Treewidth-Aware Complexity for Evaluating Epistemic Logic Programs

Jorge Fandinno[1] , **Markus Hecher**[2]
[1]University of Nebraska Omaha, NE, USA
[2]Massachusetts Institute of Technology, MA, USA
jfandinno@unomaha.edu, hecher@mit.edu

Abstract

Logic programs are a popular formalism for encoding many problems relevant to knowledge representation and reasoning as well as artificial intelligence. However, for modeling rational behavior it is oftentimes required to represent the concepts of knowledge and possibility. Epistemic logic programs (ELPs) is such an extension that enables both concepts, which correspond to being true in all or some possible worlds or stable models. For these programs, the parameter treewidth has recently regained popularity. We present complexity results for the evaluation of key ELP fragments for treewidth, which are exponentially better than known results for full ELPs. Unfortunately, we prove that obtained runtimes can not be significantly improved, assuming the exponential time hypothesis. Our approach defines treewidth-aware reductions between quantified Boolean formulas and ELPs. We also establish that the completion of a program, as used in modern solvers, can be turned treewidth-aware, thereby linearly preserving treewidth.

1 Introduction

The language of *epistemic specifications* [Gelfond, 1991; Gelfond and Przymusinska, 1993; Gelfond, 1994] (a.k.a. *epistemic logic programs*), proposed by Gelfond in 1991, extends disjunctive logic programs (under the *stable model* semantics; [Gelfond and Lifschitz, 1988; Gelfond and Lifschitz, 1991]) with modal constructs called *subjective literals*. The introduction of this extension was originally motivated by the need to correctly represent incomplete information in programs that have several stable models. Using subjective literals, it is possible to check whether a regular literal is true in every or some stable model of the program, those models are being collected in a set called *world view*. This allows for representing, within the language, whether some proposition should be understood accordingly to the open or the closed world assumption. See the work by [Fandinno *et al.*, 2021] for a recent survey about epistemic logic programs (ELPs).

In general, deciding whether an epistemic logic program has a world view is Σ_3^{P}-complete [Truszczyński, 2011]. However, for the *head-cycle-free* fragment, it complexity is reduced to Σ_2^{P}-complete. Though the complexity of this fragment is the same as disjunctive logic programs under the stable models semantics, the language of epistemic logic programs allows us to write more natural representations of several problems in the polynomial hierarchy than previous logic programming techniques such as saturation [Eiter and Gottlob, 1995]. Examples of this are (length-bounded) conformant planning [Kahl *et al.*, 2020; Cabalar *et al.*, 2021] and action reversibility [Faber *et al.*, 2021].

In this paper, we conduct a more fine-grained complexity analysis of epistemic logic programs in terms of parameters of a problem [Cygan *et al.*, 2015]. In particular, we focus on the influence of the parameter *treewidth* for solving *objectively tight* and head-cycle-free epistemic programs. An epistemic logic programs is *objectively tight* if its dependency graph does not have positive cycles that involve exclusively objective atoms. For these classes, we show that deciding the existence of a world view is $2^{2^{\mathcal{O}(k)}} \cdot \mathsf{poly}(|A(\Pi)|)$ and $2^{2^{\mathcal{O}(k\cdot\log(k))}} \cdot \mathsf{poly}(|A(\Pi)|)$, respectively, where k is the treewidth. That is, we can solve the consistency problem in polynomial time in the program size while being super-polynomial only in its treewidth. We also show that, under the *Exponential Time Hypothesis (ETH)* [Impagliazzo *et al.*, 2001], our characterizations are tight, that is, consistency for these classes cannot be decided in time $2^{2^{o(k)}} \cdot \mathsf{poly}(|A(\Pi)|)$ and $2^{2^{o(k\cdot\log(k))}} \cdot \mathsf{poly}(|A(\Pi)|)$, respectively.

Table 1 shows our results. So far, only the classical complexity [Shen and Eiter, 2016] as well as the parameterized complexity for treewidth on full ELPs are known [Hecher *et al.*, 2020]. Our fine-grained complexity results on the presented ELP fragments are not only exponentially better, they utilize novel reductions via *quantified Boolean formulas* that ensure treewidth-guarantees. This allows us to demonstrate treewidth-awareness of the completion used by modern solvers.

2 Preliminaries

We assume familiarity with complexity [Papadimitriou, 1994], graph theory [Bondy and Murty, 2008], and logic [Biere *et al.*, 2009]. We also assume familiarity with logic programs under the stable model semantics [Gelfond and Lifschitz, 1988].

Epistemic Logic Programs. Let A be a set of atoms. An *objective literal* is either an atom $a \in A$ (*positive literal*), an

Result \ Fragment	Non-Negative*+Tight	Non-Negative*	Tight	ι-Tight	Normal (HCF)	Full ELPs
Complexity	NP-complete	NP-complete	Σ_2^{P}-complete	Σ_2^{P}-complete	Σ_2^{P}-complete	Σ_3^{P}-complete
TW Upper Bound	$\mathbf{2}^{\mathcal{O}(\mathbf{w})}$	$\mathbf{2}^{\mathcal{O}(\mathbf{w}\cdot\log(\mathbf{w}))}$	$\mathbf{2}^{\mathbf{2}^{\mathcal{O}(\mathbf{w})}}$	$\mathbf{2}^{\mathbf{2}^{\mathcal{O}(\mathbf{w}\cdot\log(\iota))}}$	$\mathbf{2}^{\mathbf{2}^{\mathcal{O}(\mathbf{w}\cdot\log(\mathbf{w}))}}$	$2^{2^{2^{\mathcal{O}(w)}}}$
TW Lower Bound	$\mathbf{2}^{\mathbf{o}(\mathbf{w})}$	$\mathbf{2}^{\mathbf{o}(\mathbf{w}\cdot\log(\mathbf{w}))}$	$\mathbf{2}^{\mathbf{2}^{\mathbf{o}(\mathbf{w})}}$	$\mathbf{2}^{\mathbf{2}^{\mathbf{o}(\mathbf{w}\cdot\log(\iota))}}$	$\mathbf{2}^{\mathbf{2}^{\mathbf{o}(\mathbf{w}\cdot\log(\mathbf{w}))}}$	$2^{2^{2^{o(w)}}}$

Table 1: Complexity classification for world view existence over diverse ELP fragments. The first row shows existing completeness results. The second row gives upper bounds for treewidth w (without polynomial factors) and the third row states corresponding lower bounds (under ETH). New results are given in bold-face, others are known [Shen and Eiter, 2016; Hecher *et al.*, 2020]. *: Negation in constraints allowed.

atom preceded by negation $\neg a$ (*negative literal*) or a truth constant.[1] A *subjective literal* is an expression $\mathbf{K}\, a$ s.t. $a \in A$.

A *rule* r is an implication of the form $a_1 \vee \cdots \vee a_n \leftarrow L_1 \wedge \cdots \wedge L_m$ (1) with $n \geq 1$ and $m \geq 0$, where each a_i is an atom in A (or constant $\bot$) and each L_j a literal. If $n = 1$, then rule r is called *normal*. The left-hand disjunction of (1) is called the rule *head* and its set of atoms is abbreviated by H_r. The right hand side of (1) is called the rule *body* and the set of its literals is abbreviated as B_r. We denote by B_r^+ the set of all positive objective literals in B_r, by B_r^- the set of all negative objective literals in B_r, and B_r^s the set of all subjective literals in B_r. By abuse of notation, we also write B_r, B_r^+, B_r^-, and B_r^s to denote the conjunction of all literals in those sets. Furthermore B_r^c is $\bot$ if $\bot$ occurs in B_r and $\top$ otherwise. Atoms in A appearing in a rule r are given by $A(r)$. Further, $A(\Pi) := A$. Rule r is called *objective* if its literals are objective; r is *non-negative* if it is normal and either all objective literals are atoms or $H_r = \{\bot\}$.

