

# Backdoors to Tractable Answer-Set Programming\*

Johannes Klaus Fichte and Stefan Szeider

Vienna University of Technology

Vienna, Austria

fichte@kr.tuwien.ac.at, stefan@szeider.net

## Abstract

We present a unifying approach to the efficient evaluation of propositional answer-set programs. Our approach is based on backdoors which are small sets of atoms that represent “clever reasoning shortcuts” through the search space. The concept of backdoors is widely used in the areas of propositional satisfiability and constraint satisfaction. We show how this concept can be adapted to the non-monotonic setting and how it allows to augment various known tractable subproblems, such as the evaluation of Horn and acyclic programs.

In order to use backdoors we need to find them first. We utilize recent advances in fixed-parameter algorithmics to detect small backdoors. This implies fixed-parameter tractability of the evaluation of propositional answer-set programs, parameterized by the size of backdoors. Hence backdoor size provides a structural parameter similar to the treewidth parameter previously considered. We show that backdoor size and treewidth are incomparable, hence there are instances that are hard for one and easy for the other parameter. We complement our theoretical results with first empirical results.

## 1 Introduction

*Answer-Set Programming* (ASP) is an increasingly popular framework for declarative programming [Marek and Truszczyński, 1999; Niemelä, 1999]. ASP allows to describe a problem by means of rules and constraints that form a disjunctive logic program. Solutions to the program are so-called stable models or answer sets. Many important problems of AI and reasoning can be represented and successfully solved within the ASP framework. However, the main computational problems for ASP (such as deciding whether a program has a solution, or if a certain atom is contained in at least one or in all solutions) are of high worst-case complexity and are located at the second level of the Polynomial Hierarchy [Eiter and Gottlob, 1995]. The known complexity results do

not rule out the possibility for exact algorithms that work efficiently for real-world instances by exploiting the presence of a “hidden structure.”

In this paper we follow a new approach of making the vague notion of a hidden structure precise. Our approach is based on the concept of *backdoors* which is widely used in the areas of propositional satisfiability and constraint satisfaction (see, e.g., [Williams *et al.*, 2003a; Gottlob and Szeider, 2006; Samer and Szeider, 2009]), and also for quantified Boolean formulas and argumentation [Samer and Szeider, 2009a; Ordyniak and Szeider, 2011].

A backdoor is a small set of key atoms that represent a “clever reasoning shortcut” through the search space. By deciding the status of the atoms in the backdoor, we can reduce a given program to several tractable programs belonging to a *target class* of programs. Consequently the evaluation of the given program is *fixed-parameter tractable* in the size of the backdoor, i.e., polynomial for fixed backdoor size  $k$  where the order of the polynomial is independent of  $k$  [Downey and Fellows, 1999]. By allowing backdoors of increasing size  $k = 1, 2, 3, \dots$  we can gradually augment a known tractable class of programs.

### Results:

- We show that the most important computational problems of propositional answer-set programming, including credulous/skeptical reasoning (and even counting all answer sets) are fixed-parameter tractable in the size of the backdoor.
- We show that the detection of backdoors is fixed-parameter tractable for various target classes, including the class of all Horn programs and classes based on various notions of acyclicity. This way we make recent results of fixed-parameter algorithmics accessible to the field of answer-set programming.
- We show that the concept of backdoors entails fixed-parameter tractability results for answer-set programming [Ben-Eliyahu, 1996] and so provides a unifying framework.
- We compare backdoor size with respect to various base classes with each other and with the recently studied parameter incidence treewidth [Jakl *et al.*, 2009].
- We present first empirical results where we consider the backdoor size of structured programs and random programs of varied density.

\*Research supported by ERC (COMPLEX REASON 239962).

## 2 Formal Background

We consider a universe  $U$  of propositional atoms. A literal is an atom  $a \in U$  or its negation  $\neg a$ . A disjunctive logic program (or simply a program)  $P$  is a set of rules of the form  $x_1 \vee \dots \vee x_l \leftarrow y_1, \dots, y_n, \neg z_1, \dots, \neg z_m$  where  $x_1, \dots, x_l, y_1, \dots, y_n, z_1, \dots, z_m$  are atoms and  $l, n, m$  are non-negative integers. We write  $\{x_1, \dots, x_l\} = H(r)$  (the head of  $r$ ) and  $\{y_1, \dots, y_n, z_1, \dots, z_m\} = B(r)$  (the body of  $r$ ),  $B^+(r) = \{y_1, \dots, y_n\}$  and  $B^-(r) = \{z_1, \dots, z_m\}$ . We denote the sets of atoms occurring in a rule  $r$  or in a program  $P$  by  $\text{at}(r) = H(r) \cup B(r)$  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  $|H(r)| = 1$ ,  $r$  is a constraint if  $|H(r)| = 0$ ,  $r$  is disjunction-free if  $|H(r)| \leq 1$ ,  $r$  is Horn if it is negation-free and disjunction-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 by **Horn** the classes of all Horn programs.

