Answer Set Solver Backdoors

Emilia Oikarinen[1] and Matti Järvisalo[2]

[1] HIIT & Department of Information and Computer Science, Aalto University, Finland
[2] HIIT & Department of Computer Science, University of Helsinki, Finland

Abstract. The concept of backdoor variables offers a generic notion for providing insights to the surprising success of constraint satisfaction solvers in solving remarkably complex real-world instances of combinatorial problems. We study backdoors in the context of answer set programming (ASP), and focus on studying the relative size of backdoors in terms of different state-of-the-art answer set solving algorithms. We show separations of the ASP solver families in terms of the smallest existing backdoor sets for the solvers.

1 Introduction

Answer set programming (ASP) [28,4] offers an expressive rule-based declarative language for conveniently modelling hard combinatorial problems, together with highly efficient solver technology for finding solutions (answer sets) to the rule-based constraint models. Answer set solver technology [21,30,31,2,19,26,22,15,1] builds on the success of Boolean satisfiability (SAT) [3] solving techniques (DPLL [8,7], CDCL [6,27]) and implements additional inference mechanisms for native reasoning over answer set programs, most notably, well-foundedness checking. While advances in ASP and SAT solvers have improved our ability to efficiently solve and reason over a remarkably wide range of important real-world problems, our understanding for the fundamental reasons for this success, however, is still somewhat lacking. The concept of backdoor variables, as introduced originally by Williams, Gomes, and Selman [32], offers a generic notion for providing insights to the surprising success of constraint satisfaction solvers in solving remarkably large and complex real-world instances of combinatorial problems. Informally, a backdoor set B of variables is a subset of the variables in a problem instance, such that a systematic search procedure needs to non-deterministically assign values (branch) only on the variables in B in order to decide the instance. Given that a search procedure has a small backdoor to a problem instance, the procedure can in principle decide the instance efficiently.

In this paper, we study backdoors in the context of answer set programming. While several other extensions of the basic definition of backdoors have been proposed and studied [9,10,29], only recently there has been work on backdoors in the context of ASP, and mainly from the parameterized complexity perspective [13]. In contrast, we focus on studying the relative size of backdoors in terms of *practical* state-of-the-art answer set solving algorithms. Closely following the techniques implemented in different solvers, we formalize different solver variants in terms of three dimensions: (i) *well-foundedness*, (ii) *conflict-learning*, and (iii) *branching*. As explained later, these dimensions reflect algorithmic choices in answer set solvers [21,30,2,31,25,19,26,22,15,1].

Importantly, different choices along the three dimensions allow for a more fine-grained analysis than that possible in the context of SAT solvers [10], especially due to the fact that dimensions (i) and (iii) do not have direct counterparts in SAT solving. As the basis of our analysis, we define *answer set solver backdoors* extending related backdoors concepts from SAT along the three dimensions, opening a new point of view to analysing the effectiveness on different answer set solving techniques in terms of problem structure. As the main results, we show up to *exponential* separations of the ASP solver families characterized by different choices along the three dimensions in terms of the smallest existing backdoor sets for the solvers, both on satisfiable and unsatisfiable families of answer set programs.

2 Preliminaries

Answer Set Semantics A normal logic program (or an *answer set program* in this context) Π over a finite set$\mathcal{P}$ of atoms consists of a finite set of *rules* of the form r : $h \leftarrow p_1, \ldots, p_m, \sim p_{m+1}, \ldots, \sim p_n$, where $0 < m \leq n$, $h \in \mathcal{P} \cup \{\bot\}$ (where $\bot$ stands for falsity), and, for each $i = 1..n$, $p_i \in \mathcal{P}$. A rule r consists of a *head*, $\mathsf{head}(r) = h$, and a *body*, $\mathsf{body}(r) = \{p_1, \ldots, p_m, \sim p_{m+1}, \ldots, \sim p_n\}$. The symbol "$\sim$" is *default negation*. A *default literal* is an atom p or its default negation $\sim p$. The set of atoms appearing in program Π is denoted by $\mathsf{atom}(\Pi)$. The set of bodies (resp. heads) in Π is $\mathsf{body}(\Pi) = \{\mathsf{body}(r) \mid r \in \Pi\}$ (resp. $\mathsf{head}(\Pi) = \{\mathsf{head}(r) \mid r \in \Pi\}$). For each atom $p \in \mathsf{head}(\Pi)$, let $\mathsf{body}(p) = \{\mathsf{body}(r) \mid r \in \Pi,\ \mathsf{head}(r) = p\}$ to represent the set of rules bodies that share the same head p. For a rule r, let $\mathsf{body}(r)^+ = \{p_1, \ldots, p_m\}$ and $\mathsf{body}(r)^- = \{p_{m+1}, \ldots, p_n\}$ denote the sets of positive and negative (default negated) atoms in $\mathsf{body}(r)$, respectively.

