Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Backdoors to Normality for Disjunctive Logic Programs*

Johannes Klaus Fichte and **Stefan Szeider**
Vienna University of Technology, Austria
fichte@kr.tuwien.ac.at, stefan@szeider.net

Abstract

Over the last two decades, propositional satisfiability (SAT) has become one of the most successful and widely applied techniques for the solution of NP-complete problems. The aim of this paper is to investigate theoretically how SAT can be utilized for the efficient solution of problems that are harder than NP or co-NP. In particular, we consider the fundamental reasoning problems in propositional disjunctive answer set programming (ASP), BRAVE REASONING and SKEPTICAL REASONING, which ask whether a given atom is contained in at least one or in all answer sets, respectively. Both problems are located at the second level of the Polynomial Hierarchy and thus assumed to be harder than NP or co-NP. One cannot transform these two reasoning problems into SAT in polynomial time, unless the Polynomial Hierarchy collapses.

We show that certain structural aspects of disjunctive logic programs can be utilized to break through this complexity barrier, using new techniques from Parameterized Complexity. In particular, we exhibit transformations from BRAVE and SKEPTICAL REASONING to SAT that run in time $O(2^k n^2)$ where k is a structural parameter of the instance and n the input size. In other words, the reduction is fixed-parameter tractable for parameter k. As the parameter k we take the size of a smallest backdoor with respect to the class of normal (i.e., disjunction-free) programs. Such a backdoor is a set of atoms that when deleted makes the program normal. In consequence, the combinatorial explosion, which is expected when transforming a problem from the second level of the Polynomial Hierarchy to the first level, can now be confined to the parameter k, while the running time of the reduction is polynomial in the input size n, where the order of the polynomial is independent of k. We show that such a transformation is not possible if we consider backdoors with respect to tightness instead of normality.

We think that our approach is applicable to many other hard combinatorial problems that lie beyond NP or co-NP, and thus significantly enlarge the applicability of SAT.

Introduction

Over the last two decades, propositional satisfiability (SAT) has become one of the most successful and widely applied techniques for the solution of NP-complete problems. Today's SAT-solvers are extremely efficient and robust instances with hundreds of thousands of variables and clauses can be solved routinely. In fact, due to the success of SAT, NP-complete problems have lost their scariness, as in many cases one can efficiently encode NP-complete problems to SAT and solve them by means of a SAT-solver (Gomes et al. 2008; Biere et al. 2009).

We investigate transformations into SAT for problems that are harder than NP or co-NP. In particular, we consider various search problems that arise in *disjunctive answer set programming (*ASP*)*. With ASP one can describe a problem by means of rules that form a disjunctive logic program, whose solutions are answer sets. Many important problems of AI and reasoning can be represented in terms of the search for answer sets (Brewka, Eiter, and Truszczyński 2011; Marek and Truszczynski 1999; Niemelä 1999). Two of the most fundamental ASP problems are BRAVE REASONING (is a certain atom contained in at least one answer set?) and SKEPTICAL REASONING (is a certain atom contained in all answer sets?). Both problems are located at the second level of the Polynomial Hierarchy (Eiter and Gottlob 1995) and thus assumed to be harder than NP or co-NP. It would be desirable to utilize SAT-solvers for these problems. However, we cannot transform these two reasoning problems into SAT in polynomial time, unless the Polynomial Hierarchy collapses, which is believed to be unlikely.

New Contribution In this work we show how to utilize certain structural aspects of disjunctive logic programs to transform the two ASP reasoning problems into SAT. In particular, we exhibit a transformation to SAT that runs in time $O(2^k n^2)$ where k is a structural parameter of the instance and n is the input size of the instance. Thus the combinatorial explosion, which is expected when transforming problems from the second level of the Polynomial Hierarchy to the first level, is confined to the parameter k, while the running time is polynomial in the input size n and the order of the polynomial is independent of k. Such transformations are known as "*fpt-transformations*" and form the base of the completeness theory of *Parameterized Complexity* (Downey and Fellows 1999; Flum and Grohe 2006). Our reductions *break complexity barriers* as they move problems form the second to the first level of the Polynomial Hierarchy.

It is known that the two reasoning problems, when restricted to so-called *normal programs*, drop to NP and

*Research supported by ERC (COMPLEX REASON 239962).

co-NP (Bidoít and Froidevaux 1991; Marek and Truszczynski 1991a; Marek and Truszczyński 1991b), respectively. Hence, it is natural to consider a structural parameter k as the distance of a given program from being normal. We measure the distance in terms of the smallest number of atoms that need to be deleted to make the program normal. Following Williams, Gomes, and Selman (2003) we call such a set of deleted atoms a *backdoor*. We show that in time $O(2^k n^2)$ we can solve both of the following two tasks for a given program P of input size n and an atom a^*:

Backdoor Detection: Find a backdoor of size at most k of the given program P, or decide that a backdoor of size k does not exist.

Backdoor Evaluation: Transform the program P into two propositional formulas $F_{\text{Brave}}(a^*)$ and $F_{\text{Skept}}(a^*)$ such that (i) $F_{\text{Brave}}(a^*)$ is satisfiable if and only if a^* is in some answer set of P, and (ii) $F_{\text{Skept}}(a^*)$ is unsatisfiable if and only if a^* is in all answer sets of P.

Tightness is a property of disjunctive logic programs that, similar to normality, lets the complexities of BRAVE and SKEPTICAL REASONING drop to NP and co-NP, respectively (Clark 1978; Fages 1994). Consequently, one could also consider backdoors to tightness. We show, however, that the reasoning problems already reach their full complexities (i.e., completeness for the second level of the Polynomial Hierarchy) with programs of distance one from being tight. Hence, an fpt-transformation into SAT for programs of distance $k > 0$ from being tight is not possible unless the Polynomial Hierarchy collapses.

Related Work Williams, Gomes, and Selman (2003) introduced the notion of backdoors to explain favorable running times and the heavy-tailed behavior of SAT and CSP solvers on practical instances. The parameterized complexity of finding small backdoors was initiated by Nishimura, Ragde, and Szeider (2004). For further results regarding the parameterized complexity of problems related to backdoors for SAT, we refer to a recent survey paper (Gaspers and Szeider 2012). Fichte and Szeider (2012) formulated a backdoor approach for ASP problems, and obtained complexity results with respect to the target class of Horn programs and various target classes based on acyclicity; some results could be generalized (Fichte 2012). Both papers are limited to target classes where we can enumerate the set of all answer sets in polynomial time. The results do not carry over to the present work since here we consider target classes where the problem of determining an answer set is already NP-hard.

