any of*
   - *(a)* $\{[3,0,0],\ [0,2,0]\}$,
   - *(b)* $\{[2,0,1],\ [0,2,0]\}$, $\{[1,0,2],\ [0,2,0]\}$,
   - *(c)* $\{[2,0,0],\ [0,2,0],[0,0,3]\}$
   - *(d)* $\{[2,1,0],\ [0,2,0]\}^\star$, $\{[2,0,0],\ [0,3,0]\}^\star$,
   - *(e)* $\{[2,0,0],\ [0,3,0]\}^\star$, $\{[1,0,1],\ [0,3,0]\}^\star$,
   - *(f)* $\{[2,0,0],\ [0,2,1]\}^\star$,
   - *(g)* $\{[2,0,0],\ [0,2,0],[0,1,2]\}^\star$,
   - *(h)* $\{[1,2,0],\ [2,0,0],\ [0,1,0]\}^\star$,
   - *(i)* $\{[1,2,0],\ [1,0,1],\ [0,1,0]\}^\star$,
2. *is* #P*-hard by subtractive reduction if* Δ *is any of*
   - *(a)* $\{[3,0,0],[1,2,0]\}^\star$,
   - *(b)* $\{[2,0,0],[1,2,0]\}^\star$,
   - *(c)* $\{[1,0,1],[1,2,0]\}^\star$,
   - *(d)* $\{[2,0,0],[0,1,0]\}^\star$,
3. *has no* FPRAS *unless* RP = NP *(and therefore unlikely to be efficiently countable) if* Δ *is any of*
   - *(a)* $\{[2,0,0],[0,2,0]\}^\star$ *(by reduction from* #2SAT*),*
   - *(b)* $\{[1,0,1],[0,2,0]\}^\star$ *(by reduction from* #2SAT*),*
   - *(c)* $\{[2,0,0]\}^\star$ *(by reduction from* #mon-2SAT*),*
   - *(d)* $\{[1,0,1]\}^\star$ *(by reduction from* #mon-2SAT*),*
4. $\# \cdot$ coNP*-hard if* Δ *is any of*
   - *(a)* $\{[2,0,0],[1,2,0],[1,0,1]\}^\star$,
   - *(b)* $\{[2,0,0],[1,2,0],[0,0,1]\}^\star$,

Proof. For Case 1, we construct a parsimonious reduction (subtractive 1.j–i) from #3-SAT to *#AS* that transforms a #3-SAT instance into a program Π. Therefore, let F be an instance of #3-SAT. *(Case 1a):* Our resulting program Π is in $\Pi \preceq C(\{[3,0,0],[0,2,0]\})$. Π takes as atoms $\mathrm{var}(F) \cup \{\overline{x} \mid x \in \mathrm{var}(F)\}$ and rules $\Pi = R_1 \cup R_2 \cup R_3$, where $R_1 = \{x \mid \overline{x} \mid x \in \mathrm{var}(F)\}$, $R_2 = \{\leftarrow x, \overline{x} \mid x \in \mathrm{var}(F)\}$, $R_3 = \{a^* \mid$

$b^* \mid c^* \mid \{a,b,c\} \in F\}$, where $x^* := \overline{y}$ if $x = \neg y$ and $x^* := y$ if $x = y$ for $y \in \mathrm{var}(F)$. First note that every model for Π is also a minimal model and answer set for Π by construction. Observe that for each $x \in \mathrm{var}(F)$, rules R_1 enforce that at least one of $x, \overline{x}$ is contained in each model for Π and rules R_2 enforces that at most one of $x, \overline{x}$ is contained in each model for Π. Thus, each model for Π must contain exactly one of $x, \overline{x}$ for each $x \in \mathrm{var}(F)$. For each set $M \subseteq \mathrm{var}(F)$, we construct the set $I = M \cup \{\overline{x} \mid x \in \mathrm{var}(F) \setminus M\}$. We observe that I is a model for all rules of R_1 and R_2 and it is a model for all rules R_3 exactly if M is a model for F. This is a bijective mapping between models for F and answer sets for Π and we are able to follow correctness. Finally, note that the reduction is trivially computable in log-space and linear time, as each construction is local and the constructed program contains $2 \cdot |\mathrm{var}(F)|$ atoms and $2 \cdot |\mathrm{var}(F)| + |F|$ rules. We conclude #P-hardness.

(*Case 1b*): holds by Case 1a and Obs. 2.

(*Case 1c*): Our program Π uses atoms $\mathrm{var}(F) \cup \{\overline{x} \mid x \in \mathrm{var}(F)\}$ and rules $\Pi = R_1 \cup R_2 \cup R_3$, where R_1 and R_2 are defined as in Case 1a and R_3 is given by $R_3 = \{\leftarrow \sim a^*, \sim b^*, \sim c^* \mid \{a,b,c\} \in F\}$ with $x^* := \overline{y}$ if $x = \neg y$ and $x^* := y$ if $x = y$ for $y \in \mathrm{var}(F)$. Let $M \subseteq \mathrm{var}(F) \cup \{\overline{x} \mid x \in \mathrm{var}(F)\}$. Π^M is either inconsistent or contains exactly rules $R_1 \cup R_2$. In latter, all models for Π^M are also minimal models and answer sets for Π^M by construction. Observe that for each $x \in \mathrm{var}(F)$, rules R_1 enforce that at least one of $x, \overline{x}$ is contained in each model for Π^M and rules R_2 enforces that at most one of $x, \overline{x}$ is contained in each model for Π^M. Thus, each model for Π^M must contain exactly one of $x, \overline{x}$ for each $x \in \mathrm{var}(F)$. For each set $M \subseteq \mathrm{var}(F)$, we construct the set $I = M \cup \{\overline{x} \mid x \in \mathrm{var}(F) \setminus M\}$. By construction, we observe that I is a model for all rules of R_1 and R_2 and Π^M has models exactly if M is a model for F. By our above observation, this is a bijective mapping between models for F and answer sets for Π and we are able to follow correctness. Finally, note that the reduction is computable in log-space and linear time, as each construction is local and the constructed program contains $2 \cdot |\mathrm{var}(F)|$ atoms and $2 \cdot |\mathrm{var}(F)| + |F|$ rules. We conclude #P-hardness. □

Additionally, we make the following observation.

