ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230277

On the Structural Complexity of Grounding – Tackling the ASP Grounding Bottleneck via Epistemic Programs and Treewidth

Viktor Besin[a], **Markus Hecher**[b;*] **and Stefan Woltran**[a]

[a]TU Wien, Austria
[b]Massachusetts Institute of Technology, United States
ORCiD ID: Markus Hecher https://orcid.org/0000-0003-0131-6771,
Stefan Woltran https://orcid.org/0000-0003-1594-8972

Abstract. Answer Set Programming is widely applied research area for knowledge representation and for solving industrial domains. One of the challenges of this formalism focuses on the so-called grounding bottleneck, which addresses the efficient replacement of first-order variables by means of domain values. Recently, there have been several works in this direction, ranging from lazy grounding, hybrid solving, over translational approaches. Inspired by a translation from non-ground normal programs to ground disjunctive programs, we attack the grounding bottleneck from a more general angle. We provide a polynomial reduction for grounding disjunctive programs of bounded domain size by reducing to propositional epistemic logic programs (ELPs). By slightly adapting our reduction, we show new complexity results for non-ground programs that adhere to the measure treewidth. We complement these results by matching lower bounds under the exponential time hypothesis, ruling out significantly better algorithms.

1 Introduction

Answer set programming (ASP) [9, 18] is a modeling and solving framework that can be seen as an extension of propositional satisfiability (SAT), where knowledge is expressed by means of rules comprising a *(logic) program*. The solutions to those programs are sets of atoms, called *answer sets*, that obey every rule. With its rich first-order like language ASP presents an appealing tool for solving problems related to knowledge representation and reasoning and industrial applications for which efficient systems are readily available [18, 10]. These systems are mainly built on a ground-and-solve technique, replacing variables by domain constants and feeding the resulting ground program into an ASP solver. For certain types of problems, this approach leads to the well-known ASP *grounding bottleneck* [34], i.e., where the instantiation of rules yields an exponentially larger program that is beyond the capabilities to be processed by the solver efficiently. This is indeed not surprising, as already the evaluation of the well-known fragment of normal programs is NEXPTIME-complete [11]. However, for constant predicate arities, lower complexity results are known [12]. Indeed, under this assumption the hardness of answer set existence increases by one level of the polynomial hierarchy, when switching from ground to non-ground programs.

Still, in general the increased complexity of evaluating non-ground programs is the root cause of the ASP grounding bottleneck. While recent works on *body-decoupled grounding* [3] have shown that reductions between non-ground and ground programs can reduce grounding time and size for normal programs, the complexity of non-ground disjunctive programs (Σ_3^P-hard) does not allow for direct translations to ground ASP (up to Δ_3^P-complete). However, over the time ASP has been significantly extended to more expressive formalisms. Epistemic logic programs (ELPs) depict such an extension, which are capable of reasoning over multiple worlds [19], where – depending on whether objections are possible or known – certain consequences have to be derived. Interestingly, the evaluation of (ground) ELPs is Σ_3^P-complete, enabling an efficient reduction from non-ground ASP under bounded predicate arities.

In this work, we take a step further and initiate the study of grounding via *translation to formalisms beyond ASP*. We present a novel reduction that adheres to body-decoupled grounding and translates non-ground disjunctive programs of bounded predicate arity to ground ELPs. Then, to deal with the high complexity, we consider the influence of structure, thereby focusing on the prominent measure treewidth. In our studies we obtain a *surprising new result* for treewidth and the well-known fragment of normal programs: It turns out that for non-ground programs, the consistency of cyclic (normal) programs is of similar hardness as for non-cyclic (tight) programs. This is indeed in stark contrast to existing results for the ground case, where high cyclicity of programs is a known source of hardness [28, 25], also for treewidth [20, 15, 21].

Contributions. Our contributions and relations to existing results are highlighted in Table 1 and detailed below.

1. We define a new translational-based grounding approach that works for *any non-ground program*, thereby relying on the expressive formalism of epistemic programs. It is exponential only in the predicate arity (polynomial for fixed arity).
2. Then, we analyze the consequences and impact of structure in our approach. To this end, we propose a formal condition of *structure preservation* for non-ground programs that can be even fulfilled for

* Corresponding author; email: hecher@mit.edu. Authors contributed equally to this research.

Please contact the corresponding author for any appendices or supplementary material mentioned in the paper.

rules with larger bodies. It turns out that this condition is already sufficient for us to obtain *precise runtimes by adapting* our grounding approach for treewidth.

3. We complete these upper bounds by *lower bounds under the exponential time hypothesis (ETH)* [24], where we essentially rule out significant improvements of our reduction. Thereby we finally complete the complexity picture for non-ground logic programs and treewidth. Surprisingly, normal programs are of similar hardness as tight programs (non-ground case). This is different to the ground case, see Table 1 (last 2 rows).

Related Work. Eiter, Faber, and Fink established complexity results [12] for the consistency of non-ground programs under bounded predicate arities, see also Table 1. In the context of meta-programming, these results have been used for space-efficient ASP evaluation [13]. The literature distinguishes different attempts to efficient grounding, ranging from classical approaches [1, 27] numerical techniques [23] and lazy grounding [35], over ASP modulo theory, e.g., [2], and methods based on structural measures (treewidth) [4, 10, 7, 30]. Treewidth has been considered for ELPs [22], also in practice, e.g., [5].

	Tight	**Normal**	**Disjunctive**	**Epistemic**
Ground	NP-c	NP-c	Σ_2^{P}-c	$\xrightarrow{\mathcal{R}}$ Σ_3^{P}-c
Non-Ground*	Σ_2^{P}-c	Σ_2^{P}-c $\nearrow$	Σ_3^{P}-c $\nearrow$	-
Ground [tw]	$2^{\Theta(tw)}$	$2^{\Theta(tw\cdot\log(tw))}$	$2^{2^{\Theta(tw)}}$	$2^{2^{2^{\Theta(tw)}}}$
Non-Ground [tw]*	$\mathbf{2^{d^{\Theta(tw)}}}$	$\mathbf{2^{d^{\Theta(tw)}}}$ $\xrightarrow{\mathcal{R}^{\text{Normal}}_{\text{tw}}}$	$\mathbf{2^{2^{d^{\Theta(tw)}}}}$ $\xrightarrow{\mathcal{R}_{\text{tw}}}$	-

Table 1: Complexity results of consistency problem variants, where each column gives the corresponding program fragment and each row shows the respective problem (asterisk indicates fixed predicate arities). Runtimes are tight under ETH and neglect polynomial factors. **Bold-face entities** indicate new results and reductions; d refers to the domain size and tw is the treewidth of the primal graph.

2 Preliminaries

Ground ASP. Let ℓ, m, n be non-negative integers such that $\ell \leq m \leq n$; $a_1, \ldots, a_n$ be distinct propositional atoms. A *(disjunctive) program (LP) P* is a set of *(disjunctive) rules*

$$a_1 \vee \ldots \vee a_\ell \leftarrow a_{\ell+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n.$$

For a rule r, we let $H_r := \{a_1, \ldots, a_\ell\}$, $B_r^+ := \{a_{\ell+1}, \ldots, a_m\}$, and $B_r^- := \{a_{m+1}, \ldots, a_n\}$. 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 $\mathrm{at}(P) := \bigcup_{r\in P} \mathrm{at}(r)$. A rule r is *normal* if $|H_r| \leq 1$ and a program P is *normal* if all its rules are normal. The *dependency graph* $\mathcal{D}_P$ is the directed graph defined on the set $\bigcup_{r\in P} H_r \cup B_r^+$ of atoms, where for every rule $r \in P$ two atoms $a \in B_r^+$ and $b \in H_r$ are joined by edge (a, b). P is *tight* if $\mathcal{D}_P$ has no directed cycle [14].

An *interpretation* I is a set of atoms. I *satisfies* a rule r if $(H_r \cup B_r^-) \cap I \neq \emptyset$ or $B_r^+ \setminus I \neq \emptyset$. I is a *model* of P if it satisfies all rules of P. The *Gelfond-Lifschitz (GL) reduct* of P under I is the program P^I obtained from P by first removing all rules r with $B_r^- \cap I \neq \emptyset$ and then removing all $\neg z$ where $z \in B_r^-$ from the remaining rules r. I is an *answer set* of a program P if I is a *minimal model (w.r.t. $\subseteq$) of P^I*. The problem of deciding whether a program has an answer set is called *consistency*, which is Σ_2^{P}-complete. For normal programs, its complexity drops to NP-complete [6, 29], allowing simpler evaluations [25].

Non-ground ASP. We use vectors $\mathbf{X} = \langle x_1, \ldots, x_m\rangle$, $\mathbf{Y} = \langle y_1, \ldots, y_n\rangle$ in the usual way. We *combine vectors* by $\langle \mathbf{X}, \mathbf{Y}\rangle := \langle x_1, \ldots, x_m, y_1, \ldots, y_n\rangle$ and test whether x_1 *is in* $\mathbf{X}$ by $x_1 \in \mathbf{X}$. We assume elements of vectors in any fixed total order. For a set S, we *construct* its unique vector by $\langle S\rangle$.