Translations from ASP problems to SAT have been explored by several authors; existing research mainly focuses on transforming programs for which the reasoning problems already belong to NP or co-NP. In particular, translations have been considered for head cycle free programs (Ben-Eliyahu and Dechter 1994), tight programs (Fages 1994), and normal programs (Lin and Zhao 2004; Janhunen 2006).

Some authors have generalized the above translations to capture programs for which the reasoning problems are outside NP and co-NP. Janhunen et al. (2006) considered programs where the number of disjunctions in the heads of rules is bounded. They provided a translation that allows a SAT encoding of the test whether a candidate set of atoms is indeed an answer set of the input program. Lee and Lifschitz (2003) considered programs with a bounded number of cycles in the positive dependency graph. They suggested a translation that, similar to ours, transforms the input program into an exponentially larger propositional formula whose satisfying assignments correspond to answer sets of the program. As pointed out by Lifschitz and Razborov (2006), this translation produces an exponential blowup already for normal programs (we note that by way of contrast, our translation is in fact quadratic for normal programs).

Over the last few years, several SAT techniques have been integrated into practical ASP solvers. In particular, solvers for normal programs (Cmodels (Giunchiglia, Lierler, and Maratea 2006), ASSAT (Lin and Zhao 2004), Clasp (Gebser et al. 2007)) use certain extensions of Clark's completion and then utilize either black box SAT solvers or integrate conflict analysis, backjumping, and other techniques within the ASP context. ClaspD (Drescher et al. 2008) is a disjunctive ASP-solver that utilizes nogoods based on the logical characterizations of loop formulas (Lee 2005).

Preliminaries

Answer set programs We consider a universe of propositional *atoms*. A *disjunctive logic program* (or simply a *program*) P is a set of *rules* of the form $x_1 \vee \ldots \vee x_l \leftarrow y_1, \ldots, y_n, \neg z_1, \ldots, \neg z_m$ where $x_1, \ldots, x_l$, $y_1, \ldots, y_n$, $z_1, \ldots, z_m$ are atoms and l, n, m are non-negative integers. We write $H(r) = \{x_1, \ldots, x_l\}$ (the *head* of r), $B^+(r) = \{y_1, \ldots, y_n\}$ (the *positive body* of r), and $B^-(r) = \{z_1, \ldots, z_m\}$ (the *negative body* of r). We denote the sets of atoms occurring in a rule r or in a program P by $\text{at}(r) = H(r) \cup B^+(r) \cup B^-(r)$ and $\text{at}(P) = \bigcup_{r \in P} \text{at}(r)$, respectively. A rule r is *negation-free* if $B^-(r) = \emptyset$, r is *normal* if $|H(r)| \leq 1$, r is a *constraint* if $|H(r)| = 0$, r is *constraint-free* if $|H(r)| > 0$, r is *Horn* if it is negation-free and normal, r is *positive* if it is Horn and constraint-free, and r is *tautological* if $B^+(r) \cap (H(r) \cup B^-(r)) \neq \emptyset$. We say that a program has a certain property if all its rules have the property. We denote the class of all normal programs by **Normal** and the class of all Horn programs by **Horn**. In the following, we restrict ourselves to programs that do *not* contain any tautological rules. This restriction is not significant as tautological rules can be omitted from a program without changing its answer sets (Brass and Dix 1998).

A set M of atoms *satisfies* a rule r if $(H(r) \cup B^-(r)) \cap M \neq \emptyset$ or $B^+(r) \setminus M \neq \emptyset$. M is a *model* of P if it satisfies all rules of P. The *GL reduct* of a program P under a set M of atoms is the program P^M obtained from P by first, removing all rules r with $B^-(r) \cap M \neq \emptyset$ and second, removing all $\neg z$ where $z \in B^-(r)$ from all remaining rules r (Gelfond and Lifschitz 1991). M is an *answer set* (or *stable set*) of a program P if M is a minimal model of P^M. The Emden-Kowalski operator of a program P and a subset A of atoms of P is the set $T_P(A) := \{\, a \mid a \in H(r), B^+(r) \subseteq A, r \in P \,\}$. The *least model* $LM(P)$ is the least fixed point of $T_P(A)$ (Van Emden and Kowalski 1976). Note that every positive program P has a unique minimal model which equals the least model $LM(P)$ (Gelfond and Lifschitz 1988).

Example 1. Consider the program $P = \{\, a \vee c \leftarrow b;\ b \leftarrow c, \neg g;\ c \leftarrow a;\ b \vee c \leftarrow e;\ h \vee i \leftarrow g, \neg c;\ a \vee b;\ g \leftarrow \neg i;\ c \,\}$. The set $A = \{b, c, g\}$ is an answer set of P since $P^A = \{\, a \vee c \leftarrow b;\ c \leftarrow a;\ b \vee c \leftarrow e;\ a \vee b;\ g;\ c \,\}$ and the minimal models of P^A are $\{b, c, g\}$ and $\{a, c, g\}$.

The main reasoning problems for ASP are BRAVE REASONING (given a program P and an atom $a \in \text{at}(P)$, is a contained in some answer set of P?) and SKEPTICAL REASONING (given a program P and an atom $a \in \text{at}(P)$, is a contained in all answer sets of P?). BRAVE REASONING is Σ_2^P-complete, SKEPTICAL REASONING is Π_2^P-complete (Eiter and Gottlob 1995).

Parameterized Complexity We give some basic background on parameterized complexity. For more detailed information we refer to other sources (Downey and Fellows 1999; Flum and Grohe 2006). A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. For an instance $(I, k) \in \Sigma^* \times \mathbb{N}$ we call I the *main part* and k the *parameter*. L is *fixed-parameter tractable* if there exists a computable function f and a constant c such that there exists an algorithm that decides whether $(I, k) \in L$ in time $O(f(k)\|I\|^c)$ where $\|I\|$ denotes the size of I. Such an algorithm is called an *fpt-algorithm*. FPT is the class of all fixed-parameter tractable decision problems.

Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq \Sigma'^* \times \mathbb{N}$ be two parameterized problems for some finite alphabets Σ and Σ'. An *fpt-reduction* r from L to L' is a many-to-one reduction from $\Sigma^* \times \mathbb{N}$ to $\Sigma'^* \times \mathbb{N}$ such that for all $I \in \Sigma^*$ we have $(I, k) \in L$ if and only if $r(I, k) = (I', k') \in L'$ such that $k' \leq g(k)$ for a fixed computable function $g : \mathbb{N} \to \mathbb{N}$ and there is a computable function f and a constant c such that r is computable in time $O(f(k)\|I\|^c)$ where $\|I\|$ denotes the size of I (Flum and Grohe 2006). Thus, an fpt-reduction is, in particular, an fpt-algorithm. It is easy to see that the class FPT is closed under fpt-reductions. We would like to note that the theory of fixed-parameter intractability is based on fpt-reductions (Flum and Grohe 2006).

Propositional satisfiability A *truth assignment* is a mapping $\tau : X \to \{0, 1\}$ defined for a set X of atoms. For $x \in X$ we put $\tau(\neg x) = 1 - \tau(x)$. By ta$(X)$ we denote the set of all truth assignments $\tau : X \to \{0, 1\}$. We usually say *variable* instead of atom in the context of formulas. Given a propositional formula F, the problem SAT asks whether F is satisfiable. We can consider SAT as a parameterized problem by simply associating with every formula the parameter 0.

Backdoors of Programs

In the following we give the main notions concerning backdoors for answer set programming, as introduced by Fichte and Szeider (2012). Let P be a program, X a set of atoms, and $\tau \in \text{ta}(X)$. The *truth assignment reduct* of P under τ is the logic program P_τ obtained from P by removing all rules r for which at least one of the following holds: (i) $H(r) \cap \tau^{-1}(1) \neq \emptyset$, (ii) $H(r) \subseteq X$, (iii) $B^+(r) \cap \tau^{-1}(0) \neq \emptyset$, and (iv) $B^-(r) \cap \tau^{-1}(1) \neq \emptyset$, and then removing from the heads and bodies of the remaining rules all literals $v, \neg v$ with $v \in X$. In the following, let $\mathcal{C}$ be a class of programs. We call $\mathcal{C}$ to be *rule induced* if for each $P \in \mathcal{C}$, $P' \subseteq P$ implies $P' \in \mathcal{C}$. A set X of atoms is a *strong $\mathcal{C}$-backdoor* of a program P if $P_\tau \in \mathcal{C}$ for all truth assignments $\tau \in \text{ta}(X)$. Given a strong $\mathcal{C}$-backdoor X of a program P, the answer sets of P are among the answer sets we obtain from the truth assignment reducts P_τ where $\tau \in X$, more formally $\text{AS}(P) \subseteq \{\, M \cup \tau^{-1}(1) \mid \tau \in \text{ta}(X \cap \text{at}(P)), M \in \text{AS}(P_\tau) \,\}$ where AS(P) denotes the set of all answer sets of P. For a program P and a set X of atoms we define $P - X$ as the program obtained from P by deleting all atoms contained in X and their negations from the heads and bodies of all the rules of P. A set X of atoms is a *deletion $\mathcal{C}$-backdoor* of a program P if $P - X \in \mathcal{C}$.

Example 2. Consider the program P from Example 1. The set $X = \{b, c, h\}$ is a strong **Normal**-backdoor since the truth assignment reducts $P_{b=0,c=0,h=0} = P_{000} = \{\, i \leftarrow g;\ a;\ g \leftarrow \neg i \,\}$, $P_{001} = P_{010} = P_{011} = P_{101} = \{\, a;\ g \leftarrow \neg i \,\}$, $P_{100} = \{\, a;\ i \leftarrow g;\ g \leftarrow \neg i \,\}$, and $P_{110} = P_{111} = \{\, g \leftarrow \neg i \,\}$ are in the class **Normal**.

In the following we refer to $\mathcal{C}$ as the *target class* of the backdoor. For most target classes $\mathcal{C}$, deletion $\mathcal{C}$-backdoors are strong $\mathcal{C}$-backdoors. For $\mathcal{C} =$ **Normal** even the opposite direction is true.

Proposition 1 (Fichte and Szeider, 2012). *If $\mathcal{C}$ is rule induced, then every deletion $\mathcal{C}$-backdoor is a strong $\mathcal{C}$-backdoor.*

Lemma 1. *Let P be a program. A set X is a strong* **Normal**-*backdoor of a program P if and only if it is a deletion* **Normal**-*backdoor of P.*

Proof. We observe that the class of all normal programs is rule-induced. Thus the if direction holds by Proposition 1. We proceed to show the only-if direction. Assume X is a strong **Normal**-backdoor of P. Consider a rule $r' \in P - X$ which is not tautological. Let $r \in P$ be a rule from which r' was obtained in forming $P - X$. We define $\tau \in \text{ta}(X)$ by setting all atoms in $H(r) \cup B^-(r)$ to 0, all atoms in $B^+(r)$ to 1, and all remaining atoms in $X \setminus \text{at}(r)$ arbitrarily to 0 or 1. Since r is not tautological, this definition of τ is sound. It remains to observe that $r' \in P_\tau$. Since X is a strong **Normal**-backdoor of P, the rule r' is normal. Hence the lemma follows. ☐

Each target class $\mathcal{C}$ gives rise to the following problems:

- $\mathcal{C}$-BACKDOOR-ASP-CHECK: *Given* a program P, a strong $\mathcal{C}$-backdoor X of P, a set $M \subseteq \text{at}(P)$, and the parameter size of the backdoor $k = |X|$. *Is* M an answer set of P?
- $\mathcal{C}$-BACKDOOR-BRAVE-REASONING: *Given* a program P, a strong $\mathcal{C}$-backdoor X of P, an atom $a^* \in \text{at}(P)$, and the parameter size of the backdoor $k = |X|$. Does a^* belong to *some* answer set of P?
- $\mathcal{C}$-BACKDOOR-SKEPTICAL-REASONING: *Given* a program P, a strong $\mathcal{C}$-backdoor X of P, an atom $a^* \in \text{at}(P)$,

and the parameter size of the backdoor $k = |X|$. Does a^* belong to *all* answer sets of P?

Problems for *deletion $\mathcal{C}$-backdoors* can be defined similarly.

Using Backdoors

In this section, we show results regarding the use of backdoors with respect to the target class **Normal**.

Theorem 1. *The problem* **Normal**-BACKDOOR-ASP-CHECK *is fixed-parameter tractable. More specifically, given a program P of input size n, a strong* **Normal***-backdoor N of P of size k, and a set $M \subseteq$ at(P) of atoms, we can check in time $O(2^k n)$ whether M is an answer set of P.*

The most important part for establishing Theorem 1 is to check whether a model is a minimal model. In general, this is a co-NP-complete task, but in the context of Theorem 1 we can achieve fixed-parameter tractability based on the following construction and lemma.

