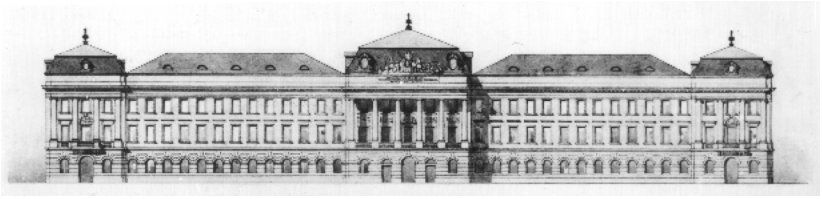
dbai

TECHNICAL REPORT



INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG DATENBANKEN UND ARTIFICIAL INTELLIGENCE

Backdoor Trees for Answer Set Programming

DBAI-TR-2016-98

Johannes K. Fichte and Stefan Szeider

DBAI TECHNICAL REPORT

2016

Institut für Informationssysteme
Abteilung Datenbanken und
Artificial Intelligence
Technische Universität Wien
Favoritenstr. 9
A-1040 Vienna, Austria
Tel: +43-1-58801-18403
Fax: +43-1-58801-18493
sek@dbai.tuwien.ac.at
www.dbai.tuwien.ac.at

TU WIEN
TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DBAI TECHNICAL REPORT
DBAI TECHNICAL REPORT DBAI-TR-2016-98, 2016

Backdoor Trees for Answer Set Programming

Johannes K. Fichte[1] **Stefan Szeider**[2]

Abstract. *Answer set programming* (ASP) is a popular framework for declarative modelling and problem solving. It has successfully been used to solve a wide variety of problems in artificial intelligence and reasoning. Many problems in propositional disjunctive ASP are of high computational complexity, such as reasoning, counting, and enumeration; in particular, the reasoning problems are complete for the second level of the Polynomial Hierarchy and thus even "harder than NP".
In this paper, we introduce backdoor trees for ASP and present a parameterized complexity analysis that takes the input size of an instance along with a composed parameter, which is based on backdoor trees, into account. When using backdoors for a parameterized complexity analysis one only considers the size k of a backdoor as a parameter. Evaluating a given backdoor results in 2^k assignments and thus 2^k programs the assignments. However, an assignment to fewer than k atoms can already yield a program under assignment that belongs to the fixed target class. Therefore, we consider binary decision trees, which make gradually assigning truth values to backdoor atoms in a program precise and lead us to the notion of backdoor trees, originally defined for propositional satisfiability. In this way, backdoor trees provide a more precise approach to the evaluation of backdoors, where we take the interaction of the assignments in the evaluation into account.

[1]Technische Universität Wien, Institute für Informationssystem (E184/2), Arbeitsgruppe Datenbanken und Künstliche Intelligenz, Favoritenstrae 9-11, 1040 Wien, Austria. E-mail: fichte@dbai.tuwien.ac.at
[2]Technische Universität Wien, Institut fr Computergraphik und Algorithmen (E186/1), Arbeitsgruppe Algroithmen und Komplexität, Favoritenstrae 9-11, 1040 Wien, Austria. E-mail: sz@ac.tuwien.ac.at

Acknowledgements: The work has been supported by the Austrian Science Fund (FWF), Grant Y698. The first author is also affiliated with the Institute of Computer Science and Computational Science at University of Potsdam, Germany.

Contents

1 Introduction

Answer set programming (ASP) is a popular framework for declarative modelling and problem solving [29, 31, 21]. It has successfully been used to solve a wide variety of problems in artificial intelligence and reasoning, e.g., match making [19], optimization of packaging of Linux distributions [20], reasoning in robots [4], shift design [1]. In ASP, problems are usually modelled by means of rules and constraints that form a logic program. The solutions to the program are the so-called answer sets (or stable models). Solving a problem means to search for answer sets of logic programs. In this paper, we are mainly interested in computational decision problems for propositional disjunctive ASP such as deciding whether a program has a solution (CONSISTENCY), or whether a certain atom is contained in at least one (BRAVE REASONING) or in all solutions (SKEPTICAL REASONING). Further, we consider a counting (COUNTING) and an enumeration problem (ENUM).

Developers of modern solvers such as Clasp [22] or Wasp [3] have demonstrated in several competitions [10, 9, 2, 23, 24] that ASP solving can be efficiently used to solve a wide variety of instances. However from the perspective of classical worst case complexity, many decision problems for disjunctive ASP are "harder than NP" and have a higher worst-case complexity than CSP and SAT. More precisely, the problems CONSISTENCY, BRAVE REASONING, and SKEPTICAL REASONING are complete for the second level of the Polynomial Hierarchy [13].

In the literature, more fine-grainted results on computational complexity of the ASP decision problems have been established. Syntactic properties where the input is restricted to certain fragments have been identified under which the computational complexity drops and where the problems can be solved more efficiently [25, 5, 6, 34, 16]. Parameterized complexity analyses, which take the input size of an instance along with a certain "hidden structure" (parameter), have been carried out [28, 7, 15, 14]. The central idea of parameterized complexity is that instances originating in practical applications are often structured in a way that facilitates obtaining a solution relatively fast. An interesting parameter for answer set programming are backdoors, which can be used as clever reasoning shortcuts through the search space. For a backdoor one usually fixes a class of programs, commonly called target class, where the problem under consideration is computationally easier. Then, a *strong backdoor* into a certain target class $\mathcal{C}$ is a (preferably small) set of atoms for which the given program with respect to any assignment $\tau \in 2^X$ belongs to $\mathcal{C}$. Exploiting backdoors usually consists of two steps (i) finding a backdoor of the given instance (backdoor detection) and (ii) applying the backdoor to the instance and determining a candidate solution and verifying its minimality (backdoor evaluation). The computational blowup is confined to the size of the backdoor in both steps. Hence, the size of the backdoor can be seen as a distance measure that indicates how far the instance is from the target class.

In this paper, we consider backdoor trees, which provide a more precise approach to the evaluation of strong backdoors, where we take the interaction of the assignments in the evaluation into account. When using backdoors for a parameterized complexity analysis one only considers the size k of a backdoor as a parameter. Evaluating a given backdoor results in 2^k assignments and thus 2^k programs with respect the possible assignments. However, an assignment to fewer than k atoms can already yield a program, with respect to a considered assignment, that belongs to the

fixed target class. Therefore, we consider binary decision trees, which make gradually assigning truth values to backdoor atoms in a program precise and lead us to the notion of backdoor trees. We investigate under which conditions (i) we need to consider significantly fewer than 2^k assignment reducts and (ii) we can significantly improve parts of the backdoor evaluation (minimality check) if those assignments set only a small number of atoms to true.

Our main contributions are as follows:

1. We define backdoor trees for answer set programming, extend the concept of backdoors from sets to trees (with similar steps detection and evaluation), and establish that the reducts that we obtain from a backdoor tree are sufficient to find all answer sets.

2. We show that backdoor tree evaluation is fixed-parameter tractable when parameterized by a composed parameter, which incorporates considerations of (i) and (ii) from above of a given backdoor tree.

3. We establish fixed-parameter tractability for backdoor tree detection for backdoor trees into the target class **Horn**.

Related Work Backdoor trees have been introduced by Samer and Szeider [32] in the context of propositional satisfiability. Backdoor trees provide a more refined concept of backdoor evaluation and take the interaction of variables that form a backdoor into account. The propositional satisfiability problem can be solved by means of a backdoor tree and is fixed parameter tractable when parameterized by the number of leaves in a backdoor tree. The problems of detecting backdoor trees into 2CNF and Horn formulas are fixed parameter tractable. Gallo-Scutellà [18] have considered generalized classes $(\Gamma_1, \Gamma_2, \ldots)$ of propositional Horn formulas by investigating the maximal number of positive literals that have be set to true to obtain a propositional Horn formula. The notion provides a parameter that measures in a certain sense the distance of a propositional formula from being Horn. The problem to decide whether a propositional formula belongs to a class Γ_k (parameter detection) has been shown to be non-uniform polynomial-time solvable where the order of the polynomial depends on the parameter (XP). However, it is an open research question whether it is also uniform polynomial-time solvable (FPT).