Let $p_1, \ldots, p_n$ be predicates; each one takes *arity* $|p_i|$ many variables for $1 \leq i \leq n$. A *(non-ground) program* Π is a set of *(non-ground) rules* of the form

$$p_1(\mathbf{X_1}) \vee \ldots \vee p_\ell(\mathbf{X}_\ell) \leftarrow p_{\ell+1}(\mathbf{X_{\ell+1}}), \ldots, p_m(\mathbf{X_m}), \neg p_{m+1}(\mathbf{X_{m+1}}), \ldots, \neg p_n(\mathbf{X}_n), \quad (1)$$

where for every *variable vector* $\mathbf{X_i}$ we have $|\mathbf{X_i}| = |p_i|$, and whenever $x \in \langle \mathbf{X}_1, \ldots, \mathbf{X}_\ell, \mathbf{X}_{m+1}, \ldots, \mathbf{X}_n\rangle$, then $x \in \langle \mathbf{X}_{\ell+1}, \ldots, \mathbf{X}_m\rangle$ (*safeness*). For a non-ground rule r, we let $H_r := \{p_1(\mathbf{X_1}), \ldots, p_\ell(\mathbf{X}_\ell)\}$, $B_r^+ := \{p_{\ell+1}(\mathbf{X}_{\ell+1}), \ldots, p_m(\mathbf{X_m})\}$, $B_r^- := \{p_{m+1}(\mathbf{X_{m+1}}), \ldots, p_n(\mathbf{X_n})\}$, and $\mathrm{var}(r) := \{x \in \mathbf{X} \mid p(\mathbf{X}) \in H_r \cup B_r^+ \cup B_r^-\}$. We use $\mathrm{heads}(\Pi) := \{p(\mathbf{X}) \in H_r \mid r \in \Pi\}$, $\mathrm{preds}(\Pi) := \{p(\mathbf{X}) \in (H_r \cup B_r^+ \cup B_r^-) \mid r \in \Pi\}$, and $\mathrm{pnam}(\Pi) := \{p \mid p(\mathbf{X}) \in \Pi\}$. Without loss of generality, we assume *variables are unique per rule*, i.e., for every two rules $r, r' \in \Pi$, $\mathrm{var}(r) \cap \mathrm{var}(r') = \emptyset$. Attributes *disjunctive*, *normal*, and *tight* carry over to non-ground rules (programs). Rule size $\|r\|$ amounts to $|B_r^+| + |B_r^-| + |H_r|$ and program size $\|\Pi\| := \sum_{r\in\Pi} \|r\|$.

In order to *ground* Π, we require a given set $\mathcal{F}$ of *facts*, i.e., atoms of the form $p(\mathbf{D})$ with p being a predicate of Π and $\mathbf{D}$ being a vector over *domain values* of size $|\mathbf{D}| = |p|$. We say that $\mathbf{D}$ is part of the *domain* of Π, defined by $\mathrm{dom}(\Pi) := \{d \in \mathbf{D} \mid p(\mathbf{D}) \in \mathcal{F}\}$. We refer to the *domain vectors* over $\mathrm{dom}(\Pi)$ for a variable vector $\mathbf{X}$ of size $|\mathbf{X}|$ by $\mathrm{dom}(\mathbf{X})$. Let $\mathbf{D}$ be a domain vector over variable vector $\mathbf{X}$ and vector $\mathbf{Y}$ containing only variables of $\mathbf{X}$. We refer to the *domain vector of* $\mathbf{D}$ *restricted to* $\mathbf{Y}$ by $\mathbf{D_Y}$. The *grounding* $\mathcal{G}(\Pi)$ consists of $\mathcal{F}$ and ground rules obtained by replacing each rule r of Form (1) for every domain vector $\mathbf{D} \in \mathrm{dom}(\langle \mathrm{var}(r)\rangle)$ by

$$p_1(\mathbf{D}_{\mathbf{X_1}}) \vee \ldots \vee p_\ell(\mathbf{D}_{\mathbf{X}_\ell}) \leftarrow p_{\ell+1}(\mathbf{D}_{\mathbf{X}_{\ell+1}}), \ldots, p_m(\mathbf{D}_{\mathbf{X}_m}), \neg p_{m+1}(\mathbf{D}_{\mathbf{X}_{m+1}}), \ldots, \neg p_n(\mathbf{D}_{\mathbf{X}_n}).$$

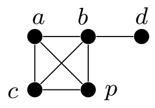
Example 1. *Consider the non-ground program* $\Pi := \{r\}$ *with* $r = a(X, Y) \leftarrow b(X), c(Y, Z)$ *and* $\mathcal{F} := \{b(1).\ c(1, 2).\}$. *Observe that* $\mathrm{dom}(X) = \{1\}$ *and* $\mathrm{dom}(Y) = \mathrm{dom}(Z) = \{1, 2\}$. *The grounding* $P = \mathcal{G}(\Pi)$ *of* Π *consists of:*

$$\{a(1, 1) \leftarrow b(1), c(1, 1).\ a(1, 1) \leftarrow b(1), c(1, 2).\ a(1, 2) \leftarrow b(1), c(2, 1).\ a(1, 2) \leftarrow b(1), c(2, 2).\}$$

The only answer set of Π *is* $\{b(1), c(1, 2), a(1, 1)\}$.

Epistemic Logic Programs. An *epistemic logic program* (ELP) P extends an LP, where rule bodies may contain *epistemic literals* of the form $\mathbf{not}\ell$ using literal ℓ and *epistemic negation* $\mathbf{not}$. Then, $\mathsf{at}(r)$ denotes the atoms occurring in a rule r and $\mathsf{e\text{-}at}(r)$ denotes the *epistemic atoms*, i.e., those used in epistemic literals of r. These notions naturally extend to programs. In a rule we write $\mathbf{K}\ell$ and $\mathbf{M}\ell$ for a literal ℓ, which refers to expressions $\neg\mathbf{not}\ell$ and $\mathbf{not}\neg\ell$, respectively.

While there are different semantics [19, 33, 26, 16, 32], we follow the approach of [19], syntactically denoted according to [31]. Note that the base semantics are equivalent, but certain extensions



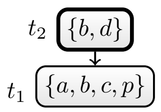


Figure 1: Primal graph G_{Π_2} (left, see Ex. 6); TD $\mathcal{T}$ of G_{Π_2} (right).

on top might result in different behavior. A *world view interpretation (WVI)* I for P is a consistent set I of literals over a set $A \subseteq \mathsf{at}(\mathsf{P})$ of atoms. Intuitively, every $\ell \in I$ is considered "known" and every $a \in A$ with $\{a, \neg a\} \cap I = \emptyset$ is treated as "possible". The *epistemic reduct* [19] of program P w.r.t. a WVI I over A, denoted P^I, is defined as $\mathsf{P}^I = \{r^I \mid r \in \mathsf{P}\}$ where r^I denotes rule r where each epistemic literal $\mathbf{not}\ell$, whose atom is also in A, is replaced by $\bot$ if $\ell \in I$, and by $\top$ otherwise. Note that P^I is an LP with no epistemic negation.

Let $\mathcal{I}$ be a set of interpretations over a set A of atoms. Formally, a WVI I is *compatible* with $\mathcal{I}$ if: (1.) $\mathcal{I} \neq \emptyset$; (2.) for each atom $a \in I$, it holds that for each $J \in \mathcal{I}$, $a \in J$; (3.) for each $\neg a \in I$, we have for each $J \in \mathcal{I}$, $a \notin J$; and (4.) for each atom $a \in A$ with $\{a, \neg a\} \cap I = \emptyset$, there are $J, J' \in \mathcal{I}$, such that $a \in J$, but $a \notin J'$. Then, a WVI I over $\mathsf{at}(\mathsf{P})$ is a *world view (WV)* of P iff I is compatible with the set $AS(\mathsf{P}^I)$. Deciding *WV existence* is Σ_3^{P}-complete [33].

Definitions for non-ground LPs carry over to ELPs.

Example 2. *Consider the ground* ELP $\mathsf{P} := P \cup \{\, na \leftarrow \neg\mathbf{K}a(1,1).\ ka \leftarrow \mathbf{K}a(1,1).\,\}$ *with P of Example 1.*

When constructing a WVI I over $\mathsf{e\text{-}at}(\mathsf{P})$ one guesses for each atom $a \in \mathsf{e\text{-}at}(\mathsf{P})$ either (1) $a \in I$, (2) $\neg a \in I$ or (3) $\{a, \neg a\} \cap I = \emptyset$ as described earlier. Each WVI I can be checked with the corresponding epistemic reduct P^I by verifying Compatibility (1.)–(4.) for $AS(\mathsf{P}^I)$.

Consider $I_1 = \{\, a(1,1) \,\}$ with its epistemic reduct $\mathsf{P}^{I_1} := P \cup \{na \leftarrow \bot.\ ka.\}$, which follows from $na \leftarrow \neg\neg\bot.$ and $ka \leftarrow \neg\bot.$ Note that the epistemic reduct is an LP, as rules r with $\bot \in B_r^+$ or $\top \in B_r^-$ can be dropped. Since $AS(\mathsf{P}^{I_1}) = \{\, \{c(1,2), b(1), a(1,1), ka\} \,\}$, compatibility of I_1 holds, which validates I_1 as (the only) WV of P.