A *program* Π is a set of rules and it is said to be *objective*, *normal*, or *non-negative* if all its rules are objective, normal, or non-negative, respectively. The *(positive) dependency digraph* D_Π of a program Π is the directed graph defined on the set of atoms from $\bigcup_{r\in\Pi} H_r \cup B_r^+$, where there is a directed edge from vertex a to vertex b iff there is a rule $r \in \Pi$ with $a \in B_r^+$ and $b \in H_r$. Recall that an objective program is called *tight* [Lifschitz, 1996] if its positive dependency graph contains no cycle. We generalize this notion to programs with subjective literals as follows. A program Π is *(objectively) tight* if D_Π contains no cycle. Further, program Π is called *head-cycle-free (HCF)* if D_Π contains no cycle consisting of at least two atoms of the head H_r of a rule r.

Note that according to our definition, the expression $a \leftarrow \mathbf{K}\, \neg b$ (2) is not a valid rule. This restriction simplifies the narrative of the paper and does not limit the expressiveness of the language, because we can simulate these more general rules using auxiliary atoms. In our particular example, we can replace (2) with the set $\{a \leftarrow \mathbf{K}\, not\_b, not\_b \leftarrow \neg b\}$, where not_b is a fresh auxiliary atom. Similarly, $a \leftarrow \neg\mathbf{K}\, b$ is not a rule according to our narrow definition, but it can be encoded by rules $a \leftarrow \neg k\_b$ and $k\_b \leftarrow \mathbf{K}\, b$, where k_b is an auxiliary atom not occurring anywhere else in the program.

For a non-empty set of propositional interpretations W, we write $W \models \mathbf{K}\, a$ if $a \in I$ for all $I \in W$. We write $W \not\models \mathbf{K}\, a$ otherwise. Given a program Π, by Π^W we denote the objective program obtained from Π by replacing each subjective literal L by $\top$ if $W \models L$ and by $\bot$ otherwise. W is called a *world view (WV)* of Π if and only if the set of stable models of Π^W is W.

[1]For brevity, we use constants with their usual meaning.

Tree Decompositions and Treewidth. For a rooted (directed) tree $T = (N, A)$ with *root* $\mathrm{root}(T)$ and a node $t \in N$, we let $\mathsf{chldr}(t)$ be the set of all nodes t', which have an edge $(t, t') \in A$. Let $G = (V, E)$ be a graph. A *tree decomposition (TD)* of a graph G is a pair $\mathcal{T} = (T, \chi)$, where T is a rooted tree, and χ is a mapping that assigns to each node t of T a set $\chi(t) \subseteq V$, called a *bag*, such that:

1. $V = \bigcup_{t \text{ of } T} \chi(t)$ and $E \subseteq \bigcup_{t \text{ of } T} \{\{u, v\} \mid u, v \in \chi(t)\}$
2. for each s lying on any r-t-path: $\chi(r) \cap \chi(t) \subseteq \chi(s)$.

Then, define $\mathsf{width}(\mathcal{T}) := \max_{t \text{ of } T} |\chi(t)| - 1$. The *treewidth* $\mathsf{tw}(G)$ of G is the minimum $\mathsf{width}(\mathcal{T})$ over all tree decompositions $\mathcal{T}$ of G. Observe that for every vertex $v \in V$, there is a unique node t^* with $v \in \chi(t^*)$ such that either $t^* = \mathrm{root}(T)$ or there is a node t of T with $\mathsf{chldr}(t) = \{t^*\}$ and $v \notin \chi(t)$. We refer to the node t^* by $\mathsf{last}(v)$. For arbitrary but fixed $w \geq 1$, it is feasible in linear time to decide if a graph has treewidth at most w and, if so, to compute a TD of width w [Cygan *et al.*, 2015]. In this work, we assume only TDs (T, χ), where for every node t of T, $|\mathsf{chldr}(t)| \leq 2$. Such a TD can be obtained from any TD in linear time without increasing the width.

Treewidth-Dependent Tightness. The *primal graph* $\mathcal{G}_\Pi$ of a program Π has as vertices the atoms $A(\Pi)$ with an edge between two vertices, whenever these vertices appear together in $A(r)$ of a rule $r \in \Pi$. For an atom $a \in A(\Pi)$ we denote the *strongly-connected component (SCC) of* a by $\mathrm{scc}(a)$, which is the $\subseteq$-largest set $C \subseteq A(\Pi)$ with $a \in C$ such that for every two distinct atoms u, v in C there is a directed path from u to v in D_Π. Then, the *tightness width* of a TD $\mathcal{T} = (T, \chi)$ of $\mathcal{G}_\Pi$, is $\max_{t \text{ of } T} \max_{x \in \chi(t)} |\chi(t) \cap \mathrm{scc}(x)|$. The *tightness treewidth* ι of Π is the smallest tightness width among every TD of width $\mathcal{O}(\mathsf{tw}(\mathcal{G}_\Pi))$. If Π has tightness treewidth ι, we call Π "*ι-tight*" [Fandinno and Hecher, 2021].

Quantified Boolean Formulas. Let ℓ be a positive integer, which we call *(quantifier) rank* later, and $\top$ and $\bot$ be the constant evaluating to 1 and 0, respectively. For a Boolean formula F, we abbreviate by $\mathsf{var}(F)$ the variables occurring in F and $F(X_1, \ldots, X_l)$ to indicate that $X_1, \ldots, X_l \subseteq \mathsf{var}(F)$. A *quantified Boolean formula* ϕ (in prenex normal form), *QBF* for short, is an expression of the form $\phi = Q_1 X_1.Q_2 X_2. \cdots Q_\ell X_\ell.F(X_1, \ldots, X_\ell)$, where for $1 \leq i \leq \ell$, we have $Q_i \in \{\forall, \exists\}$ and $Q_i \neq Q_{i+1}$, the X_i are disjoint, non-empty sets of Boolean variables, and F is a Boolean formula. We let $\mathsf{matrix}(\phi) := F$ and we say that ϕ is *closed* if $\mathsf{var}(\mathsf{matrix}(F)) = \bigcup_{i\in\ell} X_i$. We evaluate ϕ by $\exists x.\phi \equiv \phi[x \mapsto 1] \vee \phi[x \mapsto 0]$ and $\forall x.\phi \equiv \phi[x \mapsto 1] \wedge \phi[x \mapsto 0]$ for a variable x. We assume that $\mathsf{matrix}(\phi) = \psi_{\text{CNF}} \wedge \psi_{\text{DNF}}$, where ψ_{CNF} is in CNF (disjunction of conjunctions of literals) and ψ_{DNF} is in DNF (conjunction of disjunctions of literals). Then, depending on Q_ℓ, either ψ_{CNF} or ψ_{CNF} is optional, more precisely, ψ_{CNF} might be $\top$, if $Q_\ell = \forall$, and ψ_{DNF} is allowed to be $\top$, otherwise. The problem ℓ-QSAT asks, given a closed QBF ϕ of rank ℓ, whether $\phi \equiv 1$.

Treewidth for QBFs. For a given QBF ϕ with $\mathsf{matrix}(\phi) = \psi_{\text{CNF}} \wedge \psi_{\text{DNF}}$, we define the *primal graph* $\mathcal{G}_\phi = \mathcal{G}_{\mathsf{matrix}(\phi)}$, whose vertices are $\mathsf{var}(\mathsf{matrix}(\phi))$. Two vertices of $\mathcal{G}_\phi$ are adjoined by an edge, whenever the corresponding variables share a clause of ψ_{CNF} or a term of ψ_{DNF}, respectively.

Let $\mathsf{tower}(i,p)$ be $\mathsf{tower}(i-1, 2^p)$ if $i > 0$ and p otherwise. Further, we assume that $\mathsf{poly}(n)$ is any polynomial for a given positive integer n. The following result is known for QSAT.

Proposition 1 ([Chen, 2004])**.** *For any arbitrary QBF ϕ of quantifier rank $\ell > 0$, the problem ℓ-QSAT can be solved in time* $\mathsf{tower}(\ell, \mathcal{O}(\mathsf{tw}(\mathcal{G}_\varphi))) \cdot \mathsf{poly}(|\mathsf{var}(\phi)|)$.

Assuming the *exponential time hypothesis (ETH)* [Impagliazzo *et al.*, 2001], one cannot significantly improve this runtime in the worst case. Intuitively, ETH implies that SAT=1-QSAT can not be decided in time better than $2^{o(|\mathsf{var}(\varphi)|)}$ for an arbitrary formula φ.