2 Preliminaries

Answer Set Programming

We consider a universe U of propositional *atoms*. A *literal* is an atom $a \in U$ or its negation $\bar{a}$. A *disjunctive logic program* (or simply a *program*) P is a set of *rules* of the form $a_1 \vee \ldots \vee a_l \leftarrow b_1, \ldots, b_n, \bar{c}_1, \ldots, \bar{c}_m$ where $a_1, \ldots, a_l, b_1, \ldots, b_n, c_1, \ldots, c_m$ are atoms and l, n, m are non-negative integers. We write $H(r) = \{a_1, \ldots, a_l\}$ (the *head* of r), $B^+(r) = \{b_1, \ldots, b_n\}$ (the *positive body* of r), and $B^-(r) = \{c_1, \ldots, c_m\}$ (the *negative body* of r). We denote the sets of atoms occurring in a rule r or in a program P by $\mathrm{at}(r) = H(r) \cup B^+(r) \cup B^-(r)$ and $\mathrm{at}(P) = \bigcup_{r \in P} \mathrm{at}(r)$, respectively. We denote the number of rules of P by $|P| = |\{\, r : r \in P \,\}|$. The *size* $\|P\|$ of a program P is

defined as $\sum_{r \in P} |H(r)| + |B^+(r)| + |B^-(r)|$. A rule r is is *normal* if $|H(r)| \leq 1$, r is a *constraint* (integrity rule) if $|H(r)| = 0$, r is *Horn* if it is positive and normal or a constraint. We say that a program has a certain property if all its rules have the property. **Horn** refers to the class of all Horn programs. We denote the class of all normal programs by **Normal**. Let P and P' be programs. We say that P' is a *subprogram* of P (in symbols $P' \subseteq P$) if for each rule $r' \in P'$ there is some rule $r \in P$ with $H(r') \subseteq H(r)$, $B^+(r') \subseteq B^+(r)$, $B^-(r') \subseteq B^-(r)$. Let $\mathcal{C}$ be a class of programs. We call a class $\mathcal{C}$ of programs *hereditary* if for each $P \in \mathcal{C}$ all subprograms of P are in $\mathcal{C}$ as well.

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 *Gelfond-Lifschitz (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 then removing all $\bar{z}$ where $z \in B^-(r)$ from the remaining rules r [26]. M is an *answer set* (or *stable model*) of a program P if M is a minimal model of P^M. We denote by $\mathrm{AS}(P)$ the set of all answer sets of P. A class $\mathcal{C}$ of programs is *enumerable* if for each $P \in \mathcal{C}$ we can compute $\mathrm{AS}(P)$ in polynomial time.

In this paper, we consider the following fundamental ASP problems:

Problem: CONSISTENCY
Input: Program P.
Question: Decide whether P has an answer set.

Problem: BRAVE REASONING
Input: Program P.
Question: Decide whether a belongs to *some* answer set of P.

Problem: SKEPTICAL REASONING
Input: Program P.
Question: Decide whether a belongs to *all* answer sets of P.

Problem: COUNTING
Input: Program P.
Question: Compute the number of answer sets of P.

Problem: ENUM
Input: Program P.
Question: List all answer sets of P.

We denote by $\mathcal{A}sp\mathcal{F}ull$ the family of all problems CONSISTENCY, BRAVE REASONING, SKEPTICAL REASONING, COUNTING, and ENUM.

Parameterized Complexity

We briefly give some background on parameterized complexity. For more detailed information we refer to other sources [11, 12, 17, 27, 30]. An instance of a *parameterized problem* L is a

pair $(I, k) \in \Sigma^* \times \mathbb{N}$ 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*. $\|I\|$ denotes the size of I. L is *fixed-parameter tractable* if there exist a computable function f and a constant c such that we can decide by an algorithm whether $(I, k) \in L$ in time $\mathcal{O}(f(k)\|I\|^c)$. Such an algorithm is called an *fpt-algorithm*. FPT is the class of all fixed-parameter tractable decision problems.

Backdoors of Answer Set Programs

In the following, we briefly summarize the concept of ASP backdoors by Fichte and Szeider [15]. A *(truth) assignment* is a mapping $\tau : X \to \{0, 1\}$ defined for a set $X \subseteq U$ of atoms. For $x \in X$, we define $\tau(\bar{x}) = 1 - \tau(x)$. By 2^X we denote the set of all assignments $\tau : X \to \{0, 1\}$. By $\tau^{-1}(b)$ we denote the preimage $\tau^{-1}(b) := \{\, a : a \in X, \tau(a) = b \,\}$ of the assignment τ for some truth value $b \in \{0, 1\}$.

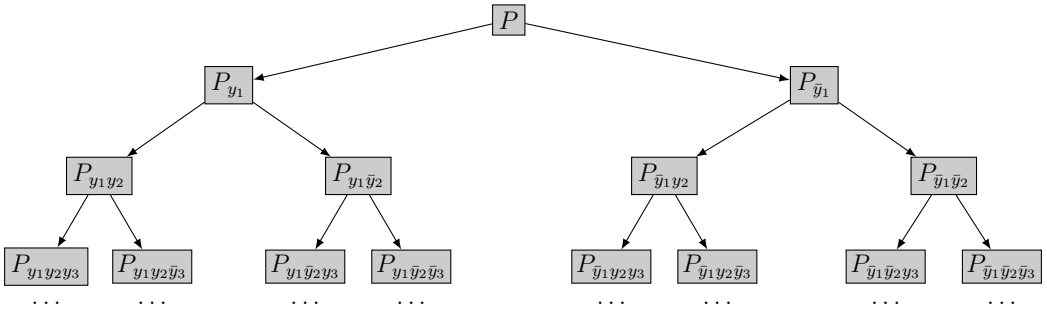
Definition 1 (Strong $\mathcal{C}$-Backdoor [15])**.** *Let P be a program, X a set of atoms, and $\tau \in 2^X$ an assignment. The* reduct *of P under τ is the logic program P_τ obtained from P by (i) removing all rules r with $H(r) \cap \tau^{-1}(1) \neq \emptyset$ or $H(r) \subseteq X$; (ii) removing all rules r with $B^+(r) \cap \tau^{-1}(0) \neq \emptyset$; (iii) removing all rules r with $B^-(r) \cap \tau^{-1}(1) \neq \emptyset$; (iv) removing from the heads and bodies of the remaining rules all literals $a, \bar{a}$ with $a \in X$. 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 assignments $\tau \in 2^X$.*

By a *minimal* strong $\mathcal{C}$-backdoor of a program P we mean a strong $\mathcal{C}$-backdoor of P that does not properly contain a smaller strong $\mathcal{C}$-backdoor of P; a *smallest* strong $\mathcal{C}$-backdoor of P is one of smallest cardinality.

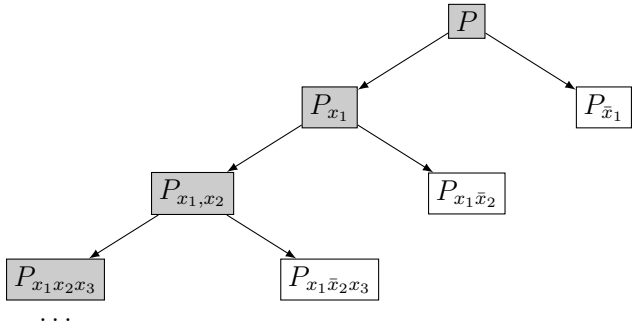
The a result by Fichte and Szeider [15] states that all ASP problems that are of interest in this paper are fixed-parameter tractable when parameterized by the size of a given strong $\mathcal{C}$-backdoor for an enumerable target class $\mathcal{C} \subseteq$ **Normal** and that finding a strong backdoor is also fixed-parameter tractable for various target classes.

3 Backdoor Trees of Answer Set Programs

When we exploit a backdoor X of a program P to find answer sets according to a backdoor-based approach [15], the exponential blowup of the running time depends only on the size of the backdoor X. Thus, the focus when considering backdoors for a complexity analysis is to efficient algorithms to find smallest backdoors into certain target classes. The actual algorithm then consists of two steps: (i) computing the answer sets of the program P under the assignment τ for all $\tau \in 2^X$, which produces candidates for answer sets of P, and (ii) checking for each $M \in \text{AS}(P, X)$ whether $M \in \text{AS}(P, X)$ is a minimal model of P^M. In Step (i) we determine for each $\tau \in 2^X$ the answer sets of the reducts P_τ and then check whether these answer sets give rise to an answer set of P. Consequently, we have to consider all the $2^{|X|}$ assignments in the worst case. However, there are answer set programs where we can find a backdoor X for which we do not need all $2^{|X|}$ assignments, as "shorter" assignments already yield a reduct that belongs to the considered target class. More



(a) Constructed from the strong **Horn**-backdoor Y



(b) Constructed from the strong **Horn**-backdoor X

Figure 1 Illustration of reducts of the program P and the strong **Horn**-backdoors X and Y from Example 1. A gray colored note indicates that the respective program does not belong to **Horn**. A white colored note indicates that the respective program belongs to **Horn**.

formally, there is an assignment τ' such that $\tau'^{-1} \subsetneq \tau^{-1}$ for some $\tau \in 2^X$ and the reduct $P_{\tau'}$ already belongs to the target class $\mathcal{C}$. Hence, when we gradually assign truth values to atoms (similar to binary search) instead of taking an assignment $\tau \in 2^X$, some atoms in τ can be irrelevant for the question whether the reduct belongs to $\mathcal{C}$.