Observation 5. *If Δ is one of:*
$\{[\infty,\infty,0],[0,0,0]\}$,
$\{[i,0,k],[0,j,k] \mid i+k \leq 2, j+k \leq 1\}$,
$\{[i,0,k],[0,j,k] \mid i+j+k \leq 2\}$,
$\{[i,0,k],[0,j,k'] \mid i+k \leq 3, k \leq 2j+k' \leq 1\}$,
$\{[1,\infty,\infty],[0,\infty,\infty]\}$,
$\{[\infty,1,\infty],[0,\infty,\infty]\}$,
$\{[\infty,\infty,0],[0,\infty,0]\}$,
then the #AS$_C$ problem for $C(\Delta)$ is in #P.

Proof. For every variable x, we can branch between an answer set containing x or not. Let p be a disjunctive logic program with variables $\mathrm{var}(F)$. To check if $M \subseteq \mathrm{var}(F)$ is an answer set for p, we add for each variable $x \in \mathrm{var}(F)$ the rule $\{\leftarrow \sim x\}$ to p and for each other variable $x \in \mathrm{var}(F) \setminus M$, we add the rules $\{\leftarrow x\}$ to p, enforcing that it does not appear. This creates the augmented new program p'. If p' now still has an answer set, then M must be an answer set for p. We note that $\Delta_{p'} = \Delta_p \cup \{[0,1,0],[0,0,1]\}$ and membership in the respective classes has not changed by this augmentation unless $\Delta_p \subseteq \{[\infty,\infty,0],[0,\infty,\infty]\}$. For the case that $\Delta_p \preceq \{[\infty,\infty,0],[0,\infty,\infty]\}$, we construct analogously and add for each variable $x \in \mathrm{var}(F)$ the rule $\{x\}$ to p instead and have $\Delta_{p'} = \Delta_p \cup \{[0,1,0],[0,0,1]\}$. In conclusion, the Cons for p' is still in P and NP respectively if p matches the respective classes and we infer that the respective counting problems are in #P. □

Now, we are in position to establish Theorem 3.

Proof of Theorem 3. Let $\Delta_\Pi \preceq \{[1,\infty,0],[0,\infty,\infty]\}$. Then, $\Pi \in$ Horn and has exactly one answer set exactly if it has a model. Thus, *#AS*(Π) can be reduced to the satisfiability of Horn-formula in this case, which is known to solvable in polynomial time. Next, assume that $\{[1,\infty,0],[0,\infty,\infty]\} \preceq \Delta_\Pi$, but not $\Delta_\Pi \preceq \{[1,\infty,0],[0,\infty,\infty]\}$. Then, either $\{[1,0,1]\} \preceq \Delta_\Pi$ or $\{[2,0,0]\} \preceq \Delta_\Pi$. In both cases, *#AS*$(\Pi)$ is #P-hard by Lemma 4.1e. Now, assume that $\{[i,j,k] \mid i+j+k \leq 2\} \preceq \Delta_\Pi$, but not $\Delta_\Pi \preceq \{[i,j,k] \mid i+j+k \leq 2\}$. Then $S \preceq \Delta_\Pi$ for some $S \in \{[3,0,0],[0,3,0],[0,0,3],[1,2,0],[2,1,0],[1,0,2],[2,0,1],[0,1,2],[0,2,1]\}$. These cases are covered by Lemma 4.1a–1g, 4b, resp. If we have $\Delta_\Pi \preceq \{[1,\infty,\infty],[0,\infty,\infty]\}$, $\Delta_\Pi \preceq \{[\infty,1,\infty],[0,\infty,\infty]\}$, or $\Delta_\Pi \preceq \{[\infty,\infty,0],[0,\infty,0]\}$. Then *#AS*$(\Pi) \in$ #P by Observation 5. Following Truszczyński (2011), assume that none of the above inclusions hold. By $\Delta_\Pi \not\preceq \{[1,\infty,\infty],[0,\infty,\infty]\}$, we must have $\{[2,0,0]\} \preceq \Delta_\Pi$. By $\Delta_\Pi \not\preceq \{[\infty,1,\infty],[0,\infty,\infty]\}$, we must have $\{[1,2,0]\} \preceq \Delta_\Pi$. By $\Delta_\Pi \not\preceq \{[\infty,\infty,0],[0,\infty,0]\}$, we must have either $\{[0,0,1]\} \preceq \Delta_\Pi$ or $\{[1,0,1]\} \preceq \Delta_\Pi$. In the former, we obtain $\# \cdot$ coNP hardness by Lemma 4.4b. In the latter, we have $\# \cdot$ coNP hardness by Lemma 4.4a. Lastly, *#AS*$(\Pi) \in \# \cdot$ coNP for arbitrary Π (Fichte et al. 2017). □

4 Reducing #Normal to #Horn-Krom

Our main contribution in this section are novel complexity results for #Choice-Horn-Krom-ASP. In this section, we assume that programs are normal. Recall that the consistency problem for Horn is in P (Truszczyński 2011) and it is a quick observation that it remains in P when including guesses. Moreover, Krom with guesses is NP-hard. Recall, a rule is *Horn* if the number of head atoms plus negative body literals is at most one, and it is *Krom* if it contains at most two literals. It turns out that the ASP counting setting is significantly more difficult than in the propositional setting, as the known reductions from #SAT to #Krom-SAT are *Turing reductions*, i.e., they produce *multiple instances*. We show that two calls are sufficient in the normal ASP setting, and that the corresponding programs can be build natively in ASP without a complex chain of SAT reductions.

Theorem 6. *The subtraction of two #Choice-Horn-Krom-ASP calls is* #P*-hard, even if both programs contain no negation and head atoms (apart from guesses).*

Note that a single call (no postprocessing) is insufficient.

Proposition 7 (Parsimonious Insufficient)**.** *If there was a parsimonious Karp reduction from #ASP to a fragment of #ASP whose consistency can be decided in* P*, then* P = NP.

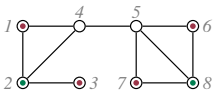
Proof. We could use this reduction to decide the problem of ASP in polynomial time, which is NP-hard. □

However, given the subtraction result of Theorem 6, we can still obtain the result using a single counting call if we allow arithmetic postprocessing.

Corollary 8. *A single #Choice-Horn-Krom-ASP call with arithmetic postprocessing is* #P*-hard, even if the program contain no negation and head atoms (apart from guesses).*

We continue with a running example.