Proposition 2 ([Fichte *et al.*, 2020])**.** *Under ETH, for any arbitrary QBF φ of quantifier rank $\ell > 0$, problem ℓ-QSAT cannot be solved in time* $\mathsf{tower}(\ell, o(\mathsf{tw}(\mathcal{G}_\varphi))) \cdot \mathsf{poly}(|\mathsf{var}(\varphi)|)$.

The result still holds for QBFs whose treewidth comprises variables V_ℓ of the inner-most quantifier, i.e., we may assume every TD bag contains constantly many variables not in V_ℓ.

3 Epistemic Logic Programs as QBFs

In this section, we show how we can encode ELPs as QBFs.

Programs as formulas. We present a way to compute the stable models of a program by translating it into a propositional formula. These translations are straightforward generalizations of the *completion* [Clark, 1977] and *level mappings* [Niemelä, 2008] to programs with subjective literals.

Let us denote by $K := \{k_a \mid a \in A(\Pi)\}$ a set of fresh new atoms that do not occur in program Π, that is $K \cap A = \emptyset$, where we refer to $A(\Pi)$ by A. Given any expression E (program, set, formula, etc.), by $k(E)$ we denote the result of replacing each occurrence of a subjective literal of the form $\mathbf{K}\, a$ by atom k_a.

Given a program Π, by COMP$[\Pi]$ we refer to the *completion*, denoting the conjunction of the following formulas:

$$a_r \leftrightarrow B_r^+ \wedge B_r^- \wedge B_r^c \wedge k(B_r^s) \wedge H_r(a) \quad \text{for every } r \in \Pi,\ a \in H_r \tag{3}$$

$$a \leftrightarrow \bigvee_{a \in H(r)} a_r \quad \text{for every } a \in A(\Pi) \tag{4}$$

where $H_r(a) := \bigwedge_{h \in H_r \setminus \{a\}} \neg h$. Note that COMP$[\Pi]$ is analogous to the completion used by ASP solvers [Gebser *et al.*, 2012; Alviano *et al.*, 2019; Lin and Zhao, 2004; Lierler and Maratea, 2004], but with the extra atoms in $k(B_r^s)$ to represent the subjective literals. These atoms behave as "externals" that can be fixed in order to compute the world views of the program. We formalize now this intuition.

Recall that for objective programs that are *tight* [Lifschitz, 1996], its stable models coincide with the classical models of its completion. We extend this to our notion of objectively tight programs below. Note that objective tightness only considers positive objective literals and is different from the notion of epistemically tight programs defined by [Cabalar *et al.*, 2020], so positive dependencies among subjective literals are allowed.

Proposition 3. *Let W be a non-empty set of propositional interpretations and Π be a program such that Π is objectively tight. Then, W is a world view of Π iff W is the set of all classical models of* COMP$[\Pi^W]$.

Proof. If Π is objectively tight, then Π^W is tight. The result follows immediately from Fages' theorem [Fages, 1994]. □

Definition of SM$[\Pi]$**.** We extend this to normal (HCF) programs as follows. If Π is tight, we define SM$[\Pi]$ to be COMP$[\Pi]$. Otherwise, SM$[\Pi]$ is the conjunction of COMP$[\Pi]$, where Formulas (3) are replaced as follows, using a strict partial ordering $\prec$[2] among atoms [Lin and Zhao, 2003; Janhunen, 2006].

$$a_r \leftrightarrow B_r^+ \wedge B_r^- \wedge B_r^c \wedge k(B_r^s) \wedge H_r(a) \wedge \bigwedge_{b \in B_r^+} (b \prec a) \quad \text{for every } r \in \Pi,\ a \in H_r \tag{5}$$

Proposition 4. *Let W be a non-empty set of propositional interpretations and Π be a program. Then, W is a world view of Π iff W is the set of classical models of* SM$[\Pi^W]$.

Proof. If Π is objectively tight, the result follows from Proposition 3. Otherwise, it holds by [Niemelä, 2008, Thm. 1]. □

We can also rewrite the right hand side of the equivalence of this proposition in terms of the auxiliary atoms in K as follows. If W is a non-empty set of propositional interpretations, then by $k(W)$ we denote $k(W) := \{k_a \mid a \in I \text{ for all } I \in W\}$.

Furthermore, if L is a subset of K, by F^L, we denote the formula obtained from F by replacing each occurrences of atom k_a by $\top$ if k_a is in L and by $\bot$ otherwise. Intuitively, F^L formalizes the epistemic reduct applied to the formula corresponding to a program.

Corollary 5. *Let W be a non-empty set of propositional interpretations and Π be a program. Then, W is a world view of Π iff W is the set of classical models of* SM$[\Pi]^{k(W)}$.

Proof. Note that SM$[\Pi^W]$ is equivalent to SM$[\Pi]^{k(W)}$. □

World views as QBFs. We now show how we can characterize the world views of a program Π as a QBF formula. Our characterization is similar to the characterization of Autoepistemic Logic given by [Egly *et al.*, 2000], but makes use of three modules that use the SM$[\Pi]$ formula.

$$\mathcal{F}_c[\Pi] = \exists A.\ \text{SM}[\Pi]$$

$$\mathcal{F}_k[\Pi] = \forall A.\ \Big(\text{SM}[\Pi] \rightarrow \bigwedge_{a \in A} (k_a \rightarrow a)\Big)$$

$$\mathcal{F}_p[\Pi] = \bigwedge_{a \in A} \Big(\neg k_a \rightarrow (\exists A. (\text{SM}[\Pi] \wedge \neg a))\Big)$$

Proposition 6. *Let L be a subset of K. The following holds:*

1. *L satisfies $\mathcal{F}_c[\Pi]$ iff* SM$[\Pi]^L$ *has a stable model;*
2. *L satisfies $\mathcal{F}_k[\Pi]$ iff, for all $k_a \in L$, every stable model of* SM$[\Pi]^L$ *satisfies a;*

[2] Ordering $\prec$ has to be irreflexive, asymmetric, and transitive.

3. *L satisfies $\mathcal{F}_p[\Pi]$ iff, for all $k_a \in K \setminus L$, some stable model of* $\mathrm{SM}[\Pi]^L$ *does not satisfy a.*

The following formula characterizes world views:

$$\mathcal{F}[\Pi] = \mathcal{F}_c[\Pi] \wedge \mathcal{F}_k[\Pi] \wedge \mathcal{F}_p[\Pi]$$

Proposition 7. *If W is a world view of Π, then $k(W)$ satisfies $\mathcal{F}[\Pi]$. Conversely, if L satisfies $\mathcal{F}[\Pi]$, then the set of classical models of* $\mathrm{SM}[\Pi]^L$ *is a world view of Π.*

Proof. Assume that W is a world view of Π. Then, from Corollary 5, W is the set of classical models of $\mathrm{SM}[\Pi]^{k(W)}$ and it is non-empty. From Proposition 6, this implies that $k(W)$ satisfies the three conjuncts of $\mathcal{F}[\Pi]$ and, thus, $\mathcal{F}[\Pi]$. Conversely, assume that L satisfies $\mathcal{F}[\Pi]$ and let W be the set of classical models of $\mathrm{SM}[\Pi]^L$. Then, from Corollary 5, W is non-empty and $k(W) = L$. That is, W is the set of classical models of $\mathrm{SM}[\Pi]^{k(W)}$. From Corollary 5, this implies that W is a world view of Π. $\square$

Formula $\mathcal{F}[\Pi]$ can be transformed to a closed QBF in prenex normal form as follows, where K and A, as used in $\mathcal{F}_c[\Pi]$, are existentially quantified. Since the existentially quantified variables A in $\mathcal{F}_p[\Pi]$ depend on the variable $a \in A$ of the outer-most conjunction, each conjunct has to be over fresh variables. Thereby every $b \in A$ with $b \neq a$ appearing in the conjunct is replaced by a fresh variable b_a, resulting in fresh sets A_a of existentially quantified variables. Finally, every universally quantified variable $a \in A$ appearing in $\mathcal{F}_k[\Pi]$ has to be replaced by a fresh copy a', resulting in a formula $\mathcal{F}'_k[\Pi]$ over variables K and A' (universally quantified).

4 Decomposition-Guided Reductions for ELPs

Inspired by related work [Hecher, 2022], a *decomposition-guided (DG) reduction* $\mathcal{R}$ is a function that takes both a program Π and a TD $\mathcal{T} = (T, \chi)$ of $\mathcal{G}_\Pi$, and returns a QBF φ.