Let P be a given program, X a strong **Normal**-backdoor of P of size k, and let $M \subseteq$ at(P). For a set $X_1 \subseteq M \cap X$ we construct a program $P_{X_1 \subseteq X}$ as follows: (i) remove all rules r for which $H(r) \cap X_1 \neq \emptyset$ and (ii) replace for all remaining rules r the head $H(r)$ with $H(r) \setminus X$ and the positive body $B^+(r)$ with $B^+(r) \setminus X_1$.

Recall that by definition we exclude programs with tautological rules. Since X is a strong **Normal**-backdoor of P, it is also a deletion **Normal**-backdoor of P by Lemma 1. Hence $P - X$ is normal. Let r be an arbitrarily chosen rule in P. Then there is a corresponding rule $r' \in P - X$ and a corresponding rule $r'' \in P_{X_1 \subseteq X}$. Since we remove in both constructions exactly the same literals from the head of every rule, $H(r') = H(r'')$ holds. Consequently, $P_{X_1 \subseteq X}$ is normal and $P^M_{X_1 \subseteq X}$ is Horn (here $P^M_{X_1 \subseteq X}$ denotes the GL-reduct of $P_{X_1 \subseteq X}$ under M).

For any program P' let Constr(P') denote the set of constrains of P' and Pos$(P') = P' \setminus$ Constr(P'). If P' is Horn, Pos(P') has a least model L and P' has a model if and only if L is a model of Constr(P') (Dowling and Gallier 1984).

Let X be a strong **Normal**-backdoor of P and $X_1 \subseteq X$. Given $M \subseteq$ at(P), the algorithm MINCHECK(X_1) below performs the following steps:

1. Return *True* if X_1 is not a subset of M.
2. Compute the Horn program $P^M_{X_1 \subseteq X}$.
3. Compute the least model L of Pos$(P^M_{X_1 \subseteq X})$.
4. Return *True* if at least one of the following conditions holds:
   (a) L is not a model of Constr$(P^M_{X_1 \subseteq X})$.
   (b) L is not a subset of X,
   (c) $L \cup X_1$ is not a proper subset of M,
   (d) $L \cup X_1$ is not a model of P^M.
5. Otherwise return *False*.

Lemma 2. *Let X be a strong* **Normal***-backdoor. A model $M \subseteq$ at(P) of P^M is a minimal model of P^M if and only if* MINCHECK(X_1) *returns True for each set $X_1 \subseteq X$.*

Because of space constraints we omit the lengthy proof.

We are now in a position to establish Theorem 1.

Proof of Theorem 1. First, we check whether M is a model of P^M. If it is, we run the algorithm MINCHECK(X_1) for each set $X_1 \subseteq X$. By Lemma 2 the algorithm decides whether M is an answer set of P. The check of whether M is a model of P^M can clearly be carried out in linear time. The algorithm MINCHECK(X_1) runs in linear time since we can compute the least model of a Horn program in linear time (Dowling and Gallier 1984). As there are at most 2^k sets X_1 to consider, the total running time is $O(2^k n)$ where n denotes the input size of P and $k = |X|$. We obtain fixed-parameter tractability for the parameter k. ☐

Example 3. Consider the program P from Example 1 and the backdoor $X = \{b, c, h\}$ from Example 2. Let $M = \{b, c, g\} \subseteq$ at(P). Since M satisfies all rules in P, the set M is a model of P. We apply the algorithm MINCHECK for each subset of $\{b, c, h\}$. For $X_1 = \emptyset$ we obtain $P^M_{X_1 \subseteq X} = \{\, a \leftarrow b;\ \leftarrow a;\ \leftarrow e;\ a;\ g\,\}$. The set $L = \{a, g\}$ is the least model of Pos$(P^M_{X_1 \subseteq X})$. Since Condition 4a holds, the algorithm returns *True* for X_1. For $X_2 = \{b\}$ we have $P^M_{X_2 \subseteq X} = \{\, a;\ \leftarrow a;\ g;\ \leftarrow \}$ and the least model $L = \{a, g\}$ of Pos$(P^M_{X_2 \subseteq X})$. Since Condition 4a holds, MINCHECK returns *True* for X_2. For $X_3 = \{c\}$ we gain $P^M_{X_3 \subseteq X} = \{\, a;\ g\,\}$ and the least model $L = \{a, g\}$ of Pos$(P^M_{X_3 \subseteq X})$. Since Condition 4c holds, the algorithm returns *True* for X_3. For $X_4 = \{b, c\}$ we obtain $P^M_{X_4 \subseteq X} = \{g\}$. The set $L = \{g\}$ is the least model of Pos$(P^M_{X_4 \subseteq X})$. Since Condition 4c holds, the algorithm returns *True* for X_4. For all remaining subsets of X the Algorithm MINCHECK returns *True* according to Condition 1. Consequently, M is a minimal model of P^M and thus an answer set of P.

Next, we state and prove that there are fpt-reductions from **Normal**-BACKDOOR-BRAVE-REASONING and **Normal**-BACKDOOR-SKEPTICAL-REASONING to SAT which is the main result of this paper.

Theorem 2. *Given a disjunctive logic program P of input size n, a strong* **Normal***-backdoor X of P of size k, and an atom $a^* \in$ at(P), we can produce in time $O(2^k n^2)$ propositional formulas $F_{Brave}(a^*)$ and $F_{Skept}(a^*)$ such that (i) $F_{Brave}(a^*)$ is satisfiable if and only if a^* is in some answer set of P, and (ii) $F_{Skept}(a^*)$ is unsatisfiable if and only if a^* is in all answer sets of P.*

Proof. We would like to use a similar approach as in the proof of Theorem 1. However, we cannot consider all possible models M one by one, as there could be too many of them. Instead, we will show that it is possible to implement MINCHECK(X_1) for each set $X_1 \subseteq X$ nondeterministically in such a way that we do not need to know M in advance. Possible sets M will be represented by the truth values of certain variables, and since the truth values do not need to be known in advance, this will allow us to consider all possible sets M without enumerating them.

Next, we describe the construction of the formulas $F_{\text{Brave}}(a^*)$ and $F_{\text{Skept}}(a^*)$ in detail.

Among the variables of our formulas will be a set $V := \{\, v[a] \mid a \in$ at$(P)\,\}$ containing a variable for each atom of P. The truth values of the variables in V represent a subset $M \subseteq \text{at}(P)$, such that $v[a]$ is true if and only if $a \in M$.