Example 9. *A* dominating set *in a graph G is a set of vertices $D \subseteq V(G)$ such that every vertex $v \in V(G)$ is either in D or adjacent to a vertex D. An* independent set *is a set $I \subseteq V(G)$ such that for all $u, v \in I$ with $u \neq v$ we have $uv \notin E(G)$, i.e., the vertices in I are pairwise not connected. The following graph has a dominating set of size two (in green) and an independent set of size four (in red):*



Finding a small dominating set or a large independent set are both NP*-hard problems, which is commonly implemented by a cardinality constraint requiring large rules (≥ 3) in the decision world. Note that the corresponding counting problems, i.e., counting the number of independent and dominating sets of graph, are expressible in #Krom-SAT and #Horn-SAT.*

Next, we first discuss the reduction to Horn-Krom programs. To this end, let Π be a completed program, as defined in the preliminaries. We require auxiliary atoms in our encoding, where $\overline{f}_{r_i}$ are regular atoms, i.e., these are not negated atoms but symbols. For every rule $r \in \Pi$, we use atom f_r and $\overline{f}_r$, which indicates that we make r false and true, respectively. To indicate the parity of the number of (guaranteed) *false rules up to* $i \in [|\Pi|] = \{1, \ldots, |\Pi|\}$, we use atoms e_i or o_i to express that there is an even or an odd number, respectively. For the i-th rule, atoms o_i^f/e_i^f indicate that the number is odd/even up to i *because of* f_r, respectively. Similarly, $o_i^{\overline{f}}/e_i^{\overline{f}}$ indicates an odd/even number up to i *due to* $\overline{f}_r$, respectively.

Then, with these auxiliary atoms at hand, we refer by $\mathfrak{R}(\Pi)$ to the program below, comprising Rules (1)–(12). The intuition of this reduction is as follows. Rule (1) guesses atom presence in answer sets and Rules (2),(3) execute consequences of a false rule r (if f_r holds). Then, Rules (4)–(7) guess the parity of the number of f_r up to rule r_i with $i \in [|\Pi|]$, as well as the reason of this parity (either because of f_{r_i} or $\overline{f}_{r_i}$). While the additional choice of whether o_i and/or e_i holds seems redundant, it will be required for symmetry later. Rules (8)–(11) execute the decisions of whether we are even or odd at i and whether this holds due f_{r_i} or $\overline{f}_{r_i}$ as guessed by Rules (4)–(7). Finally, Rule (12) ensures there are zero initial false rules.

Guess truth values; ensure: if f_r holds, rule r is false

$$\{a\} \leftarrow \quad \text{for every } a \in \text{at}(\Pi) \tag{1}$$
$$\bot \leftarrow f_r, b \quad \text{for every } r \in \Pi, b \in H_r \cup B_r^- \tag{2}$$
$$b \leftarrow f_r \quad \text{for every } r \in \Pi, b \in B_r^+ \tag{3}$$

Guess parity of # of false rules at index i due to false r_i

$$\{o_i^f\} \leftarrow \quad \{o_i\} \leftarrow \quad \text{Odd due } f_{r_i}; i \in [|\Pi|] \tag{4}$$
$$\{o_i^{\overline{f}}\} \leftarrow \quad \text{Odd due } \overline{f}_{r_i}; i \in [|\Pi|] \tag{5}$$
$$\{e_i^f\} \leftarrow \quad \text{Even due } f_{r_i}; i \in [|\Pi|] \tag{6}$$
$$\{e_i^{\overline{f}}\} \leftarrow \quad \{e_i\} \leftarrow \quad \text{Even due } \overline{f}_{r_i}, i \in [|\Pi|] \tag{7}$$

Propagate consequences of even/odd # of false rules

$$e_{i-1} \leftarrow o_i^f \quad f_{r_i} \leftarrow o_i^f \quad o_i \leftarrow o_i^f \quad \text{Odd due } f_{r_i}; i{\in}[|\Pi|] \tag{8}$$
$$o_{i-1} \leftarrow o_i^{\overline{f}} \quad \overline{f}_{r_i} \leftarrow o_i^{\overline{f}} \quad o_i \leftarrow o_i^{\overline{f}} \quad \text{Odd due } \overline{f}_{r_i}; i{\in}[|\Pi|] \tag{9}$$
$$o_{i-1} \leftarrow e_i^f \quad f_{r_i} \leftarrow e_i^f \quad e_i \leftarrow e_i^f \quad \text{Even due } f_{r_i}; i{\in}[|\Pi|] \tag{10}$$
$$e_{i-1} \leftarrow e_i^{\overline{f}} \quad \overline{f}_{r_i} \leftarrow e_i^{\overline{f}} \quad e_i \leftarrow e_i^{\overline{f}} \quad \text{Even due } \overline{f}_{r_i}; i \in [|\Pi|] \tag{11}$$
$$\bot \leftarrow o_1^{\overline{f}} \quad \bot \leftarrow e_1^f \quad \text{Initially: zero (even) rules} \tag{12}$$

Observe that this program has many trivial answer sets by construction, as deciding consistency is trivial. In particular, any assignment to at(Π) can be extended to an answer set of $\mathfrak{R}(\Pi)$. Indeed, we can easily derive all e_i, $\overline{f}_{r_i}$, and $e_i^{\overline{f}}$.

Example 10. *Suppose we want to count dominating sets of a graph $G = (V, E)$. Then we can construct a program Π from the ideas in Example 9. We arbitrarily order all rules in Π from 1 to $|\Pi|$ and construct Rules (1)–(12) for every $i \in [|\Pi|]$, resulting in Horn-Krom program Π'. Every subset of $\{x_v \mid v \in V\}$* can be extended to an answer set *of Π' and we do not need rules of size* 3 *any more due to the computational power of counting.*

The previous example demonstrates that there are indeed many answer sets we do not want to have. Below, we will see a technique that is fault-tolerant regarding these sets.

4.1 Counting by Exploiting Symmetry

Given the reduction $\mathfrak{R}(\Pi)$ above, surprisingly, we can show that this is almost enough for counting the number of answer sets of Π. Indeed, we require arithmetic (subtraction) on top of this approach to precisely obtain the number of answer sets. More concretely, we can count as follows:

$$(\#[\Pi_1] - \#[\Pi_2]) \quad \text{where } \Pi_1 = \mathfrak{R}(\Pi) \cup \{\bot \leftarrow o_{|\Pi|}\}, \; \Pi_2 = \mathfrak{R}(\Pi) \cup \{\bot \leftarrow e_{|\Pi|}\} \tag{13}$$

Equation (13) is enough to capture the number of answer sets as we will present during the course of this work, but the actual proof is involved. It mostly relies on exploiting symmetry between both counting calls, as well as on a combinatorial extension of the inclusion-exclusion principle.