In ASP, we are interested in *stable models* [17] (or *answer sets*) of a given program Π. A truth assignment for an answer set program Π is a function τ that maps atoms in Π to $\{0, 1\}$. An assignment τ extends implicitly to default literals by requiring that $\tau(\sim p) = 1 - v$ for each atom p such that $\tau(p) = v \in \{0, 1\}$. An assignment τ can be extended over a set of literals β: $\tau(\beta) = 1$ if $\tau(p) = 1$ for all $p \in \beta^+$ and $\tau(p) = 0$ for each $p \in \beta^-$; otherwise $\tau(\beta) = 0$. τ satisfies a rule $r \in \Pi$ iff $\tau(\mathsf{body}(r)) = 1$ implies $\tau(\mathsf{head}(r)) = 1$. An assignment τ that satisfies all rules of a program Π is an answer set of Π if and only if there is no complete assignment τ' distinct from τ such that (i) $\tau(p) = 0$ implies $\tau'(p) = 0$, and (ii) τ' satisfies each rule in the program

Relation to Boolean Satisfiability (SAT) For a Boolean variable x, there are two *literals*, the positive literal x and the negative literal $\neg x$. A *clause* is a disjunction of literals and a CNF formula a conjunction of clauses. A truth assignment for a CNF formula F is a function τ that maps variables in F to $\{0, 1\}$. An assignment τ extends implicitly to literals by requiring that $\tau(\neg x) = 1 - v$ for each variable x such that $\tau(x) = v \in \{0, 1\}$. A clause C is satisfied by τ if $\tau(l) = 1$ for some literal $l \in C$. An assignment τ satisfies F if it satisfies every clause in F.

Clark [5] defines the completion of a given answer set program Π, mapping Π to a CNF formula $\mathsf{comp}(F)$ as follows. For a body $\beta = \{p_1, \ldots, p_m, \sim p_{m+1}, \ldots, \sim p_n\} \in \mathsf{body}(\Pi)$, let $B(\beta)$ stand for $\beta \leftrightarrow p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \cdots \wedge \neg p_n$ interpreted as a

CNF formula where β and all p_i's are viewed as Boolean variables.[3] $B(\beta)$ characterizes that (i) the body of a rule is 1 if all its literals are 1, and (ii) some literal in the body must be 0 if the head is 0 For an atom $p \in \mathsf{head}(\Pi)$ with $\mathsf{body}(p) = \{\beta_1, \ldots, \beta_k\}$, $H(p)$ stands for $p \leftrightarrow \beta_1 \vee \cdots \vee \beta_k$, characterizing that (i) a head atom must be 0 if all of the bodies of the rules defining it are 0, and (ii) the head atom must be 1 if there is a rule such that the body is 1. The completion of Π is then the CNF formula $\mathsf{comp}(\Pi) = \bigwedge_{\beta \in \mathsf{body}(\Pi)} B(\beta) \wedge \bigwedge_{p \in \mathsf{head}(\Pi)} H(p)$. For any Π, the satisfying assignments of $\mathsf{comp}(\Pi)$ capture the *supported models* of Π. In general, every answer set of Π is also a supported model. However, the supported models coincide with answer sets only when Π is *tight*[4] [12,11]. If Π is tight, an assignment τ satistying $\mathsf{comp}(\Pi)$ corresponds to the answer set $A_\tau(p) = \tau(p)$ if and only if $p \in \mathsf{atom}(\Pi)$ of Π, obtained basically by restricting τ to $\mathsf{atom}(\Pi)$.

In case Π is non-tight, A_τ might not be an answer set of Π. This is due to *loops* in Π that induce cyclic support among atoms assigned to 1 in τ. *Loop formulas* can be used to prohibit such cyclic support. For a given program Π and a set $U \subseteq \mathsf{atom}(\Pi)$ of atoms, the set of *external bodies* of U in Π, denoted by $EB(U, \Pi)$, is $\{\mathsf{body}(r) \mid r \in \Pi,\ \mathsf{head}(r) \in U,\ \mathsf{body}(r)^+ \cap U = \emptyset\}$. The loop formula induced by U for Π, where $EB(U, \Pi) = \{\beta_1, \ldots, \beta_k\}$, is $L(U, \Pi) = \bigwedge_{p \in U}(p \rightarrow \beta_1 \vee \cdots \vee \beta_k)$. For any non-tight program Π and satisfying truth assignment τ for $\mathsf{comp}(\Pi)$, we know that A_τ is an answer set of Π if and only if τ satisfies the loop formulas induced by each non-empty $U \subseteq \mathsf{atom}(\Pi)$, i.e., all loop formulas for Π: $L(\Pi) = \bigwedge_{\emptyset \subset U \subseteq \mathsf{atom}(\Pi)} L(U, \Pi)$. There is an exponential number of loop formulas in the worst-case [24], which makes the direct approach of answer set solving Π by satisfiability checking $\mathsf{comp}(\Pi) \wedge L(U, \Pi)$ infeasible in practice.