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  $P^M$ . We denote by  $\text{AS}(P)$  the set of all answer sets of  $P$ .

We also need some notions from propositional satisfiability. A clause is a finite set of literals, a CNF formula is a finite set of clauses. A truth assignment is a mapping  $\tau : X \rightarrow \{0, 1\}$  defined for a set  $X \subseteq U$  of atoms. For  $x \in X$  we put  $\tau(\neg x) = 1 - \tau(x)$ . By  $\text{ta}(X)$  we denote the set of all truth assignments  $\tau : X \rightarrow \{0, 1\}$ . The truth assignment reduct of a CNF formula  $F$  with respect to  $\tau \in \text{ta}(X)$  is the CNF formula  $F_\tau$  obtained from  $F$  by first removing all clauses  $c$  that contain a literal set to 1 by  $\tau$ , and second removing from the remaining clauses all literals set to 0 by  $\tau$ .  $\tau$  satisfies  $F$  if  $F_\tau = \emptyset$ , and  $F$  is satisfiable (in symbols  $\text{sat}(F)$ ) if it is satisfied by some  $\tau$ .

**ASP Problems** The main computational problems for ASP are as follows. CONSISTENCY: given a program  $P$ , does  $P$  have an answer-set? CREDULOUS/SKEPTICAL REASONING: given a program  $P$  and an atom  $a \in \text{at}(P)$ , is  $a$  contained in some/all answer-set(s) of  $P$ ? AS COUNTING: how many answer sets does  $P$  have? AS ENUMERATION: list all answer sets of  $P$ . CONSISTENCY and CREDULOUS REASONING are  $\Sigma_2^P$ -complete, SKEPTICAL REASONING is  $\Pi_2^P$ -complete [Eiter and Gottlob, 1995]. The problems remain NP (or co-NP) hard for normal programs [Marek and Truszczyński, 1991], but are polynomial-time solvable for Horn programs [Gelfond and Lifschitz, 1988]. AS COUNTING is easily seen to be  $\#P$ -hard even for normal programs in view of the  $\#P$ -completeness of  $\#\text{SAT}$ .

**Fixed-Parameter Tractability** We give some basic background on parameterized complexity. For more detailed information we refer to other sources [Downey and Fellows, 1999; Gottlob and Szeider, 2006]. A parameterized problem

$L$  is a subset of  $\Sigma^* \times \mathbb{N}$  for some finite alphabet  $\Sigma$ . 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 exist a computable function  $f$  and a constant  $c$  such that we can decide whether  $(I, k) \in L$  in time  $O(f(k) \|I\|^c)$  where  $\|I\|$  denotes the size of  $I$ . FPT is the class of all fixed-parameter tractable decision problems. The Weft Hierarchy consists of parameterized complexity classes  $\text{W}[1] \subseteq \text{W}[2] \subseteq \dots$  which are defined as the closure of certain parameterized problems under parameterized reductions. There is strong theoretical evidence that parameterized problems that are hard for classes  $\text{W}[i]$  are not fixed-parameter tractable.

## 3 Backdoors

Before we introduce the notion of backdoors to the ASP domain we review it in the domain where it originates from.

**Satisfiability Backdoors** Let  $F$  be a CNF formula and  $X$  a set of atoms. The following is obvious from the definitions:

- (\*)  $F$  is satisfiable if and only if  $F_\tau$  is satisfiable for at least one truth assignment  $\tau \in \text{ta}(X)$ .

This observation leads to the definition of a strong backdoor relative to a class  $\mathcal{C}$  of polynomially solvable CNF formulas: a set  $X$  of atoms is a strong  $\mathcal{C}$ -backdoor of a CNF formula  $F$  if  $F_\tau \in \mathcal{C}$  for all truth assignments  $\tau \in \text{ta}(X)$ . Assume that the satisfiability of formulas  $F \in \mathcal{C}$  of size  $\|F\| = n$  can be decided in time  $O(n^c)$ . Then we can decide the satisfiability of an arbitrary formula  $F$  for which we know a strong  $\mathcal{C}$ -backdoor of size  $k$  in time  $O(2^k n^c)$  which is efficient as long as  $k$  remains small.

Before we can use the strong backdoor we need to find it first. For most reasonable target classes  $\mathcal{C}$  the detection of a strong backdoor of size at most  $k$  is NP-hard if  $k$  is part of the input. However, as we are interested in finding small backdoors, it makes sense to parameterize the backdoor search by  $k$  and consider the parameterized complexity of backdoor detection. Indeed, with respect to the classes of Horn CNF formulas and 2-CNF formulas, the detection of strong backdoors of size  $\leq k$  is fixed-parameter tractable [Nishimura et al., 2004; Samer and Szeider, 2009]. For other target classes (clustering formulas and renamable Horn formulas) the detection of deletion backdoors (a subclass of strong backdoors) of size at most  $k$  is fixed-parameter tractable [Nishimura et al., 2007; Razgon and O'Sullivan, 2008].