Example 11. *Subtraction indeed fixes the problem of Example 10. By subtracting the answer sets with an odd number of (guaranteed) false rules from the answer sets modulo an even number of false rules, we obtain the answer sets of Π:*

$$\#(|\Pi|) = \#(|\Pi_1|) - \#(|\Pi_2|) = 309213 - 309070 = 143,$$

which we can compute within seconds using $\mathfrak{R}$ *and clingo (Gebser et al. 2012). Intuitively, we can show a bijection between unwanted answer sets of both programs.*

Correctness: A Novel Combinatorial Approach. For establishing correctness we require to extend the well-known principle of inclusion-exclusion. Indeed, in our case of Equation (13) we also need to consider the symmetry between two counting operations relying on inclusion-exclusion. To this end, we aim to prove that there will be a lot of solutions (answer set) that both answer set counting operations share. In more details, the goal is that unwanted answer sets are subtracted out. Note that this is different to subtractive reductions (Durand, Hermann, and Kolaitis 2005), because these are not the same answer sets and therefore we do not perform set difference. In fact, recall Proposition 7, where we demonstrate that a single call is insufficient, as otherwise this would prove P = NP.

However, we will define the concept of *rogue answer sets* and show how both (highly symmetric) counting operations share precisely the same number of rogue answer sets.

Definition 12 (Rogue Answer Set)**.** *Let* Π *be a program constructed using* $\mathfrak{R}$ *and* $1 \leq i \leq |\Pi|$*. Then, we call an answer set* M rogue *at* i *if at least one of the following holds.*

1. $|M \cap \{o_i^f, o_i^{\overline{f}}, e_i^f, e_i^{\overline{f}}\}| \neq 1$, 2. $|M \cap \{o_i, e_i\}| \neq 1$, *or*
3. $|M \cap \{f_{r_i}, \overline{f}_{r_i}\}| \neq 1$

We say an answer set M *is rogue, if it is rogue at some* i*.*

We observe that if there were no rogue answer sets, our reduction of Equation (13) was correct.

Proposition 13. *Let* Π_1, Π_2 *be the programs constructed in Equation (13) such that both programs additionally exclude rogue answer sets. Then, the number of answer sets of* Π_1 *minus those of* Π_2 *yields the number of answer sets of* Π*.*

Proof. Indeed, Π_1 counts even numbers of guaranteed false rules of Π (including answer sets of Π, i.e., 0 false rules), whereas Π_2 counts odd guaranteed false rules of Π. We are summing up answer sets of Π, subtract where we allow at least 1 dissatisfied rule, add those, where we allow at least 2 dissatisfied rules, subtract those with at least 3 dissatisfied rules, and so on. So, the even (positive) terms of this sum are due to Π_1 and the odd (negative) terms are due Π_2. □

In fact, we can easily characterize non-rogue answer sets.

Lemma 14 (Non-Rogue AS)**.** *Let* Π *be a completed program and let* Π_1*,* Π_2 *be defined as in Equation (13). (1) For every answer set* M *of* Π_1 *or* Π_2 *that is not rogue,* $M \cap \text{at}(\Pi)$ *is an interpretation of* Π *that invalidates at least* $n = |\{r \in \Pi \mid f_r \in M\}|$ *rules. (2)* n *is odd iff* M *is an answer set of* Π_2*.*

Proof. By construction, we know that M invalidates (at least) n rules. Since M is not rogue, n is odd (even) iff M is an answer set of Π_2 (Π_1), respectively. □

It remains to establish a bijection between rogue answer sets of Π_1 and those of Π_2. The bijection itself is somewhat intricate, and our construction relies heavily on the following definition. The central idea behind this definition is to ensure that only necessary modifications are made in establishing the bijection. Moreover, these modifications must be canonical and structured such that reapplying the same rules restores the original answer set.

Definition 15 (Symmetric Rogue Answer Set)**.** *Let* Π *be a completed program,* $\Pi' \supseteq \mathfrak{R}(\Pi)$*,* M *be an answer set of* Π' *that is rogue at* i*, and* M *is not rogue at* j *for some* $j>i$*. The* symmetric rogue answer set M' *(of* M*) is defined as*

1. *For every* $j \geq i$*, we replace* $o_i \in M$ *by* $e_i \in M'$ *(and vice versa,* $e_i \in M$ *by* $o_i \in M'$*).*
2. *For every* $j > i$*, we replace*
   - $o_j^f \in M$ *by* $e_j^f \in M'$, $\quad o_j^{\overline{f}} \in M$ *by* $e_j^{\overline{f}} \in M'$,
   - $e_j^f \in M$ *by* $o_j^f \in M'$*, and* $\quad e_j^{\overline{f}} \in M$ *by* $o_j^{\overline{f}} \in M'$

   (and vice versa)
3. *If* $|M \cap \{o_i, e_i\}|=1$ *and* $|M \cap \{o_i^f, o_i^{\overline{f}}, e_i^f, e_i^{\overline{f}}\}| \geq 1$*, replace*
   - $o_i^f \in M$ *by* $e_i^{\overline{f}} \in M'$, $\quad o_i^{\overline{f}} \in M$ *by* $e_i^f \in M'$,
   - $e_i^f \in M$ *by* $o_i^{\overline{f}} \in M'$*, and* $\quad e_i^{\overline{f}} \in M$ *by* $o_i^f \in M'$

   (and vice versa)

Intuitively, this definition ensures that for every rogue answer set in Π_1 there is exactly one unique rogue answer set in Π_2 (and vice versa). Due to the construction of the symmetric rogue answer set, we always obtain a corresponding rogue answer set with analogous properties, such that applying the construction a second time recovers the original rogue answer set. This works by swapping even and odd, thereby preserving rogueness at some i, see also Figure 1. However, there are some subtleties attached. Indeed, property 1 of Definition 15 ensures that we transition a rogue answer set in Π_1 from odd to even (or even to add) in Π_2, and vice versa. Then, the second property propagates this information from i towards $|\Pi|$. Finally, property 3 fixes non-trivial cases, where M only contains either o_i (odd) or e_i (even). There, we also have to swap the reason of why we end up even or odd in a case $c = |M \cap \{o_i^f, o_i^f, e_i^f, e_i^f\}|$ with $c > 0$. This is safe, as then for M to be rogue it contains either f_{r_i} and $\overline{f}_{r_i}$ (which implies $c \geq 2$) or $c = 0$.

Example 16. *Suppose there is an answer set* M *of* Π_1 *that is rogue only at* $|\Pi|-1$ *with* $e_{|\Pi|} \in M$*. Due to selecting a single wrong reason at* $|\Pi|$*, say* $e_{|\Pi|}^f \in M$*, we have both* $o_{|\Pi|-1} \in M$ *and* $e_{|\Pi|-1} \in M$*. Then,* $M' = (M \setminus \{e_{|\Pi|}^f, e_{|\Pi|}\}) \cup \{o_{|\Pi|}^f, o_{|\Pi|}\}$ *is* the symmetric rogue answer set *of* Π_2*. This might seem surprising, as we do not change parities for* $|\Pi|-1$ *and, thus,* $\{e_{|\Pi|-1}, o_{|\Pi|-1}\} \subseteq M'$*. However, constructing* M' *were impossible without guessing* $o_{|\Pi|-1}$ *in Rule (4) or guessing* $e_{|\Pi|-1}$ *in Rule (7), to go back from* M' *to* M*.*

With this definition, we are ready to show the combinatorial result by means of the following lemma. Crucially, we establish the desired bijection as follows.