Interestingly, in some cases it is more effective to use a backdoor that is not a smallest backdoor into the target class $\mathcal{C}$. We show this in the following example.

Example 1. *Let n be some integer. Consider the following program:*

$$
\begin{aligned}
P := {} & \{\, y_0 \vee x_0 \leftarrow x_1, \ldots, x_{2n} \,\} \cup \\
& \{\, y_j \vee x_i \leftarrow x_0 \ldots, x_{i-1}, x_{i+1}, \ldots, x_{2n-1} : j = (i \bmod n), 1 \leq i < 2n \,\} \cup \\
& \{\, x_0 \leftarrow x_1, \ldots, x_{2n-1}, \bar{y}_j \,\} \\
& \{\, x_i \leftarrow x_0, x_{i-1}, x_{i+1}, x_{2n-1}, \bar{y}_j : j = (i \bmod n), 0 \leq i < 2n \,\}.
\end{aligned}
$$

We observe that $Y = \{y_0, \ldots, y_{n-1}\}$ is a smallest strong **Horn***-backdoor. Figure 1 (a) visualizes the assignments that we obtain when gradually constructing the reducts for $\tau \in 2^Y$. Obviously, we need all $2^{|Y|}$ reducts since "removing" any atom from an assignment τ results in $P_\tau \notin$* **Horn**. *The set $X = \{x_0, \ldots, x_{2n-1}\}$ is also a strong backdoor into* **Horn**. *The set X is larger than*

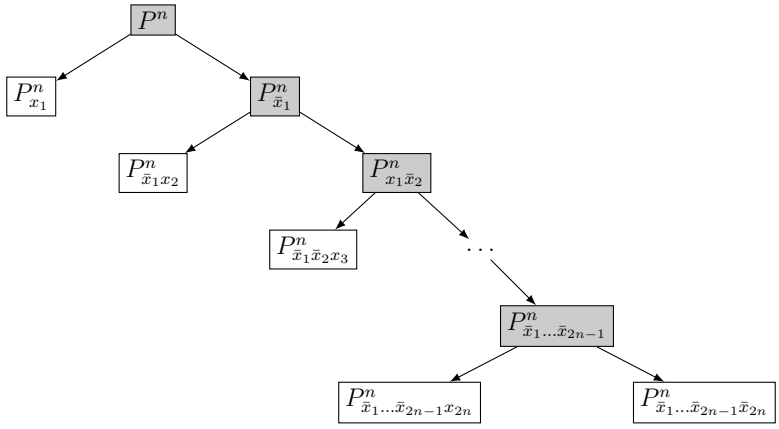


Figure 2 A **Horn**-backdoor tree $BT = (T, \chi)$ of program P^n from Example 2.

the set Y, but already for "shorter" assignments τ' than the assignment reducts $\tau \in 2^X$ we obtain that the reduct belongs to **Horn**. *For instance, the assignment $\tau' = \{\bar{x}_1\}$ yields the reduct $P_{\bar{x}_1} = \{ y_1 \leftarrow x_0, x_2, \ldots, x_{2n-1} \}$, which belongs to* **Horn**. *Hence, we obtain only $2n + 1$ reducts, see Figure 1 (b).*

Example 1 shows that gradually assigning backdoors can "earlier" yield reducts that belong to the considered target class and that larger backdoors can yield an exponentially smaller number of such reducts. A main part for backdoor evaluation is to check whether a model is a minimal model ("minimality check"). The task is co-NP-complete in general, but fixed-parameter tractable when parameterized by the size of a smallest backdoor into a subclass of normal programs. For the minimality check we have to consider all backdoor atoms that have been set to true by any assignment. Hence the backdoor Y from Example 1 yields a significantly smaller number of reducts, however for the minimality check we still have to consider all subsets of Y. Conversely, we construct subsequently in Example 2 a program where the number of assignments that we obtain from a smallest strong backdoor can be arbitrarily larger than the maximum number of atoms in a backdoor that have been set to true by any assignment on a much larger number of atoms.

Example 2. *Let n be some large integer. We define the following program:*

$$\begin{aligned} P := &\{ y_0 \vee \bar{x}_0 \leftarrow \bar{x}_1, \ldots, \bar{x}_{2n-1} \} \cup \\ &\{ y_j \vee x_i \leftarrow \bar{x}_0 \ldots, \bar{x}_{i-1}, \bar{x}_{i+1}, \ldots, \bar{x}_{2n-1} : j = (i \bmod n), 1 \leq i < 2n \} \cup \\ &\{ x_0 \leftarrow x_1, \ldots, x_{2n-1}, \bar{y}_j \} \\ &\{ x_i \leftarrow \bar{x}_0, \bar{x}_{i-1}, \bar{x}_{i+1}, \bar{x}_{2n-1}, \bar{y}_j : j = (i \bmod n), 0 \leq i < 2n \}. \end{aligned}$$

We observe that $Y = \{y_0, \ldots, y_{n-1}\}$ is a smallest strong **Horn***-backdoor. Gradually constructing the reducts as carried out above yields a complete tree with 2^n leaves and a maximum number n of atoms that might be set to true. Obviously, we need all $2^{|Y|}$ reducts since "removing" any atom from an assignment τ results in $P_\tau \notin$* **Horn**. *Further, we easily observe that $X = \{y_1, \ldots, y_n\}$ is a smallest strong* **Horn***-backdoor. Figure 2 visualizes the assignments that we obtain when gradually*

constructing the reducts $\tau \in 2^X$. There we have only $2n+1$ reducts where at most one atom is set to true.

Before we can make the observations from the previous examples precise, we provide some basic definitions. Let X be a set of atoms, $T = (N, E, r)$ a binary tree, and χ a labeling that maps any node $t \in N$ to a set $\chi(t) \subseteq \{ a, \bar{a} : a \in X \}$. We denote by $X_1(t)$ the positive literals of the labeling $\chi(t)$, i.e., $X_1(t) := \chi(t) \cap X$. The *corresponding assignment* $\tau_{\chi(t)}$ of t is the assignment $\tau_{\chi(t)}$ where $\tau_{\chi(t)}(a) = 1$ if $a \in \chi(t)$ and $\tau_{\chi(t)}(a) = 0$ if $\bar{a} \in \chi(t)$. The pair $BT = (T, \chi)$ is a *binary decision tree* of P if the following conditions hold: (i) for the root r we have $\chi(r) = \emptyset$, (ii) for any two nodes $t, t' \in N$, if t' is a child of t, then either $\chi(t') = \chi(t) \cup \{\bar{a}\}$ or $\chi(t') = \chi(t) \cup \{a\}$ for some atom $a \in X \setminus \tau^{-1}_{\chi(t)}$, and (iii) for any three nodes $t, t_1, t_2 \in N$, if t_1 and t_2 are children of t, then $\chi(t_1) \neq \chi(t_2)$. We denote by $\mathrm{at}(BT)$ the atoms *occurring* in assignments of BT, i.e., $\mathrm{at}(BT) := \bigcup_{t \in N} \tau^{-1}_{\chi(t)}$.

Next, we give a definition for backdoor trees of answer set programs.

Definition 2. *Let P be a program, $X = \mathrm{at}(P)$, and $BT = (T, \chi)$ be a binary decision tree and $T = (N, E)$. The pair $BT = (T, \chi)$ is a* $\mathcal{C}$-backdoor tree *of P if $P_\tau \in \mathcal{C}$ for every leaf $t \in N$ and $\tau = \tau_{\chi(t)}$. We denote by* #leaves*(BT) the number of leaves of T, i.e.,* $\#\mathrm{leaves}(BT) := |\{ t : t \text{ is a leaf of } T \}|$. *We denote by* $\mathrm{gs}(BT)$ *the maximum number of atoms that have been set to true by a corresponding assignment of any leaf of T, more specifically,* $\mathrm{gs}(BT) := \max\{ |X_1(t)| : t \text{ is a leaf of } T \}$. *For reasons explained below, we call* $\mathrm{gs}(BT)$ *the* Gallo-Scutellà parameter *of* BT.

In other words, a backdoor tree of a program P is a binary decision tree where the nodes of the tree are labeled by assignments $\tau \in 2^X$ on subsets $X \subseteq \mathrm{at}(P)$, the corresponding partial assignment τ of an inner node yields a reduct P_τ that does not belong to the considered target class, and the corresponding assignment τ of a leaf yields a reduct P_τ that belongs to the considered target class.

Relationship to a Parameter by Gallo and Scutellà

The maximal number of positive variables in a propositional formula has been considered as parameter by Gallo and Scutellà [18] to measure in a certain sense the distance from being Horn. Recall that a formula is Horn if each clause has at most one positive literal. Gallo and Scutellà have also established an XP-algorithm to determine the parameter of a propositional formula.