We define

$F_{\text{Brave}}(a^*) := F^{\text{mod}} \wedge F^{\text{min}} \wedge v[a^*]$ and

$F_{\text{Skept}}(a^*) := F^{\text{mod}} \wedge F^{\text{min}} \wedge \neg v[a^*]$,

where F^{mod} and F^{min} are formulas, defined below, that check whether the truth values of the variables in V represent a model M of P^M, and whether M is a minimal model of P^M, respectively.

The definition of F^{mod} is easy:

$$F^{\text{mod}} := \bigwedge_{r\in P} \Big(\bigwedge_{b\in B^-(r)} \neg v[b] \rightarrow \big(\bigvee_{b\in B^+(r)} \neg v[b] \vee \bigvee_{b\in H(r)} v[b]\big)\Big).$$

The definition of F^{min} is more involved. First we define:

$$F^{\text{min}} := \bigwedge_{1\leq i\leq 2^k} F_i^{\text{min}},$$

where F_i^{min}, defined below, encodes the Algorithm MINCHECK(X_i) for each set X_i where $X_1, \ldots, X_{2^k}$ is an enumeration of all the subsets of X.

The formula F_i^{min} will contain, in addition to the variables in V, p distinct variables for each atom of P, $p := \min\{|P|, |\text{at}(P)|\}$. In particular, the set of variables of F_i^{min} is the disjoint union of V and U_i where $U_i := \{\, u_i^j[a] \mid a \in \text{at}(P),\ 1 \leq j \leq p\,\}$. We write U_i^j for the subset of U_i containing all the variables $u_i^j[a]$. We assume that for $i \neq i'$ the sets U_i and $U_{i'}$ are disjoint. For each $a \in \text{at}(P)$ we also use the propositional constants $X(a)$ and $X_1(a)$ that are true if and only if $a \in X$ and $a \in X_1$, respectively.

We define the formula F_i^{min} by means of the following auxiliary formulas.

The first auxiliary formula checks whether the truth values of the variables in V represent a set M that contains X_i:

$$F_i^{\subseteq} := \bigwedge_{a\in X} X_i(a) \rightarrow v[a].$$

The next auxiliary formula encodes the computation of the least model ("lm") L of $\text{Pos}(P^M_{X_i\subseteq X})$ where M and L are represented by the truth values of the variables in V and U_i^p, respectively.

$F_i^{\text{lm}} := \bigwedge_{a\in \text{at}(P), 0\leq i\leq p} F_i^{(a,i)}$, where

$F_i^{(a,0)} := u_i^0[a] \leftrightarrow \textit{false}$,

$$F_i^{(a,j)} := u_i^j[a] \leftrightarrow \big[u_i^{j-1}[a] \vee \bigvee_{r\in P_{X_i\subseteq X}, a\in H(r)} \big(\bigwedge_{b\in B^+(r)} u_i^{j-1}[b] \wedge \bigwedge_{b\in B^-(r)} \neg v[b]\big)\big]$$
(for $1 \leq j \leq p-1$).

The idea behind the construction of F_i^{lm} is to simulate the linear-time algorithm of Dowling and Gallier (1984). Initially, all variables are set to false. This is represented by variables $u_i^0[a]$. Now we flip a variable from false to true if and only if there is a Horn rule where all the variables in the rule body are true. We iterate this process until a fixed-point is reached, then we have the least model. The flipping is represented in our formula by setting a variable $u_i^j[a]$ to true if and only if either $u_i^{j-1}[a]$ is true, or there is a rule $r \in \text{Pos}(P^M_{X_i\subseteq X})$ such that $H(r) = \{a\}$ and $u_i^j[b]$ is true for all $b \in B^+(r)$. The truth values of the variables u_i^p now represent the least model of $\text{Pos}(P^M_{X_i\subseteq X})$.

The next four auxiliary formulas check whether the respective condition (a)–(d) in Step 4 of algorithm MINCHECK(X_i) does not hold for L.

$F_i^{\text{(a)}}$ expresses that there is a rule in $\text{Constr}(P^M_{X_i\subseteq X})$ that is not satisfied by L:

$$F_i^{\text{(a)}} := \bigvee_{r\in P_{X_i\subseteq X}, H(r)\subseteq X} \big(\bigwedge_{b\in B^-(r)} \neg v[b] \wedge \bigwedge_{b\in B^+(r)} u_i^p[b]\big).$$

$F_i^{\text{(b)}}$ expresses that L contains an atom that is not in $M \setminus X$:

$$F_i^{\text{(b)}} := \bigvee_{a\in \text{at}(P)\setminus X} (\neg v[a] \wedge u_i^p[a]).$$

$F_i^{\text{(c)}}$ expresses that $L \cup X_i$ equals M or $L \cup X_i$ contains an atom that is not in M:

$$F_i^{\text{(c)}} := \Big(\bigwedge_{a\in \text{at}(P)} v[a] \leftrightarrow (u_i^p[a] \vee X_i(a))\Big) \vee \Big(\bigvee_{a\in \text{at}(P)} (u_i^p[a] \vee X_i(a)) \wedge \neg v[a]\Big).$$

$F_i^{\text{(d)}}$ expresses that P^M contains a rule that is not satisfied by $L \cup X_i$:

$$F_i^{\text{(d)}} := \bigvee_{r\in P} \big[\bigwedge_{a\in B^-(r)} \neg v[a] \wedge \bigwedge_{a\in H(r)} (\neg u_i^p[a] \wedge \neg X_i(a)) \wedge \bigwedge_{b\in B^+(r)} (u_i^p[b] \vee X_i(b))\big].$$

Now we can put the auxiliary formulas together and obtain

$$F_i^{\text{min}} := \neg F_i^{\subseteq} \vee (F_i^{\text{lm}} \wedge (F_i^{\text{(a)}} \vee F_i^{\text{(b)}} \vee F_i^{\text{(c)}} \vee F_i^{\text{(d)}})).$$

It follows by Lemma 2 and by the construction of the auxiliary formulas that (i) $F_{\text{Brave}}(a^*)$ is satisfiable if and only if a^* is in some answer set of P, and (ii) $F_{\text{Skept}}(a^*)$ is unsatisfiable if and only if a^* is in all answer sets of P.