Tree Decompositions and Treewidth. We assume that graphs are undirected, simple, and free of self-loops. Let $G = (V, E)$ be a graph, T a rooted tree with *root node* $\mathrm{root}(T)$, and χ a labeling function that maps every node t of T to a subset $\chi(t) \subseteq V$ called the *bag* of t. The pair $\mathcal{T} = (T, \chi)$ is called a *tree decomposition (TD)* of G iff (i) for each $v \in V$, there exists a t in T, such that $v \in \chi(t)$; (ii) for each $\{v, w\} \in E$, there exists t in T, s.t. $\{v, w\} \subseteq \chi(t)$; and (iii) for each r, s, t of T, s.t. s lies on the unique path from r to t, we have $\chi(r) \cap \chi(t) \subseteq \chi(s)$. The *width* of $\mathcal{T}$ is the largest bag size minus one and the *treewidth* of G is the smallest width among all TDs of G.

In order to illustrate dependencies of programs we define graphs. The *primal graph* $G_P = (V, E)$ *of a ground program* P is a graph with $V = \mathrm{at}(P)$ and $\{a, b\} \in E$ iff atoms a and b with $a \neq b$ appear together in a rule in P. Similar, the *primal graph* $G_\Pi = (V, E)$ *of* Π (non-ground) is a graph with $V = \mathrm{pnam}(\Pi)$, where $\{p_1, p_2\} \in E$ iff predicates p_1 and p_2 with $p_1 \neq p_2$ appear together in a rule in Π.

Example 3. *Consider the non-ground, disjunctive program* $\Pi_2 := \{a(X,Y) \vee p(X,Y) \leftarrow b(X), c(Y,Z).\ d(X) \leftarrow b(X).\}$. *Figure 1 depicts the primal graph G_{Π_2} and a corresponding TD $\mathcal{T}$ of G_{Π_2} of the width* 3.

3 Grounding LPs via Reduction to ELPs

Before we focus on the complexity of non-ground ELPs, we present a way to bypass the grounding-bottleneck of LPs via ground ELPs. This is inspired by recent works [3] on body-decoupled grounding for normal programs.

To this end, we assume a given non-ground program Π and a set $\mathcal{F}$ of facts. Then, for each predicate $p(\mathbf{X})$ in $\mathrm{heads}(\Pi)$, we use every instantiation of $p(\mathbf{X})$ and its negation $\dot{p}(\mathbf{X})$ over $\mathrm{dom}(\Pi)$, resulting in atoms $\mathrm{AtPred} := \{p(\mathbf{D}), \dot{p}(\mathbf{D}) \mid p(\mathbf{X}) \in \mathrm{heads}(\Pi), \mathbf{D} \in \mathrm{dom}(\mathbf{X})\}$. Similar, we define $\mathrm{AtPredC} := \{p^{\mathsf{c}}(\mathbf{X}), \dot{p}^{\mathsf{c}}(\mathbf{X}) \mid p(\mathbf{X}) \in \mathrm{AtPred}\}$.

In the accordance with the semantics of ASP, we require to ensure (i) satisfiability of a potential model M of Π and (ii) non-existence of counter witness, i.e., a model $C \subset M$ (satisfying (i)) of Π^M. Notice that, by performing subset minimization, we do not require a foundedness check as in earlier works. For (i) computing models of rules, we require atoms $\mathrm{AtSat}^{\mathtt{l}} := \{\mathrm{sat}^{\mathtt{l}}, \mathrm{sat}_r^{\mathtt{l}}, \mathrm{gr}_x^{\mathtt{l}}(d) \mid r \in \Pi, x \in \mathrm{var}(r), d \in \mathrm{dom}(x)\}$, where $\mathrm{sat}^{\mathtt{l}}$ (sat_r) indicates satisfiability (of non-ground rule r) for a potential model $\mathtt{l} \in \{\epsilon, \mathtt{c}\}$, where we use ϵ for model M and $\mathtt{c}$ for a potentially smaller model N. An atom of the form $\mathrm{gr}_x^{\mathtt{l}}(d)$ indicates that for checking satisfiability, we assign variable x of non-ground rule r to domain value $d \in \mathrm{dom}(x)$. For (ii) non-existence of a model $N \subset M$, we require proper subsets. For this purpose we use $\mathrm{AtEq} := \{\mathrm{eq}_{p(\mathbf{X})} \mid p(\mathbf{X}) \in \mathrm{AtPred}\} \cup \{\mathrm{nempty}\}$ to intuitively derive equality of a predicate instantiation for potential models M and N while considering empty candidates.

The overall idea consists of the following parts, which are encoded as our reduction $\mathcal{R}$, which transforms a disjunctive, non-ground program Π into to an epistemic, ground program $\mathcal{R}(\Pi)$ consisting of $\mathcal{F}$ and the rules given in Figure 2. To generate all possible world view candidates, i.e., our targeted models M, we use Rules (2). For (ii) we guess possible smaller models N throughout these candidates, achieved by Rule (3), which results (for each world view candidate) in an answer set candidate for all possible subsets of the guessed world view candidates. Then, Rules (4)–(7) trivially derive the equality of the guessed head predicates ensuring that the guessed models N only contain proper subsets of models M, where we use Rules (6) to consider the empty set candidate (which would be removed since no proper subset exists).

We ensure (i) satisfiability of models M through Rules (8), by Rules (11)–(13), (15) and (17) of Figure 3, where we yield sat_r whenever there is an assignment of variables to domain values for a non-ground rule $r \in \Pi$. If such an atom can be derived for all non-ground rules $r \in \Pi$, we follow sat by Rule (15), which is mandatory throughout the world view (cf. Rule (9)). Rule (17) applies *saturation*, causing the assignment of all domain values to every variable. This way, any WV candidate M is removed if not complying with (i).

In the same way, we ensure (ii) satisfiability for the potential models N (proper subsets of M), removing every answer set candidate (within each world view candidate) that does not conform with (i). Notice that, in addition to Rules (11)–(13) and (17), we use Rules (14) to "satisfy" rules of Π that are removed by the construction of the GL reduct Π^M. Instead of Rules (15), we use Rules (16) to consider a potentially empty model for which we do not need to check for satisfiability of (non-existing) proper subsets. Then, Rule (10) removes any world view candidate M, where there is an answer set candidate deriving $\mathrm{sat}^{\mathsf{c}}$ such that we ensure (ii) non-existence of a model $N \subset M$ of Π^M. Intuitively, the only world view candidates remaining depict models of Π, where none of the

Guess Answer-Set Candidates

$$a \leftarrow \mathbf{not}\ \dot{a} \qquad \dot{a} \leftarrow \mathbf{not}\ a \qquad \text{for every } h(\mathbf{X}) \in \text{heads}(\Pi), \mathbf{D} \in \text{dom}(\mathbf{X}), a = h(\mathbf{D}) \tag{2}$$

Guess Potentially $\subseteq$-Smaller Models

$$a^{\mathsf{c}} \vee \dot{a}^{\mathsf{c}} \leftarrow a \qquad \text{for every } h(\mathbf{X}) \in \text{heads}(\Pi), \mathbf{D} \in \text{dom}(\mathbf{X}), a = h(\mathbf{D}) \tag{3}$$

Prevent Spuriously $\subseteq$-Smaller Models

$$\text{eq}_a \leftarrow a, a^{\mathsf{c}} \qquad \text{for every } h(\mathbf{X}) \in \text{heads}(\Pi), \mathbf{D} \in \text{dom}(\mathbf{X}), a = h(\mathbf{D}) \tag{4}$$

$$\text{eq}_a \leftarrow \neg a \qquad \text{for every } h(\mathbf{X}) \in \text{heads}(\Pi), \mathbf{D} \in \text{dom}(\mathbf{X}), a = h(\mathbf{D}) \tag{5}$$

$$\text{nempty} \leftarrow a \qquad \text{for every } h(\mathbf{X}) \in \text{heads}(\Pi), \mathbf{D} \in \text{dom}(\mathbf{X}), a = h(\mathbf{D}) \tag{6}$$

$$\leftarrow \text{nempty}, \text{eq}_{a_1}, \dots, \text{eq}_{a_n} \qquad \text{for every } h(\mathbf{X}) \in \text{heads}(\Pi), \{a_1, \dots, a_n\} = \{h(\mathbf{D}) \mid \mathbf{D} \in \text{dom}(\mathbf{X})\}, 1 \leq i \leq n \tag{7}$$

Ensure Satisfiability of Non-Ground Rules (see Figure 3)

$$r \qquad r' \qquad \text{for every rule } r \in \mathcal{S}(\Pi), r' \in \mathcal{S}^{\mathsf{c}}(\Pi), \text{where } \mathcal{S} \text{ and } \mathcal{S}^{\mathsf{c}} \text{ are defined in Figure 3} \tag{8}$$

Prevent Unsatisfied Rules and $\subseteq$-Smaller Models

$$\leftarrow \mathbf{not}\ \text{sat} \tag{9}$$

$$\leftarrow \mathbf{not}\ \neg\text{sat}^{\mathsf{c}} \tag{10}$$

Figure 2: Reduction $\mathcal{R}(\Pi)$ from a disjunctive, non-ground program Π to an epistemic, ground program; it relies on Figure 3 through Rules (8).