3 Search for Answer Sets

We now describe formalizations of answer set solver variants, closely related to actual available answer set solver implementations, which we analyze in terms of the relative size of backdoors. We base the formalization on the fact that, for any program Π, the answer sets of Π correspond to the satisfying assignments for $\mathsf{comp}(\Pi) \wedge L(\Pi)$.[5] Indeed, the various answer set solvers available today can be characterized as implementing variants of the classical the Davis–Putnam–Logemann–Loveland (DPLL) procedure [8,7] or the *conflict-driven clause learning* (CDCL) algorithm [6,27], with additional propagation techniques for performing well-foundedness checks over $L(\Pi)$.

DPLL implements a standard backtracking depth-first search for satisfiability, with *unit propagation* over clauses for extending deterministically the current partial assignment τ making decisions (branching) on variables. Unit propagation over F and τ refers applying the following rules until fixpoint: if there is a clause $(l \vee l_1 \vee \cdots \vee l_k)$ such that $\tau(l_i) = 0$ for all $i = 1..k$, let $\tau(l) = 1$. Unit propagation on the completion $\mathsf{comp}(\Pi)$ and loop formulas $L(\Pi)$ of an answer set program Π is tightly connected with native

[3] When clear, we liberally refer to atoms and Boolean variables interchangeably.

[4] A logic program Π is tight iff its dependency graph has no positive cycles.

[5] $L(\Pi)$ is not generated explicitly by solvers: the native propagation rules equivalent to unit propagation on $L(\Pi)$ and τ can be applied on Π to check for conflicts.

propagation rules [16] on the level of the answer set program. For a thorough discussion characterizing ASP propagation in terms of unit propagation, see [14].

CDCL search is also based on making decisions and employing unit propagation. However, CDCL in contrast to DPLL does not implement standard backtracking, but rather uses a conflict analysis scheme for learning *conflict clauses* from seen conflicting assignments, and performs *non-chronological backtracking* after learning a conflict clause to erase more than one decision from the current assignment. For detailed accounts on conflict-driven answer set solving, see [23,15], and e.g. [18,20] for accounts on the relation of ASP and SAT solving.

Concretely, given a program Π as input, our formalizations of answer set solvers differ in three dimensions:

1. whether well-foundedness checks over the loop formulas $L(\Pi)$ are performed *eagerly* (**EWF**) after each decision during search under the current partial assignment τ, or *lazily* (**LWF**) after reaching a satisfying assignment for $\mathsf{comp}(\Pi)$);
2. whether a form of *conflict learning* is employed (**CL**), in analogy with CDCL, or not (**noCL**), in analogy with DPLL;
3. whether the solver makes decisions on all atoms in $\mathsf{comp}(\Pi)$ (**B**), or only on atoms in $\mathsf{atom}(\Pi)$ (**noB**), i.e., *not* on the atoms of the form β, which would correspond to making decision on the bodies of rules);

yielding eight solvers variants $\{(X, Y, Z)$ where $X \in \{\textbf{EWF}, \textbf{LWF}\}, Y \in \{\textbf{CL}, \textbf{noCL}\}$, and $Z \in \{\textbf{B}, \textbf{noB}\}$. The different variants are closely related to techniques implemented in state-of-the-art answer set solvers. For examples, the DLV [21] and Smodels [30] systems relate with (**EWF**, **noCL**, **noB**); Nomore++ [2] with (**EWF**, **noCL**, **B**); Smodels$_{cc}$ [31] (a conflict-learning variant of Smodels) with (**EWF**, **CL**, **noB**); the SAT-based answer set solvers ASSAT [25], Cmodels [19], and SUP [22], incorporating variants of **LWF**, relate with (**LWF**, **CL**, **B**); and finally, Clasp [15] relates most closely with (**EWF**, **CL**, **B**), together with WASP [1] and SAG [26], both of which employ forms of (partial) **EWF**.

4 Backdoors

We continue by defining backdoors in the context of answer set solving. In general, backdoors are defined in terms of tractable (polynomial-time decidable) subclasses $\mathcal{C}$, which may either syntactically-defined classes such as Horn programs or 2-SAT, or, more closely related to solvers, subclasses defined via *subsolvers*, such as unit propagation in the context of SAT. Here our focus is on the latter type of backdoors in the context of answer set solving.

We start with the traditional definition for backdoors w.r.t. using unit propagation as the subsolver. This definition fits with the non-conflict-learning DPLL-style solvers. Given a CNF formula F on variables X, a subset of variables $B \subseteq X$, and a (partial) truth assignment $\tau : B \to \{0, 1\}$, let $F|_\tau$ denote the simplified formula obtained by assigning values to variables in B according to τ.