Lemma 17 (Well-Defined Symmetric Rogue Answer Set)**.** *Let* Π *be a completed program and let* Π_1*,* Π_2 *be as in Equation (13). The symmetric rogue answer set* M' *of any rogue answer set* M *of* Π_1 *is an answer set of* Π_2 *(and vice versa).*

Finally, this lemma in combination with Proposition 13 and Lemma 14 indeed suffices for establishing correctness.

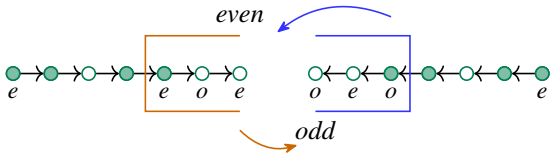


Figure 1: Abstract visualization of the symmetric rogue answer set and its construction. Rules are ordered, which is represented as a directed path. Nodes filled in green indicate that the answer set is rogue at this rule. The intuition is that it suffices to change the parity of the rule (that is the source of rogueness) closest to the root, which then also causes parity changes of remaining nodes upwards. These changes enable the transition from odd to even parity (and vice versa) and they are visualized in blue (and orange). Intuitively, every rogue node enables a free change of parity, which is then propagated towards the last rule of the program.

Theorem 18 (Correctness of $\mathfrak{R}$)**.** *Equation (13) is correct for any completed program* Π*, i.e.,* $\#(\Pi) = \#(\Pi_1) - \#(\Pi_2)$

Proof. The construction simulates the principle of inclusion-exclusion. We count the answer sets of Π as:

$$2^n - \sum_{\substack{M\subseteq 2^{\mathrm{at}(\Pi)},\exists r\in\Pi,\\ M\not\models\{r\}}} 1 \;+\; \sum_{\substack{M\subseteq 2^{\mathrm{at}(\Pi)},\exists r,r'\in\Pi,\\ r\neq r',M\not\models\{r\}\text{or}M\not\models\{r'\}}} 1 \;+\cdots+\; \sum_{\substack{M\subseteq 2^{\mathrm{at}(\Pi)},\\ \nexists r\in\Pi,M\models\{r\}}} (-1)^{|\Pi|},$$

which simplifies to:

$$\sum_{\substack{M\subseteq 2^{\mathrm{at}(\Pi)},0\leq n\leq|\Pi|,\\ M\text{ does not satisfy}\geq n\text{ rules in }\Pi}} (-1)^n.$$

Therefore, we can split this term into two parts, where we compute interpretations that do not satisfy at least n clauses with n being even (Π_1) and then subtract those interpretations with n being odd (Π_2). As a result, the goal is to encode the result as an exclusive-or, i.e., whether an assignment does not satisfy at least n clauses with n being even (odd). We refer to those interpretations where n is even by

$$E = \{M \subseteq 2^{\mathrm{at}(\Pi)} \mid 0 \leq n \leq |\Pi|, M \text{ does not satisfy} \geq n \text{ rules in } \Pi, n \equiv 0 (\mathrm{mod} 2)\}.$$

The interpretations with n being odd are referred to by

$$O = \{M \subseteq 2^{\mathrm{at}(\Pi)} \mid 0 \leq n \leq |\Pi|, M \text{ does not satisfy} \geq n \text{ rules in } \Pi, n \equiv 1 (\mathrm{mod} 2)\}.$$

By Lemma 14, every answer set in E can be extended to an answer set of Π_1, i.e., $|E| \leq \#(\Pi_1)$. Analogously, $|O| \leq \#(\Pi_2)$. By Lemma 17, there is a bijection between the rogue answer sets R of Π_1 and those of Π_2. Consequently,

$$\#(\Pi) = (|E| + |R|) - (|O| + |R|) = \#(\Pi_1) - \#(\Pi_2). \qquad \square$$

4.2 Reduction to #Horn-Krom-ASP

We can further restrict the reduction to positive rules with empty heads (apart from guesses). However, this requires us to duplicate atoms, so for every atom a in the program we will use atoms a and $\overline{a}$. While this might seem to be a minor detail, the problem is that there is no easy way of extending rogue answer sets to include such cases. However, there is a nice solution: From now on, we will not only consider false rules, but also false (unassigned) atom assignments, which are those where we neither assign an atom to be in the answer set (i.e., $a \notin M$) nor to be out (i.e., $\overline{a} \notin M$). This requires us to use auxiliary atoms f_i and $\overline{f}_i$ for every $i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$. In our construction, for the ease of notation we will first consider all false rules $r_i \in \Pi$ with $1 \leq i \leq |\Pi|$ and then all false (unassigned) atoms $a_{j-|\Pi|} \in \mathrm{at}(\Pi)$ with $|\Pi|+1 \leq j \leq |\Pi| + |\,\mathrm{at}(\Pi)|$.

We refer by $\mathfrak{R}'$ to the reduction obtained from Rules (14)–(27). Intuitively, Rules (1)–(2) are adapted variants of Rules (14)–(16) and Rule (17) is the additional case of a false atom assignment. Note that we also require that an answer set does not contain both $\{a, \overline{a}\}$ for an atom $a \in \mathrm{at}(\Pi)$, which we cannot easily add to rogue answer sets without usage of negation, but can excluded directly (18). Then, Rules (4)–(7) are as (19)–(22), but also add guesses for the parity state and regarding f_i or $\overline{f}_i$. Finally, Rules (23)–(26) are constraint variants of Rules (8)–(11); Rule (27) ensures zero initial errors.