**ASP Backdoors** In order to translate the notion of backdoors to the domain of ASP, we first need to come up with a suitable concept of a reduction with respect to a truth assignment. The following is a natural definition which generalizes a concept of Gottlob et al. [2002].

**Definition 1.** Let  $P$  be a program,  $X$  a set of atoms, and  $\tau \in \text{ta}(X)$ . The truth assignment reduct of  $P$  under  $\tau$  is the logic program  $P_\tau$  obtained from  $P$  by

1. removing all rules  $r$  with  $H(r) \cap \tau^{-1}(1) \neq \emptyset$  or  $H(r) \subseteq X$ ;
2. removing all rules  $r$  with  $B^+(r) \cap \tau^{-1}(0) \neq \emptyset$ ;
3. removing all rules  $r$  with  $B^-(r) \cap \tau^{-1}(1) \neq \emptyset$ ;
4. removing from the heads and bodies of the remaining rules all literals  $v, \neg v$  with  $v \in X$ .

**Definition 2.** Let  $\mathcal{C}$  be a class of programs. 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  $\tau \in \text{ta}(X)$ .

**Example 1.** Consider the program  $P = \{s \leftarrow w; u \leftarrow s, q; r \leftarrow w, s; t \leftarrow \neg r; q \leftarrow \neg s, u; w \leftarrow \neg r, u\}$ . The set  $X = \{r, s\}$  is a strong **Horn**-backdoor since all four truth assignments  $P_{r=0, s=0} = P_{00} = \{t \leftarrow; q \leftarrow u; w \leftarrow u\}$ ,  $P_{01} = \{u \leftarrow q; t \leftarrow; w \leftarrow u\}$ ,  $P_{10} = \{q \leftarrow u\}$ , and  $P_{11} = \{u \leftarrow q\}$  are Horn programs.

A direct equivalence similar to (\*) does not hold for ASP, even if we consider the most basic problem CONSISTENCY. Take for example the program  $P = \{x \leftarrow \neg x; y \leftarrow\}$  and the set  $X = \{x\}$ . Both reducts  $P_{x=0} = \{y\}$  and  $P_{x=1} = \{y\}$  have answer sets, but  $P$  has no answer set. However, we can show a somewhat weaker asymmetric variant of (\*), where we can map each answer set of  $P$  to an answer set of  $P_\tau$  for some  $\tau \in \text{ta}(X)$ . This is made precise by the following definition and lemma.

**Definition 3.** Let  $P$  be a program and  $X$  a set of atoms. We define  $\text{AS}(P, X) = \{M \cup \tau^{-1}(1) : \tau \in \text{ta}(X \cap \text{at}(P)), M \in \text{AS}(P_\tau)\}$ .

**Lemma 1.**  $\text{AS}(P) \subseteq \text{AS}(P, X)$  holds for every program  $P$  and every set  $X$  of atoms.

Owing to space restrictions we omit the proof which can be found in a longer version of this paper at <http://arxiv.org/abs/1104.2788>.

In view of Lemma 1, we can compute  $\text{AS}(P)$  by (i) computing  $\text{AS}(P_\tau)$  for all  $\tau \in \text{ta}(X)$  and (ii) checking for each  $M \in \text{AS}(P_\tau)$  whether it gives rise to an answer-set of  $P$ . Observe that in particular any constraint contained in  $P$  is removed in the truth assignment reduct  $P_\tau$  but is then considered in step (ii). If we can compute  $\text{AS}(P_\tau)$  in polynomial time for each  $\tau \in \text{ta}(X)$ , we have a fixed-parameter algorithm for computing  $\text{AS}(P)$  for parameter  $k = |X|$ . This consideration leads to the following definition and result.

**Definition 4.** A class  $\mathcal{C}$  of programs is enumerable if for each  $P \in \mathcal{C}$  we can compute  $\text{AS}(P)$  in polynomial time.

For any class  $\mathcal{C}$  of programs we denote by  $\mathcal{C}^*$  the class containing all programs that belong to  $\mathcal{C}$  after removal of tautological rules and constraints. It is easy to see that whenever  $\mathcal{C}$  is enumerable, then so is  $\mathcal{C}^*$ . Note that all classes considered in this paper are enumerable.

**Theorem 1.** Let  $\mathcal{C}$  be an enumerable class of programs. Problems CONSISTENCY, CREDULOUS and SKEPTICAL REASONING, AS COUNTING and AS ENUMERATION are all fixed-parameter tractable when parameterized by the size of a smallest strong  $\mathcal{C}$  backdoor, assuming that the backdoor is given as an input.

*Proof.* Let  $X$  be the given backdoor. Since we have  $|\text{AS}(P, X)| \leq 2^{|X|}$ , we can solve each listed problem by making at most  $2^{|X|}$  polynomial checks.  $\square$

If we know that each program in  $\mathcal{C}$  has at most one answer set, and  $P$  has a strong  $\mathcal{C}$ -backdoor of size  $k$ , then we can conclude that  $P$  has at most  $2^k$  answer sets. Thus, we obtain an upper bound on the number of answer sets of  $P$  by computing a small strong  $\mathcal{C}$ -backdoor of  $P$ .