The way a DG reduction is constructed for Π, it yields a TD $\mathcal{T}' = (T, \chi')$ of $\mathcal{G}_\varphi$ of the resulting QBF φ. So, the idea of such a DG reduction is to construct φ from a TD node's point of view. Thereby, for each node t of T, the constructed bag $\chi'(t)$ depends on the original bag $\chi(t)$, but also on the constructed bags of its child nodes. This gives rise to a function $f_\mathcal{R}$ that takes a TD node t, its bag $\chi(t)$ and a set $\chi'(\mathsf{chldr}(t)) := \{\chi'(t_i) \mid t_i \in \mathsf{chldr}(t)\}$ of constructed bags for the child nodes of t. Figure 1 illustrates such a function $f_\mathcal{R}$ taking node t, its original bag $\chi(t)$, as well as $\chi'(\mathsf{chldr}(t))$, to construct each bag $\chi'(t)$, which then corresponds to $f(t, \chi(t), \chi'(\mathsf{chldr}(t)))$.

Then, since $\mathsf{width}(\mathcal{T})$ is bounded by $\mathcal{O}(\max_{t \text{ of } T}(|\chi(t)|))$, also the treewidth of the resulting QBF is at most $\mathcal{O}(\max_{t \text{ of } T}(|f(t, \chi(t), \chi'(\mathsf{chldr}(t)))|))$. So, these DG reductions are *TD-guided*; their construction adheres to dynamic programming, thereby giving treewidth guarantees.

Decomposition-Guided Program Completion. As an example of DG reductions, we establish that Clark's completion [Clark, 1977], as created by ASP solvers, e.g., [Gebser *et al.*, 2012; Alviano *et al.*, 2019; Lin and Zhao, 2004; Lierler and Maratea, 2004] can be turned treewidth-aware. The completion ensures that rules are not only satisfied, but we have justifications for every atom in a stable model. For

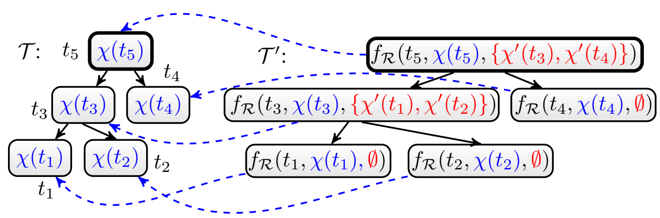

Figure 1: Illustration of a DG reduction $\mathcal{R}$ taking a program Π and a TD $\mathcal{T} = (T, \chi)$ of $\mathcal{G}_\Pi$, to construct a QBF φ. The DG reduction immediately yields a TD $\mathcal{T}' = (T, \chi')$ of $\mathcal{G}_\varphi$. Each bag $\chi'(t)$ of a node t of T functionally depends on t, $\chi(t)$, and $\chi'(\mathsf{chldr}(t))$.

tight programs this suffices to characterize stable models. So, let Π be a tight program and recall the completion $\mathsf{COMP}[\Pi]$.

In terms of treewidth overhead, Formulas (3) are not an issue and can be easily converted into CNF without auxiliary variables. However, in case there are many rules containing a, Formulas (4) might be problematic. We resolve this issue by means of auxiliary variables introduced in the decomposition-guided reduction. To this end, let $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$. We use auxiliary variables of the form $a_{\leq t}$ for every atom $a \in A(\Pi)$ and node t of T with $a \in \chi(t)$, and we guide evaluations along TD $\mathcal{T}$. By definition of TDs, for every rule $r \in \Pi$ there is a node t of T with $A(r) \subseteq \chi(t)$. For every node t, we refer by Π_t to a subset of those rules at t, i.e., $\Pi_t \subseteq \{r \in \Pi \mid A(r) \subseteq \chi(t)\}$. Then, instead of Formulas (4), we create the following.

$$a_{\leq t'} \leftarrow a \qquad \text{for every } t \text{ in } T, t' \in \mathsf{chldr}(t), a \in \chi(t') \setminus \chi(t) \quad (6)$$

$$\bigvee_{r \in \Pi_t} a_r \vee \bigvee_{\substack{t' \in \mathsf{chldr}(t),\\ a \in \chi(t')}} a_{\leq t'} \leftarrow a_{\leq t} \qquad \text{for every } t \text{ in } T, a \in \chi(t) \quad (7)$$

Intuitively, Formulas (6) ensure that whenever a holds, $a_{\leq t'}$ holds as well for a node t' of T whose bag contains a such that the parent bags do not contain a. Then, Formulas (7) take care that whenever $a_{\leq t}$ holds for a node t, either some a_r holds with $r \in \Pi_t$ or $a_{\leq t'}$ holds for $t' \in \mathsf{chldr}(t)$ below t. We refer to this version of the completion, comprising Formulas (3), (6), and (7), by $\mathsf{COMP}[\Pi, \mathcal{T}]$, which can be converted to CNF without additional auxiliary variables.

This slightly adapted construction of Clark's completion as used by ASP solvers indeed linearly preserves the treewidth.

Proposition 8 (TW-Aware Completion)**.** *Let Π be a program and $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$. Then, the completion $\Pi' = \mathcal{C}(\Pi, \mathcal{T})$ linearly preserves the (tree)width, i.e., there is a TD $\mathcal{T}$ of $\mathcal{G}_{\Pi'}$ with $|\mathsf{width}(\mathcal{T}')| \leq |\mathsf{width}(\mathcal{T})|$.*

Proof. Without loss of generality, we assume that $|\Pi_t| \leq 1$ for every node t of T and that for every rule r there is a unique node t with $r \in \Pi_t$, which can be established by adding intermediate auxiliary TD nodes. Further, we assume for every node t of T that $|\mathsf{chldr}(t)| \leq 2$, which can be also obtained via constructing auxiliary TD nodes. Since Π' was constructed by means of a DG reduction, it easily gives rise to a TD $\mathcal{T}' := (T, \chi')$ of $\mathcal{G}_{\Pi'}$. Precisely, we define $\chi'(t) := \chi(t) \cup \{a_r \mid r \in \Pi_t, a \in H_r\} \cup \{a_{\leq t} \mid a \in \chi(t)\} \cup \{a_{\leq t'} \mid t' \in \mathsf{chldr}(t), a \in \chi(t') \cap \chi(t)\}$ for every node t of T. Indeed $\mathcal{T}'$ is a well-defined TD of $\mathcal{G}_{\Pi'}$. Further, since $|\Pi_t| \leq 1$ and $|\mathsf{chldr}(t)| \leq$

2, we have $|\chi(t')| \leq 4\,|\chi(t)| + 1$. Consequently, also for $\mathsf{width}(\mathcal{T}) = \mathsf{tw}(\mathcal{G}_\Pi)$, we have $\mathsf{tw}(\mathcal{G}_{\Pi'}) \in \mathcal{O}(\mathsf{tw}(\mathcal{G}_\Pi))$. ☐

Note that while treewidth-aware variants of the completion has been presented before [Fandinno and Hecher, 2021; Eiter *et al.*, 2021], here we provide a different viewpoint, showing that the existing built-in reduction used by ASP solvers is almost treewidth-aware, i.e., only additional auxiliary variables are required to break-up long clauses. As above, one could extend our technique to HCF programs using a strict partial ordering $\prec_t$ among bag atoms [Fandinno and Hecher, 2021] for a TD node t. Below we pursue a different approach.

Definition of $\mathrm{SM}[\Pi, \mathcal{T}]$**.** If Π is a tight program, we define $\mathrm{SM}[\Pi, \mathcal{T}]$ to be $\mathrm{COMP}[\Pi, \mathcal{T}]$. Otherwise, i.e., for HCF programs Π, we first create a tight program Π' in a treewidth-aware way, which will be discussed afterwards in Section 5.4.

5 Treewidth-Aware QBF Encodings for ELPs

In order to design treewidth-aware encodings, we follow the basic modules from Section 3, thereby utilizing decomposition-guided reductions of the previous section. Let Π be a tight program over atoms $A = A(\Pi)$ and K be the set of fresh atoms for Π as defined in the previous section. Then, we solve the world view existence problem by $\exists K.\mathcal{F}[\Pi]$ using the QBF defined above. However, there is no guarantee on the treewidth of the primal graph of this formula.