Definition 1. *[32] Given a CNF formula F on variables X, a subset $B \subseteq X$ of variables is a* strong unit-propagation backdoor *if for every truth assignment $\tau : B \to \{0, 1\}$, unit propagation on F and τ returns a satisfying assignment for F or concludes that $F|_\tau$ is unsatisfiable.*

We will from now on refer to strong unit-propagation backdoors simply as strong backdoors. Due to the differences in how DPLL and CDCL solvers explore search trees, the traditional definition of backdoors does not as such fit with CDCL. Hence in [10] a more natural definition of *learning-sensitive backdoors* was proposed for the context of CDCL solvers.

Definition 2. *[10] Given a CNF formula F on variables X, a subset of variables $B \subseteq X$ is a* learning-sensitive (unit-propagation) backdoor *for F if there exists a search tree exploration order such that a CDCL SAT solver branching only on the variables in B, with this order and with unit propagation as the sub-solver at the leaves of the search tree, either finds a satisfying assignment for F or proves that F is unsatisfiable.*

Backdoors in ASP As natural counterparts of strong backdoors and learning-sensitive backdoors in SAT, we now define $(X, \textbf{noCL}, Z)$-backdoors and $(X, \textbf{CL}, Z)$-backdoors, respectively, in the context of ASP. We start with $(X, \textbf{noCL}, Z)$-backdoors, which serve as the counterparts of strong backdoors. Similarly as for CNF formulas, $\Pi|_\tau$ denotes the simplified program obtained by assigning values to atoms according to τ.

Definition 3. *Given an answer set program Π, a subset $B \subseteq \mathsf{atom}(\Pi) \cup \mathsf{body}(\Pi)$ is a $(X, \textbf{noCL}, Z)$-*backdoor*, where $X \in \{\textbf{\textit{EWF}}, \textbf{\textit{LWF}}\}$ and $Z \in \{\textbf{\textit{B}}, \textbf{\textit{noB}}\}$, if the following conditions hold:*

- *If $X = \textbf{\textit{EWF}}$, then for every truth assignment $\tau : B \to \{0, 1\}$, unit propagation on $\mathsf{comp}(\Pi) \wedge L(\Pi)$ and τ returns a satisfying assignment for $\Pi|_\tau$ or concludes that $\Pi|_\tau$ is unsatisfiable.*
- *If $X = \textbf{\textit{LWF}}$, then for every truth assignment $\tau : B \to \{0, 1\}$, unit propagation on $\mathsf{comp}(\Pi)$ and τ returns a satisfying assignment for $\mathsf{comp}(\Pi)$ or concludes that $\Pi|_\tau$ is unsatisfiable.*
- *If $Z = \textbf{\textit{noB}}$, then $B \subseteq \mathsf{atom}(\Pi)$.*

Since $\mathsf{comp}(\Pi)$ over-approximates the answer sets of Π, in the case $X = \textbf{LWF}$, unit propagation can restricted to $\mathsf{comp}(\Pi)$ without loss of generality: If unit propagation on $\mathsf{comp}(\Pi)$ and τ determines that $\mathsf{comp}(\Pi)|_\tau$ is unsatisfiable, then $\Pi|_\tau$ is also unsatisfiable. If unit propagation on $\mathsf{comp}(\Pi)$ and τ returns a satisfying assignment for $\mathsf{comp}(\Pi)$, we know that the assignment is either an answer set of Π, or unit propagation on $\mathsf{comp}(\Pi) \wedge L(\Pi)$ and τ concludes unsatisfiability.

We continue by defining $(X, \textbf{CL}, Z)$-backdoors as natural counterparts of learning-sensitive backdoors in SAT.

Definition 4. *Given an answer set program Π, a subset $B \subseteq \mathsf{atom}(\Pi) \cup \mathsf{body}(\Pi)$ is a $(X, \textbf{CL}, Z)$-*backdoor *for Π if there exists a search tree exploration order for the $(X, \textbf{\textit{CL}}, Z)$-solver such that the following conditions hold:*

- *The solver branches only on the variables in B.*
- *The solver uses unit propagation on $\mathsf{comp}(\Pi) \wedge L(\Pi)$ when all variables in B are assigned.*
- *The solver either finds a satisfying assignment for Π or proves Π unsatisfiable.*

– *If $X = \textbf{\textit{LWF}}$, then the solver uses $L(\Pi)$ for unit propagation only when the current assignment is complete over* $\mathsf{atom}(\Pi) \cup \mathsf{body}(\Pi)$.
– *If $Z = \textbf{\textit{noB}}$, then $B \subseteq \mathsf{atom}(\Pi)$.*

Notice that, in contrast to $(X, \textbf{noCL}, Z)$-backdoors, here the additional unit propagation enabled by $L(\Pi)$ can play a critical role in terms of causing a conflict, which would then allow the solver to learn from the conflict. Thus, in connection with the lazy well-foundedness checking employed in SAT-based ASP solvers which employ CDCL SAT-solvers, in case $X = \textbf{LWF}$ unit propagation on $L(\Pi)$ is postponed until a complete assignment is reached on $\mathsf{comp}(\Pi)$ alone.