We consider the parameter in its original context and definition as nested classes of families of sets on a family to generalize Horn formulas. Let S be a family of sets $S_1, \ldots, S_m$, $\mathsf{S}_X = \mathsf{S} \setminus \{ Y \in \mathsf{S} : X \subseteq Y \}$, and $\mathsf{S} - X := \{ S \setminus X : S \in \mathsf{S} \}$ for some set X. Moreover, (i) $\mathsf{S} \in \Sigma_0$ if and only if $|S| \leq 1$ for each $S \in \mathsf{S}$ and (ii) $\mathsf{S} \in \Sigma_k$ if and only if there is some $v \in \bigcup_{1 \leq i \leq m} S_i$ such that $S_{\{v\}} \in \Sigma_{k-1}$ and $\mathsf{S} - \{v\} \in \Sigma_k$. Then, the class Γ_k consists of all propositional formulas F such that $F' \in \Sigma_k$ where F' is obtained from F by removing all negative literals (note that we consider F' as a set of clauses and a clause is a set of variables). Observe that Γ_0 consists of all Horn formulas.

A *backdoor tree of F into Horn formulas* is a binary decision tree where the formula F_τ is Horn for each leaf t and its corresponding assignment τ.

Proposition 3. *A propositional formula F belongs to Γ_k if and only if there is a backdoor tree $BT = (T, \chi)$ into Horn formulas of F such that* $\mathrm{gs}(BT) \leq k$.

Proof. We refer to the appendix (Appendix A, p. 18). □

4 Backdoor Tree Evaluation

In this section, we establish an analogue to backdoor evaluation for backdoor trees. Again we consider the reducts P_τ together with the atoms that are set to true and extend this notion to the corresponding assignments of the leaves for binary decision trees.

Definition 4. *Let P be a program, $X = \mathrm{at}(P)$, and $BT = (T, \chi)$ a binary decision tree.*

$$\mathrm{AS}(P, \tau) := \{\, M \cup \tau^{-1}(1) : M \in \mathrm{AS}(P_\tau) \,\} \quad \textit{and}$$
$$\mathrm{AS}(P, BT) := \{\, M : \textit{ t is a leaf of } T, \tau = \chi(t), M \in \mathrm{AS}(P, \tau) \,\}.$$

In other words, the sets in $\mathrm{AS}(P, BT)$ are answer sets of P_τ for assignments τ to $\chi(t) \cap \mathrm{at}(P)$ extended by those atoms which are set to true by τ. In the following lemma we will see that the elements in $\mathrm{AS}(P, BT)$ are "answer set candidates" of the original program P. The concepts are similar to ASP backdoors, but slightly more sophisticated.

Lemma 5. *Let P be a program, $BT = (T, \chi)$ a binary decision tree of P, and $X := \mathrm{at}(BT)$. Then* $\mathrm{AS}(P) \subseteq \mathrm{AS}(P, BT)$.

Proof. We refer to the appendix (Appendix A, p. 18). □

The subsequent observation states that we obtain less "answer set candidates" when evaluating ASP backdoor trees than by evaluation ASP backdoors.

Observation 6. *Let P be a program, $BT = (T, \chi)$ a binary decision tree of P, $X := \mathrm{at}(BT)$, and* $\mathrm{AS}(P, X) := \{\, M \cup \tau^{-1}(1) : \tau \in 2^{X \cap \mathrm{at}(P)}, M \in \mathrm{AS}(P_\tau) \,\}$. *Then* $\mathrm{AS}(P, BT) \subseteq \mathrm{AS}(P, X)$.

Proof. We refer to the appendix (Appendix A, p. 18). □

Theorem 7. *Let $\mathcal{C} \subseteq$* **Normal** *be an enumerable class. The problems in $\mathcal{AspFull}$ are all fixed-parameter tractable when parameterized by* $\mathrm{gs}(BT) + \#\mathrm{leaves}(BT)$ *of a $\mathcal{C}$-backdoor tree BT, assuming that the backdoor tree is given as input.*

Before proving this Theorem, we need to make some observations. In view of Lemma 5 we have to consider the corresponding reducts of the leaves in the backdoor tree. For each assignment $\tau \in \mathrm{ta}(T)$ we construct the reduct P_τ and compute the set $\mathrm{AS}(P_\tau)$. Then, we obtain the set $\mathrm{AS}(P)$ by checking for each $M \in \mathrm{AS}(P_\tau)$ whether it gives rise to an answer set of P. The crucial part is again to consider minimality with respect to the Gelfond-Lifschitz reduct. For the leaf t and its corresponding assignment τ we can guarantee minimality with respect to the reduct $(P_\tau)^M$. Setting atoms to true by the assignment τ does apparently not guarantee minimality with respect to P^M (cf. Lemma 5). Hence, we have to check for each atom in $\tau^{-1}(1)$ whether there is a "justification" to set the atom to true.

We establish the following result.

Proposition 8. *Let $\mathcal{C} \subseteq$ **Normal**. Given a program P of input size n, a $\mathcal{C}$-backdoor tree $BT = (T, \chi)$ of P of Gallo-Scutellà parameter $k = \text{gs}(BT)$, a leaf t of T, and a set $M \subseteq \text{AS}(P, \tau_{\chi(t)})$ of atoms, we can check in time $\mathcal{O}(2^k \cdot n)$ whether M is an answer set of P.*

Proof. We refer to the appendix (Appendix A, p. 19). □

Now, we are in position to establish Theorem 7.

Proof of Theorem 7. Let $BT = (T, r, \chi)$ be the given $\mathcal{C}$-backdoor tree, $g = \text{gs}(BT)$, $l = \#\text{leaves}(BT)$, $T = (N, E, r)$, and n the input size of P. According to Lemma 5, $\text{AS}(P) \subseteq \text{AS}(P, BT)$. Since $P_\tau \in \mathcal{C}$ and $\mathcal{C}$ is enumerable, we can compute $\text{AS}(P_\tau)$ in polynomial time for each leaf $t \in N$ and $\tau = \tau_{\chi(t)}$, say in time $\mathcal{O}(n^c)$ for some constant c. Hence, $|\text{AS}(P_\tau)| \leq \mathcal{O}(n^c)$ for each leaf $t \in N$ and $\tau = \tau_{\chi(t)}$. By Proposition 8, we can decide whether $M \in AS(P)$ in time $\mathcal{O}(2^g \cdot n^c)$ and $|\text{AS}(P, \tau)| \leq \mathcal{O}(2^g \cdot n^c)$ for each $M \in \text{AS}(P, \tau)$ where $\tau = \tau_{\chi(t)}$ and t is a leaf of T. Since there are at most l many leaves, we can compute $\text{AS}(P, T)$ and check whether for $M \in \text{AS}(P, T)$ also $M \in \text{AS}(P)$ holds in time $\mathcal{O}(l \cdot 2^g \cdot n^c)$ and $|\text{AS}(P, T)| \leq \mathcal{O}(l \cdot 2^g \cdot n^c)$. Then we can also solve all problems in $\mathcal{A}sp\mathcal{F}ull$ from $\text{AS}(P)$ within polynomial time. Consequently, the problem is fixed-parameter tractable when parameterized by $g + l$.

□

There are two factors for hardness of ASP problems when parameterized by the Gallo-Scutellà parameter plus the size a backdoor tree (i) atoms that are set to true which yield potential candidates and are hence important for the minimality check in each leaf; and (ii) leaves in a backdoor tree which yield the reducts we have to consider. Both factors of hardness are "used" in the proof of Theorem 7. Hence, in contrast to SAT backdoor trees we do not simply parameterize the reasoning problems in $\mathcal{A}sp\mathcal{F}ull$ by $\#\text{leaves}(BT)$ of a given backdoor tree $BT = (T, \chi)$ of P to obtain a more refined view on backdoors. Instead we also consider $\text{gs}(BT)$ which is the maximum number of atoms that are set to true in a leaf of T. This is attributed to the minimality check where we have to consider the number of atoms that are set to true.

5 Relation to Backdoors

In this section, we investigate on connections between backdoors and backdoor trees. We show that our composed parameter based on backdoor trees is more general than the size of a backdoor.

Lemma 9. *Let P be a program and $\mathcal{C}$ be an hereditary class of programs. If BT is a $\mathcal{C}$-backdoor tree of P, then $\text{at}(BT)$ is a strong $\mathcal{C}$-backdoor of P.*

Proof. We refer to the appendix (Appendix A, p. 20). □

We make the following observations about binary decision trees.

Observation 10. *Let BT be a binary decision tree. Then, $\text{gs}(BT) \leq |\text{at}(BT)| \leq \#\text{leaves}(BT) - 1$.*

Observation 11. *Let BT be a binary decision tree, $n = |\text{at}(BT)|$, $g = \text{gs}(BT)$, and $l = \#\text{leaves}(BT)$. Then, $l \leq (1+n)^g$.*

We establish that every strong backdoor of size k yields a backdoor tree consisting of at least $k+1$ leaves and at most 2^k leaves.