In the following, we provide for each of the formulas $\mathcal{F}_c[\Pi]$, $\mathcal{F}_k[\Pi]$, $\mathcal{F}_p[\Pi]$ a decomposition-guided variant. To this end, let $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$ in order to define three DG reductions constructing QBFs for solving WV existence on Π with the help of $\mathcal{T}$. This will allow us then to evaluate problem parts locally for each node of T, thereby ensuring that the (tree)width is not increased arbitrarily. Precisely, we let $\mathcal{F}[\Pi, \mathcal{T}] := \exists K.\mathcal{F}_c[\Pi, \mathcal{T}] \wedge \mathcal{F}_k[\Pi, \mathcal{T}] \wedge \mathcal{F}_p[\Pi, \mathcal{T}]$ and we define the three conjuncts in the following.

For the consistency part, i.e., in order to encode $\mathcal{F}_c[\Pi]$ in a way that is decomposition-guided, we define $\mathcal{F}_c[\Pi, \mathcal{T}] := \exists A.\ \mathrm{SM}[\Pi, \mathcal{T}]$. Indeed, $\mathcal{F}_c[\Pi, \mathcal{T}]$ linearly preserves treewidth.

Lemma 9. *Given a tight program Π and a TD $\mathcal{T} = (T, \chi)$ of $\mathcal{G}_\Pi$. Then, the treewidth of $\mathcal{G}_{\mathcal{F}_c[\Pi,\mathcal{T}]}$ is in $\mathcal{O}(\mathsf{width}(\mathcal{T}))$.*

Proof. We define a TD $\mathcal{T}' = (T, \chi')$ of $\mathcal{G}_{\mathcal{F}_c[\Pi,\mathcal{T}]}$, where for every t in T we let $\chi'(t) := \chi(t) \cup \{a_r \mid r \in \Pi_t, a \in H_r\} \cup \{a_{\leq t} \mid a \in \chi(t)\} \cup \{a_{\leq t'} \mid t' \in \mathsf{chldr}(t), a \in \chi(t) \cap \chi(t')\} \cup \{k_a \mid k_a \in k(B_r^s), r \in \Pi_t\}$. Observe that $\mathcal{T}'$ is well-defined; indeed all rules of $\mathrm{SM}[\Pi, \mathcal{T}]$ are covered, i.e., the variables of every clause of $\mathrm{COMP}[k(\Pi), \mathcal{T}]$ appear in at least one common bag of $\mathcal{T}'$. ☐

The definition of the remaining formulas $\mathcal{F}_k[\Pi, \mathcal{T}]$ and $\mathcal{F}_p[\Pi, \mathcal{T}]$ is more involved. Intuitively, in order to locally evaluate both formulas for each node of T, we require additional information that is guided along $\mathcal{T}$, which we develop next.

5.1 Encoding Knowledge

Observe that in $\mathcal{F}_k[\Pi]$, the formula uses the result of $\mathrm{SM}[\Pi]$, which, when solving parts locally along the tree decomposition $\mathcal{T}$ (bottom-up), is not available until the root of T has been processed. Consequently, we require auxiliary variables that store and maintain this information.

To this end, we use auxiliary variables $usat_{\leq t}$ for every node t of T, that intuitively holds the information whether up to t, those parts of Π that have been encountered so far, are inconsistent. We define $U := \{usat_{\leq t} \mid t \text{ in } T\}$ and we construct $\mathcal{F}_k[\Pi, \mathcal{T}] := \forall(A \cup U).\mathcal{F}_{up}[\Pi, \mathcal{T}] \wedge \mathcal{F}_u[\Pi, \mathcal{T}]$, where $\mathcal{F}_{up}[\Pi, \mathcal{T}]$ is defined by means of Formulas (8)–(11), resulting in a DNF.

$$usat_{\leq t} \wedge \bigwedge_{t' \in \mathsf{chldr}(t)} \neg usat_{\leq t'} \qquad \text{for every } t \text{ in } T, \Pi_t = \emptyset \tag{8}$$

$$usat_{\leq t} \wedge \bigwedge_{t' \in \mathsf{chldr}(t)} \neg usat_{\leq t'} \wedge l \qquad \text{for every } t \text{ in } T, \{r\} = \Pi_t,\ l \in B_r^+ \cup \overline{k(B_r^s)} \cup B_r^- \cup H_r \tag{9}$$

$$\bigwedge_{l \in B_r^+ \cup k(B_r^s) \cup B_r^- \cup \overline{H_r}} l \wedge \neg usat_{\leq t} \qquad \text{for every } t \text{ in } T, \{r\} = \Pi_t \tag{10}$$

$$usat_{\leq t'} \wedge \neg usat_{\leq t} \qquad \text{for every } t \text{ in } T, t' \in \mathsf{chldr}(t) \tag{11}$$

Intuitively, Formulas (8) ignore cases for nodes t with Π_t, where the inconsistency is not properly propagated from a node to its child nodes. Then, Formulas (9) do the same for nodes t with $\Pi_t \neq \emptyset$, i.e., inconsistency for a node t requires inconsistency below t or in t. Formulas (10) and (11) encode the other direction, where inconsistency is underclaimed, i.e., not set in a node or not propagated from a node to its parent.

Further, we encode $\mathcal{F}_u[\Pi, \mathcal{T}]$ in CNF, comprising a conjunction of Formulas (12).

$$usat_{\leq \mathrm{root}(T)} \vee \neg k_a \vee a \qquad \text{for every } a \in A(\Pi) \tag{12}$$

Formulas (12) ensure that either the current assignment proves inconsistency or whenever an atom $a \in A(\Pi)$ is known, it has to be set to true.

Theorem 10 (Correctness)**.** *Let Π be a tight ELP and $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$. Then, given any assignment $\alpha : K \rightarrow \{0, 1\}$ we have that $\mathcal{F}_k[\Pi][\alpha]$ is valid iff $\mathcal{F}_k[\Pi, \mathcal{T}][\alpha]$ is valid.*

Proof (Sketch). $\Longrightarrow$: Assume that $\mathcal{F}_k[\Pi][\alpha]$ is valid. Then for any assignment $\beta : A \rightarrow \{0, 1\}$, we have that $\mathcal{F}_k[\Pi][\alpha][\beta]$ evaluates to true. From this we show that for any assignment $\beta' : U \rightarrow \{0, 1\}$ we have $\mathcal{F}_k[\Pi, \mathcal{T}][\alpha][\beta][\beta']$ evaluates to true as well. Assume towards a contradiction that this is not the case. However, Formulas (8) and (9) ensure that assignments β' are valid, where $\beta'(usat_{\leq t}) = 1$ for a node t but neither $\beta'(usat_{\leq t'}) = 1$ for any node $t' \in \mathsf{chldr}(t)$, nor a rule in Π_t is dissatisfied. Further, Formulas (10) and (11) ensure that whenever despite inconsistency up to a node t, $\beta'(usat_{\leq t}) = 0$. Consequently, we only need to consider remaining cases, where $usat_{\leq t}$ is maintained properly for every node t of T. Then, assuming that any Formula (12) is dissatisfied due to β', we have $\mathcal{F}_k[\Pi][\beta]$ is dissatisfied, contradicting our assumption. $\Longleftarrow$: The other direction works similarly. ☐

Further, we show that it is treewidth-aware and that the reduction causes at most a linear overhead.

Lemma 11 (Linear TW-Awareness)**.** *Let Π be a tight ELP and $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$. Then, the treewidth of $\mathcal{G}_{\mathcal{F}_k[\Pi,\mathcal{T}]}$ is in $\mathcal{O}(\mathsf{width}(\mathcal{T}))$.*

Proof. We construct a TD $\mathcal{T}' = (T, \chi')$, where we define χ' as follows. For every node t of T, we let $\chi'(t) := \chi(t) \cup k(\chi(t)) \cup \{usat_{\leq t}, usat_{\leq \text{root}(T)}\} \cup \{usat_{\leq t'} \mid t' \in \mathsf{chldr}(t)\}$. Observe that $\mathcal{T}'$ is well-defined and that since $|\mathsf{chldr}(t)| \leq 2$, we have $\mathsf{width}(\mathcal{T}') \in \mathcal{O}(\mathsf{width}(\mathcal{T}))$. $\square$

5.2 Encoding Possibility

In order to check whether an assumption on known atoms in K is fulfilled, it suffices to find counterexamples among all these modal atoms. This is different from the assumptions that are not known, since one has to find an individual witness model, over dedicated variables, as seen in the "possibility" part $\mathcal{F}_p[\Pi]$. However, this is problematic for treewidth, since these (linearly) many models are not bounded by the treewidth. To resolve this issue, we are going to guide these witness models along the tree decomposition. In other words, in the following we are utilizing synergies between all witness models, restricted to the bags of a node.