Hence, it remains to observe that for each $i \leq 2^k$ the auxiliary formula F_i^{lm} can be constructed in quadratic time, whereas the auxiliary formulas $F_i^{\subseteq}$ and $F_i^{\text{(a)}} \vee F_i^{\text{(b)}} \vee F_i^{\text{(c)}} \vee F_i^{\text{(d)}}$ can be constructed in linear time. Since $|X| = k$ by assumption, we need to construct $O(2^k)$ auxiliary formulas in order to obtain $F_{\text{Skept}}(a^*)$ and $F_{\text{Brave}}(a^*)$. Hence, the running time as claimed in Theorem 2 follows. ☐

We would like to note that Theorem 2 remains true if we require that the formulas $F_{\text{Skept}}(a^*)$ and $F_{\text{Brave}}(a^*)$ are in Conjunctive Normal Form (CNF), as we can transform in linear time any propositional formula into a satisfiability-equivalent formula in CNF, e.g., using the well-known transformation due to Tseitin (1968).

Furthermore, the SAT encoding can be improved. For instance, one could share parts between the formulas F_i^{min} or replace the quadratic formula F_i^{lm} for the computation of least models with a smaller and more sophisticated SAT encoding (Janhunen 2004) or a SAT(DL) encoding (Janhunen, Niemela, and Sevalnev 2009) for the SMT framework which combines propositional logic and linear constraints.

We would like to point out that our approach directly extends to more general problems, when we look for answer sets that satisfy a certain global property which can be expressed by a propositional formula F^{prop} on the variables in V. We just check the satisfiability of $F^{\text{mod}} \wedge F^{\text{min}} \wedge F^{\text{prop}}$.

Example 4. Consider the program P from Example 1 and the strong **Normal**-backdoor $X = \{b, c, h\}$ of P from Example 2. We ask whether the atom b is contained in at least one answer set. To decide the question, we check that $F_{\text{brave}}(b)$ is satisfiable and we answer the question positively. Since $M = \{b, c, g\}$ is model of P^M we can satisfy F^{mod} with a truth assignment τ that maps 1 to each variable $v[x]$ where $x \in \{b, c, g\}$ and 0 to each variable $v[x]$ where $x \in \text{at}(P) \setminus \{b, c, g\}$. For $i = 1$ let $X_1 = \emptyset$. Then we have for the constants $X_1(x) = 0$ where $x \in \{b, c, h\}$. Observe that τ already satisfies $F_i^{\subseteq}$ and that F_i^{lm} encodes the computation of the least model L of $\text{Pos}(P^M_{X_1 \subseteq X})$ where L is represented by the truth values of the variables in $U_i^P = \{\, u_i^p[x] \mid x \in \text{at}(P)\,\}$. Thus τ also satisfies F_i^{lm} if τ maps $u_i^p[a]$ to 1, $u_i^p[g]$ to 1, and $u_i^p[x]$ to 0 where $x \in \text{at}(P) \setminus \{a, g\}$. As τ satisfies $F_1^{(\text{a})}$, the truth assignment τ satisfies the formula F_1^{min}. It is not hard to see that F_i^{min} is satisfiable for other values of i. Hence the formula $F_{\text{brave}}(b)$ is satisfiable and b is contained in at least one answer set.

Completeness for paraNP and co-paraNP

The parameterized complexity class paraNP contains all parameterized decision problems L such that $(I, k) \in L$ can be decided *nondeterministically* in time $O(f(k)\|I\|^c)$, for some computable function f and constant c (Flum and Grohe 2006). By co-paraNP we denote the class of all parameterized decision problems whose complement (the same problem with yes and no answers swapped) is in paraNP.

As a corollary to Theorem 2 we obtain the following result:

Corollary 1. **Normal**-BACKDOOR-BRAVE-REASONING *is* paraNP-*complete, and* **Normal**-BACKDOOR-SKEPTICAL-REASONING *is* co-paraNP-*complete.*

Proof. If a parameterized problem L is NP-hard when we fix the parameter to a constant, then L is paraNP-hard (Flum and Grohe, 2006, Th. 2.14). As **Normal**-BACKDOOR-BRAVE-REASONING is NP-hard for backdoor size 0, we conclude that **Normal**-BACKDOOR-BRAVE-REASONING is paraNP-hard. A similar argument shows that **Normal**-BACKDOOR-SKEPTICAL-REASONING is co-paraNP-hard. SAT, considered as a parameterized problem with constant parameter 0, is clearly paraNP-complete, this also follows from the mentioned result of Flum and Grohe (2006); hence UNSAT is co-paraNP-complete. As Theorem 2 provides fpt-reductions from **Normal**-BACKDOOR-BRAVE-REASONING to SAT, and from **Normal**-BACKDOOR-SKEPTICAL-REASONING to UNSAT, we conclude that **Normal**-BACKDOOR-BRAVE-REASONING is in paraNP, and **Normal**-BACKDOOR-SKEPTICAL-REASONING is in co-paraNP. ☐

Finding Backdoors

In this section, we study the problem of finding backdoors, formalized in terms of the following parameterized problem: STRONG $\mathcal{C}$-BACKDOOR-DETECTION: *Given* a (disjunctive) program P, and the parameter integer k. *Question:* Find a strong $\mathcal{C}$-backdoor X of P of size at most k, or report that such X does not exist. We also consider the problem DELETION $\mathcal{C}$-BACKDOOR-DETECTION, defined similarly.

Let P be a program. Let the *head dependency graph* U_P^H be the undirected graph $U_P^H = (V, E)$ defined on the set $V = \text{at}(P)$ of atoms of the given program P, where two atoms x, y are joined by an edge $xy \in E$ if and only if P contains a non-tautological rule r with $x, y \in H(r)$. A *vertex cover* of a graph $G = (V, E)$ is a set $X \subseteq V$ such that for every edge $uv \in E$ we have $\{u, v\} \cap X \neq \emptyset$.

Lemma 3. *Let P be a program. A set $X \subseteq \text{at}(P)$ is a deletion* **Normal**-*backdoor of P if and only if X is a vertex cover of U_P^H.*

Due to space limitations we omit the proof.

Theorem 3. *The problems* STRONG **Normal**-BACKDOOR-DETECTION *and* DELETION **Normal**-BACKDOOR-DETECTION *are fixed-parameter tractable.*

Proof. In order to find a deletion **Normal**-backdoor of a given program P, we use Lemma 3 and find a vertex cover of size at most k in the head dependency graph U_P^D. A vertex cover of size k, if it exists, can be found in time $O(1.2738^k + kn)$ (Chen, Kanj, and Xia 2006). Thus the theorem holds for deletion **Normal**-backdoors. Lemma 1 states that the strong **Normal**-backdoors of P are exactly the deletion **Normal**-backdoors of P (as we assume that P does not contain any tautological rules). The theorem follows. ☐

In Theorem 2 we assume that a strong **Normal**-backdoor of size at most k is given when solving the problems STRONG **Normal**-BACKDOOR-BRAVE-REASONING and SKEPTICAL-REASONING. As a direct consequence of Theorem 3, this assumption can be dropped, and we obtain the following corollary.