$$\bigvee_{d \in \text{dom}(x)} \text{gr}_x^{\mathtt{l}}(d) \leftarrow \qquad \text{for every } r \in \Pi, x \in \text{var}(r) \tag{11}$$

$$\text{sat}_r^{\mathtt{l}} \leftarrow \text{gr}_{x_1}^{\mathtt{l}}(\mathbf{D}_{\langle x_1 \rangle}), \dots, \text{gr}_{x_\ell}^{\mathtt{l}}(\mathbf{D}_{\langle x_\ell \rangle}), \neg p^{\mathtt{l}}(\mathbf{D}) \qquad \text{for every } r \in \Pi, p(\mathbf{X}) \in B_r^+, \mathbf{D} \in \text{dom}(\mathbf{X}), \mathbf{X} = \langle x_1, \dots, x_\ell \rangle \tag{12}$$

$$\text{sat}_r^{\mathtt{l}} \leftarrow \text{gr}_{x_1}^{\mathtt{l}}(\mathbf{D}_{\langle x_1 \rangle}), \dots, \text{gr}_{x_\ell}^{\mathtt{l}}(\mathbf{D}_{\langle x_\ell \rangle}), p^{\mathtt{l}}(\mathbf{D}) \qquad \text{for every } r \in \Pi, p(\mathbf{X}) \in H_r \cup \{b \in B_r^- \mid \mathtt{l}{=}\epsilon\}, \mathbf{D} \in \text{dom}(\mathbf{X}), \mathbf{X}{=}\langle x_1, \dots, x_\ell \rangle \tag{13}$$

$$\text{sat}_r^{\mathsf{c}} \leftarrow \text{gr}_{x_1}^{\mathsf{c}}(\mathbf{D}_{\langle x_1 \rangle}), \dots, \text{gr}_{x_\ell}^{\mathsf{c}}(\mathbf{D}_{\langle x_\ell \rangle}), p(\mathbf{D}) \qquad \text{if } \mathtt{l} = \mathtt{C}, \text{for every } r \in \Pi, p(\mathbf{X}) \in B_r^-, \mathbf{D} \in \text{dom}(\mathbf{X}), \mathbf{X}{=}\langle x_1, \dots, x_\ell \rangle \tag{14}$$

$$\text{sat} \leftarrow \text{sat}_{r_1}, \dots, \text{sat}_{r_n} \qquad \text{if } \mathtt{l} = \epsilon, \text{where } \Pi = \{r_1, \dots, r_n\} \tag{15}$$

$$\text{sat}^{\mathsf{c}} \leftarrow \text{nempty}, \text{sat}_{r_1}^{\mathsf{c}}, \dots, \text{sat}_{r_n}^{\mathsf{c}} \qquad \text{if } \mathtt{l} = \mathtt{C}, \text{where } \Pi = \{r_1, \dots, r_n\} \tag{16}$$

$$\text{gr}_x^{\mathtt{l}}(d) \leftarrow \text{sat}^{\mathtt{l}} \qquad \text{for every } r \in \Pi, x \in \text{var}(r), d \in \text{dom}(x) \tag{17}$$

Figure 3: Reduction $\mathcal{S}^{\mathtt{l}}(\Pi)$ for encoding satisfiablity of a non-ground program Π into a ground program requiring the use of disjunction through Rules (11). Thereby, $\mathtt{l} \in \{\epsilon, \mathtt{C}\}$ is a potentially empty label allowing the usage of $\mathcal{S}$ in different contexts.

$a(1,1) \leftarrow \mathbf{not}\ \dot{a}(1,1).\ a(1,2) \leftarrow \mathbf{not}\ \dot{a}(1,2).$ (2)
$p(1,1) \leftarrow \mathbf{not}\ \dot{p}(1,1).\ p(1,2) \leftarrow \mathbf{not}\ \dot{p}(1,2).$
$\dot{a}(1,1) \leftarrow \mathbf{not}\ a(1,1).\ \dot{a}(1,2) \leftarrow \mathbf{not}\ a(1,2).$
$\dot{p}(1,1) \leftarrow \mathbf{not}\ p(1,1).\ \dot{p}(1,2) \leftarrow \mathbf{not}\ p(1,2).$
$a^{\mathsf{c}}(1,1) \vee \dot{a}^{\mathsf{c}}(1,1) \leftarrow a(1,1).\ a^{\mathsf{c}}(1,2) \vee \dot{a}^{\mathsf{c}}(1,2) \leftarrow a(1,2).$ (3)
$p^{\mathsf{c}}(1,1) \vee \dot{p}^{\mathsf{c}}(1,1) \leftarrow p(1,1).\ p^{\mathsf{c}}(1,2) \vee \dot{p}^{\mathsf{c}}(1,2) \leftarrow p(1,2).$
$\text{eq}_{a(1,1)} \leftarrow a(1,1), \neg a^{\mathsf{c}}(1,1).\ \text{eq}_{a(1,2)} \leftarrow a(1,2), \neg a^{\mathsf{c}}(1,2).$ (4)
$\text{eq}_{p(1,1)} \leftarrow p(1,1), \neg p^{\mathsf{c}}(1,1).\ \text{eq}_{p(1,2)} \leftarrow p(1,2), \neg p^{\mathsf{c}}(1,2).$
$\text{eq}_{a(1,1)} \leftarrow \neg a(1,1).\ \text{eq}_{a(1,2)} \leftarrow \neg a(1,2).$ (5)
$\text{eq}_{p(1,1)} \leftarrow \neg p(1,1).\ \text{eq}_{p(1,2)} \leftarrow \neg p(1,2).$
$\text{nempty} \leftarrow a(1,1).\ \text{nempty} \leftarrow a(1,2).$ (6)
$\text{nempty} \leftarrow p(1,1).\ \text{nempty} \leftarrow p(1,2).$
$\leftarrow \text{nempty}, \text{eq}_{a(1,1)}, \text{eq}_{a(1,2)}, \text{eq}_{p(1,1)}, \text{eq}_{p(1,2)}.$ (7)
$\leftarrow \mathbf{not}\ \text{sat}. \qquad \leftarrow \mathbf{not}\ \neg\text{sat}^{\mathsf{c}}.$ (9), (10)

Figure 4: $\mathcal{R}(\Pi)$ with Π from Example 4; $\mathcal{R}$ is given in Figure 2.

answer set candidates contains sat^{c}, i.e., there is no proper subset N of the guessed model M (represented by the world view candidate) which represents a model of Π^M.

Example 4. *Consider the non-ground, disjunctive program* $\Pi := \{a(X,Y) \vee p(X,Y) \leftarrow b(X), c(Y,Z).\}$ *and facts* $\mathcal{F} := \{b(1).\ c(1,2).\}$. *Grounding the program as described above and shown in Figure 2, results in the ground program* $P = \mathcal{R}(\Pi)$ *of Figures 4–6. The world view candidates that are guessed by the resulting ground program are checked by the procedure described above, resulting in the two WVs* $\{a(1,1)\}$ *and* $\{p(1,1)\}$ *(when restricted to symbols of* Π*) which are equatable with the answer sets of* Π.

Theorem 1 (Correctness, $\star$[1]). *Let* Π *be any disjunctive, non-ground program. Then, the grounding procedure* $\mathcal{R}$ *on* Π *is correct, i.e., the world views of* $\mathcal{R}(\Pi)$ *restricted to* $\text{at}(\mathcal{G}(\Pi))$ *match with the answer sets of* $\mathcal{G}(\Pi)$. *Precisely, for every world view* W *of* $\mathcal{R}(\Pi)$ *there is exactly one answer set* M *of* $\mathcal{G}(\Pi)$, *such that* $M = \{a \mid a \in W \cap \text{at}(\mathcal{G}(\Pi))\}$ *holds.*

Proof (Idea). $\Longrightarrow$: Let W be a WV of $\mathcal{R}(\Pi)$. Then, by Rules (9), we require that sat $\in W$. From this, one can construct a set $M := \{a \mid a \in W \cap \text{at}(\mathcal{G}(\Pi))\}$. Then, it remains to show that (i) M is indeed a model of $\mathcal{G}(\Pi)$ and (ii) that M is a subset-minimal model of $\mathcal{G}(\Pi)^M$, i.e., there does not exist a model N of $\mathcal{G}(\Pi)^M$ with $N \subsetneq M$.

$\Longleftarrow$: Let M be an answer set of $\mathcal{G}(\Pi)$. First, we construct $N := \{\neg a, \dot{a}^{\mathsf{c}}, \neg a^{\mathsf{c}}, \dot{a} \mid h(\mathbf{X}) \in \text{heads}(\Pi), \mathbf{D} \in \text{dom}(\mathbf{X}), a = h(\mathbf{D}), a \notin M\}$, which collects those head atoms that are not in M. From this we construct a set $I := M \cup N \cup \{\neg\text{sat}^{\mathsf{c}}\} \cup \{\text{sat}, \text{sat}_r, \text{gr}_x(d) \mid r \in \Pi, x \in \text{var}(r), d \in \text{dom}(x)\} \cup \{\text{nempty} \mid M \neq \emptyset\} \cup \{\text{eq}_a \mid a \in \text{at}(\mathcal{G}(\Pi)) \setminus M\}$. It remains to show that I can be extended by a subset S' of $S := \{\text{sat}_r^{\mathsf{c}}, \neg\text{sat}_r^{\mathsf{c}} \mid r \in \Pi\}$, i.e., $I \cup S'$ is a WV of $\mathcal{R}(\Pi)$. $\square$

Theorem 2 (Runtime). *Let* Π *be any disjunctive, non-ground program, where every predicate has arity at most* a. *Then, the grounding procedure* $\mathcal{R}$ *on* Π *is polynomial, i.e., runs in time* $\mathcal{O}(\|\Pi\| \cdot |\text{dom}(\Pi)|^a)$.