Again, we focus on the easier case assuming that Π is tight. Towards defining $\mathcal{F}_p[\Pi, \mathcal{T}]$, we use auxiliary variables of the form I_t for every node t of T and every interpretation $I \subseteq 2^{\chi(t)}$ restricted to $\chi(t)$, which we address by $I_\star := \{I_t \mid t \text{ in } T, I \subseteq 2^{\chi(t)}\}$. Observe that therefore the number of variables in $I_\star$ is exponential in the bag size $|\chi(t)|$. However, since we do not use these variables under an innermost universal quantifier block, our approach therefore does not the complexity, as we will see later. Precisely, we construct $\mathcal{F}_p[\Pi, \mathcal{T}] := \exists I_\star . \mathcal{F}_s[\Pi, \mathcal{T}]$, where $\mathcal{F}_s[\Pi, \mathcal{T}]$ is defined by means of Formulas (13)–(17) below.

$$\neg I_t \qquad \text{for every } t \text{ in } T, I \in 2^{\chi(t)}, I \not\models k(\Pi_t) \tag{13}$$

$$\bigvee_{k_a \in k(B_r^c)} \neg k_a \leftarrow I_t \qquad \text{for every } t \text{ in } T, I \in 2^{\chi(t)}, \Pi_t = \{r\}, (I \cup k(B_r^c)) \not\models k(\Pi_t) \tag{14}$$

$$\bigvee_{\substack{I' \in 2^{\chi(t')}, \\ I' \cap \chi(t) = I \cap \chi(t')}} I'_{t'} \leftarrow I_t \qquad \text{for every } t \text{ in } T, t' \in \mathsf{chldr}(t), I \in 2^{\chi(t)} \tag{15}$$

$$I'_{t'} \rightarrow \bigvee_{\substack{I \in 2^{\chi(t)}, \\ I \cap \chi(t') = I' \cap \chi(t)}} I_t \qquad \text{for every } t \text{ in } T, t' \in \mathsf{chldr}(t), I' \in 2^{\chi(t')} \tag{16}$$

$$\bigvee_{I \in 2^{\chi(t)}, a \notin I} I_t \vee k_a \qquad \text{for every } t \text{ in } T, a \in \chi(t) \tag{17}$$

Intuitively, our approach is to not maintain an individual assignment for every variable in K, which might increase the treewidth due to many copies. Instead, using $I_\star$, we store all potential assignments restricted to a bag, which indicates assignments that satisfy the program up to t. Then, Formulas (13) ensure that assignments not satisfying Π_t cannot be claimed stable models. Formulas (14) additionally handles the case where the assignment of specific atoms in K is needed in order to reach satisfiability. Formulas (15) and (16) ensure that whenever an assignment I is claimed to hold, there is a corresponding matching assignment in every child node as well as parent node (if exist). Thereby, Formulas (15) propagate from a node t to compatible predecessor interpretations, and Formulas (16) propagate to the parent node. Finally, Formulas (17) model the requirement that whenever $\neg k_a$ holds, some stable model that does not contain a is ultimately forced to hold.

Theorem 12 (Correctness)**.** *Let Π be a tight ELP and $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$. Then, given any assignment $\alpha : K \rightarrow \{0, 1\}$ we have that $\mathcal{F}_p[\Pi][\alpha]$ is valid iff $\mathcal{F}_p[\Pi, \mathcal{T}][\alpha]$ is valid.*

Proof (Sketch). $\Longrightarrow$: Assume that $\mathcal{F}_p[\Pi][\alpha]$ is valid. Then, for every k_a with $\alpha(k_a) = 0$, there is an assignment $\beta_a : A(\Pi) \rightarrow \{0, 1\}$ such that $\text{SM}[\Pi][\alpha][\beta_a] = \text{COMP}[\Pi][\alpha][\beta_a]$ evaluates to true. From this we construct an assignment β' such that $\mathcal{F}_p[\alpha][\beta']$ evaluates to true. For every node t and $a \in A(\Pi)$, we set $\beta'(I_t) := 1$, whenever $I = \chi(t) \cap \beta_a^{-1}(1)$ and $\beta'(I_t) := 0$ otherwise. Then, by construction of β', we have that $\mathcal{F}_p[\Pi, \mathcal{T}][\alpha][\beta']$ evaluates to true. $\Longleftarrow$: The other direction works as follows. We assume an assignment β' such that $\mathcal{F}_p[\Pi, \mathcal{T}][\alpha][\beta']$ evaluates to true and for every $k_a \in K$ with $\alpha(k_a) = 0$, we construct an assignment β_a such that $\text{SM}[\Pi][\alpha][\beta_a] = \text{COMP}[\Pi][\alpha][\beta_a]$ evaluates to true. To this end, we construct an assignment I, consisting of the union over one assignment I_t for every node t of T such that $\beta'(I_{tt}) = 1$ and $I_t \cap \chi(t') \cap I_{t'} \cap \chi(t)$ for every child node $t' \in \mathsf{chldr}(t)$. Since $\mathcal{F}_p[\Pi, \mathcal{T}][\alpha][\beta']$ evaluates to true, such an assignment I with $a \notin I$ has to exist by construction of this formula. Consequently, one can show that then $\text{SM}[\Pi][\alpha][\beta_a] = \text{COMP}[\Pi][\alpha][\beta_a]$ evaluates to true. $\square$

The reduction is treewidth-aware, i.e., while it causes an overhead in terms of treewidth, this is bounded by $2^{\mathcal{O}(w)}$.

Lemma 13 (TW-Awareness (single exponential))**.** *Let Π be a tight ELP and $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$. Then, the treewidth of $\mathcal{G}_{\mathcal{F}_p[\Pi, \mathcal{T}]}$ is in $2^{\mathcal{O}(\mathsf{width}(\mathcal{T}))}$.*

Proof. We construct a TD $\mathcal{T}' = (T, \chi')$, where we define χ' as follows. For every node t of T, we let $\chi'(t) := \chi(t) \cup k(\chi(t)) \cup \{I_t \mid I \in 2^{\chi(t)}\} \cup \{I'_{t'} \mid t' \in \mathsf{chldr}(t), I' \in 2^{\chi(t')}, I \in 2^{\chi(t)}, I \cap \chi(t') = I' \cap \chi(t)\}$. It is easy to see that $\mathcal{T}'$ is well-defined and that indeed since $|\mathsf{chldr}(t)| \leq 2$, we have that $\mathsf{width}(\mathcal{T}') \in 2^{\mathcal{O}(\mathsf{width}(\mathcal{T}))}$. $\square$

5.3 Merging Consistency, Knowledge & Possibility

Overall, we obtain the following runtime result.

Lemma 14 (Runtime)**.** *Let Π be any tight program and $\mathcal{T} = (T, \chi)$ be a TD of $\mathcal{G}_\Pi$ of width w. Then, the formula $\mathcal{F}[\Pi, \mathcal{T}]$ can be computed in time $2^{\mathcal{O}(w)} \cdot \mathsf{poly}(|A(\Pi)|)$.*

Proof. The runtime is due to the 2^w many different variables of the form I_t in the construction of formula $\mathcal{F}_p[\Pi, \mathcal{T}]$. $\square$

The formula $\mathcal{F}[\Pi, \mathcal{T}]$ can then be used to solve world view existence and to obtain the following upper bound result.

