Treewidth-Aware Complexity for Evaluating Epistemic Logic Programs

Jorge Fandinno\textsuperscript{1}, Markus Hecher\textsuperscript{2}

\textsuperscript{1}University of Nebraska Omaha, NE, USA
\textsuperscript{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 $\Sigma^p_3$-complete [Truszczynski, 2011]. However, for the head-cycle-free fragment, it complexity is reduced to $\Sigma^p_2$-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 program 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 $2^{o(k)} \cdot \text{poly}(|A(\Pi)|)$ and $2^{O(k \log(k))} \cdot \text{poly}(|A(\Pi)|)$, respectively, where $k$ is the treewidth.

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
atom preceded by negation \( \neg a \) (negative literal) or a truth constant.\(^1\) A subjective literal is an expression \( K \) a s.t. \( a \in A \).

A rule \( r \) is an implication of the form \( a_1 \lor \cdots \lor a_n \leftarrow L_1 \land \cdots \land L_m \) (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 \( \delta_r \). 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 \( B_r^0 \) the set of all subjective literals in \( B_r \). By abuse of notation, we also write \( B_r \), \( B_r^+ \), \( B_r^- \), and \( B_r^0 \) to denote the conjunction of all literals in those sets. Furthermore \( B_r^+ \) 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, \( A(P) = A \). 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 \( B_r = \{ \bot \} \).

A program \( P \) 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 \( D_{H_1} \) of a program \( P \) is the directed graph defined on the set of atoms from \( \bigcup_{r \in P} H_r \cup B_r^+ \), where there is a directed edge from vertex \( a \) to vertex \( b \) iff there is a rule \( r \in P \) with \( a \in B_r^+ \) and \( b \in H_r \). Recall that an objective program is called tight [Litschitz, 1996] if its positive dependency graph contains no cycle. We generalize this notion to programs with subjective literals as follows. A program \( P \) is (objectively) tight if \( D_{H_1} \) contains no cycle. Further, program \( P \) is called head-cycle-free (HCF) if \( D_{H_1} \) 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 \( a \leftarrow K \neg b (2) \) 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 \( \{ a \leftarrow K \neg b, \neg b, b \leftarrow \bot \} \), where \( \neg b \) is a fresh auxiliary atom. Similarly, \( a \leftarrow \neg K b \) is not a rule according to our narrow definition, but it can be encoded by rules \( a \leftarrow \neg k.b \) and \( k.b \leftarrow K b \), where \( k.b \) is an auxiliary atom not occurring anywhere else in the program.

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

### Tree Decompositions and Treewidth.

For a rooted (directed) tree \( T = (N, A) \) with root \( \text{root}(T) \) and a node \( t \in N \), we let \( \text{child}(t) \) 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 \( T = (T, \chi) \), where \( T \) is a rooted tree, and \( \chi \) is a mapping that assigns to each node \( t \) of \( T \) a set \( \chi(t) \subseteq V \) called a bag, such that:

1. \( \forall v \in V \triangledown \chi(t) \) and \( \forall \) \( e \in E \) \( \chi(t) \cap \chi(s) \neq \emptyset \).
2. For each \( s \) lying on any \( r-t-path: \chi(r) \cap \chi(t) \subseteq \chi(s) \).

Then, define width \( T := \max_{t \in T} |\chi(t)| - 1 \). The treewidth \( tw(G) \) of \( G \) is the minimum width \( T \) over all tree decompositions \( T \) of \( G \). Observe that for every vertex \( v \in V \), there is a unique node \( t^* \) with \( v \in \chi(t^*) \) such that \( v = \text{root}(T) \) or there is a node \( t \) of \( T \) with \( \text{child}(t) \cap \chi(t^*) \neq \emptyset \) and \( v \notin \chi(t) \). We refer to the node \( t^* \) by last \( v \). For arbitrary but fixed \( v \), 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 \( (T, \chi) \), where for the node \( t \) of \( T \), \( |\text{child}(t)| \leq 2 \). Such a TD can be obtained from any TD in linear time without increasing the width.

### Tree-Dependent Tightness.

The primal graph \( \mathcal{G}_1 \) of program \( P \) has as vertices the atoms \( A(\Pi) \) with an edge between two vertices, whenever these vertices appear together in \( A(\Pi) \) of a rule \( r \in \Pi \). For an atom \( a \in A(\Pi) \) we denote the strongly-connected component (SCC) of \( a \) by \( \text{sc}(a) \), which is the 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 \( P \). Then, the tightness width of a TD \( T = (T, \chi) \) of \( \mathcal{G}_1 \) is \( \max_{t \in T} \max_{v \in \chi(t)} |\chi(t) \cap \text{sc}(v)| \).

The treewidth \( tw(\Pi) \) of \( \Pi \) is the smallest tightness width among every TD of width \( \mathcal{O}(tw(G_{\Pi})) \). If \( \Pi \) has tightness treewidth \( t \), we call \( \Pi \) “\( t \)-tight” [Fandinno and Hecher, 2021].