Guess truth values; ensure: if f_i holds, rule/atom error

$\{a\} \leftarrow \quad \{\overline{a}\} \leftarrow$ for every $a \in \mathrm{at}(\Pi)$ (14)

$\bot \leftarrow f_i, b$ for every $r_i \in \Pi, b \in H_r \cup B_r^-$ (15)

$\bot \leftarrow f_i, \overline{b}$ for every $r_i \in \Pi, b \in B_r^+$ (16)

$\bot \leftarrow f_{i+|\Pi|}, a_i \quad \bot \leftarrow f_{i+|\Pi|}, \overline{a_i}$ for every $a_i \in \mathrm{at}(\Pi)$ (17)

$\bot \leftarrow a_i, \overline{a_i}$ for every $a_i \in \mathrm{at}(\Pi)$ (18)

Guess parity of # of false rules at index i

$\{o_i^f\} \leftarrow \quad \{o_i\} \leftarrow$ Odd due $f_i; i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (19)

$\{o_i^{\overline{f}}\} \leftarrow \quad \{\overline{f}_i\} \leftarrow$ Odd due $\overline{f}_i; i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (20)

$\{e_i^f\} \leftarrow \quad \{f_i\} \leftarrow$ Even due $f_i; i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (21)

$\{e_i^{\overline{f}}\} \leftarrow \quad \{e_i\} \leftarrow$ Even due $\overline{f}_i, i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (22)

Check consequences of even/odd # of errors

$\bot \leftarrow o_i^f, o_{i-1} \quad \bot \leftarrow o_i^f, \overline{f}_i \quad \bot \leftarrow o_i^f, e_i$ Odd due $f_i; i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (23)

$\bot \leftarrow o_i^{\overline{f}}, e_{i-1} \quad \bot \leftarrow o_i^{\overline{f}}, f_i \quad \bot \leftarrow o_i^{\overline{f}}, e_i$ Odd due $\overline{f}_i; i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (24)

$\bot \leftarrow e_i^f, e_{i-1} \quad \bot \leftarrow e_i^f, \overline{f}_i \quad \bot \leftarrow e_i^f, o_i$ Even due $f_i; i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (25)

$\bot \leftarrow e_i^{\overline{f}}, o_{i-1} \quad \bot \leftarrow e_i^{\overline{f}}, f_i \quad \bot \leftarrow e_i^{\overline{f}}, o_i$ Even due $\overline{f}_i; i \in [|\Pi| + |\,\mathrm{at}(\Pi)|]$ (26)

$\bot \leftarrow o_1^{\overline{f}} \quad \bot \leftarrow e_1^f$ Initially: zero (even) errors (27)

Finally, this allows us to obtain the resulting count of Π via an adaptation of Equation (13) as follows:

$$(\#[\Pi_1] - \#[\Pi_2]) \quad \text{where } \Pi_1 = \mathfrak{R}'(\Pi) \cup \{\bot \leftarrow o_{|\Pi|+|\,\mathrm{at}(\Pi)|}\}, \; \Pi_2 = \mathfrak{R}'(\Pi) \cup \{\bot \leftarrow e_{|\Pi|+|\,\mathrm{at}(\Pi)|}\} \quad (28)$$

Example 19. *Recall Example 11 and let us apply* $\mathfrak{R}'$*. If we allowed negation in Rule (16) and do not duplicate atoms as in Rule (14), we obtain* $143 = 46771348 - 46771491$*. If we execute* $\mathfrak{R}'$ *as is, we have huge counts and use translations (Janhunen 2006; Bomanson and Janhunen 2013) to #SAT, and an efficient model counter, e.g., d4 (Lagniez and Marquis 2017). Within a second we obtain the result as:* $143 = 60334257524677568427 2 - 60334257524677568412 9$.

It remains to adapt the construction of rogue answer sets, which is more involved. Intuitively, if we assign an atom

$a \in \text{at}(\Pi)$ to true and to false, we can freely switch parity. We must add this case (4. below) to rogue answer sets.

Definition 20 (Rogue Answer Set with Empty Heads)**.** *Let* Π *be a program constructed using* $\mathfrak{R}'$ *and let further i be an index* $1 \leq i \leq |\Pi|+|\text{at}(\Pi)|$*. Then, we call an answer set M* rogue *at i if at least one of the following holds.*

1. $|M \cap \{o_i^f, o_i^{\overline{f}}, e_i^f, e_i^{\overline{f}}\}| \neq 1$,
2. $|M \cap \{o_i, e_i\}| \neq 1$,
3. $|M \cap \{f_i, \overline{f}_i\}| \neq 1$, *or*
4. $a_{i-|\Pi|} \in \text{at}(\Pi)$ *and* $M \cap \{a_{i-|\Pi|}, \overline{a_{i-|\Pi|}}\} = \emptyset$,

With this definition, we can characterize non-rogue answer sets, similarly to Lemma 14 above.

In fact, we can easily characterize non-rogue answer sets.

Lemma 21 (Non-Rogue AS (Empty Heads))**.** *Let* Π *be a completed program and let* Π_1, Π_2 *be defined as in Equation (13). (1) For every answer set M of* Π_1 *or* Π_2 *that is not rogue,* $M \cap \text{at}(\Pi)$ *is an interpretation of* Π *that invalidates at least* $n = |\{r_i \in \Pi \mid f_i \in M\}| + |\{a \in \text{at}(\Pi) \mid f_{|\Pi|+i} \in M\}|$ *rules and atoms. (2) n is odd iff M is an answer set of* Π_2*.*

Proof. By construction, M invalidates $\geq n$ rules and atoms (left unassigned). Since M is not rogue, n is odd (even) iff M is an answer set of Π_2 (Π_1), respectively. □

This also implies that the concept of symmetric rogue answer set needs to be extended accordingly. Indeed, there is more freedom in deriving atoms, so this is also more effort.

Definition 22 (Symmetric Rogue AS (Empty Heads))**.** *Let* Π *be completed,* $\Pi' \supseteq \mathfrak{R}'(\Pi)$ *and* M' *be an answer set of* Π' *that is rogue at i such that* M' *is not rogue at some j>i. We define the* symmetric rogue answer set M' *(of M) by*

1. *For every* $j \geq i$*, we replace* $o_i \in M$ *by* $e_i \in M'$ *(and vice versa,* $e_i \in M$ *by* $o_i \in M'$*).*
2. *For every* $j > i$*, we replace*
   - $o_j^f \in M$ *by* $e_j^f \in M'$, $\quad o_j^{\overline{f}} \in M$ *by* $e_j^{\overline{f}} \in M'$,
   - $e_j^f \in M$ *by* $o_j^f \in M'$*, and* $\quad e_j^{\overline{f}} \in M$ *by* $o_j^{\overline{f}} \in M'$

   (and vice versa)
3. *If* $|M \cap \{o_i, e_i\}|=1$ *and* $|M \cap \{o_i^f, o_i^{\overline{f}}, e_i^f, e_i^{\overline{f}}\}| \geq 1$*, replace*
   - $o_i^f \in M$ *by* $e_i^{\overline{f}} \in M'$, $\quad o_i^{\overline{f}} \in M$ *by* $e_i^f \in M'$,
   - $e_i^f \in M$ *by* $o_i^{\overline{f}} \in M'$*, and* $\quad e_i^{\overline{f}} \in M$ *by* $o_i^f \in M'$