Theorem 15 (Upper Bound for tight ELPs)**.** *Let Π be any tight program, whose treewidth of primal graph $\mathcal{G}_\Pi$ is w. Then, WV existence for Π can be decided in time $2^{2^{\mathcal{O}(w)}} \cdot \mathsf{poly}(|A(\Pi)|)$.*

Proof. We construct both a TD $\mathcal{T}$ of $\mathcal{G}_\Pi$ of width $5 \cdot w$ [Bodlaender *et al.*, 2016] as well as the formula $\mathcal{F}[\Pi, \mathcal{T}]$, in time $2^{\mathcal{O}(w)} \cdot \mathsf{poly}(|A(\Pi)|)$. The constructed formula is an instance of 2-QSAT can be solved in the desired runtime, namely $2^{(2^{\mathcal{O}(w)})} + 2^{(2^{\mathcal{O}(w)})} \cdot 2^{2^{\mathcal{O}(w)}} \cdot \mathsf{poly}(|A(\Pi)|)$, see Proposition 1, since by Lemma 11 the treewidth over the inner-most (universally) quantified variables is linear in w. $\square$

However, for tight programs it is not expected that we can significantly improve this result, which we show below.

Theorem 16 (Lower Bound for tight ELPs)**.** *Let* Π *be any tight program, whose treewidth of the primal graph* $\mathcal{G}_\Pi$ *is* w. *Then, unless ETH fails, WV existence for* Π *cannot be decided in time* $2^{2^{o(w)}} \cdot \mathsf{poly}(|A(\Pi)|)$.

Proof. Let $\exists X.\forall Y.\varphi$ be any QBF with φ being in DNF and $\mathsf{var}(\varphi) = X \cup Y$. From this we construct an ELP Π over the set $\{sat, x, \tilde{x} \mid x \in \mathsf{var}(\varphi)\}$ of atoms as follows. For every variable $x \in X$, we construct the rules $x \leftarrow \mathbf{K}\, x$, $\tilde{x} \leftarrow \mathbf{K}\, \tilde{x}$, $\bot \leftarrow x, \tilde{x}$, and $\bot \leftarrow \neg x, \neg\tilde{x}$. For every $y \in Y$, we construct rules $y \leftarrow \neg\tilde{y}$ and $\tilde{y} \leftarrow \neg y$. Further, we build the rule $\bot \leftarrow \neg\mathbf{K}\, sat$, which can be easily transformed to our narrow syntax using $\mathbf{K}$ without negation, by means of auxiliary atoms, as explained in the preliminaries. Then, for each term $d \in \varphi$ consisting of literals $l_1, \ldots, l_i$, we construct: $sat \leftarrow \hat{l}_1, \ldots \hat{l}_i$, where for a literal l we let $\hat{l} := \tilde{a}$ if $l = \neg a$ and $\hat{l} := l$ otherwise. Observe that Π is normal and even tight. The reduction clearly runs in polynomial time.

Indeed, the reduction is correct. We show that there is a one-to-one correspondence between satisfying assignments over X of Q and world views of Π. Let $I \in 2^X$ be any assignment with $Q[I]$ being valid. Then, we construct a WV W by defining $W := \{\{sat\} \cup \{\tilde{x} \mid I(x) = 0\} \cup \{x \mid I(x) = 1\} \cup \{J \mid J \in 2^Y\}\}$. Assume towards a contradiction that W is not a WV of Π. But then, W does not coincide with set W' of stable models of Π^W. Take any $J \in W$. Observe that by construction of Π and since $Q[I]$ is valid, we have that J is also an answer set of Π^W. Further, $W' \subseteq W$ since W contains every subset over variables Y, which yields $W = W'$.

Take any WV W of Π. Then, by construction of Π, we have that $sat \in W'$ for every $W' \in W$ and consequently, for any stable model N of Π^W, $sat \in N$. From this we construct an assignment $I : X \to \{0, 1\}$ from some $W' \in W$ by $I(x) := 1$ for every $x \in X$ with $x \in W'$ and $I(x) := 0$ for every $x \in X$ with $\tilde{x} \in W'$. Assume towards a contradiction that $Q[I]$ is invalid. Then, there is an assignment $J : Y \to \{0, 1\}$ with $Q[I][J] = \{\emptyset\}$. As a consequence, we can define a stable model N' of Π^W with $N' := (W' \setminus \{sat\}) \cup \{y \mid J(y) = 1\} \cup \{\tilde{y} \mid J(y) = 0\}$. This contradicts W being a WV.

Further, the reduction is treewidth-aware and it even linearly preserves the treewidth. To this end, take any TD $\mathcal{T} = (T, \chi)$ of $\mathcal{G}_Q$. From this we construct a TD $\mathcal{T}' = (T, \chi')$ of $\mathcal{G}_\Pi$ as follows. For every node t of T, we define $\chi'(t) := \chi(t) \cup \{sat\} \cup \{\tilde{x} \mid x \in \chi(t)\}$. Observe that indeed $\mathcal{T}'$ is a well-defined TD of $\mathcal{G}_Q$. Obviously, we have that $|\chi'(t)| \leq 2\,|\chi(t)| + 1$ and therefore we follow that $\mathsf{tw}(\Pi) \in \mathcal{O}(\mathsf{tw}(Q))$. Then, assuming WV existence on Π can be decided in time $2^{2^{o(w)}} \cdot \mathsf{poly}(|A(\Pi)|)$ with $w = \mathsf{tw}(\mathcal{G}_\Pi)$ contradicts Proposition 2. $\square$

5.4 Normal (HCF) and ι-Tight Programs

For HCF programs Π, we first construct a tight program Π', using a treewidth-aware translation $\mathcal{R}$ from $k(\Pi)$ to $\Pi' = \mathcal{R}(k(\Pi))$ [Fandinno and Hecher, 2021]. Then, the treewidth of $\mathcal{G}_{\Pi'}$ is in $\mathcal{O}(\mathsf{tw}(\mathcal{G}_\Pi) \cdot \log(\mathsf{tw}(\mathcal{G}_\Pi)))$, an increase that is in line with known lower bounds under ETH [Hecher, 2022]. So, a significant improvement of this increase is unexpected.

After constructing tight program Π' and obtaining an adapted TD $\mathcal{T}'$ of $\mathcal{G}_{\Pi'}$, we apply $\mathcal{F}[\Pi', \mathcal{T}']$, as defined in the previous subsections, but where $K = k(A(\Pi))$.

Theorem 17 (Upper Bound for normal ELPs)**.** *Let* Π *be any normal ELP, whose treewidth of the primal graph* $\mathcal{G}_\Pi$ *is* w. *WV existence can be decided in time* $2^{2^{\mathcal{O}(w \cdot \log(w))}} \cdot \mathsf{poly}(|A(\Pi)|)$.

Proof (Sketch). Correctness of our approach follows from correctness of $\mathcal{R}$ and $\mathcal{F}$. $\mathcal{R}$ increases treewidth from w to $w \cdot \log(w)$ [Fandinno and Hecher, 2021]; applying Theorem 15 yields the result. $\square$

Theorem 18 (Tight Lower Bound for normal ELPs, $\star$[3])**.** *Let* Π *be any normal program, whose treewidth of the primal graph* $\mathcal{G}_\Pi$ *is* w. *Then, unless ETH fails, WV existence for* Π *cannot be decided in time* $2^{2^{o(w \cdot \log(w))}} \cdot \mathsf{poly}(|A(\Pi)|)$.

For ι-tight programs Π, the construction works analogously, but instead of $\mathcal{R}$, we use an adapted reduction $\mathcal{R}'$ [Hecher, 2023] to construct a tight program Π'. There, the treewidth of $\mathcal{G}_{\Pi'}$ is in $\mathcal{O}(\mathsf{tw}(\mathcal{G}_\Pi) \cdot \log(\iota))$. Consequently, we obtain results analogously to Theorems 17 and 18.

6 Discussion & Conclusion

We have focused here on the G94 semantics of ELPs by [Gelfond, 1994]. This was the original semantics that satisfy interesting properties from a knowledge representation point of view. Besides, many of our results carry on to the K15 semantics [Kahl *et al.*, 2015] due to the existence of known translations [Fandinno *et al.*, 2021]. For instance, any ELP interpreted under the K15 semantics can be understood as a different ELP under G94 semantics, which is obtained by replacing each expression of the form $\mathbf{K}\, a$ by conjunction $a \wedge \mathbf{K}\, a$.