Corollary 2. *The results of Theorem 2 and Corollary 1 still hold if the backdoor is not given as part of the input.*

Backdoors to Tightness

We associate with each program P its *positive dependency graph* D_P^+. It has the atoms of P as vertices and a directed edge (x, y) between any two atoms $x, y \in \text{at}(P)$ for which there is a rule $r \in P$ with $x \in H(r)$ and $y \in B^+(r)$. A program is called *tight* if D_P^+ is acyclic (Lee and Lifschitz 2003). We denote the class of all tight programs by **Tight**.

It is well known that the main ASP reasoning problems are in NP and co-NP for tight programs; in fact, a reduction to SAT based on the concept of *loop formulas* has been proposed by Lin and Zhao (2004). This was then generalized by Lee and Lifschitz (2003) with a reduction that takes as input a disjunctive normal program P together with the set S of all directed cycles in the positive dependency graph of P, and produces a CNF formula F such that answer sets of P correspond to the satisfying assignments of F. This provides an fpt-reduction from the problems BRAVE REASONING and SKEPTICAL REASONING to SAT, when parameterized by the number of all cycles in the positive dependency graph of a given program P, assuming that these cycles are given as part of the input.

The number of cycles does not seem to be a very practical parameter, as this number can quickly become very large even for very simple programs. Lifschitz and Razborov (2006) have shown that already for normal programs an exponential blowup may occur, since the number of cycles in a normal program can be arbitrarily large. Hence, it would be interesting to generalize the result of Lee and Lifschitz (2003) to a more powerful parameter. In fact, the size k of a deletion **Tight**-backdoor would be a candidate for such a parameter, as it is easy to see, it is at most as large as the number of cycles, but can be exponentially smaller. This is a direct consequence of the following two observations: (i) If a program P has exactly k cycles in D_P^+, we can construct a deletion **Tight**-backdoor X of P by taking one element from each cycle into X. (ii) If a program P has a deletion **Tight**-backdoor of size 1, it can have arbitrarily many cycles that run through the atom in the backdoor.

Next, we show that this parameter k is of little use, as the reasoning problems already reach their full complexity for programs with a deletion **Tight**-backdoor of size 1.

Theorem 4. *The problems* **Tight**-BACKDOOR-BRAVE-REASONING *and* **Tight**-BACKDOOR-SKEPTICAL-REASONING *are Σ_2^P-hard and Π_2^P-hard, respectively, even for programs that admit a strong* **Tight***-backdoor of size* 1*, and the backdoor is provided with the input. The problems remain hard when we consider a deletion* **Tight***-backdoor instead of a strong* **Tight***-backdoor.*

Proof. Eiter and Gottlob (1995) give a reduction from a Σ_2^P-complete (Π_P^2-complete) problem to Brave (Skeptical) Reasoning, where the produced program has rules of the form $x_i \vee v_i$; $y_i \vee z_j$; $y_j \leftarrow w$; $z_j \leftarrow w$; $w \leftarrow y_j, z_j$; $w \leftarrow g(l_{k,1}), g(l_{k,2}), g(l_{k,3})$; $w \leftarrow \neg w$. The set $\{w\}$ is a deletion **Tight**-backdoor (strong **Tight**-backdoor) of size 1. □

Experiments

Although our main results are theoretical, we have performed first experiments to determine the size of smallest strong **Normal**-backdoors for answer set programs representing *structured* and *random* sets of instances. Our experimental results summarized in Table 1 indicate, as expected, that structured instances have smaller backdoors than random instances. As instances from `ConformantPlanning` have rather small backdoors our translation seems to be feasible for these instances. Furthermore, we have compared the size of a smallest strong **Normal**-backdoor with the size of a smallest strong **Horn**-backdoor (Fichte and Szeider 2012) for selected sets. It turns out that for `ConformantPlanning` smallest strong **Normal**-backdoors are significantly smaller (0.7% vs. 8.8% of the total number of atoms).

Conclusion

We have shown that backdoors of small size capture structural properties of disjunctive ASP instances that yield to a reduction of problem complexity. In particular, small backdoors to normality admit an fpt-translation from ASP to SAT and thus reduce the complexity of the fundamental ASP problems from the second level of the Polynomial Hierarchy to the first level.

instance set	atoms	bd (%)	stdev
`ConformantPlanning`	1378.21	0.69	0.39
`MinimalDiagnosis`	97302.5	14.19	3.19
`MUS`	49402.3	1.90	0.35
`StrategicCompanies`	2002.0	6.03	0.04
`Mutex`	6449.0	49.94	0.09
`RandomQBF`	160.1	49.69	0.00

Table 1: Size of smallest strong **Normal**-backdoor (bd) for benchmark sets, given as % of the total number of atoms by the mean over the instances.
`ConformantPlanning`: secure planning under incomplete initial states (To, Pontelli, and Son 2009) encodings provided by Gebser and Kaminski (2012). `MinimalDiagnosis`: an application in systems biology (Gebser et al. 2008) instances provided by Calimeri et al. (2011). `MUS`: problem whether a clause belongs to some minimal unsatisfiable subset (Janota and Marques-Silva 2011) encoding provided by Gebser and Kaminski (2012). `StrategicCompanies`: encoding the Σ_2^P-complete problem of producing and owning companies and strategic sets between the companies (Gebser *et al.* 2007). `Mutex`: equivalence test of partial implementations of circuits, provided by Maratea *et al.* (2008) based on QBF instances of Ayari and Basin (2000). `RandomQBF`: translations of randomly generated 2-QBF instances using the method by Chen and Interian (2005) instances provided by Gebser (2007).

Thus, the size of a smallest **Normal**-backdoor is a structural parameter that admits a fixed-parameter tractable complexity reduction without making the problem itself fixed-parameter tractable.

Our *complexity barrier breaking reductions* provide a new way of using fixed-parameter tractability and enlarges its applicability. In fact, our approach as exemplified above for ASP is very general and might be applicable to a wide range of other hard combinatorial problems that lie beyond NP or co-NP. We hope that our work stimulates further investigations into this direction such as the application to abduction very recently established by Pfandler, Rümmele, and Szeider (2013).