   (and vice versa)
4. *If preconditions of (3.) and* $M \cap \{a_{i-|\Pi|}, \overline{a_{i-|\Pi|}}\} = \emptyset$*, we replace* $f_i \in M$ *by* $\overline{f}_i \in M'$ *(and vice versa)*

Lemma 23 (Well-Defined Symmetric Rogue Answer Sets)**.** *Let* Π *be a completed program and* $\Pi_1 = \mathfrak{R}'(\Pi) \cup \{\leftarrow o_{|\Pi|+|\text{at}(\Pi)|}\}$, $\Pi_2 = \mathfrak{R}'(\Pi) \cup \{\leftarrow e_{|\Pi|+|\text{at}(\Pi)|}\}$*. Then, the symmetric rogue answer set* M' *of any rogue answer set M of* Π_1 *is a rogue answer set of* Π_2 *(and vice versa).*

For $\mathfrak{R}'$ we then obtain the following correctness result.

Theorem 24 (Correctness of $\mathfrak{R}'$)**.** *Rule (28) is correct, i.e.,* $\#(\Pi) = (\#(\Pi_1) - \#(\Pi_2))$.

Proof. The proof works similarly as in Theorem 18, thereby mainly resting on Lemmas 21 and 23, as well as a slight adaptation of Proposition 13. □

This allows us to finally establish Theorem 6, as presented in the introduction. However, we can even go beyond that and push everything into a single call (see supplemental material).

The construction is quadratic, which we can avoid via an additional guess, then extracting c_1, c_2 from $c_1 + c_1 \cdot c_2 \cdot 2^{|\text{at}(\Pi_1)+1|}$. We can extend the construct to Krom rules, where we prohibit rules with a single atom (apart from guesses).

Extension to binary rules. We can adapt $\mathfrak{R}'$ such that only guesses and binary constraints are used. This works by modifying Rule (27) as well as Rule (28). This works by adding an auxiliary atom x and a choice rule $\{x\} \leftarrow$ and adding x to the positive bodies of the rules in Rule (27) and Rule (28). Definition 20 can be extended to cover the rogue case where *x is not in* an answer set, which is its symmetric rogue answer set. For this we need to be able to freely derive x, hence we add the guess.

Example 25. *Recall Example 11. The adapted reduction yields* 143 *as the difference of* 278961612294505869533 1 *and* 27896161229450586951 88 *answer sets.*

Using Definition 22, we can adapt reduction $\mathfrak{R}$ to only use guesses and binary rules without integrity rules. The details of this constructions are presented in the appendix.

5 Conclusion and Outlook

We study the counting complexity of answer set programming in the arity framework of Truszczyński (2011) and beyond. There we obtain a comprehensive picture of hardness and membership results and reductions.

Moreover, we study the power of answer set programs with guesses in which all rules are both Horn *and* Krom. This is a strong syntactic restriction from which we may expect that its expressiveness is limited. However, we show that under counting, this fragment is powerful enough to capture the expressiveness of #ASP. From a knowledge representation perspective, a key feature of these translations is that they can directly be expressed in ASP, without a chain of reductions through other problems. From a complexity theoretic point of view it is worth pointing out that all our reductions can be implemented in *linear time* or alternatively in *log-space*.

While our work is mainly of theoretical nature, we expect a positive impact of our translations on practical tools, possible (counting) proof systems for ASP (Alviano et al. 2019; Fichte, Hecher, and Roland 2022), and applications that require multiple counting (Fichte, Hecher, and Nadeem 2022). For #SAT, it was was already empirically observed that #Horn-SAT solvers (Dubray, Schaus, and Nijssen 2023; Dubray, Schaus, and Nijssen 2024) solvers can outperform traditional tools on various benchmarks. We expect similar effects for through-thought implementations of the reductions presented within this article. Already a proof-of-concept for #Horn-Krom-ASP, which we implemented to generate examples, shows promising speedups.

AI Declaration

During the preparation of this work, the authors used generative AI tools for grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

Acknowledgements

The authors are listed in alphabetical order. Research supported by the Austrian Science Fund (FWF) grant J4656; the French Agence nationale de la recherche (ANR) grant ANR-25-CE23-7647; and ELLIIT funded by the Swedish government and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Part of the research was carried out while Hecher was a postdoc at MIT and while he was at UC Berkeley while visiting the Simons institute for the theory of computing.

References

Altay-Kern, O.; Dietz, E.; Kuhlmann, I.; and Thimm, M. 2025. Exploring Desirable Configurations in Global Logistics with Heuristic Search in Answer Set Programming. In *Proc. of KR*, 578–587.

Alviano, M., and Reiners, L. A. R. 2025. ASP chef for water waste monitoring. In *Proc. of the 40th Italian Conference on Computational Logic*, CEUR Workshop Proc. CEUR-WS.org.

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

Azzolini, D., and Riguzzi, F. 2025. Probabilistic answer set programming with discrete and continuous random variables. *Theory Pract. Log. Program.* (1):1–32.

Bannach, M.; Demaine, E. D.; Gomez, T.; and Hecher, M. 2025. #P is sandwiched by one and two #2DNF calls: Is subtraction stronger than we thought? In *Proc. of LICS*, 31–43.

Bannach, M.; Demaine, E. D.; Gomez, T.; and Hecher, M. 2026. A novel reduction from #SAT to #2SAT based on symmetry: Simply drop the large clauses. In *Proc. of SOSA*, 254–265.

Baumeister, J.; Herud, K.; Ostrowski, M.; Reutelshöfer, J.; Rühling, N.; Schaub, T.; and Wanko, P. 2024. Towards industrial-scale product configuration. In *Proc. of LPNMR*, 71–84.

Ben-Eliyahu, R., and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *Ann. Math. Artif. Intell.* (1):53–87.

Bogaerts, B., and Van den Broeck, G. 2015. Knowledge compilation of logic programs using approximation fixpoint theory. *Theory Pract. Log. Program.* (4–5):464–480.

Bomanson, J., and Janhunen, T. 2013. Normalizing Cardinality Rules Using Merging and Sorting Constructions. In *Proc. of LPNMR*, 187–199.

Bomanson, J.; Gebser, M.; Janhunen, T.; Kaufmann, B.; and Schaub, T. 2016. Answer Set Programming Modulo Acyclicity. *Fundamenta Informaticae* (1):63–91.

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

Cadoli, M., and Lenzerini, M. 1994. The complexity of propositional closed world reasoning and circumscription. *Journal of Computer and System Sciences* (2):255–310.