Lemma 12. *Let P be a program, $\mathcal{C}$ an hereditary class of programs, X a strong $\mathcal{C}$-backdoor of smallest size of P, and $BT = (T, \chi)$ a $\mathcal{C}$-backdoor tree of smallest number of leaves of P. Then, $|X| + 1 \leq \#\text{leaves}(BT) \leq 2^{|X|}$.*

Proof. We refer to the appendix (Appendix A, p. 20). □

Lemma 13. *Let P be a program, $\mathcal{C}$ a hereditary class of programs, X a strong $\mathcal{C}$-backdoor of smallest size of P, and $BT = (T, \chi)$ a $\mathcal{C}$-backdoor tree of smallest Gallo-Scutellà parameter $\text{gs}(BT)$ of P. Then, $\text{gs}(BT) \leq |X|$.*

Proof. We refer to the appendix (Appendix A, p. 20). □

6 Backdoor Tree Detection

In this section, we pay attention to the detection of backdoor trees. We first define the following decision problem:

$\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES)

Given:	A program P, an integer $g \geq 0$, and an integer $l \geq 0$.
Parameter:	The integer $g + l$.
Task:	Decide whether P has a $\mathcal{C}$-backdoor tree BT of Gallo-Scutellà parameter $\text{gs}(BT) \leq g$ and $\#\text{leaves}(BT) \leq l$.

By self-reduction (or self-transformation) [33, 11, 12], we can use a decision algorithm for $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) to actually find the backdoor. Again we only require the target class to be hereditary.

Lemma 14. *Let $\mathcal{C}$ be a hereditary class of programs. If $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) is fixed-parameter tractable, then also finding a $\mathcal{C}$-backdoor tree of a given program P of Gallo-Scutellà parameter at most g and at most l leaves is fixed-parameter tractable (when parameterized by $g + l$).*

Proof. We refer to the appendix (Appendix A, p. 20). □

In the following, we consider backdoor tree detection when parameterized by the Gallo-Scutellà parameter and the number of leaves of a backdoor tree. Therefore, we consider notions coined by Samer and Szeider [32] in the setting of propositional satisfiability and apply it to answer set programming.

Theorem 15. *The problem* **Horn**-BACKDOOR-TREE DETECTION(GS,LEAVES) *is fixed-parameter tractable.*

Before we can establish the result, we introduce the notion of a loss-free kernelization for answer set programming and establish how we can use loss-free kernelizations to solve the backdoor tree detection problem.

Definition 16. *Let $\mathcal{C}$ be a class of programs. A* loss-free kernelization *of the problem* STRONG $\mathcal{C}$-BACKDOOR DETECTION *is a polynomial-time algorithm that given an instance (I, k), either correctly decides that I does not have a strong $\mathcal{C}$-backdoor of size at most k, or computes a set $K \subseteq \text{at}(P)$ such that the following conditions hold:*

1. *$X \subseteq K$ for every minimal strong $\mathcal{C}$-backdoor X of size at most k and*
2. *there is a computable function f such that $|K| \leq f(k)$.*

We establish the following proposition about target classes $\mathcal{C}$ that admit loss-free kernelizations.

Proposition 17. *Let $\mathcal{C}$ be a class of programs. If the problem* STRONG $\mathcal{C}$-BACKDOOR DETECTION *has a loss-free kernelization, then the problem $\mathcal{C}$-*BACKDOOR-TREE DETECTION(GS,LEAVES) *is fixed-parameter tractable.*

Proof. We refer to the appendix (Appendix A, p. 21). □

The following is a direct consequence of results presented by Samer and Szeider [32].

Lemma 18. *The problem* STRONG **Horn**-BACKDOOR DETECTION *has a loss-free kernelization with loss-free kernels of size $k^2 + k$.*

Proof. We refer to the appendix (Appendix A, p. 22). □

We are now in position to establish Theorem 15.

Proof of Theorem 15. It follows directly from Proposition 17 and Lemma 18. □

Then we can drop the assumption in Theorem 7 that the backdoor is given.

Corollary 19. *Let $\mathcal{C} \subseteq$* **Normal** *be an enumerable class. The problems in AspFull are all fixed-parameter tractable when parameterized by* $\text{gs}(BT) + \#\text{leaves}(BT)$ *of a smallest* $\text{gs}(BT) + \#\text{leaves}(BT)$ *$\mathcal{C}$-backdoor tree BT.*

Proof. Let P be a program and k an integer. Since there are only linear many combinations for $k = g + l$, we can use Lemma 14 to find a $\mathcal{C}$-backdoor tree BT of smallest $\text{gs}(BT) + \#\text{leaves}(BT)$ where $\text{gs}(BT) \leq g$ and $\#\text{leaves}(BT) \leq l$ or to decide that no such backdoor tree exists. The remainder follows from Theorem 7. □

7 Discussion and Future Work

We have introduced backdoor trees to answer set programming. The general concepts are similar to the propositional setting. We also take the number of leaves of a backdoor tree into account. However, the minimality check, which is necessary to verify minimality of potential answer set candidates with respect to the Gelfond-Lifschitz reduct, yields an additional hardness factor. Therefore, we parameterize the problem of backdoor tree evaluation by the composed parameter number of leaves of a backdoor tree and maximum number of atoms that are set to true by a corresponding assignment in a leaf. The former parameter is crucial to bound the number of potential reducts and hence to bound the number of answer set candidates. The latter parameter is crucial to bound the number of atoms in any assignment, which we additionally have to consider for the minimality check.

Our parameterization raises the question of whether we can drop one parameter from the composed parameter. On the one hand, one could parameterize the evaluation problem just by the number of leaves of the backdoor tree, which yields fixed-parameter tractability, but then the evaluation algorithm does not necessarily yield any speedup in the algorithm since we still have to consider the minimality check where a bound on the number of leaves does not pay off when using our minimality check approach. In other words, the evaluation problem is fixed-parameter tractable when parameterized by the number of leaves of backdoor tree. We obtain a parameter that might be significantly smaller, but the running time of the evaluation algorithm can be significantly worse (exponentially). On the other hand, one could parameterize the evaluation problem just by the Gallo-Scutellà parameter (the maximal number of atoms that we have to set to true in any leaf) of the backdoor tree. Since the Gallo-Scutellà parameter of a backdoor tree can be arbitrarily small compared to the number of leaves of a backdoor tree (and hence the size of a smallest backdoor), we obtain an arbitrarily smaller parameter. However, since our upper bound for the number of reducts is $(1 + n)^g$, where n is the number of atoms of the given program and g the Gallo-Scutellà parameter of the backdoor tree, the number of reducts remains non-uniformly bounded. Hence, it remains open whether we obtain fixed-parameter tractability. Moreover, the problem backdoor tree detection when parameterized by the Gallo-Scutellà parameter is only known to be in XP and the question of whether it can be carried out in fixed-parameter tractable time is currently an open research question.

Finally, we would like to note that it is unlikely that the problem backdoor tree detection is fixed-parameter tractable for the classes based on the absence of certain cycles since backdoor detection is already W[2]-hard and co-para-NP-hard [15], respectively.

References

[1] Abseher, M., Gebser, M., Musliu, N., Schaub, T., Woltran, S.: Shift design with answer set programming. In: Calimeri, F., Ianni, G., Truszczyński, M. (eds.) Proceedings of the 13th International Conference Logic Programming and Nonmonotonic Reasoning (LPNMR'15). pp. 32–39. Springer Verlag, Lexington, KY, USA (Sep 2015)

[2] Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F.: WASP: A native ASP solver based on constraint learning. In: Cabalar, P., Son, T. (eds.) Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13), Lecture Notes in Computer Science, vol. 8148, pp. 54–66. Springer Verlag, Corunna, Spain (Sep 2013)

[3] Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: Calimeri, F., Ianni, G., Truszczyński, M. (eds.) Proceedings of the 13th International Conference Logic Programming and Nonmonotonic Reasoning (LPNMR'15). pp. 40–54. Springer Verlag, Lexington, KY, USA (2015)

[4] Andres, B., Rajaratnam, D., Sabuncu, O., Schaub, T.: Integrating ASP into ROS for reasoning in robots. In: Calimeri, F., Ianni, G., Truszczyński, M. (eds.) Proceedings of the 13th International Conference Logic Programming and Nonmonotonic Reasoning (LPNMR'15). pp. 69–82. Springer Verlag, Lexington, KY, USA (Sep 2015)

[5] Apt, K.R., Blair, H.A., Walker, A.: Towards a theory of declarative knowledge. Foundations of deductive databases and logic programming pp. 89–148 (1988)

[6] Ben-Eliyahu, R., Dechter, R.: Propositional semantics for disjunctive logic programs. Ann. Math. Artif. Intell. 12(1), 53–87 (1994)

[7] Bliem, B., Ordyniak, S., Woltran, S.: Clique-width and directed width measures for answer-set programming. In: Fox, M., Kaminka, G. (eds.) Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI'16). The Hague, Netherlands (Aug 2016), to appear.