Proof (Sketch). Reduction $\mathcal{R}$ constructs $\mathcal{O}(\|\Pi\| \cdot |\text{dom}(\Pi)|^a)$ many Rules (2) and (3) of constant size for guessing candidates. For preventing spuriously smaller models $\mathcal{O}(\|\Pi\| \cdot |\text{dom}(\Pi)|^a)$ many Rules (4), (5) and (6) of constant size, as well as one Rule (7) of size

[1] Proofs marked with "$\star$" are detailed in the appendix of the paper.

$\text{gr}_X(1).\ \text{gr}_Y(1) \vee \text{gr}_Y(2).\ \text{gr}_Z(1) \vee \text{gr}_Z(2).$ (11)

$\text{sat}_r \leftarrow \text{gr}_X(1), \neg b(1).$ (12)

$\text{sat}_r \leftarrow \text{gr}_Y(1), \text{gr}_Z(1), \neg c(1,1).\ \text{sat}_r \leftarrow \text{gr}_Y(1), \text{gr}_Z(2), \neg c(1,2).$

$\text{sat}_r \leftarrow \text{gr}_Y(2), \text{gr}_Z(1), \neg c(2,1).\ \text{sat}_r \leftarrow \text{gr}_Y(2), \text{gr}_Z(2), \neg c(2,2).$

$\text{sat}_r \leftarrow \text{gr}_X(1), \text{gr}_Y(1), a(1,1).$ (13)

$\text{sat}_r \leftarrow \text{gr}_X(1), \text{gr}_Y(2), a(1,2).\ \text{sat}_r \leftarrow \text{gr}_X(1), \text{gr}_Y(1), p(1,1).$

$\text{sat}_r \leftarrow \text{gr}_X(1), \text{gr}_Y(2), p(1,2).$

$\text{sat} \leftarrow \text{sat}_r.$ (14)

$\text{gr}_X(1) \leftarrow \text{sat}.\ \text{gr}_Y(1) \leftarrow \text{sat}.\ \text{gr}_Y(2) \leftarrow \text{sat}.$ (17)

$\text{gr}_Z(1) \leftarrow \text{sat}.\ \text{gr}_Z(2) \leftarrow \text{sat}.$

Figure 5: $\mathcal{S}(\Pi)$ with Π from Example 4, guided by $\mathcal{S}$ of Figure 3.

$\text{gr}^{\mathsf{c}}_X(1).\ \text{gr}^{\mathsf{c}}_Y(1) \vee \text{gr}^{\mathsf{c}}_Y(2).\ \text{gr}^{\mathsf{c}}_Z(1) \vee \text{gr}^{\mathsf{c}}_Z(2).$ (11)

$\text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_X(1), \neg b^{\mathsf{c}}(1).$ (12)

$\text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_Y(1), \text{gr}^{\mathsf{c}}_Z(1), \neg c^{\mathsf{c}}(1,1).\ \text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_Y(1), \text{gr}^{\mathsf{c}}_Z(1), \neg c^{\mathsf{c}}(1,2).$

$\text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_Y(2), \text{gr}^{\mathsf{c}}_Z(1), \neg c^{\mathsf{c}}(2,1).\ \text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_Y(2), \text{gr}^{\mathsf{c}}_Z(1), \neg c^{\mathsf{c}}(2,2).$

$\text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_X(1), \text{gr}^{\mathsf{c}}_Y(1), a^{\mathsf{c}}(1,1).$ (13)

$\text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_X(1), \text{gr}^{\mathsf{c}}_Y(2), a^{\mathsf{c}}(1,2).\text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_X(1), \text{gr}^{\mathsf{c}}_Y(1), p^{\mathsf{c}}(1,1).$

$\text{sat}^{\mathsf{c}}_r \leftarrow \text{gr}^{\mathsf{c}}_X(1), \text{gr}^{\mathsf{c}}_Y(2), a^{\mathsf{c}}(1,2).$

$\text{sat}^{\mathsf{c}} \leftarrow \text{sat}^{\mathsf{c}}_r, \text{nempty}.$ (16)

$\text{gr}^{\mathsf{c}}_X(1) \leftarrow \text{sat}^{\mathsf{c}}.\ \text{gr}^{\mathsf{c}}_Y(1) \leftarrow \text{sat}^{\mathsf{c}}.\ \text{gr}^{\mathsf{c}}_Y(2) \leftarrow \text{sat}^{\mathsf{c}}.$ (17)

$\text{gr}^{\mathsf{c}}_Z(1) \leftarrow \text{sat}^{\mathsf{c}}.\ \text{gr}^{\mathsf{c}}_Z(2) \leftarrow \text{sat}^{\mathsf{c}}.$

Figure 6: $\mathcal{S}^{\mathsf{c}}(\Pi)$ with Π from Example 4, guided by $\mathcal{S}$ of Figure 3.

$\mathcal{O}(|\Pi|)$ are constructed. For the sub procedure $\mathcal{S}^1$ in Rules (8), we require: $\mathcal{O}(\|\Pi\| \cdot a)$ many Rules (11) of size $|\text{dom}(\Pi)|$, $\mathcal{O}(\|\Pi\| \cdot |\text{dom}(\Pi)|^a)$ many Rules (12), (13) and (14) of size $\mathcal{O}(a)$, one of each Rule (15) and (16) of size $\mathcal{O}(\|\Pi\|)$, as well as $\mathcal{O}(\|\Pi\| \cdot |\text{dom}(\Pi)|)$ many Rules (17) of constant size. Further, the reduction $\mathcal{R}$ constructs one Rule (9) and (10) of constant size. □

4 Treewidth-Aware Grounding and its Limits

Despite the high complexity of grounding, there is a way to circumvent hardness, by tackling the complexity by means of structural parameters. First, we provide precise treewidth guarantees for grounding, where we show how grounding increases the treewidth of a non-ground (E)LP. Then, we further improve this result. Finally, we show that this treewidth-increase is ETH-tight, i.e., assuming ETH, a significant improvement would be unexpected. As a result, this yields precise runtime bounds for the consistency of non-ground (E)LPs, when considering treewidth, and matching lower bounds that are asymptotically tight under ETH.

4.1 *Utilizing Treewidth for Grounding*

In order to mitigate the complexity results of the previous section, we adapt our above grounding procedure to obtain treewidth guarantees for programs over fixed arity a and with respect to a certain domain size d. This allows us to show the impact of treewidth for body-decoupled grounding, thereby also establishing new general results for grounding and treewidth. To this end, we assume that each predicate only appears constantly often in a rule, formalized as follows.

Definition 1 (Structure Preservation). *Let $c \geq 0$ be a constant and Π be a non-ground logic program. Then Π is* structure-preserving, *whenever for every $r \in \Pi$ and every predicate $p \in \text{pnam}(\text{preds}(\Pi))$, $|\{p(\mathbf{X}) \mid p(\mathbf{X}) \in \text{preds}(\{r\})\}| \leq c$.*

Note that this structure preservation is indeed not a hard restriction, since one can easily replace each such predicate occurrence $p(\mathbf{X})$ in a rule r by a fresh predicate $p'(\mathbf{X})$. Then one adds the rule $p(\mathbf{X}) \leftarrow p'(\mathbf{X})$ if $p(\mathbf{X}) \in H_r$, otherwise, i.e., if $p(\mathbf{X}) \in B^+_r \cup B^-_r$, rule $p'(\mathbf{X}) \leftarrow p(\mathbf{X})$ is added. However, by requiring structure preservation, we prevent structural dependencies to become "hidden" in a way where extra dependencies cannot be broken or avoided (not even by body-decoupled grounding). This is demonstrated below.

Example 5. *Consider a program Π where a predicate p occurs frequently in the body of a rule, as shown in Figure 7. The primal graph for the output yielded by traditional grounders forms a connection between each instantiated body-predicate p as well as each instantiated head-predicate. This way all atoms will appear in a clique in the primal graph, yielding tree decomposition bags of size linear in the instance size (cf. Fig. 7 (middle)). While reduction $\mathcal{R}$ decouples body-predicates by design (cf. Fig. 7 (right)), variable allocations still emerge together through decoupled rules. This can lead to high treewidth as any tree decomposition requires a bag for all variable allocations appearing together in a rule. However, as long as single predicates occur only constantly often in the body of a rule, this is mitigated to a constant boundary.*

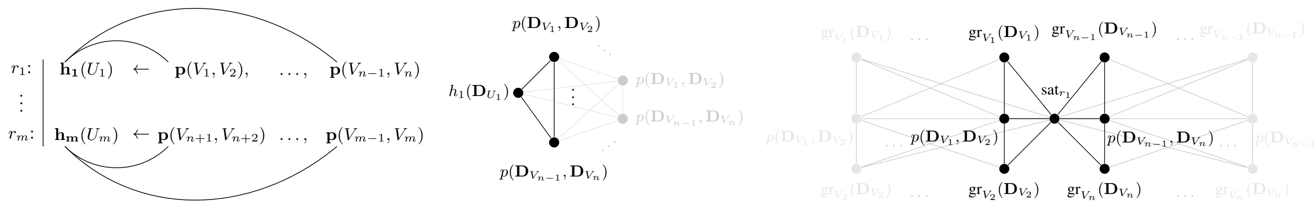
In the remainder of this section we assume a given structure-preserving program Π and a TD $\mathcal{T} = (T, \chi)$ of G_Π. Before we commence with the formal description, we define for a node t in T the *(non-ground) bag program* $\Pi_t \subseteq \Pi$ as any fixed subset of the program $\{r \in \Pi \mid \text{preds}(r) \subseteq \chi(t)\}$ such that $\Pi = \bigcup_{t' \text{ in } T} \Pi_{t'}$, i.e., (i) every rule $r \in \Pi$ appears in at least one bag program. Further, for simplicity we assume (ii) $|\Pi_t| \leq 1$, which can be achieved by copying nodes in T.