**Example 2.** We consider program  $P$  of Example 1. The answer sets of  $P_\tau$  are  $\text{AS}(P_{00}) = \{\{t\}\}$ ,  $\text{AS}(P_{01}) = \{\{t\}\}$ ,  $\text{AS}(P_{10}) = \{\emptyset\}$ , and  $\text{AS}(P_{11}) = \{\emptyset\}$  for  $\tau \in \text{ta}(\{r, s\})$ .  $\text{AS}(P, X) = \{\{t\}, \{t, s\}, \{r\}, \{r, s\}\}$ , and since only  $\{t\} \in \text{AS}(P, X)$  is an answer set of  $P$ , we obtain  $\text{AS}(P) = \{\{t\}\}$ .

**Deletion Backdoors** We will see below that the following variant of strong backdoors is often useful. 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  $X$  from all the rules (heads and bodies) of  $P$ .

**Definition 5.** Let  $\mathcal{C}$  be a class of programs. A set  $X$  of atoms is a deletion  $\mathcal{C}$ -backdoor of a program  $P$  if  $P-X \in \mathcal{C}$ .

In general, not every strong  $\mathcal{C}$ -backdoor is a deletion  $\mathcal{C}$ -backdoor, and not every deletion  $\mathcal{C}$ -backdoor is a strong  $\mathcal{C}$ -backdoor. We call  $\mathcal{C}$  to be *rule induced* if for each  $P \in \mathcal{C}$ ,  $P' \subseteq P$  implies  $P' \in \mathcal{C}$ . Note that many natural classes of programs (and all classes considered in this paper) are rule induced.

**Proposition 1.** If  $\mathcal{C}$  is rule induced, then every deletion  $\mathcal{C}$ -backdoor is a strong  $\mathcal{C}$ -backdoor.

*Proof.* The statement follows from the fact that  $P_\tau \subseteq P-X$  for every  $\tau \in \text{ta}(X)$  and every program  $P$ .  $\square$

**Backdoor Detection** In order to use Theorem 1 we need to find the backdoor first. Each class  $\mathcal{C}$  of programs gives rise to the following parameterized problem: STRONG  $\mathcal{C}$ -BACKDOOR DETECTION: given a program  $P$  and an integer  $k$ , 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 (which is in some cases easier to solve).

## 4 Target Class Horn

We first consider the important case **Horn**<sup>\*</sup> as the target class for backdoors. It is well known that normal Horn programs have a unique answer set and this set can be found in linear time [Dowling and Gallier, 1984], hence **Horn** and **Horn**<sup>\*</sup> are enumerable. The following lemma shows that **Horn**<sup>\*</sup> is particularly well suited as a target class.

**Lemma 2.** A set  $X$  is a strong **Horn**<sup>\*</sup>-backdoor of a program  $P$  if and only if it is a deletion **Horn**<sup>\*</sup>-backdoor of  $P$ .

Owing to space restrictions we omit the proof.

**Theorem 2.** STRONG **Horn**<sup>\*</sup>-BACKDOOR DETECTION is fixed-parameter tractable in  $k$ . In fact, given a program  $P$  with  $n$  atoms we can find a strong **Horn**<sup>\*</sup>-backdoor of size  $\leq k$  in time  $O(1.2738^k + kn)$  or decide that no such backdoor exists.

*Proof.* (Sketch) Let  $G$  be the undirected graph defined on the set of atoms of the given program  $P$ , where two atoms  $x, y$  are joined by an edge if and only if  $P$  contains a non-tautological rule  $r$  with  $x, y \in H(r)$  or  $x \in H(r)$  and  $y \in B^-(r)$ . Now it is easy to see that a set  $X \subseteq \text{at}(P)$  is a vertex cover of  $G$  if and only if  $X$  is a deletion **Horn**<sup>\*</sup>-backdoor of  $P$ . A vertex cover of size  $\leq k$ , if it exists, can be found in time  $O(1.2738^k + kn)$  [Chen *et al.*, 2006]. The theorem follows by Lemma 2.  $\square$

For instance, the undirected graph  $G$  of the program  $P$  of Example 1 consists of the two paths  $(w, r, t)$  and  $(s, q)$ . Then  $\{r, s\}$  is a vertex cover of  $G$ . We observe easily that there exists no vertex cover of size 1. Thus  $\{r, s\}$  is a smallest strong **Horn\***-backdoor of  $P$ .

Ben-Eliyahu [1996] showed that evaluation of normal logic programs is fixed-parameter tractable in (i) the number of atoms that appear in negative rule bodies, and (ii) the total number of non-Horn rules. It is not difficult to see that both numbers are greater or equal to the size of a smallest strong **Horn\***-backdoor, and so entailed by our approach.