5 Analysis

As the main results of this paper, we will now analyze the relative size of (X, Y, Z)-backdoors that exist for different answer set solver variants. We begin with relatively simple observations.

Proposition 1. *The following claims hold for any program Π, $B \subseteq \mathsf{atom}(\Pi) \cup \mathsf{body}(\Pi)$, and $X \in \{\textbf{\textit{EWF,LWF}}\}$, $Y \in \{\textbf{\textit{CL}}, \textbf{\textit{noCL}}\}$, $Z \in \{\textbf{\textit{B}}, \textbf{\textit{noB}}\}$.*

(a) If B is a $(\textbf{\textit{LWF}}, Y, Z)$-backdoor for Π, then it is a $(\textbf{\textit{EWF}}, Y, Z)$-backdoor for Π.
(b) If B is a $(X, \textbf{\textit{noCL}}, Z)$-backdoor, then it is a $(X, \textbf{\textit{CL}}, Z)$-backdoor for Π.
(c) If B is a $(X, Y, \textbf{\textit{noB}})$-backdoor, then it is a $(X, Y, \textbf{\textit{B}})$-backdoor for Π.

Proposition 2. *For any tight program Π, $B \subseteq \mathsf{atom}(\Pi) \cup \mathsf{body}(\Pi)$, and $Y \in \{\textbf{\textit{CL}}, \textbf{\textit{noCL}}\}$, $Z \in \{\textbf{\textit{B}}, \textbf{\textit{noB}}\}$, it holds that B is a $(\textbf{\textit{LWF}}, Y, Z)$-backdoor for Π if and only if B is a $(\textbf{\textit{EWF}}, Y, Z)$-backdoor for Π.*

In many cases, bounds on the sizes of backdoors in SAT can be mapped into bounds on the sizes of backdoors in ASP. For this, we use a straightforward encoding $\mathsf{cnf2asp}(F)$ of a CNF formula F as

$$\{\bot \leftarrow \sim x_1, \ldots, \sim x_m, x_{m+1}, \ldots, x_n \mid (x_1 \vee \ldots \vee x_m \vee \neg x_{m+1} \vee \ldots \vee \neg x_n) \in F\} \cup$$
$$\{x \leftarrow \sim \hat{x} \mid \text{ variable } x \text{ occurs in } F\} \cup \{\hat{x} \leftarrow \sim x \mid \text{ variable } x \text{ occurs in } F\},$$

where the first set of rules encode the clauses in F, and the latter two enforce the classical semantics over the variables (atoms) using a new atom $\hat{x}$ for each x.

Proposition 3. *Let F be a CNF formula and B a subset of variables in F.*

(a) If B is a strong backdoor for F, then B is a $(X, \textbf{\textit{noCL}}, Z)$-backdoor for $\mathsf{cnf2asp}(F)$ for any $X \in \{\textbf{\textit{EWF,LWF}}\}$ and $Z \in \{\textbf{\textit{B}}, \textbf{\textit{noB}}\}$.
(b) If B is a learning-sensitive backdoor for F, then B is a $(X, \textbf{\textit{CL}}, Z)$-backdoor for $\mathsf{cnf2asp}(F)$ for any $X \in \{\textbf{\textit{EWF,LWF}}\}$ and $Z \in \{\textbf{\textit{B}}, \textbf{\textit{noB}}\}$.

Proposition 4. *For any CNF formula F, if the smallest strong (resp. learning-sensitive) backdoors for F are of size k, then the smallest $(X, \textbf{\textit{noCL}}, Z)$-backdoors (resp. $(X, \textbf{\textit{CL}}, Z)$-backdoors) for $\mathsf{cnf2asp}(F)$ are of size at least k for any $X \in \{\textbf{\textit{EWF,LWF}}\}$ and $Z \in \{\textbf{\textit{B}}, \textbf{\textit{noB}}\}$.*

In addition to being able to carry over results from SAT to ASP, in particular cases—especially, when conflict-learning is not enabled—results for unsatisfiable programs carry over to satisfiable programs, using the following transformation:

$$\mathsf{trsat}(\Pi) = \{e \leftarrow \sim d\} \cup \{d \leftarrow \sim e\} \cup \{\mathsf{head}(r) \leftarrow \mathsf{body}(r), \sim d \mid r \in \Pi\}.$$

This translation essentially encodes an exclusive-or choice between d and e, and each of the rules in Π is conditioned on $\sim d$.