Example 6. *Recall the program Π_2 from Example 3 and Figure 1 depicting a corresponding TD $\mathcal{T}$. The bag programs are $\Pi_{t_1} = \{a(X,Y) \vee p(X,Y) \leftarrow b(X), c(Y,Z).\}$ and $\Pi_{t_2} = \{d(X) \leftarrow b(X).\}$.*

For our treewidth-aware reduction $\mathcal{R}_{tw}$, we assume a given non-ground program Π, a set $\mathcal{F}$ of facts and a tree decomposition $\mathcal{T}$ of the given program. Based on the idea of *dynamic programming*, the tree can then be traversed in post-order, where at each node information is gathered, i.e., at every node the answer sets of the corresponding bag programs are computed, which is then interpreted accordingly. To this end, we illustrate treewidth-aware variants $\mathcal{S}_{tw}$ and $\mathcal{R}_{tw}$ in Figure 8 and 9.

While the overall idea remains as earlier described, the following rules are replaced to incorporate the idea of tree decomposition. For checking satisfiability, we replace Rules (15) (and (16)) of $\mathcal{S}$ with Rules (18) (and (19) respectively) in $\mathcal{S}_{tw}$, where, in comparison to the earlier introduced reduction, rules of the bag program as well as the satisfiablity of child nodes of the TD are incorporated. Similar, we replace Rules (7) of $\mathcal{R}$ with Rules (20)–(21) in $\mathcal{R}_{tw}$, which additionally includes the equality of child nodes. Lastly, the treewidth-aware sub-procedure $\mathcal{S}_{tw}$ is incorporated through (22).

Theorem 3 (Treewidth-Aware Decoupling)**.** *Let Π be a non-ground, disjunctive, structure-preserving program over domain size $d = |\text{dom}(\Pi)|$ and constant arity a, where $\mathcal{T}$ is a TD of G_Π of width k. Reduction $\mathcal{R}_{tw}(\Pi, \mathcal{T})$ is treewidth-aware, i.e., the treewidth of $G_{\mathcal{R}_{tw}(\Pi)}$ is bounded by $\mathcal{O}(d \cdot k)$.*



r_1: $\mathbf{h_1}(U_1) \leftarrow \mathbf{p}(V_1, V_2), \ldots, \mathbf{p}(V_{n-1}, V_n)$
r_m: $\mathbf{h_m}(U_m) \leftarrow \mathbf{p}(V_{n+1}, V_{n+2}) \ldots, \mathbf{p}(V_{m-1}, V_m)$
$p(\mathbf{D}_{V_1}, \mathbf{D}_{V_2})$
$h_1(\mathbf{D}_{U_1})$
$p(\mathbf{D}_{V_{n-1}}, \mathbf{D}_{V_n})$
$\mathrm{gr}_{V_1}(\mathbf{D}_{V_1})$
$\mathrm{gr}_{V_{n-1}}(\mathbf{D}_{V_{n-1}})$
sat_{r_1}
$\mathrm{gr}_{V_2}(\mathbf{D}_{V_2})$
$\mathrm{gr}_{V_n}(\mathbf{D}_{V_n})$

Figure 7: (Left): Example of a program $\Pi := \{r_1, \ldots, r_m\}$ where a predicate p occurs frequently in the body of a rule. (Middle): Snippet of the primal graph for the grounding yielded by $\mathcal{G}(r_1)$. (Right): Snippet of the primal graph for the output yielded by reduction $\mathcal{R}(r_1)$.

Treewidth-aware variant of $\mathcal{S}$ by replacing Rules (15) and (16) by (18) and (19), respectively

$$\mathrm{sat}_t \leftarrow \mathrm{sat}_{t_1}, \ldots, \mathrm{sat}_{t_\ell}, \mathrm{sat}_{r_1}, \ldots, \mathrm{sat}_{r_n} \quad \text{for every } t \text{ in } T, \text{ if } \mathtt{l} = \epsilon, \mathrm{chldr}(t) = \{t_1, \ldots, t_\ell\}, \Pi_t = \{r_1, \ldots, r_n\} \quad (18)$$

$$\mathrm{sat}^{\mathsf{c}}_t \leftarrow \mathrm{nempty}, \mathrm{sat}^{\mathsf{c}}_{t_1}, \ldots, \mathrm{sat}^{\mathsf{c}}_{t_\ell}, \mathrm{sat}^{\mathsf{c}}_{r_1}, \ldots, \mathrm{sat}^{\mathsf{c}}_{r_n} \quad \text{for every } t \text{ in } T, \text{if } \mathtt{l} = \mathsf{c}, \mathrm{chldr}(t) = \{t_1, \ldots, t_\ell\}, \Pi_t = \{r_1, \ldots, r_n\} \quad (19)$$

Figure 8: Treewidth-aware reduction $\mathcal{S}^{\mathtt{l}}_{tw}(\Pi, \mathcal{T})$ for encoding satisfiablity of a non-ground program Π and a TD $\mathcal{T}{=}(T, \chi)$ of G_Π into a ground program requiring the reuse of Rules (11)–(14), and (17). Again, $\mathtt{l} \in \{\epsilon, \mathsf{c}\}$ is a label allowing usage in different contexts.

Proof. We assume that $\mathcal{T}{=}(T, \chi)$ is a nice TD of G_Π of width k, since one can easily construct a nice TD from any TD without increasing its width. From this, we will construct a TD $\mathcal{T}'{=}(T, \chi')$ of $G_{\mathcal{R}_{tw}(\Pi,\mathcal{T})}$ as follows. We define χ' by showing how each bag $\chi(t)$ for every node t is modified accordingly. Let $\chi'(t) := P \cup A \cup S$, where $P := \{p(\mathbf{D}), p(\mathbf{D})^{\mathsf{c}}, \dot{p}(\mathbf{D}), \dot{p}(\mathbf{D})^{\mathsf{c}}, \mathrm{eq}_{p(\mathbf{D})} \mid p \in \chi(t), r \in \Pi, p(\mathbf{X}) \in \mathrm{preds}(r), \mathbf{D} \in \mathrm{dom}(\mathbf{X})\}$, $A := \{\mathrm{gr}_{x_i}(\mathbf{D}_{\langle x_i \rangle}), \mathrm{gr}^{\mathsf{c}}_{x_i}(\mathbf{D}_{\langle x_i \rangle}) \mid p \in \chi(t), r \in \Pi, p(\mathbf{X}) \in \mathrm{preds}(r), \mathbf{D} \in \mathrm{dom}(\mathbf{X}), \mathbf{X}{=}\langle x_1, \ldots, x_\ell \rangle, 1 \leq i \leq \ell\}$, $S := \{\mathrm{sat}_r, \mathrm{sat}^{\mathsf{c}}_r, \mathrm{sat}_t, \mathrm{sat}^{\mathsf{c}}_t, \mathrm{sat}_{t'}, \mathrm{sat}^{\mathsf{c}}_{t'}, \mathrm{nempty} \mid r \in \Pi_t, t' \in \mathrm{chldr}(t)\}$. It is easy to see that $\mathcal{T}'$ is indeed a TD of $G_{\mathcal{R}_{tw}(\Pi,\mathcal{T})}$. Indeed, connectedness holds by construction. Further, the atoms of Rules (2)–(7) and (9), (10) appear together in at least one TD node of $\mathcal{T}'$. Also for Rules (8), one can easily check that the atoms of the corresponding Rules (11)–(17) appear in a TD node of $\mathcal{T}'$.

It remains to show that the (tree)width bound holds. By structure preservation of Π, we have that ever predicate appears at most c times in a rule. Consequently, we have that $|P| = 5 \cdot c \cdot k \cdot d^a$, $|A| = 2 \cdot c \cdot k \cdot d^a$, $|S| = |\Pi_t| \cdot 3 \cdot 2 + 1 = 6 \cdot 1 + 1$, since $|\Pi_t| \leq 1$ and $|\mathrm{chldr}(t)| \leq 2$. In total, this adds up to $\mathcal{O}(d \cdot k)$, since c and a are constant. □

While the increase of treewidth is from k to $\mathcal{O}(k \cdot d)$, it turns out that those parts of the "treewidth-causing" bags that are relevant for solving can be further decreased to $\mathcal{O}(k \cdot \log(d))$. Later, we will show that for fixed arity a, one cannot significantly improve, i.e., the treewidth increase is not expected to be in $o(k \cdot \log(d))$, unless ETH fails, cf., Corollary 2.