## 5 Acyclicity-Based Target Classes

There are two causes for a program to have a large number of answer sets: (i) disjunctions in the heads of rules, and (ii) certain cyclic dependencies between rules. Disallowing both causes yields so-called *stratified* programs [Gelfond and Lifschitz, 1988]. In the following we will study backdoor detection for various classes of stratified programs. We define the classes by requiring normality and acyclicity (the absence of certain types cycles). In order to define acyclicity we associate with each normal program  $P$  its *directed dependency graph*  $D_P$  [Apt et al., 1988], and its *undirected dependency graph*  $U_P$  [Gottlob et al., 2002].  $D_P$  has as vertices the atoms of  $P$  and a directed edge  $(x, y)$  between any two atoms  $x, y$  for which there is a rule  $r \in P$  with  $x \in H(r)$  and  $y \in B(r)$ ; if there is a rule  $r \in P$  with  $x \in H(r)$  and  $y \in B^-(r)$ , then the edge  $(x, y)$  is called a *negative edge*.  $U_P$  is obtained from  $D_P$  by replacing each negative edge  $e = (x, y)$  with two undirected edges  $\{x, v_e\}, \{v_e, y\}$  where  $v_e$  is a new *negative vertex*, and by replacing each remaining directed edge  $(u, v)$  with an undirected edge  $\{u, v\}$ . By a *directed cycle of  $P$*  we mean a directed cycle in  $D_P$ , by an *undirected cycle of  $P$*  we mean an undirected cycle in  $U_P$ . Figure 1 visualizes  $D_P$  and  $U_P$  of the program  $P$  of Example 1. A directed (undirected) cycle is *bad* if it contains a negative edge (a negative vertex), otherwise it is *good*. Various classes of programs arise by requiring the programs to have no directed bad cycles (**DBC-Acyc**), no undirected bad cycles (**BC-Acyc**), no directed cycles (**DC-Acyc**), and no undirected cycles (**C-Acyc**). **DBC-Acyc**, the largest class among the considered classes, is exactly the class **Strat** of stratified programs [Apt et al., 1988].

For instance in the program  $P$  of Example 1,  $(r, v(w,r), w, r)$  is an undirected cycle,  $(u, q, u)$  is a directed cycle,  $(s, v(q,s), q, u, s)$  is an undirected bad cycle, and  $(w, r, w)$  is a directed bad cycle (see Figure 1).

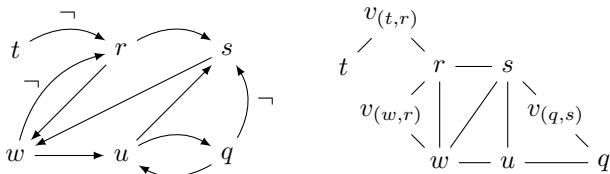


Figure 1: Directed dependency graph  $D_P$  (left) and undirected dependency graph  $U_P$  (right) of the program  $P$  of Example 1.

In order to compare the size of backdoors for the various classes, we need to compare the classes themselves (ev-

idently, if  $\mathcal{C} \subseteq \mathcal{C}'$ , then every strong (deletion)  $\mathcal{C}'$ -backdoor is also a strong (deletion)  $\mathcal{C}$ -backdoor, but not necessarily the other way around). By definition we have **DC-Acyc**  $\subseteq$  **DBC-Acyc** and **C-Acyc**  $\subseteq$  **BC-Acyc**  $\subseteq$  **DC-Acyc**; it is easy to see that the inclusions are proper. However, contrary to what one expects, **C-Acyc**  $\not\subseteq$  **DC-Acyc**, which can be seen by considering the program  $P_0 = \{x \leftarrow y, y \leftarrow x\}$ , hence **C-Acyc** and **DC-Acyc** are incomparable. Requiring that a program has no directed cycles but may have directed good cycles of length 2 (as in  $P_0$ ) gives rise to the class **DC2-Acyc**, which generalizes both classes **C-Acyc** and **DC-Acyc**. The diagram in Figure 2 shows the relationship between the various program classes.

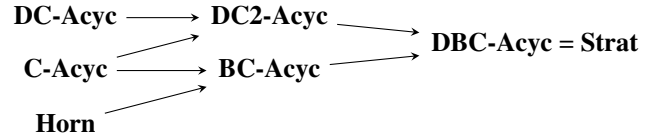


Figure 2: Relationship between classes of programs. An arrow from  $\mathcal{C}$  to  $\mathcal{C}'$  indicates that  $\mathcal{C}$  is a proper subset of  $\mathcal{C}'$ . If there is no arrow between two classes (or the arrow does not follow by transitivity of set inclusion), then the two classes are incomparable.