[8] Buss, J., Goldsmith, J.: Nondeterminism within p^*. SIAM J. Comput. 22(3), 560–572 (1993)

[9] Calimeri, F., Ianni, G., Ricca, F.: The third open answer set programming competition. Theory Pract. Log. Program. 14, 117–135 (1 2014)

[10] Denecker, M., Vennekens, J., Bond, S., Gebser, M., Truszczyński, M.: The second answer set programming competition. In: Erdem, E., Lin, F., Schaub, T. (eds.) Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09), Lecture Notes in Computer Science, vol. 5753, pp. 637–654. Springer Verlag, Potsdam, Germany (Sep 2009)

[11] Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science, Springer Verlag, New York, NY, USA (1999)

[12] Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer Verlag, London, UK (2013)

[13] Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Ann. Math. Artif. Intell. 15(3–4), 289–323 (1995)

[14] Fichte, J.K., Szeider, S.: Backdoors to normality for disjunctive logic programs. ACM Trans. Comput. Log. 17(1) (Oct 2015), extended and updated version of a paper that appeared in *Proc. of the 27th AAAI Conference on Artificial Intelligence (AAAI'13)*.

[15] Fichte, J.K., Szeider, S.: Backdoors to tractable answer-set programming. Artificial Intelligence 220(0), 64–103 (2015), extended and updated version of a paper that appeared in *Proc. of the 22nd International Conference on Artificial Intelligence (IJCAI'11)*.

[16] Fichte, J.K., Truszczyński, M., Woltran, S.: Dual-normal programs – the forgotten class. Theory Pract. Log. Program. (2015), coRR: 1507.05388. To appear in Proceedings of the 31st International Conference on Logic Programming (ICLP'15).

[17] Flum, J., Grohe, M.: Parameterized Complexity Theory, Theoretical Computer Science, vol. XIV. Springer Verlag, Berlin (2006)

[18] Gallo, G., Scutellà, M.G.: Polynomially solvable satisfiability problems. Information Processing Letters 29(5), 221–227 (1988)

[19] Gebser, M., Glase, T., Sabuncu, O., Schaub, T.: Matchmaking with answer set programming. In: Cabalar, P., Son, T.C. (eds.) Proceedings of 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13). Lecture Notes in Computer Science, vol. 8148, pp. 342–347. Springer Verlag, Corunna, Spain (Sep 2013)

[20] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-Criteria Optimization in Answer Set Programming. In: Gallagher, J., Gelfond, M. (eds.) Technical Communications of the 27th International Conference on Logic Programming (ICLP'11). Leibniz International Proceedings in Informatics (LIPIcs), vol. 11, pp. 1–10. Dagstuhl Publishing, Lexington, KY, USA (Jul 2011)

[21] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Morgan & Claypool (2012)

[22] Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam answer set solving collection. AI Communications 24(2), 107–124 (2011)

[23] Gebser, M., Maratea, M., Ricca, F.: The design of the sixth answer set programming competition. In: Calimeri, F., Ianni, G., Truszczynski, M. (eds.) Proceedings of the 13th International Conference Logic Programming and Nonmonotonic Reasoning (LPNMR'15). pp. 531–544. Springer Verlag, Lexington, KY, USA (Sep 2015)

[24] Gebser, M., Maratea, M., Ricca, F.: What's hot in the answer set programming competition. In: Schuurmans, D., Wellman, M.P. (eds.) Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16). pp. 4327–4329. The AAAI Press, Phoenix, Arizona, USA (Feb 2016), http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12233

[25] Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) Proceedings of the 5th International Conference and Symposium on Logic Programming (ICLP/SLP'88). vol. 2, pp. 1070–1080. MIT Press, Seattle, WA, USA (August 1988)

[26] Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Comput. 9(3/4), 365–386 (1991)

[27] Gottlob, G., Szeider, S.: Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. The Computer Journal 51(3), 303–325 (2008)

[28] Jakl, M., Pichler, R., Woltran, S.: Answer-set programming with bounded treewidth. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09). vol. 2, pp. 816–822. The AAAI Press, Pasadena, CA, USA (Jul 2009)

[29] Marek, V.W., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: Apt, K.R., Marek, V.W., Truszczyński, M., Warren, D.S. (eds.) The Logic Programming Paradigm: a 25-Year Perspective, pp. 375–398. Artificial Intelligence, Springer Verlag (1999)

[30] Niedermeier, R.: Invitation to Fixed-Parameter Algorithms, Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, New York, NY, USA (2006)

[31] Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. 25(3), 241–273 (1999)

[32] Samer, M., Szeider, S.: Backdoor trees. In: Holte, R.C., Howe, A.E. (eds.) Proceedings of 23rd Conference on Artificial Intelligence (AAAI'08). pp. 363–368. The AAAI Press, Chicago, IL, USA (July 2008)

[33] Schnorr, C.P.: On self-transformable combinatorial problems. In: König, H., Korte, B., Ritter, K. (eds.) Mathematical Programming at Oberwolfach, Mathematical Programming Studies, vol. 14, pp. 225–243. Springer Verlag (1981)

[34] Truszczyński, M.: Trichotomy and dichotomy results on the complexity of reasoning with disjunctive logic programs. Theory Pract. Log. Program. 11, 881–904 (11 2011)

A Appendix: Omitted Proofs

Proof of Proposition 3

Proof. Let F be a propositional formula and $BT = (T, \chi)$ be a backdoor tree BT into Horn formulas of F of Gallo-Scutellà parameter k. Observe that the labelings χ of the paths in T from the root to a leaf provide witnesses for Conditions (i) and (ii). Hence, $F \in \Gamma_k$. Conversely, we construct a $\mathcal{C}$-backdoor tree from the fact that $F \in \Gamma_k$. Take the binary decision tree (T, χ) where $T = (N, E, r)$ and $E = \emptyset$. Since $F \in \Gamma_k$, the formula $F' \in \Sigma_k$ where F' is obtained from F by removing all negative literals. By Condition (ii) there is a variable v such that $F'_{\{v\}} \in \Sigma_{k-1}$ and $F' - \{v\} \in \Sigma_k$. We add two fresh nodes n_1 and n_2 to N, edges (r, n_1) and (r, n_2) to E, and extend the mapping χ by labelings $\chi(n_1) := \chi(r) \cup \{v\}$ and $\chi(n_2) := \chi(r) \cup \{\bar{v}\}$. We proceed inductively with $F'_{\{v\}}$ and Σ_{k-1} for the leaf n_1 of T and $F' - \{v\}$ and Σ_k for the leaf n_2 of T. We easily observe that by construction BT is a $\mathcal{C}$-backdoor tree and since at most k variables are set to true $\mathrm{gs}(BT) \leq k$. □

Proof of Lemma 5

Proof. Let $M \in \mathrm{AS}(P)$ be chosen arbitrarily. We consider the assignments $\tau = \chi(t)$ for each leaf t of T. Let $M' = M \setminus \tau^{-1}(1)$. Observe that $M' \in \mathrm{AS}(P_\tau)$ implies $M \in \mathrm{AS}(P, BT)$ since $M = M' \cup \tau^{-1}(1)$ by definition. Hence, to establish the lemma, it suffices to show that $M' \in \mathrm{AS}(P_\tau)$. We have to show that M' is a model of $P_\tau^{M'}$, and that no proper subset of M' is a model of $P_\tau^{M'}$ (which we already carried out in the proof for ASP backdoors [15, Lemma 3.7]. Consequently, $\mathrm{AS}(P) \subseteq \mathrm{AS}(P, BT)$ and the lemma is established. □