Theorem 1. *For any unsatisfiable program Π for which the smallest $(X, \textbf{\textit{noCL}}, Z)$-backdoors are of size k, it holds that (i) $\mathsf{trsat}(\Pi)$ has an answer set, and that (ii) the smallest $(X, \textbf{\textit{noCL}}, Z)$-backdoors for $\mathsf{trsat}(\Pi)$ are at least of size k, for any $X \in \{\textbf{\textit{EWF,LWF}}\}$ and $Z \in \{\textbf{\textit{B}}, \textbf{\textit{noB}}\}$.*

Proof. (sketch) It is straightforward to verify that the assignment τ such that $\tau(d) = 1$ and $\tau(a) = 0$ for all $a \in \mathsf{atom}(\mathsf{trsat}(\Pi)) \setminus \{d\}$ is the unique answer set of $\mathsf{trsat}(\Pi)$. However, the smallest $(X, \textbf{noCL}, Z)$-backdoors for $\mathsf{trsat}(\Pi)$ can be shown to be at least of size k. □

We will next focus on analyzing the effects of different choices for $X \in \{\textbf{EWF}, \textbf{LWF}\}$ and $Y \in \{\textbf{CL}, \textbf{noCL}\}$ on the relative sizes of (X, Y, Z)-backdoors. We begin by focusing on unsatisfiable programs. First, we exploit a result from [10] for backdoors in SAT via the connections between backdoors in SAT and ASP we established in Sect. 5.

Theorem 2. *[10] There are unsatisfiable CNF formulas for which the smallest learning-sensitive backdoor are exponentially smaller than the smallest strong backdoors.*

This result carries over to ASP as follows.

Proposition 5. *There are unsatisfiable programs for which the smallest $(\textbf{\textit{LWF}}, \textbf{\textit{CL}}, \textbf{\textit{noB}})$-backdoors are exponentially smaller than the smallest $(\textbf{\textit{EWF}}, \textbf{\textit{noCL}}, \textbf{\textit{B}})$-backdoors.*

Proof. Take any CNF formula F witnessing Theorem 2. By Proposition 3, any smallest learning-sensitive backdoor B for F is a $(\textbf{LWF}, \textbf{CL}, \textbf{noB})$-backdoor for $\mathsf{cnf2asp}(F)$. By Theorem 2 and Proposition 4, the smallest $(\textbf{EWF}, \textbf{noCL}, \textbf{B})$-backdoors are exponentially larger than B. □

To compare the differences between lazy and eager propagation, we need to consider non-tight programs, thus involving a more fine-grained analysis than what is possible in the context of SAT. Interestingly, there are programs which have *exponentially* smaller $(\textbf{EWF}, \textbf{noCL}, \textbf{noB})$-backdoors than $(\textbf{LWF}, \textbf{CL}, \textbf{noB})$-backdoors; that is, lazy well-foundedness checking can cause an exponential blow-up in the size of backdoor.

Theorem 3. *There are unsatisfiable programs for which the smallest $(\textbf{\textit{EWF}}, \textbf{\textit{noCL}}, \textbf{\textit{noB}})$-backdoors are exponentially smaller than the smallest $(\textbf{\textit{LWF}}, \textbf{\textit{noCL}}, \textbf{\textit{B}})$-backdoors.*

Proof. (sketch) Consider the unsatisfiable program

$$\begin{aligned}\Pi_n = &\{f \leftarrow \sim f, \sim p_{i,1}, \ldots, \sim p_{i,n-1} \mid 1 \leq i \leq n\} \cup \\ &\{f \leftarrow \sim f, p_{i,k}, p_{j,k} \mid 1 \leq i, j \leq n, 1 \leq k \leq n-1, i \neq j\} \cup \\ &\{p_{i,k} \leftarrow p_{i,k} \mid 1 \leq i \leq n, 1 \leq k \leq n-1\},\end{aligned}$$

from [18], where $n = 2^m$ for some m. Notice that there is no external support for atoms $p_{i,k}$ (only rule with an atom $p_{i,k}$ in the head is a self-loop). This and the first rule cause the unsatisfiability of the program. Now $\{f\}$ is a (**EWF**, **noCL**, **noB**)-backdoor for Π_n (note that $\tau(p_{i,j}) = 0$ for all i, j can directly be propagated with **EWF**).

However, for (**LWF**, **CL**, **B**)-backdoor, in order to check the loops of the form $p_{i,k} \leftarrow p_{i,k}$, one must first have a complete truth assignment, which can be shown to require assigning at least n atoms. Thus, a (**LWF**, **noCL**, **B**)-backdoor has to be at least of size n. □

We now turn our attention to backdoors for satisfiable programs.