**Theorem 3.** *For each class  $\mathcal{C} \in \{\mathbf{C-Acyc}, \mathbf{BC-Acyc}, \mathbf{DC-Acyc}, \mathbf{DC2-Acyc}, \mathbf{Strat}\}$  the problem STRONG  $\mathcal{C}$ -BACKDOOR DETECTION is W[2]-hard and therefore unlikely to be fixed-parameter tractable.*

*Proof.* (Sketch) We give a reduction from the W[2]-complete problem HITTING SET [Downey and Fellows, 1999]. An instance of this problem is a pair  $(S, k)$  where  $S = \{S_1, \dots, S_m\}$  is a family of sets and  $k$  is an integer (the parameter). The question is whether there exists a set  $H$  of size at most  $k$  which intersects with all the  $S_i$ ; such  $H$  is a hitting set. We construct a program  $P$  as follows. As atoms we take the elements of  $X = \bigcup_{i=1}^m S_i$  and new atoms  $a_i^j$  and  $b_i^j$  for  $1 \leq i \leq m$ ,  $1 \leq j \leq k + 1$ . For each  $1 \leq i \leq m$  and  $1 \leq j \leq k + 1$  we take two rules  $r_i^j, s_i^j$  where  $H(r_i^j) = \{a_i^j\}$ ,  $B^-(r_i^j) = S_i$ ,  $B^+(r_i^j) = S_i \cup \{b_i^j\}$ ;  $H(s_i^j) = \{b_i^j\}$ ,  $B^-(s_i^j) = \{a_i^j\}$ ,  $B^+(s_i^j) = \emptyset$ . The result now follows by showing that  $S$  has a hitting set of size  $\leq k$  if and only if  $P$  has a strong  $\mathcal{C}$ -backdoor of size  $\leq k$  where  $\mathcal{C}$  is any of the classes mentioned.  $\square$

For **DC-Acyc**, **DC2-Acyc**, and **Strat** we can avoid the use of tautological rules in the reduction and so strengthen Theorem 3 (it would be interesting to know if this is also possible for the remaining two classes mentioned in Theorem 3).

The W[2]-hardness results suggest to relax the considered problems and to look for *deletion* backdoors: Which of the classes mentioned in Theorem 3 admit fixed-parameter tractable detection of deletion backdoors? Using very recent results from fixed-parameter algorithmics we can answer this question positively for all considered classes except for **Strat** whose complexity remains open.

**Theorem 4.** *For each class  $\mathcal{C} \in \{\mathbf{C-Acyc}, \mathbf{BC-Acyc}, \mathbf{DC-Acyc}, \mathbf{DC2-Acyc}\}$  the problem DELETION  $\mathcal{C}^*$ -BACKDOOR DETECTION is fixed-parameter tractable.*

*Proof.* (1) DELETION **C-Acyc**-BACKDOOR DETECTION can be solved by solving the FEEDBACK VERTEX SET (FVS) problem for  $U_p$ , which is well-known to be FPT [Downey and Fellows, 1999], this was already observed by Gottlob *et al.* [2002]. (2) DELETION **DC-Acyc**-BACKDOOR DETECTION is equivalent to the DIRECTED FVS problem on  $D_p$ . The parameterized complexity of DIRECTED FVS remained open for many years and was recently shown FPT by Chen *et al.* [2008] with a break-through result. (3) DELETION **BC-Acyc**-BACKDOOR DETECTION can be solved by solving the EDGE SUBSET FVS problem, where only cycles need to be covered that contain an edge from a given set  $S$ . This problem was recently shown FPT by Cygan *et al.* [2010] and Kawarabayashi and Kobayashi [2010]. (4) Finally, the problem DELETION **DC2-Acyc**-BACKDOOR DETECTION can be solved by finding a feedback vertex set in a *mixed graph* (a graph containing directed and undirected edges), as we can replace a good cycle on two edges by one undirected edge. FVS FOR MIXED GRAPHS was recently shown FPT by Bon-sma and Lokshtanov [2010].  $\square$

The classes mentioned in Theorem 3 are rule-induced. Hence we can use this theorem to strengthen the fixed-parameter tractability result of Theorem 1 by dropping the assumption that the backdoor is given.

The considered classes can be generalized by taking the parity of the number of negative edges on bad cycles into account. In recent research Fichte [2011] generalized the tractability results of Lin and Zhao [2004] by considering backdoors with respect to such parity classes.

## 6 Theoretical Comparison of Parameters

In this section we compare ASP parameters in terms of their generality.

Jakl *et al.* [2009] applied the graph parameter treewidth to ASP and showed that the main reasoning problems for ASP are fixed-parameter tractable by the treewidth of the *incidence graph* of the program. The incidence graph of a program  $P$  is the bipartite graph on the rules and atoms of  $P$ , where a rule and an atom are joined by an edge if and only if the atom occurs in the rule. It turns out that incidence treewidth is incomparable with backdoor size for various target classes considered.