Proof of Observation 6

Proof. We show that $\mathrm{AS}(P, BT) \subseteq \mathrm{AS}(P, X)$. By Definition 4 we have $\mathrm{AS}(P, X) = \bigcup_{\tau \in 2^X} \mathrm{AS}(P, \tau)$. Let $U_\tau := \{\, \tau' : \tau' \in 2^X, \tau'(a) = \tau(a) \text{ for every } a \in \tau^{-1} \,\}$, in other words, U_τ contains all assignments $\tau' \in 2^X$ such that $\tau^{-1} \subseteq \tau'^{-1}$ for a possible assignment $\tau' \in 2^X$. It remains to observe that $\mathrm{AS}(P, \tau) \subseteq \bigcup_{\tau' \in U_\tau} \mathrm{AS}(P, \tau')$. We define $l := |\tau^{-1} \setminus \tau'^{-1}|$ for some assignments τ and τ' where $\tau^{-1} \subseteq \tau'^{-1}$ and proceed an induction proof on l. The case $l = 0$ is obvious. The case $l = 1$ follows simply from [15, Lemma 3.7] since $\mathrm{AS}(P, \tau) = \mathrm{AS}(P, \tau_0) \cup \mathrm{AS}(P, \tau_1)$ where $\tau_0(a) = 0$ and $\tau_1(a) = 1$ for the atom $a \in \tau^{-1} \setminus \tau'^{-1}$. Consider the case $l > 1$. Let τ'' be an assignment such that $\tau'^{-1} = \tau''^{-1} \setminus \{a\}$. Then from known results [15, Lemma 3.7] we obtain $\mathrm{AS}(P, \tau'') \subseteq \mathrm{AS}(P, \tau'_0) \cup \mathrm{AS}(P, \tau'_1)$ where $\tau'_0(a) = \tau'_1(a) = \tau''(a)$ for each $a \in \tau''^{-1}$, $\tau'_0(b) = 0$ and $\tau'_1(b) = 1$ for some $b \notin \tau^{-1}$. Since, $\mathrm{AS}(P, \tau) \subseteq \mathrm{AS}(P, \tau'')$ by induction, we conclude $\mathrm{AS}(P, \tau) \subseteq \bigcup_{\tau' \in U_\tau} \mathrm{AS}(P, \tau')$. □

Proof of Proposition 8

Lemma 20 (Folkore)**.** *Every Horn program has at most one minimal model which can be found in linear time.*

Lemma 21 (Minimality Check [14])**.** *Let $\mathcal{C}$ be a class of normal programs. Given a program P of input size n, a strong $\mathcal{C}$-backdoor X of P of size k, and a set $M \subseteq \text{at}(P)$ of atoms, we can check in time $\mathcal{O}(2^k n)$ whether M is an answer set of P.*

of Proposition 8. We would like to use a similar construction as in [15, Lemma 3.7] and therefore need slightly stronger arguments.

Let $BT = (T, \chi)$ be a $\mathcal{C}$-backdoor tree of P. We first check whether M is a model of P^M. If M is not a model of P^M, then M cannot be an answer set of P. Hence, assume that M is a model of P^M.

We construct from P^M a program $P^M_{Y \subseteq X_1(t)}$ by (i) removing all rules r for which $H(r) \cap Y \neq \emptyset$, and (ii) replacing for all remaining rules r the head $H(r)$ with $H(r) \setminus X_1(t)$, and the positive body $B^+(r)$ with $B^+(r) \setminus Y$.

Claim: $P^M_{Y \subseteq X_1(t)}$ is Horn.

We first establish that $P_{Y \subseteq X_1(t)}$ is normal. Since $BT = (T, \chi)$ is a backdoor tree, the partial reduct P_τ is normal for each leaf t of T and $\chi := \tau_{\chi(t)}$. Let r' be an arbitrarily chosen rule in P_τ. Then there is a corresponding rule $r \in P$ and a corresponding rule $r'' \in P_{Y \subseteq X_1(t)}$. Since we remove in both constructions exactly the same literals from the head of every rule, $H(r') = H(r'')$ holds. Consequently, $P_{Y \subseteq X_1(\tau)}$ is normal and $P^M_{Y \subseteq X_1(\tau)}$ is Horn.

> If $P^M_{Y \subseteq X_1(t)}$ has no model, then stop and return *True*.
>
> Otherwise, compute the unique minimal model L of the Horn program $P^M_{Y \subseteq X_1(t)}$. If $L \subseteq M \setminus X$, $L \cup Y \subsetneq M$, and $L \cup Y$ is a model of P^M, then return *False*. Otherwise return *True*.

For each set $Y \subseteq M \cap X_1(t)$ the above procedure runs in linear time by Lemma 20. By Lemma 5 we consider only the atoms that have been set to true in M, i.e., $Y \subseteq X_1(t)$ and run the algorithm from above for each $X_1 \subseteq X \cap M$ and M is a minimal model of P^M if and only if the algorithm returns *True* for each $Y \subseteq X_1(t)$. Consequently, we have to consider at most $2^{|X_1(t)|}$ many subsets of $X_1(t)$. Since $k = \max\{\, |X| : X \in X_1(T) \,\}$, we obtain a running time of $\mathcal{O}(2^k \|P\|)$.

Claim: M is a minimal model of P^M if and only if the algorithm returns True for each $Y \subseteq M \cap X_1(t)$.

The claim follows directly from the proof of Lemma 21 and establishes the correctness of the above procedure.

Consequently the problem is fixed-parameter tractable when parameterized by k.

□

Proof of Lemma 9

Proof. Let $X = \text{at}(BT)$. For every leaf $t \in T$ we have $P_\tau \in \mathcal{C}$ where $\tau = \tau_{\chi(t)}$ according to Definition 2. Then, we observe that one obtains all assignments $2^{|X|}$ simply by extending the assignments $\tau = \tau_{\chi(t)}$ by assignments τ' on the atoms $\text{at}(BT) \setminus \tau^{-1}$. Since $\mathcal{C}$ is hereditary and already $P_\tau \in \mathcal{C}$, also $P_{\tau \cup \tau'} \in \mathcal{C}$. Hence, for all possible assignments $\tau \in 2^X$ we have $P_\tau \in \mathcal{C}$. Consequently, X is a strong $\mathcal{C}$-backdoor of P and the lemma holds. □

Proof of Observation 11

Proof. Let $BT = (T, \chi)$ a binary decision tree, $T = (N, E, r)$, $n = |\text{at}(BT)|$, $g = \text{gs}(BT)$, and $l = \#\text{leaves}(BT)$. According to Definition 2, for every leaf t of T we have $|\chi(t)| \leq n$ and at most g atoms are set to true in the corresponding assignment $\tau_{\chi(t)}$. Hence, we have at most $\sum_{i=0}^{g} \binom{n}{i}$ possible combinations of assignments. Thus, $l \leq \sum_{i=0}^{g} \binom{n}{i}$. By binomial expansion we obtain $\sum_{i=0}^{g} \binom{n}{i} \leq \sum_{i=0}^{g} (n^i \cdot 1^{g-i}) \leq (1+n)^g$. □

Proof of Lemma 12

Proof. Let P be a program, X a strong $\mathcal{C}$-backdoor of smallest size of P, and $BT = (T, \chi)$ a $\mathcal{C}$-backdoor tree of smallest number of leaves of P. According to Lemma 9 the set $\text{at}(BT)$ is also a strong $\mathcal{C}$-backdoor of P. By Observation 10 and the definition a backdoor tree we have $|\text{at}(BT)| \leq \#\text{leaves}(BT) - 1$. It remains to observe that we can construct from X a complete binary decision tree (T', χ) of P with $2^{|X|}$ leaves by labeling the root of T' by $\emptyset$, and for each level the nodes by an a or $\bar{a}$ (which did not occur in a lower level) for $a \in X$ in an arbitrary fixed order. We obtain for a leaf t that $\chi(t) = \tau$ if and only if $\tau \in 2^X$. Hence, $P_\tau \in \mathcal{C}$ for every $\tau = \chi(t)$ and leaf t. Then (T', χ) is a $\mathcal{C}$-backdoor tree of P where T has $2^{|X|}$ many leaves. Since the number of leaves of T is less or equal the number of leaves of T', we conclude $\#\text{leaves}(BT) \leq 2^{|X|}$. Consequently, the lemma holds. □

Proof of Lemma 13

Proof. Let P be a program, X a strong $\mathcal{C}$-backdoor of smallest size of P, and $BT = (T, \chi)$ a $\mathcal{C}$-backdoor tree of smallest Gallo-Scutellà parameter $\text{gs}(BT)$ of P. We can construct from X a $\mathcal{C}$-backdoor tree (T', χ) of P which is a complete binary tree (cf. proof of Lemma 12). Since $\chi(t) = \tau$ if and only if $\tau \in 2^X$ for every leaf t, $|\{\, X_1(t) : t \text{ is a leaf of } T \,\}| = 2^{|X|}$. Hence, $\max\{\, |X_1(t)| : t \text{ is a leaf of } T \,\} = |X|$ and we conclude $\text{gs}(BT) \leq |X|$. Consequently, the proposition holds. □

Proof of Lemma 14

Proof. Given a program P of input size n and integers $g \geq 0$ and $l \geq 0$. We check whether P has a $\mathcal{C}$-backdoor tree of Gallo-Scutellà parameter at most g and at most l leaves by means of Algorithm 1. Assume that the decision problem $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) is

fixed-parameter tractable and runs in time $\mathcal{O}(f(g+l)\cdot n^c)$ for some constant c and some computable function f.