4.2 *Further Decreasing Structural Dependencies*

In order to manifest an increase of solving-relevant parts[2] of treewidth from k to $\mathcal{O}(k \cdot \log(d))$, one can utilize *bit manipulations*. There, instead of atoms of the form $\mathrm{gr}_x(e)$ for each non-ground variable x and domain value $e \in \mathrm{dom}(x)$ (see Figure 3), we use combinations of fresh atoms $\mathrm{gr}^i_x(0)$ and $\mathrm{gr}^i_x(1)$ for bit indices $1 \leq i \leq \lceil \log(|\mathrm{dom}(\Pi)|) \rceil$, which indicate that the i-th bit in the binary representation of e is set to false and true, respectively.

[2] Intuitively, these are atoms subject to saturation, see Theorem 4.

This results in larger rules, but smaller bags and in addition to applying the treewidth-aware variant of Figure 8, one also needs to adjust Rules (11)–(14) and (17) accordingly. Intuitively, the concept stays the same, but instead of using $\mathrm{gr}_x(e)$, one addresses a sequence of $\lceil \log(|\mathrm{dom}(\Pi)|) \rceil$ many atoms of the form $\mathrm{gr}^i_x(0)$ and $\mathrm{gr}^i_x(1)$.

The complete adjustment is sketched in the appendix of the paper, which shows reduction $\mathcal{S}'_{tw}(\Pi, \mathcal{T})$ that is then used in Rules (22), resulting in reduction $\mathcal{R}'_{tw}(\Pi, \mathcal{T})$. Overall, with this adjusted reduction $\mathcal{R}'_{tw}(\Pi, \mathcal{T})$ we obtain the following runtime results for a non-ground program.

Theorem 4 (Upper Bound, $\star$). *Let Π be a non-ground, disjunctive, structure-preserving program over domain size $d = |\mathrm{dom}(\Pi)|$ and constant arity a, where the treewidth of G_Π is k. Answer set existence of $\mathcal{G}(\Pi)$ can be decided in time $2^{2^{d^{\mathcal{O}(k)}}} \cdot \mathrm{poly}(|\mathrm{preds}(\Pi)|)$, by treewidth-aware decoupling.*

Proof (Sketch). First, we compute a TD $\mathcal{T} = (T, \chi)$ of G_Π of width $5 \cdot k$ in time $2^{\mathcal{O}(k)} \cdot \mathrm{poly}(|\mathrm{preds}(\Pi)|)$, see [8]. Then, we use reduction $\mathsf{P} := \mathcal{R}'_{tw}(\Pi, \mathcal{T})$ to construct an ELP. This ELP can be solved using a DP algorithm [22, Listing 2]. This algorithm works for each node t of T as follows. First, it computes WVIs among those atoms that appear under epistemic negation, i.e., it is exponential in the number of atoms in $|\chi(t) \cap \mathsf{e\text{-}at}(\mathsf{P})|$. Then, for each computed WVI I, the algorithm decides for interpretations M in P^I; finally, it computes interpretations N of $(\mathsf{P}^I)^M$. So, the resulting worst-case runtime for t boils down to $R(t) := 2^{|\chi(t) \cap \mathsf{e\text{-}at}(\mathsf{P})|} \cdot 2^{2^{|\chi(t)|}} \cdot 2^{2^{2^{|\chi(t)|}}}$, see [22, Theorem 14].

However, by construction of P, we can draw the following crucial observation. The atoms in I, which are those appearing in Rules (2), always appear also in M and N. In other words, there is no need to decide for those atoms in M and N (just consider I), since the epistemic reduct of Rules (2) results in facts. Similarly, atoms a^{c} and $\dot{a}^{\mathsf{c}}$ of Rules (3), which are assigned in M, always appear in N by the semantics via GL reduct; hence, these atoms do not need to be decided for, when computing N (just take I and M). The same argument holds for atoms eq_a and nempty in Rules (4)–(7).

As a result, when deciding for N, which is the inner-most choice causing hardness, the only remaining atoms one needs to decide upon, are atoms of the form sat, $\mathrm{sat}^{\mathsf{c}}$, sat_r, $\mathrm{sat}^{\mathsf{c}}$ as well as atoms over predicates of the form gr^i_x and $\mathrm{gr}^{i,\mathsf{c}}_x$. We may assume that the former

Treewidth-aware variant of $\mathcal{R}$ by replacing Rules (7) and (8) by (20)–(21) and (22), respectively

$$\mathrm{eq}_t \leftarrow \mathrm{eq}_{t_1}, \ldots, \mathrm{eq}_{t_\ell}, \mathrm{eq}_{a_1}, \ldots, \mathrm{eq}_{a_n} \qquad \text{for every } t \text{ in } T, \{t_1, \ldots, t_\ell\} = \mathrm{chldr}(t), h(\mathbf{X}) \in \chi(t) \cap \mathrm{heads}(\Pi), \{a_1, \ldots, a_n\} = \{h(\mathbf{D}) \mid \mathbf{D} \in \mathrm{dom}(\mathbf{X})\}, 1 \leq i \leq n \tag{20}$$

$$\leftarrow \mathrm{eq}_t, \mathrm{nempty} \qquad \text{for } t = \mathrm{root}(T) \tag{21}$$

$$r \qquad r' \qquad \text{for every rule } r \in \mathcal{S}_{tw}(\Pi, \mathcal{T}), r' \in \mathcal{S}^{\mathsf{c}}_{tw}(\Pi, \mathcal{T}), \text{as defined in Figure 8} \tag{22}$$

Figure 9: Treewidth-aware reduction $\mathcal{R}_{tw}(\Pi, \mathcal{T})$ from a non-ground program Π and a TD $\mathcal{T}=(T, \chi)$ of G_Π to an epistemic, ground program.

atoms appear only constantly often in $\chi(t)$, see A of the proof of Theorem 3. For the latter atoms, there only exist $\mathcal{O}(k \cdot \log(d))$ many in $\chi(t)$. Consequently, we obtain that $R(t) = 2^k \cdot 2^{2^{k \cdot d}} \cdot 2^{2^{2^{k \cdot \log(d)}}} = 2^{2^{d^{\mathcal{O}(k)}}}$. By applying $R(t)$ for every node in T, the claim follows. □

For tight and normal programs, the runtime for treewidth decreases by one level of exponentiality. This can be shown by modifying existing constructions [3], such that one obtains a treewidth-aware grounding procedure ensuring mild treewidth increases.

Theorem 5 ($\star$)**.** *Let Π be a non-ground, normal, structure-preserving program over domain size $d = |\mathrm{dom}(\Pi)|$ and constant arity a, s.t. the treewidth of G_Π is k. Then, answer set existence of $\mathcal{G}(\Pi)$ can be decided in time $2^{d^{\mathcal{O}(k)}} \cdot \mathrm{poly}(|\mathrm{preds}(\Pi)|)$, using treewidth-aware decoupling.*

4.3 Hardness and Limits of Treewidth

Interestingly, under reasonable assumptions in complexity theory, it is not expected that the result of Theorem 4 can be significantly improved. We show this by reducing from quantified CSP, with known runtime guarantees and limitations.

Theorem 6 (Lower Bound, $\star$)**.** *Let Π be a non-ground, disjunctive program over domain size $d = |\mathrm{dom}(\Pi)|$ and constant arity a, where the treewidth of G_Π is k. Then, unless the ETH fails, the existence of an answer set of $\mathcal{G}(\Pi)$ can not be computed in time $2^{2^{d^{o(k)}}} \cdot \mathrm{poly}(|\mathrm{preds}(\Pi)|)$.*

Proof (Idea). The lower bound can be shown by a reduction from QCSP, where the treewidth is linearly preserved. □

The proof construction carrys over to simpler fragments.

Corollary 1 (LB for Tight LPs)**.** *Let Π be a non-ground, tight (or normal) program over domain size $d = |\mathrm{dom}(\Pi)|$ and constant arity a, where the treewidth of G_Π is k. Then, unless the ETH fails, the existence of an answer set of $\mathcal{G}(\Pi)$ can not be computed in time $2^{d^{o(k)}} \cdot \mathrm{poly}(|\mathrm{preds}(\Pi)|)$.*

Theorem 6 yields a general grounding lower bound.

Theorem 7 (TW LB)**.** *Let Π be a non-ground, disjunctive program over domain size $d = |\mathrm{dom}(\Pi)|$ and constant arity a, where the treewidth of G_Π is k. Then, unless ETH fails, there cannot be a grounding procedure, whose running time is in $d^{\mathcal{O}(k)} \cdot \mathrm{poly}(\mathrm{preds}(\Pi))$, that transforms Π into a ground program P such that the treewidth of G_P is in $d^{o(k)}$.*

Proof. Assume towards a contradiction that such a grounding procedure exists and the ETH still holds. Then, however, we apply this procedure on Π obtaining a ground program, where the treewidth of G_P is in $d^{o(k)}$. In turn, we can then decide the consistency of the ground disjunctive programs in time $2^{2^{d^{o(k)}}} \cdot \mathrm{poly}(|\mathrm{at}(P)|)$ using a known algorithm [17]. So we decide whether an answer set of Π exists in time $2^{2^{d^{o(k)}}} \cdot \mathrm{poly}(|\mathrm{preds}(\Pi)| \cdot d^a)$. In cosequence, by Theorem 6, ETH fails. □

This also yields a lower bound of the treewidth-increase by Theorem 3. The following bound matches the solving-relevant treewidth part of our reduction, see Theorem 4.