**Theorem 5.** *Let  $\mathcal{C} \in \{\text{Horn, C-Acyc, DC-Acyc, DC2-Acyc, BC-Acyc, Strat}\}$ . There are programs whose incidence graphs have constant treewidth but require arbitrarily large strong and deletion  $\mathcal{C}$ -backdoors, and there are programs where the converse prevails.*

*Proof.* (Sketch) Clearly each of the considered classes contains programs of arbitrary high treewidth of the incidence graph. Conversely, we consider a program  $P \notin \mathcal{C}$ . We denote by  $nP$  the program consisting of the union of  $n$  atom-disjoint copies of  $P$ . By basic properties of treewidth it follows that the treewidth of the incidence graph of  $P$  equals to that of  $nP$ , however, smallest  $\mathcal{C}$ -backdoors are of size  $\geq n$ .  $\square$

The treewidth approach is based on dynamic programming which is of high space complexity and therefore only practical for instances of treewidth below 10 [Jakl *et al.*, 2009]. The

instance set	vars	bd (%)	stdev
Daimler-Chrysler-MT	1785.14	21.46	1.56
Daimler-Chrysler-NC	1793.0	22.94	2.92
Daimler-Chrysler-RZ	1562.5	11.53	2.21
Daimler-Chrysler-SZ	1567.53	13.97	3.47
Daimler-Chrysler-UC	1781.74	21.4	2.01
Daimler-Chrysler-UT	1781.23	23.53	3.95
Mutex	6449.0	49.94	0.09
RLP-3	150.0	58.28	1.37
RLP-4	150.0	64.53	0.92
RLP-5	150.0	68.4	0.97
RLP-6	150.0	70.9	0.82
RLP-7	150.0	73.68	0.87
RLP-8	150.0	75.54	0.74
RG-40	40.0	93.5	1.24
RG-50	50.0	94.05	0.96
RG-60	60.0	94.38	0.82

Table 1: Size of smallest strong **Horn**-backdoors (bd) for various benchmark sets, given as % of the total number of variables (vars) by the mean over the instances. Daimler-Chrysler-(test): 554 Real-world instances encoding logistics problems from car configurations. The disjunctive programs have been compiled from SAT instances provided by Sinz *et al.* [2003] grouped by the kind of consistency test. The instances are produced using the simple encoding where a clause  $\{a, b, \neg c, \neg d\}$  becomes the rule  $a, b \leftarrow c, d$ . Mutex: Disjunctive programs that encode the equivalence test of partial implementations of circuits, provided by Maratea *et al.* [2008] based on QBF instances of Ayari and Basin [2000] RLP- $\langle\rho\rangle$ : Randomly generated normal programs provided by Zhao and Lin [2003] of various density  $\rho$  (number of rules divided by the number of variables) with 10 instances per step. RG- $\langle n\rangle$ : Randomly generated instances provided by Gebser [Asp, 2009] with  $n = 40, 50$ , and 60 variables, respectively with 40 instances per step.

backdoor approach is more space efficient since for each partial truth assignment  $\tau \in \text{ta}(X)$  of a backdoor  $X$ , the computations of  $\text{AS}(P_\tau)$  and the corresponding elements of  $\text{AS}(P_\tau)$  can be carried out independently.

One might ask whether it makes sense to consider restrictions on the treewidth of the undirected dependency graph, defined above. However, this restriction does not yield tractability, as the reduction of [Eiter and Gottlob, 1995] produces programs with undirected dependency graphs of treewidth 2.

## 7 Empirical Comparison of Parameters

We have determined the size of smallest backdoors for various programs, including *structured* real-world instances and *random* instances. The results are summarized in Table 1. It is known that so-called tight programs are closely related to SAT [Lin and Zhao, 2003]. The QueensEqTest instances and two of the Daimler-Chrysler instances are tight, all other instances considered are not tight. For pragmatic reasons we have used **Horn\*** as the target class as smallest backdoors are easy to compute, even for large inputs. Our experimental results indicate that structured instances have smaller backdoors than random instances. It also seems that random instances with higher density have larger backdoors.

We have conducted a second series of experiments on random instances where we have analyzed how much we gain by considering the more general acyclicity-based tar-

get classes instead of **Horn**. It appears that smallest deletion **Horn**\*-backdoors are indeed significantly larger than deletion backdoors for acyclicity-based target classes. The distinction between directed and undirected cycles seems to have a significant effect on the backdoor size, whereas the distinction between good and bad cycles seems to be less significant. However these results are not fully conclusive as the considered programs were rather small.

## 8 Conclusion

We have introduced the backdoor approach to the domain of propositional answer-set programming. The backdoor approach allows to augment a known tractable class and makes the efficient solving methods for the tractable class generally applicable. Our approach makes recent results in fixed-parameter algorithmics applicable to nonmonotonic reasoning. The comparison results show that the parameters based on backdoor size are incomparable with treewidth and therefore provide fixed-parameter tractability for programs that are hard for the treewidth approach.

The results and concepts of this paper give rise to several interesting research questions. For instance, it would be interesting to consider backdoors for target classes that contain programs with an exponential number of answer-sets, but where the set of all answer-sets can be succinctly represented. A simple example is the class of programs that consist of independent components of bounded size. Other questions are concerned with alternative ways of using backdoors. For instance, by means of “backdoor trees” [Samer and Szeider, 2008] one can avoid the consideration of all  $2^k$  partial assignments of the backdoor and thus make the backdoor approach feasible for programs with larger backdoors. A further use of backdoors that seems worth exploring is the control of heuristics of ASP solvers.