Steps 1 and 2 of Algorithm 1 are trivial. In Step 3 we use the decision problem $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) to find an atom $a \in \text{at}(P)$ where (i) for the assignment τ_1 that assigns $\tau_1(a) = 1$ the program P_{τ_1} has a $\mathcal{C}$-backdoor tree BT_1 of $\text{gs}(BT_1) \leq g-1$ and $\#\text{leaves}(BT_1) \leq l_1$ and (ii) for the assignment τ_0 that assigns $\tau_0(a) = 0$ the program P_{τ_0} has a $\mathcal{C}$-backdoor tree BT_0 of $\text{gs}(BT_0) \leq g$ and $\#\text{leaves}(BT_0) \leq l_0$, and $2 \leq l_1 + l_0 \leq l$ for some integers l_1 and l_0. Such an atom a and a $\mathcal{C}$-backdoor tree $BT = (T, \chi)$ of $\text{gs}(BT) \leq g$ and $\#\text{leaves}(BT) \leq l$ exist since $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) returns *Yes* in Step 1. Then, by definition of a binary decision tree there are also trees BT_1 and BT_0 where $\#\text{leaves}(BT_1) + \#\text{leaves}(BT_0) \leq l$. There are only linear many combinations $l_1 + l_0 \leq l$ and we determine the integers l_1 and l_0 simply by means of binary search in Steps 3a–3g. We ensure that $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) yields *No* for every value of i and *Yes* for every value of j; by setting i initially to 0 (Step 3a), checking whether $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) yields *Yes* for the initial values (Step 3b), and assigning i and j accordingly in Step 3e. In Step 4 we compute the backdoor trees BT_1 and BT_0 for P_{τ_1} and P_{τ_0}. Finally, in Step 5 we merge BT_1 and BT_0 into a solution BT for the input program. Obviously, BT is a $\mathcal{C}$-backdoor tree of P of $\text{gs}(BT) \leq g$ and $\#\text{leaves}(BT) \leq l$.

By assumption Step 1 runs in time $\mathcal{O}(f(g+l)\cdot n^c)$ for some constant c, Step 2 runs in time $\mathcal{O}(n^d)$ for some constant d, since we can check in polynomial time whether $P \in \mathcal{C}$. Step 3 runs in time $\mathcal{O}(n\cdot(\lceil\log_2 l\rceil+2)\cdot f(g+l)\cdot n^c)$. Hence, Steps 1–3 run in time $\mathcal{O}(f(g+l)\cdot n^c + n^d + f(g+l)\cdot(\lceil\log_2 l\rceil+2)\cdot n^{c+1}) = \mathcal{O}(n^e\cdot(f(g+l)+f(g+l)\lceil\log_2 l\rceil))$ for some constant e. Since a binary search tree with l leaves has exactly $l-1$ inner nodes, we have to run the recursion in Step 4 for at most $l-2$ times. Thus, the algorithm runs in time $\mathcal{O}(n^e\cdot l\cdot(f(g+l)+f(g+l)\lceil\log_2 l\rceil)) = \mathcal{O}(n^e\cdot f'(g+l))$ for some computable function f'. □

Proof of Proposition 17

Proof. Let $\mathcal{C}$ be a class of programs, P a program, and $g, l > 0$ integers. We consider g as bound for the Gallo-Scutellà parameter and l for the maximum number of leaves. If $l \leq 2$ we can check in polynomial time by definition of a loss-free kernelization whether $P \in \mathcal{C}$. Assume $l \geq 2$. We compute by means of the loss-free kernelization in polynomial-time a set $K \subseteq \text{at}(P)$ (if it exists). If P has a $\mathcal{C}$-backdoor tree $BT = (T, \chi)$ of Gallo-Scutellà parameter g and at most l leaves, then $\text{at}(BT)$ is a strong $\mathcal{C}$-backdoor, $g \leq \text{at}(BT)$, and $\#\text{leaves}(BT) \leq l-1$. Since the problem STRONG $\mathcal{C}$-BACKDOOR DETECTION has a loss-free kernelization, the size of K is bounded by l and the number of binary decision trees T where $\text{at}(BT) \subseteq K$ is bounded by some computable function of l. Finally, we check for at most $f(l)\cdot|K|$ times by testing at most $f(l)$ times whether $P_\tau \in \mathcal{C}$ and $|\tau^{-1}| \leq g$ for the leaves in T and hence determine whether BT is a $\mathcal{C}$-backdoor tree of P of Gallo-Scutellà parameter g and of at most l leaves. Hence the proposition follows. □

ALGORITHM 1: $\mathcal{C}$-BDTREECOMP(P, g, l)

Input: A disjunctive program P, an integer g, an integer l, and a assignment τ.

Output: A $\mathcal{C}$-backdoor tree of Gallo-Scutellà parameter at most g and at most l leaves or None if no such $\mathcal{C}$-backdoor tree exists.

1. Return None
   if $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) returns *No* for input P, g, and l.
2. Return the backdoor tree (T, χ) where $T = (\{r\}, \emptyset, r)$ and $\chi(r) = \emptyset$ if $P \in \mathcal{C}$.
3. For each atom $a \in \text{at}(P)$ carry out the following steps:
   (a) Let $i := 0$, $j := l - 1$, and τ_1 be the assignment that assigns $\tau_1(a) = 1$
   (b) Decide $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) for P_{τ_1} and integers $g - 1$ and j
      i. Proceed with Step 3c if the answer is *Yes*.
      ii. Proceed with the next atom in Step 3 if the answer is *No*.
   (c) Let $m := \lfloor \frac{i+j}{2} \rfloor$
   (d) Let $l_1 := j$ and proceed with Step 3g if $i = m$ or $j = m$
   (e) Decide $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) for P_{τ_1} and integers $g - 1$ and m
      i. Let $j := m$ if the answer is *Yes*.
      ii. Let $i := m$ if the answer is *No*.
      iii. Proceed with Step 3c
   (f) Let τ_0 be the assignment that assigns $\tau_0(a) = 0$.
   (g) Decide $\mathcal{C}$-BACKDOOR-TREE DETECTION(GS,LEAVES) for P_{τ_0}, and integers g and $l - l_1$.
      i. Proceed with Step 4 if the answer is *Yes*.
      ii. Proceed with the next atom in Step 3 if the answer is *No*.
4. Compute BT_1 using $\mathcal{C}$-BDTREECOMP$(P_{\tau_1}, g - 1, m)$ and compute BT_0 using $\mathcal{C}$-BDTREECOMP$(P_{\tau_0}, g, l - m)$
5. Return $BT = (T, \chi)$ from $BT_1 = (T_1, \chi_1)$ where $T_1 = (N_1, E_1, r_1)$ and $BT_0 = (T_0, \chi_0)$ where $T_0 = (N_0, E_0, r_0)$ as follows:
   - let r be a fresh node, i.e., $r \notin (N_1 \cup N_0)$,
   - $N := N_1 \cup N_0 \cup \{r\}$, $E := E_1 \cup E_0 \cup \{(r, r_1), (r, r_0)\}$, $T := (N, E, r)$, and
   - $\chi(t) := \begin{cases} \chi_1(t) \cup \{a\}, & \text{if } t \in N_1, \\ \chi_0(t) \cup \{\bar{a}\}, & \text{if } t \in N_0, \end{cases}$

Proof of Lemma 18

Proof. Let P be a program and k an integer. According to results by Fichte and Szeider [15] the set $X \subseteq \text{at}(P)$ is a strong **Horn**-backdoor of P if and only if X is a vertex cover of the negation

dependency graph N_p. Thus, we consider kernelizations of VERTEX COVER for the input graph N_p. In fact the well-known algorithm by Buss and Goldsmith [8] provides a kernelization with kernels of size $k^2 + k$. Buss' kernelization works as follows: Consider instance (N_p, k). Let $X \subseteq V$ be the set of vertices with more than k neighbors. If $X > k$ then output an arbitrary instance $(I', 1) \notin$ VERTEX COVER (as $(I, k) \notin$ VERTEX COVER).Otherwise, consider the instance (N'_p, k') where N'_p is obtained from N_p by removing the vertices in X and all isolated vertices, and $k' = k - |X|$. Since each vertex in V has at most k neighbors, a vertex cover of N'_p of size k' can cover at most $k \cdot k'$ edges. Hence, there can be at most $k'(k+1) \leq k^2+k$ vertices. Then if N'_p contains more than k^2+k vertices return $(I', 1) \notin$ VERTEX COVER (as $(I, k) \notin$ VERTEX COVER). Otherwise, return (N'_p, k') and the set X. It remains to observe that the algorithm runs in time $\mathcal{O}(k + \|N_p\|)$, and yields a loss-free kernel of size at most $k^2 + k$ as $X' \subseteq N'_P \cup X$ for every minimal strong $\mathcal{C}$-backdoor X' of size at most k. Consequently, the algorithm is a loss-free kernelization and the lemma sustains. □