As a result the upper bounds for every non-tight category carry on to this semantics as well. Note that in general, if a program is objectively tight under the K15 semantics does not mean that the corresponding program to be used under the G94 semantics is also objectively tight. For instance, $a \leftarrow \mathbf{K}\, a$ is objectively tight, but its corresponding program $a \leftarrow a \wedge \mathbf{K}\, a$ is not. Conversely, an ELP under G94 semantics can be translated into a new program under the K15 semantics where expressions $\mathbf{K}\, a$ are replaced by $not\_k\_not\_k\_a$ and the rules

$$not\_k\_not\_k\_a \leftarrow \neg k\_not\_k\_a \qquad not\_k\_a \leftarrow \neg k\_a$$
$$k\_not\_k\_a \leftarrow \mathbf{K}\, not\_k\_a \qquad k\_a \leftarrow \mathbf{K}\, a$$

with $not\_k\_not\_k\_a$, $k\_not\_k\_a$, not_k_a and k_a fresh auxiliary atoms. As a result, the lower bounds for every category not restricting the use of negation, carry on to these semantics.

In the future, we want to focus on implementations carrying out and comparing the presented QBF encodings. Since there is a known empirical correspondence between formulas of small treewidth and fast SAT solving [Atserias *et al.*, 2011], this raises the question of whether similar observations can be drawn for such QBF encodings. Also, we expect further insights from comparisons with existing ELP solvers.

[3]Proofs of statements marked with "$\star$" are given in the appendix.

Acknowledgments

The work has been carried out while Hecher visited the Simons Institute for the Theory of Computing. It is supported by Austrian Science Fund (FWF) grants J4656 and P32830, by the Society for Research Funding Lower Austria (GFF) grant ExzF-0004, as well as by the Vienna Science and Technology Fund (WWTF) grant ICT19-065.

References

[Alviano *et al.*, 2019] 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*, pages 241–255. Springer, 2019.

[Atserias *et al.*, 2011] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-Learning Algorithms with Many Restarts and Bounded-Width Resolution. *J. Artif. Intell. Res.*, 40:353–373, 2011.

[Biere *et al.*, 2009] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.

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

[Bondy and Murty, 2008] John A. Bondy and Uppaluri S. R. Murty. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, New York, USA, 2008.

[Cabalar *et al.*, 2020] Pedro Cabalar, Jorge Fandinno, and Luis Fariñas del Cerro. Autoepistemic answer set programming. *Artificial Intelligence*, 289:103382, 2020.

[Cabalar *et al.*, 2021] Pedro Cabalar, Jorge Fandinno, and Luis Fariñas del Cerro. Splitting epistemic logic programs. *Theory and Practice of Logic Programming*, 21:296–316, 2021.

[Chen, 2004] Hubie Chen. Quantified Constraint Satisfaction and Bounded Treewidth. In *16th European Conference on Artificial Intelligence (ECAI'04)*, pages 161–165. IOS Press, 2004.

[Clark, 1977] Keith L. Clark. Negation as failure. In *Logic and Data Bases*, Advances in Data Base Theory, pages 293–322. Plemum Press, 1977.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Egly *et al.*, 2000] Uwe Egly, Thomas Eiter, Hans Tompits, and Stefan Woltran. Solving advanced reasoning tasks using quantified boolean formulas. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'00)*, pages 417–422. AAAI/MIT Press, 2000.

[Eiter and Gottlob, 1995] Thomas Eiter and Georg Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.

[Eiter *et al.*, 2021] Thomas Eiter, Markus Hecher, and Rafael Kiesel. Treewidth-Aware Cycle Breaking for Algebraic Answer Set Counting. In *Proceedings of the Eighteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'21)*, pages 269–279, 2021.

[Faber *et al.*, 2021] Wolfgang Faber, Michael Morak, and Luk ás Chrpa. Determining action reversibility in STRIPS using answer set and epistemic logic programming. *Theory and Practice of Logic Programming*, 21(5):646–662, 2021.

[Fages, 1994] Francois Fages. Consistency of Clark's completion and the existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.

[Fandinno and Hecher, 2021] Jorge Fandinno and Markus Hecher. Treewidth-aware complexity in ASP: not all positive cycles are equally hard. In *Proceedings of the Thirty-Fifth International Conference on Artificial Intelligence (AAAI'21)*, pages 6312–6320. AAAI Press, 2021.

[Fandinno *et al.*, 2021] Jorge Fandinno, Wolfgang Faber, and Michael Gelfond. Thirty years of epistemic specifications. *Theory and Practice of Logic Programming*, page 1–41, 2021.

[Fichte *et al.*, 2020] Johannes K. Fichte, Markus Hecher, and Andreas Pfandler. Lower Bounds for QBFs of Bounded Treewidth. In *Proceedings of the Thirty-Fifth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'20)*, pages 410–424. Assoc. Comput. Mach., New York, 2020.

[Gebser *et al.*, 2012] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Morgan & Claypool, 2012.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. MIT Press, 1988.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Gelfond and Przymusinska, 1993] Michael Gelfond and Halina Przymusinska. Reasoning on open domains. In Luís M. Pereira and Anil Nerode, editors, *Logic Programming and Non-monotonic Reasoning, Proceedings of the Second International Workshop, Lisbon, Portugal, June 1993*, pages 397–413. MIT Press, 1993.

[Gelfond, 1991] Michael Gelfond. Strong introspection. In Thomas L. Dean and Kathleen R. McKeown, editors, *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 386–391. AAAI Press / The MIT Press, 1991.

[Gelfond, 1994] Michael Gelfond. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence*, 12(1-2):89–116, 1994.

[Hecher *et al.*, 2020] Markus Hecher, Michael Morak, and Stefan Woltran. Structural decompositions of epistemic logic programs. In V. Conitzer and F. Sha, editors, *Proceedings of the Thirty-fourth National Conference on Artificial Intelligence (AAAI'20)*, pages 2830–2837. AAAI Press, 2020.

[Hecher, 2022] Markus Hecher. Treewidth-aware reductions of normal ASP to SAT - is normal ASP harder than SAT after all? *Artif. Intell.*, 304:103651, 2022.

[Hecher, 2023] Markus Hecher. Characterizing Structural Hardness of Logic Programs: What makes Cycles and Reachability Hard for Treewidth? In *Proceedings of the Thirty-Seventh International Conference on Artificial Intelligence (AAAI'23)*. AAAI Press, 2023. In Press.

[Impagliazzo *et al.*, 2001] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. of Computer and System Sciences*, 63(4):512–530, 2001.

[Janhunen, 2006] 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.

[Kahl *et al.*, 2015] Patrick Kahl, Richard Watson, Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. The language of epistemic specifications (refined) including a prototype solver. *Journal of Logic and Computation*, 09 2015.

[Kahl *et al.*, 2020] Patrick Kahl, Richard Watson, Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. The language of epistemic specifications (refined) including a prototype solver. *Journal of Logic and Computation*, 30(4):953–989, 2020.

[Lierler and Maratea, 2004] Yuliya Lierler and Marco Maratea. Cmodels-2: Sat-based answer set solver enhanced to non-tight programs. In Vladimir Lifschitz and Ilkka Niemelä, editors, *LPNMR'04*, volume 2923 of *LNCS*, pages 346–350. Springer, 2004.

[Lifschitz, 1996] Vladimir Lifschitz. Foundations of logic programming. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 69–127. CSLI Publications, 1996.

[Lin and Zhao, 2003] Fangzhen Lin and Jicheng Zhao. On tight logic programs and yet another translation from normal logic programs to propositional logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 853–858. Morgan Kaufmann, August 2003.

[Lin and Zhao, 2004] Fangzhen Lin and Xishun Zhao. On odd and even cycles in normal logic programs. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'04)*, pages 80–85. AAAI Press / The MIT Press, 2004.

[Niemelä, 2008] Ilkka Niemelä. Stable models and difference logic. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):313–329, 2008.

[Papadimitriou, 1994] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Shen and Eiter, 2016] Yi-Dong Shen and Thomas Eiter. Evaluating epistemic negation in answer set programming. *Artificial Intelligence*, 237:115–135, 2016.

[Truszczyński, 2011] Mirosław Truszczyński. Revisiting epistemic specifications. In Marcello Balduccini and Tran C. Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays Dedicated to Michael Gelfond on the Occasion of his 65th Birthday*, volume 6565 of *LNCS*, pages 315–333. Springer, 2011.