## References

- [Apt *et al.*, 1988] K. R. Apt, H. A. Blair, and A. Walker. *Towards a theory of declarative knowledge*, Morgan Kaufmann, 1988.
- [Asp, 2009] Asparagus. <http://asparagus.cs.uni-potsdam.de>, 2009.
- [Ayari and Basin, 2000] A. Ayari and D. Basin. Bounded model construction for monadic second-order logics. *CAV 2000*.
- [Ben-Eliyahu, 1996] R. Ben-Eliyahu. A hierarchy of tractable subsets for computing stable models. *J. Artif. Intell. Res.*, 5, 1996.
- [Bonsma and Lokshtanov, 2010] P. Bonsma and D. Lokshtanov. Feedback vertex set in mixed graphs. arXiv 1010.5974, 2010.
- [Chen *et al.*, 2006] J. Chen, I. A. Kanj, and G. Xia. Improved parameterized upper bounds for vertex cover. *MFCS 2006*.
- [Chen *et al.*, 2008] J. Chen, Y. Liu, S. Lu, B. O’Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5), 2008.
- [Cygan *et al.*, 2010] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. Subset feedback vertex set is fixed parameter tractable. arXiv 1004.2972, 2010.
- [Dowling and Gallier, 1984] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Logic Programming*, 1(3), 1984.
- [Downey and Fellows, 1999] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
- [Eiter and Gottlob, 1995] T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: propositional case. *Ann. Math. Artif. Intell.*, 15(3-4), 1995.
- [Fichte, 2011] J. K. Fichte. The good, the bad, the odd – cycles in answer set programming. *submitted*, 2011.
- [Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. *ICLP 1988*.
- [Gelfond and Lifschitz, 1991] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4), 1991.
- [Gottlob and Szeider, 2006] G. Gottlob and S. Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *The Computer Journal*, 51(3), 2006.
- [Gottlob *et al.*, 2002] G. Gottlob, F. Scarcello, and M. Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, 138(1-2), 2002.
- [Jakl *et al.*, 2009] M. Jakl, R. Pichler, and S. Woltran. Answer-set programming with bounded treewidth. *IJCAI 2009*.
- [Janhunen and Oikarinen, 2002] T. Janhunen and E. Oikarinen. Testing the equivalence of logic programs under stable model semantics. *Logics in AI*, 2002.
- [Kawarabayashi and Kobayashi, 2010] K. Kawarabayashi and Y. Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. Technical report, University of Tokyo, Japan, 2010.
- [Lin and Zhao, 2003] F. Lin and J. Zhao. On tight logic programs and yet another translation from normal logic programs to propositional logic. *IJCAI 2003*.
- [Lin and Zhao, 2004] F. Lin and X. Zhao. On odd and even cycles in normal logic programs. *AAAI 2004*.
- [Maratea *et al.*, 2008] M. Maratea, F. Ricca, W. Faber, and N. Leone. Look-back techniques and heuristics in DLV: Implementation, evaluation, and comparison to QBF solvers. *J. Algorithms*, 63, 2008.
- [Marek and Truszczyński, 1991] V. W. Marek and M. Truszczyński. Autoepistemic logic. *J. ACM*, 38(3), 1991.
- [Marek and Truszczyński, 1999] V. W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. *The Logic Programming Paradigm: a 25-Year Perspective*, 1999.
- [Niemelä, 1999] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. Artif. Intell.*, 25(3-4), 1999.
- [Nishimura *et al.*, 2004] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. *SAT 2004*.
- [Nishimura *et al.*, 2007] N. Nishimura, P. Ragde, and S. Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8), 2007.
- [Ordyniak and Szeider, 2011] S. Ordyniak and S. Szeider. Augmenting Tractable Fragments of Abstract Argumentation. *IJCAI 2011*.
- [Razgon and O’Sullivan, 2008] I. Razgon and B. O’Sullivan. Almost 2-sat is fixed-parameter tractable (extended abstract). *ICALP 2008*.
- [Samer and Szeider, 2008] M. Samer and S. Szeider. Backdoor trees. *AAAI 2008*.
- [Samer and Szeider, 2009] M. Samer and S. Szeider. Fixed-parameter tractability. *Handbook of Satisfiability*, ch. 13, 2009.
- [Samer and Szeider, 2009a] M. Samer and S. Szeider. Backdoor sets of quantified Boolean formulas. *JAR*, 42(1):77–97, 2009.
- [Sinz *et al.*, 2003] C. Sinz, A. Kaiser, and W. Küchlin. Formal methods for the validation of automotive product configuration data. *AI EDAM 2003*.
- [Williams *et al.*, 2003a] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. *IJCAI 2003*.
- [Zhao and Lin, 2003] Y. Zhao and F. Lin. Answer set programming phase transition: A study on randomly generated programs. *ICLP 2003*.