Chen, H. 2006. A rendezvous of logic, complexity, and algebra. *SIGACT News* (4):85–114.

Clark, K. L. 1977. Negation as failure. In *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977*, 293–322.

Coste-Marquis, S., and Marquis, P. 1999. Complexity results for propositional closed world reasoning and circumscription from tractable knowledge bases. In *Proc. of IJCAI*, 24–29.

Darwiche, A. 2020. Three modern roles for logic in AI. In *Proc. of PODS*, 229–243.

Dowling, W. F., and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.* (3):267–284.

Dubray, A.; Schaus, P.; and Nijssen, S. 2023. Probabilistic inference by projected weighted model counting on horn clauses. In *Proc. of CP*, 15:1–15:17.

Dubray, A.; Schaus, P.; and Nijssen, S. 2024. Anytime weighted model counting with approximation guarantees for probabilistic inference. In *Proc. of CP*, 10:1–10:16.

Durand, A., and Hermann, M. 2008. On the counting complexity of propositional circumscription. *Information Processing Letters* (4):164–170.

Durand, A.; Hermann, M.; and Kolaitis, P. G. 2005. Subtractive reductions and complete problems for counting complexity classes. *Theor. Comput. Sci.* (3):496–513.

Durand, A.; Hermann, M.; and Nordh, G. 2012. Trichotomies in the complexity of minimal inference. *Theory of Computing Systems* (3):446–491.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.* (3–4):289–323.

Eiter, T.; Hecher, M.; and Kiesel, R. 2024. aspmc: New frontiers of algebraic answer set counting. *Artificial Intelligence* 104–109.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Methods Log. Comput. Sci.* 1(1):51–60.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2017. Answer Set Solving with Bounded Treewidth Revisited. In *Proc. of LPNMR*, 132–145.

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

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

Fichte, J. K.; Hecher, M.; and Roland, V. 2022. Proofs for propositional model counting. In *Proc. of SAT*, volume 236, 30:1–30:24.

Fierens, D.; Van den Broeck, G.; Renkens, J.; Shterionov, D. S.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2015. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory Pract. Log. Program.* (3):358–401.

Garey, M. R.; Johnson, D. S.; and Stockmeyer, L. J. 1976. Some Simplified NP-Complete Graph Problems. *Theor. Comput. Sci.* (3):237–267.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Morgan & Claypool.

Gebser, M.; Janhunen, T.; and Rintanen, J. 2014. Answer Set Programming as SAT modulo Acyclicity. In *Proc. of ECAI*, 351–356. IOS Press.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2009. Solution enumeration for projected boolean search problems. In *Proc. of CPIOR*, 71–86.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of LP*, 1070–1080. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Comput.* (3/4):365–386.

Hahn, S.; Schaub, T.; Martens, C.; Nemes, A.; Otunuya, H.; Romero, J.; and Schellhorn, S. 2024. Reasoning about study regulations in answer set programming. *Theory Pract. Log. Program.* (4):790–804.

Hemaspaandra, L. A., and Vollmer, H. 1995. The satanic notations: Counting classes beyond #P and other definitional adventures. *SIGACT News* 26(1):2–13.

Jakl, M.; Pichler, R.; Rümmele, S.; and Woltran, S. 2008. Fast counting with bounded treewidth. In *Proc. of LPAR*, 436–450. Springer.

Jakl, M.; Pichler, R.; and Woltran, S. 2009. Answer-Set Programming with Bounded Treewidth. In *Proc. of IJCAI*, 816–822.

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

Janhunen, T. 2006. Some (in)translatability results for normal logic programs and propositional theories. *J. of Applied Non-Classical Logics* (1-2):35–86.

Kabir, M., and Meel, K. S. 2023. A fast and accurate asp counting based network reliability estimator. In *Proc. of LPAR*, 270–287. EasyChair.

Kabir, M.; Everardo, F. O.; Shukla, A. K.; Hecher, M.; Fichte, J. K.; and Meel, K. S. 2022. ApproxASP – a scalable approximate answer set counter. *Proc. of AAAI* (5):5755–5764.

Kabir, M.; Chakraborty, S.; and Meel, K. S. 2024. Exact ASP counting with compact encodings. In *Proc. of AAAI*. AAAI Press.

Kabir, M. M.; Chakraborty, S.; and Meel, K. S. 2025. Counting answer sets of disjunctive answer set programs. *Theory Pract. Log. Program.* (4):703–721.

Lagniez, J.-M., and Marquis, P. 2017. An improved decision-DNNF compiler. In *Proc. of IJCAI*, 667–673. ijcai.org.

Lau, D. 2006. *Function algebras on finite sets: a basic course on many-valued logic and clone theory*. Springer.

Lin, F., and Zhao, J. 2003. On Tight Logic Programs and Yet Another Translation from Normal Logic Programs to Propositional Logic. In *Proc. of IJCAI*, 853–858. Morgan Kaufmann.

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

Nabeshima, H.; Banbara, M.; Schaub, T.; and Soh, T. 2026. The ASP-based nurse scheduling system at the university of yamanashi hospital. *Electronic Proc. in Theoretical Computer Science* 334–348.

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

Post, E. L. 2016. The two-valued iterative systems of mathematical logic. *Annals of Mathematics Studies*.

Schaefer, T. J. 1978. The complexity of satisfiability problems. In *Proc. of STOC*, 216–226. Association for Computing Machinery.

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

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

Toda, S., and Watanabe, O. 1992. Polynomial-time 1-turing reductions from #PH to #P. *Theoretical Computer Science* 100(1):205–221.

Trevisan, L.; Sorkin, G. B.; Sudan, M.; and Williamson, D. P. 2000. Gadgets, Approximation, and Linear Programming. *SIAM J. Comput.* (6):2074–2097.

Truszczyński, M. 2011. Trichotomy and dichotomy results on the complexity of reasoning with disjunctive logic programs. *Theory Pract. Log. Program.* (6):881–904.

Valiant, L. G. 1979a. The complexity of computing the permanent. *Theoretical Computer Science* 189–201.

Valiant, L. G. 1979b. The complexity of enumeration and reliability problems. *SIAM J. Comput.* (3):410–421.

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

Zak, D.; Mei, J.; Lagniez, J.; and Laarman, A. 2025. Reducing quantum circuit synthesis to #SAT. In *Proc. of CP*, 38:1–38:21.

Zhang, H.; Kung, P.-N.; Yoshida, M.; Van den Broeck, G.; and Peng, N. 2024. Adaptable logical control for large language models. In *Proc. of NeurIPS*, 115563–115587. Curran Associates, Inc.