Theorem 4. *[10] There are satisfiable CNF formulas for which there are learning-sensitive backdoors are smaller than the smallest strong backdoors.*

Proposition 6. *There are satisfiable programs for which there are (**LWF**, **CL**, **noB**)-backdoors that are smaller than the smallest (**EWF**, **noCL**, **B**)-backdoors.*

Proof. Like Proposition 5, follows from Theorem 4 and Propositions 3 and 4. □

Theorem 5. *There are satisfiable programs for which the smallest (**EWF**, **noCL**, **noB**)-backdoors are exponentially smaller than the smallest (**LWF**, **CL**, **noB**)-backdoors.*

Proof. (sketch) Consider the program $\Pi_n = P_0 \cup P_1 \cup \cdots \cup P_n$ such that $P_0 = \{d \leftarrow \sim d, c\}$ and $P_i = \{a_i \leftarrow b_i.\ b_i \leftarrow a_i.\ e_i \leftarrow \sim a_i, \sim b_i, \sim c\}$ for all $1 \leq i \leq n$, where $n = 2^m$ for some m. The assignment $\tau(e_i) = 1$ for all $1 \leq i \leq n$ and $\tau(a) = 0$ for all $a \in \mathsf{atom}(\Pi_n) \setminus \{e_1, \ldots, e_n\}$ is the unique answer set of Π_n. Now, $\{d\}$ is a (**EWF**, **noCL**, **noB**)-backdoor for Π_n. However, the smallest (**LWF**, **noCL**, **noB**)-backdoors and (**LWF**, **CL**, **noB**)-backdoors can be shown to be of size $n + 1$. □

It turns out that a simple modification of the program Π_n in the proof of Theorem 5 gives an analogous results for unsatisfiable programs.

Theorem 6. *There are unsatisfiable programs for which the smallest (**EWF**, **noCL**, **noB**)-backdoors are exponentially smaller than the smallest (**LWF**, **CL**, **noB**)-backdoors.*

Proof. Consider the program $\Pi'_n = \Pi_n \cup \{\bot \leftarrow e_1, \ldots, e_n, \sim a_1, \ldots, \sim a_n, \sim b_1, \ldots, \sim b_n, \sim c, \sim d\}$. The additional rule disallows in a naive way exactly the only satisfying assignment for Π_n, being equivalent with the clause $\bigvee_{i=1}^{n} \neg e_i \vee \bigvee_{i=1}^{n} a_i \vee \bigvee_{i=1}^{n} b_i \vee c \vee d$. This rule has $3n + 2$ atoms in the body. Hence under any assignment over no more than $3n$ atoms, unit propagation cannot derive anything based on the rule. It follows that the arguments in the proof of Theorem 5 are valid also for Π'_n. □

Finally, we look at the question of whether allowing solvers to branch on the bodies of rules (i.e., on the β variables in $\mathsf{comp}(\Pi)$), or put another way, whether restricting solvers to branch only on atoms, has an effect on the size of smallest backdoors. It turns out that such a restriction can cause exponential separations.

Theorem 7. *There are unsatisfiable programs for which the smallest (**LWF**, **noCL**, **B**)-backdoors are exponentially smaller than the smallest (**EWF**, **noCL**, **noB**)-backdoors.*

Proof. (sketch) Consider the tight program $\Pi_k^n = \{f \leftarrow \sim f\} \cup P_1^n \cup \cdots \cup P_k^n$, where

$$P_i^n = \{f \leftarrow B_i \mid B_i = \{\sim a_{i,1}, \ldots, \sim a_{i,n}\}\} \cup \{a_{i,j} \leftarrow B_{i,j} \mid B_{i,j} = B_i \backslash \{\sim a_{i,j}\}, 1 \leq j \leq n\}$$

and $n = 2^k$. There are (**LWF**, **noCL**, **B**)-backdoors of size k for Π_k^n: Consider any set $\mathsf{BD} = \{f, B_1, \ldots, B_{j-1}, B_{j+1}, B_k\}$, a set that contains f and all except one bodies B_i from Π_k^n. On the other hand, it can be shown that the smallest (**EWF**, **noCL**, **noB**)-backdoor are of size at least $(k-1)\cdot(n-1)+1$. **!! IS THIS THE RIGHT CONSTANT, LOOK INTO iffalse** □

Additionally, we establish that in connection with eager well-foundedness checking, conflict-learning even when restricting branching on atoms can have exponentially exponentially smaller backdoors than without conflict-learning.

Theorem 8. *There are unsatisfiable programs for which the smallest* (***LWF***, ***CL***, ***noB***)*-backdoors are exponentially smaller than the smallest* (***EWF***, ***noCL***, ***B***)*-backdoors.*

Proof. Consider the CNF formula F_3 from [10, Proof of Theorem 3] with $k+3\cdot 2^k$ variables, having a learning-sensitive backdoor of size k. Now, by Proposition 3 the translation $\mathsf{cnf2asp}(F_3)$, which is a tight program, has a (**LWF**, **CL**, **noB**)-backdoor of size k. On the other hand, as shown in [10, Proof of Theorem 3], the smallest strong backdoors for F_3 are at least of size $2^k + k$. Thus, by Proposition 4 the smallest (**EWF**, **noCL**, **B**)-backdoors of $\mathsf{cnf2asp}(F_3)$ are at least of size $2^k + k$. □

6 Conclusions

Closely following the techniques implemented in different solvers, we introduced *answer set solver backdoors* defined with respect to three dimensions of answer set solving techniques. As the main results, we showed up to exponential separations of the resulting notions of answer set solver backdoors, which we believe to highlight intrinsic differences of the solver variants in terms of their behavior w.r.t. problem structure. Specific question related to this work remain open for future work. For example, not all of our separations are exponential; can the sub-exponential separations be strengthened into exponential ones?