Our first empirical results suggest that with an improved SAT encoding and preprocessing techniques to reduce the size of **Normal**-backdoors (for instance, *shifting*, Janhunen et al., 2007), our approach could be of practical use, at least for certain classes of instances, and hence might fit into a portfolio-based solver.

References

Ayari, A.; Basin, D. Bounded model construction for monadic second-order logics. 2000. In *Computer Aided Verification*, *LNCS* 1855. 99–112. Springer.

Ben-Eliyahu, R., and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence* 12(1):53–87. Springer.

Bidoít, N., and Froidevaux, C. 1991. Negation by default and unstratifiable logic programs. *Th. Comp. Sci.* 78(1):85–112. Elsevier.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, vol. 185 of *Frontiers in Artificial Intelligence and Applications* 185. IOS Press.

Brass, S., and Dix, J. 1998. Characterizations of the disjunctive wellfounded semantics: Confluent calculi and iterated GCWA. *Journal of Automated Reasoning* 20:143–165. Springer.

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

Calimeri, F.; et.al. 2011. The third answer set programming competition: Preliminary report of the system competition track. *LPNMR'11* 388–403. Springer

Chen, J.; Kanj, I.; and Xia, G. 2006. Improved parameterized upper bounds for vertex cover. *MFCS'06*. 238–249. Springer.

Chen, H.; Interian, Y. 2005 A model for generating random quantified boolean formulas. *IJCAI'05*. 66–71. Morgan Kaufmann.

Clark, K. L. 1978. Negation as failure. *Logic and data bases* 1:293–322. Plenum Publishing Corporation.

Dowling, W. F., and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal on Logic Programming* 1(3):267–284. Elsevier.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. Springer.

Drescher, C.; Gebser, M.; Grote, T.; Kaufmann, B.; König, A.; Ostrowski, M.; and Schaub, T. 2008. Conflict-driven disjunctive answer set solving. *KR'08*. 422–432. AAAI Press.

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. Springer.

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

Fichte, J. K., and Szeider, S. 2012. Backdoors to tractable answer-set programming. Updated IJCAI'11 paper. arXiv:1104.2788.

Fichte, J. K. 2012. The good, the bad, and the odd: Cycles in answer-set programs. In *New Directions in Logic, Language and Computation*, *LNCS* 7415. 78–90. Springer.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, vol. XIV of *Th. Comp. Sci.* Springer.

Gaspers, S., and Szeider, S. 2012. Backdoors to satisfaction. In *The Multivariate Algorithmic Revolution and Beyond*, *LNCS* 7370. 287–317. Springer.

Gebser, M., and Kaminski, R. 2012. Personal Communication.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. Conflict-driven answer set solving. *IJCAI'07*. 386–392. Morgan Kaufmann.

Gebser, M.; Liu, L.; Namasivayam, G.; Neumann, A.; Schaub, T.; Truszczyński, M. 2007, The first answer set programming system competition. *LPNMR'07*. 3–17. Springer.

Gebser, M.; Schaub, T.; Thiele, S.; Usadel, B.; and Veber, P. 2008. Detecting inconsistencies in large biological networks with answer set programming. In *Logic Programming*, *LNCS* 5366. 130–144. Springer.

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

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3/4):365–386. Springer.

Giunchiglia, E.; Lierler, Y.; and Maratea, M. 2006. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning* 36(4):345–377. Springer.

Maratea, M.; Ricca, F.; Faber, W.; Leone, N. 2007 Look-back techniques and heuristics in DLV: Implementation, evaluation, and comparison to QBF solvers. *Journal of Algorithms*. 63(1-3):70–89. Elsevier.

Gomes, C. P.; Kautz, H.; Sabharwal, A.; and Selman, B. 2008. Chapter 2 satisfiability solvers. In *Handbook of Knowledge Representation*, vol. 3 of *Foundations of Artificial Intelligence*. 89–134. Elsevier.

Janhunen, T.; Niemelä, I.; Seipel, D.; Simons, P.; and You, J. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Trans. Comput. Logic* 7(1):1–37. ACM.

Janhunen, T.; Oikarinen, E.; Tompits, H.; and Woltran, S. 2007. Modularity aspects of disjunctive stable models. *LPNMR'07*. 175–187. Springer.

Janhunen, T.; Niemela, I.; and Sevalnev, M. 2009. Computing stable models via reductions to difference logic. *LPNMR'09*. 142–154. Springer.

Janhunen, T. 2004. Representing normal programs with clauses. *ECAI'04*. 358–362. IOS Press.

Janhunen, T. 2006. Some (in)translatability results for normal logic programs and propositional theories. *Journal of Applied Non-Classical Logics* 16(1-2):35–86. Taylor & Francis.

Janota, M., and Marques-Silva, J. 2011. A tool for circumscription-based mus membership testing. *LPNMR'11*. 266–271. Springer

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

Lee, J., and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In *Logic Programming*, *LNCS* 2916 451–465. Springer.

Lee, J. 2005. A model-theoretic counterpart of loop formulas. *IJCAI'05*. 503–508. Professional Book Center.

Lifschitz, V., and Razborov, A. 2006. Why are there so many loop formulas? *ACM Trans. on Comput. Logic* 7(2):261–268. ACM.

Lin, F., and Zhao, Y. 2004. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence Journal* 157(1-2):115–137. Elsevier.

Marek, W., and Truszczynski, M. 1991a. Computing intersection of autoepistemic expansions. *LPNMR'91*. 37–50. MIT Press.

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

Marek, V. W., and Truszczynski, 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):241–273. Springer.

Nishimura, N.; Ragde, P.; and Szeider, S. 2004. Detecting backdoor sets with respect to Horn and binary clauses. *SAT'04*. 96–103. Springer.

Pfandler, A.; Rümmele, S.; Szeider, S. 2013. Backdoors to Abduction. *IJCAI'13*. To appear.

To, S.; Pontelli, E.; and Son, T. 2009. A conformant planner with explicit disjunctive representation of belief states. *ICAPS'09*. 305–312. AAAI Press.

Tseitin, G. S. 1968. Complexity of a derivation in the propositional calculus. *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR* 8:23–41. Russian.

Van Emden, M. H., and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM* 23:733–742. ACM.

Williams, R.; Gomes, C.; and Selman, B. 2003. Backdoors to typical case complexity. *IJCAI'03*. 1173–1178. Morgan Kaufmann.