Corollary 2. *Let Π be a non-ground, disjunctive program over domain size $d = |\mathrm{dom}(\Pi)|$ and constant arity a, where the treewidth of G_Π is k. Then, under ETH, there is no reduction R to a ground ELP, running in $2^{2^{d^{o(k)}}} \cdot \mathrm{poly}(\mathrm{preds}(\Pi))$, such that the treewidth of $G_{R(\Pi)}$ is in $o(k \cdot \log(d))$.*

5 Conclusion

This work focuses on dealing with the ASP grounding bottleneck and complexity differences between non-ground and ground logic programs – two topics that are highly researched in the last decade and drive the developement of efficient ASP systems. In particular, we first analyze the idea of generalizing a body-decoupled grounding approach from normal programs to disjunctive programs, where a set of non-ground rules can be transformed to an epistemic logic program encoding subset-minimization, thereby achieving groundings being only exponential in the maximum predicate arity. That way, we are able to *translate the Σ_3^P-hard problem* (complete for fixed predicate arity) of deciding whether a disjunctive program admits an answer set, to the problem of deciding whether a *ground* epistemic program admits a world view. This allows us to shift complexity due to grounding, in the direction of solving. Our presented reduction thereby bijectively preserves the answer sets, i.e., there is a one-to-one correspondence between answer sets of the non-ground program and world views of the epistemic program.

In a further step, we analyze the impact of structure in our grounding approach. By transforming the idea to treewidth, we achieve precise runtimes, which, surprisingly, show that some parts of the complexity hide in grounding, as tight and normal (non-ground) programs are of similar hardness compared to the ground equivalent. Lastly, we focus on the respective lower bounds, ruling out significant improvements.

For future work we plan on evaluating this technique in the practical sense. For this reason, we expect that a competition and sophisticated instances for epistemic programs, designed to reflect the high-performance requirements of grounding, may help to build and compare systems in the future. The techniques in this work could also be useful for other first-order formalisms and might lend itself well to big data. Further, this work also opens up questions on the expressiveness and complexity of non-ground ELPs, we aim to settle.

Acknowledgments

This research was funded by the Austrian Science Fund (FWF), grants J 4656 and P 32830, the Society for Research Funding in Lower Austria (GFF, Gesellschaft für Forschungsförderung NÖ) grant ExzF-0004, as well as the Vienna Science and Technology Fund (WWTF) grant ICT19-065.

References

[1] Mario Alviano, Giovanni Amendola, Carmine Dodaro, Nicola Leone, Marco Maratea, and Francesco Ricca, 'Evaluation of Disjunctive Programs in WASP', in *LPNMR'19*, volume 11481 of *LNCS*, pp. 241–255. Springer, (2019).

[2] Mutsunori Banbara, Benjamin Kaufmann, Max Ostrowski, and Torsten Schaub, 'Clingcon: The next generation', *Theory Pract. Log. Program.*, **17**(4), 408–461, (2017).

[3] Viktor Besin, Markus Hecher, and Stefan Woltran, 'Body-Decoupled Grounding via Solving: A Novel Approach on the ASP Bottleneck', in *IJCAI'22*, pp. 2546–2552. ijcai.org, (2022).

[4] Manuel Bichler, Michael Morak, and Stefan Woltran, 'lpopt: A Rule Optimization Tool for Answer Set Programming', *Fundamenta Informaticae*, **177**(3-4), 275–296, (2020).

[5] Manuel Bichler, Michael Morak, and Stefan Woltran, 'selp: A single-shot epistemic logic program solver', *Theory Pract. Log. Program.*, **20**(4), 435–455, (2020).

[6] Nicole Bidoit and Christine Froidevaux, 'Negation by default and unstratifiable logic programs', *Theoretical Computer Science*, **78**(1), 85–112, (1991).

[7] Bernhard Bliem, Michael Morak, Marius Moldovan, and Stefan Woltran, 'The Impact of Treewidth on Grounding and Solving of Answer Set Programs', *Journal of Artificial Intelligence Research*, **67**, 35–80, (2020).

[8] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk, 'A c^k n 5-Approximation Algorithm for Treewidth', *SIAM J. on Computing*, **45**(2), 317–378, (2016).

[9] G. Brewka, T. Eiter, and M. Truszczyński, 'Answer set programming at a glance', *Comm. of the ACM*, **54**(12), 92–103, (2011).

[10] Francesco Calimeri, Simona Perri, and Jessica Zangari, 'Optimizing answer set computation via heuristic-based decomposition', *Theory Pract. Log. Program.*, **19**(4), 603–628, (2019).

[11] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov, 'Complexity and expressive power of logic programming', *ACM Comput. Surv.*, **33**(3), 374–425, (2001).

[12] Thomas Eiter, Wolfgang Faber, Michael Fink, and Stefan Woltran, 'Complexity results for answer set programming with bounded predicate arities and implications', *Annals of Mathematics and Artificial Intelligence*, **51**(2-4), 123–165, (2007).

[13] Thomas Eiter, Wolfgang Faber, and Mushthofa Mushthofa, 'Space efficient evaluation of ASP programs with bounded predicate arities', in *AAAI'10*. AAAI Press, (2010).

[14] François Fages, 'Consistency of Clark's completion and existence of stable models', *Logical Methods in Computer Science*, **1**(1), 51–60, (1994).

[15] Jorge Fandinno and Markus Hecher, 'Treewidth-Aware Complexity in ASP: Not all Positive Cycles are Equally Hard', in *AAAI'21*, pp. 6312–6320. AAAI Press, (2021).

[16] Luis Fariñas del Cerro, Andreas Herzig, and Ezgi Iraz Su, 'Epistemic equilibrium logic', in *IJCAI'15*, pp. 2964–2970, (2015).

[17] Johannes K. Fichte, Markus Hecher, Michael Morak, and Stefan Woltran, 'Answer Set Solving with Bounded Treewidth Revisited', in *LPNMR'17*, volume 10377 of *LNCS*, pp. 132–145. Springer, (2017).

[18] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub, 'Multi-shot ASP solving with clingo', *Theory Pract. Log. Program.*, **19**(1), 27–82, (2019).

[19] Michael Gelfond, 'Strong Introspection', in *AAAI'91*, pp. 386–391. AAAI Press / The MIT Press, (1991).

[20] Markus Hecher, 'Treewidth-aware reductions of normal ASP to SAT - Is normal ASP harder than SAT after all?', *Artif. Intell.*, **304**, 103651, (2022).

[21] Markus Hecher, 'Characterizing Structural Hardness of Logic Programs: What makes Cycles and Reachability Hard for Treewidth?', in *AAAI'23*. AAAI Press, (2023). In Press.

[22] Markus Hecher, Michael Morak, and Stefan Woltran, 'Structural Decompositions of Epistemic Logic Programs', in *AAAI'20*, pp. 2830–2837. AAAI Press, (2020).

[23] Nicholas Hippen and Yuliya Lierler, 'Estimating grounding sizes of logic programs under answer set semantics', in *JELIA*, volume 12678 of *LNCS*, pp. 346–361. Springer, (2021).

[24] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane, 'Which Problems Have Strongly Exponential Complexity?', *Journal of Computer and System Sciences*, **63**(4), 512–530, (2001).

[25] Tomi Janhunen, 'Some (in)translatability results for normal logic programs and propositional theories', *Journal of Applied Non-Classical Logics*, **16**(1-2), 35–86, (2006).

[26] Patrick Thor Kahl, Richard Watson, Evgenii Balai, Michael Gelfond, and Yuanlin Zhang, 'The Language of Epistemic Specifications (Refined) Including a Prototype Solver', *J. Log. Comput.*, **25**, (2015).

[27] Roland Kaminski and Torsten Schaub, 'On the foundations of grounding in answer set programming', *CoRR*, **abs/2108.04769**, (2021).

[28] Vladimir Lifschitz and Alexander A. Razborov, 'Why are there so many loop formulas?', *ACM Transactions on Computational Logic*, **7**(2), 261–268, (2006).

[29] Wiktor Marek and Mirosław Truszczyński, 'Autoepistemic logic', *Journal of the ACM*, **38**(3), 588–619, (1991).

[30] David Mitchell, 'Guarded constraint models define treewidth preserving reductions', in *CP*, volume 11802 of *LNCS*, pp. 350–365. Springer, (2019).

[31] Michael Morak, 'Epistemic Logic Programs: A Different World View', in *ICLP'19*, pp. 52–64, (2019).

[32] Yi-Dong Shen and Thomas Eiter, 'Evaluating epistemic negation in answer set programming', *Artif. Intell.*, **237**, 115–135, (2016).

[33] Miroslaw Truszczyński, 'Trichotomy and dichotomy results on the complexity of reasoning with disjunctive logic programs', *Theory Pract. Log. Program.*, **11**(6), 881–904, (2011).

[34] Efthymia Tsamoura, Víctor Gutiérrez-Basulto, and Angelika Kimmig, 'Beyond the grounding bottleneck: Datalog techniques for inference in probabilistic logic programs', in *AAAI'20*, pp. 10284–10291. AAAI Press, (2020).

[35] Antonius Weinzierl, Richard Taupe, and Gerhard Friedrich, 'Advancing lazy-grounding ASP solving techniques - restarts, phase saving, heuristics, and more', *Theory Pract. Log. Program.*, **20**(5), 609–624, (2020).