Acknowledgements Work funded by Academy of Finland, grants 251170 (Centre of Excellence in Computational Inference Research), 250518 (EO), and 276412 (MJ).

References

1. Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F.: WASP: A native ASP solver based on constraint learning. In: Proc. LPNMR. LNCS, vol. 8148, pp. 54–66. Springer (2013)
2. Anger, C., Gebser, M., Linke, T., Neumann, A., Schaub, T.: The nomore++ system. In: Proc. LPNMR. LNCS, vol. 3662, pp. 422–426. Springer (2005)
3. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)

4. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM 54(12), 92–103 (2011)
5. Clark, K.: Negation as failure. In: Readings in nonmonotonic reasoning, pp. 311–325. Morgan Kaufmann Publishers (1987)
6. Darwiche, A., Pipatsrisawat, K.: Complete algorithms. In: Biere et al. [3], pp. 99–130
7. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem proving. Communications of the ACM 5(7), 394–397 (1962)
8. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the ACM 7(3), 201–215 (1960)
9. Dilkina, B.N., Gomes, C.P., Malitsky, Y., Sabharwal, A., Sellmann, M.: Backdoors to combinatorial optimization: Feasibility and optimality. In: Proc. CPAIOR. LNCS, vol. 5547, pp. 56–70. Springer (2009)
10. Dilkina, B.N., Gomes, C.P., Sabharwal, A.: Tradeoffs in the complexity of backdoors to satisfiability: dynamic sub-solvers and learning during search. Ann. Math. Artif. Intell. 70(4), 399–431 (2014)
11. Erdem, E., Lifschitz, V.: Tight logic programs. Theory and Practice of Logic Programming 3(4-5), 499–518 (2003)
12. Fages, F.: Consistency of Clark's completion and existence of stable models. Journal of Methods of Logic in Computer Science 1, 51–60 (1994)
13. Fichte, J.K., Szeider, S.: Backdoors to normality for disjunctive logic programs. In: Proc. AAAI. AAAI Press (2013)
14. Gebser, M., Schaub, T.: Characterizing ASP inferences by unit propagation. In: ICLP Workshop on Search and Logic: Answer Set Programming and SAT, Seattle, August 16, 2006. pp. 41–56 (2006)
15. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. 187, 52–89 (2012)
16. Gebser, M., Schaub, T.: Tableau calculi for logic programs under answer set semantics. ACM Trans. Comput. Log. 14(2), 15 (2013)
17. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proc. ICLP/SLP'88. pp. 1070–1080. MIT Press (1988)
18. Giunchiglia, E., Leone, N., Maratea, M.: On the relation among answer set solvers. Ann. Math. Artif. Intell. 53(1-4), 169–204 (2008)
19. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning 36(4), 345–377 (2006)
20. Järvisalo, M., Oikarinen, E.: Extended ASP tableaux and rule redundancy in normal logic programs. Theory and Practice of Logic Programming 8(5–6), 691–716 (2008)
21. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic 7(3), 499–562 (2006)
22. Lierler, Y.: Abstract answer set solvers. In: Proc. ICLP. LNCS, vol. 5366, pp. 377–391. Springer (2008)
23. Lierler, Y.: Abstract answer set solvers with backjumping and learning. TPLP 11(2-3), 135–169 (2011)
24. Lifschitz, V., Razborov, A.: Why are there so many loop formulas? ACM Transactions on Computational Logic 7(2), 261–268 (2006)
25. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence 157(1–2), 115–137 (2004)
26. Lin, Z., Zhang, Y., Hernandez, H.: Fast SAT-based answer set solver. In: Proc. AAAI. pp. 92–97. AAAI Press (2006)
27. Marques-Silva, J.P., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere et al. [3], pp. 131–153

28. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence 25(3–4), 241–273 (1999)
29. Samer, M., Szeider, S.: Backdoor sets of quantified boolean formulas. In: Proc. SAT. LNCS, vol. 4501, pp. 230–243. Springer (2007)
30. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence 138(1–2), 181–234 (2002)
31. Ward, J., Schlipf, J.: Answer set programming with clause learning. In: Proc. LPNMR. LNCS, vol. 2923, pp. 302–313. Springer (2004)
32. Williams, R., Gomes, C.P., Selman, B.: Backdoors to typical case complexity. In: Proc. IJCAI. pp. 1173–1178. Morgan Kaufmann (2003)