### Quantified Boolean Formulas.

Let \( t \) 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 \( \text{var}(F) \) the variables occurring in \( F \) and \( \text{var}(X_1, \ldots, X_i) \) to indicate that \( X_1, \ldots, X_i \subseteq \text{var}(F) \). A quantified Boolean formula \( \phi \) (in prefix normal form), QBF for short, is an expression of the form \( \phi = Q_1 X_1. Q_2 X_2. \cdots Q_i X_i.F(X_1, \ldots, X_i) \), where for \( 1 \leq i \leq t \), 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 \( \text{mat}(\phi) := F \) and we say that \( \phi \) is closed if \( \text{var}(\text{mat}(F)) = \bigcup_{X_i} X_i \). We evaluate \( \phi \) by \( \exists X_1. \phi \equiv \phi[x \mapsto 1] \lor \phi[x \mapsto 0] \) and \( \forall X_1. \phi \equiv \phi[x \mapsto 1] \land \phi[x \mapsto 0] \) for a variable \( x \). We assume that \( \text{mat}(\phi) = \psi_{\text{CNF}} \land \psi_{\text{DNNF}} \), where \( \psi_{\text{CNF}} \) is in CNF (disjunction

---

\(^1\)For brevity, we use constants with their usual meaning.

---

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].

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Non-Negative*+Tight</th>
<th>Non-Negative*</th>
<th>Tight</th>
<th>( t )-Tight</th>
<th>Normal (HCF)</th>
<th>Full ELPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TW Upper Bound</td>
<td>( 2^O(n) )</td>
<td>( 2^O(n \log \log(n)) )</td>
<td>( \Sigma_2^p )-complete</td>
<td>( \Sigma_2^p )-complete</td>
<td>( \Sigma_2^p )-complete</td>
<td>( \Sigma_2^p )-complete</td>
</tr>
<tr>
<td>TW Lower Bound</td>
<td>( 2^O(n) )</td>
<td>( 2^O(n \log \log(w)) )</td>
<td>( 2^O(w \log \log(w)) )</td>
<td>( 2^O(w \log \log(w)) )</td>
<td>( 2^O(w \log \log(w)) )</td>
<td>( 2^O(w \log \log(w)) )</td>
</tr>
</tbody>
</table>
of conjunctions of literals) and $\psi_{\text{DNF}}$ is in DNF (conjunction of disjunctions of literals). Then, depending on $Q_\ell$, either $\psi_{\text{CNF}}$ or $\psi_{\text{CNF}}$ is optional, more precisely, $\psi_{\text{CNF}}$ might be $\top$, if $Q_\ell = \forall$, and $\psi_{\text{DNF}}$ is allowed to be $\top$, otherwise. The problem $\ell$-Q$\text{SAT}$ asks, given a closed QBF $\phi$ of rank $\ell$, whether $\phi \equiv 1$.

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

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

**Proposition 1** ([Chen, 2004]). For any arbitrary QBF $\phi$ of quantifier rank $\ell > 0$, the problem $\ell$-Q$\text{SAT}$ can be solved in time tower$(\ell, \text{O}(\text{tw}(G_\phi))) \cdot \text{poly}(\text{var}(\phi))$.

Assuming the exponential time hypothesis (ETH) [Impagliazzo et al., 2001], one cannot significantly improve this runtime in the worst case. Intuitively, ETH implies that Q$\text{SAT} = 1$-Q$\text{SAT}$ cannot be decided in time better than $2^n(\text{poly}(\text{var}(\phi)))$ for an arbitrary formula $\phi$.

**Proposition 2** ([Fichte et al., 2020]). Under ETH, for any arbitrary QBF $\phi$ of quantifier rank $\ell > 0$, problem $\ell$-Q$\text{SAT}$ cannot be solved in time tower$(\ell, \text{O}(\text{tw}(G_\phi))) \cdot \text{poly}(\text{var}(\phi))$.

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

### 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 $K := \{k_a | a \in A(\Pi)\}$ a set of fresh new atoms that do not occur in program $\Pi$, 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 $K a$ by atom $k_a$.

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

$$a_r \leftrightarrow B^+_r \land B^-_r \land B^c_w \land k(B^+_w) \land H_r(a) \quad \text{for every } r \in \Pi,$$

$$a \leftrightarrow \bigvee_{a \in H_r(\ell)} a_r \quad \text{for every } a \in A(\Pi)$$

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 $k(B^+_w)$ 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 $\Pi$ be a non-empty set of propositional interpretations and $\Pi$ be a program such that $\Pi$ is objectively tight. Then, $W$ is a world view of $\Pi$ if $W$ is the set of all classical models of COMP$[\Pi]_W$.

**Definition of SM$[\Pi]$.** We extend this to normal (HCF) programs as follows. If $\Pi$ 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 $\prec$ among atoms [Lin and Zhao, 2003; Janhunen, 2006].

$$a_r \leftrightarrow B^+_r \land B^-_r \land B^c_w \land k(B^+_w) \land H_r(a) \land \bigwedge_{b \in B^+_r} (b \prec a) \quad \text{for every } r \in \Pi,$$

$$\bigwedge_{b \in B^+_r} (b \prec a) \quad \text{for every } r \in \Pi$$

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

**Corollary 5.** Let $\Pi$ be a non-empty set of propositional interpretations and $\Pi$ be a program. Then, $W$ is a world view of $\Pi$ if $W$ is the set of all classical models of SM$[\Pi]_W$.

**World views as QBFs.** We now show how we can characterize the world views of a program 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.

$$\forall a. \text{SM}[\Pi] \land \bigwedge_{a \in A} (\neg k_a \rightarrow (\exists A. \text{SM}[\Pi] \land \neg a))$$

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

1. $L$ satisfies $F_c[\Pi]$ iff SM$[\Pi]^L$ has a stable model;
2. $L$ satisfies $F_k[\Pi]$ 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 $F_P[I]$ iff, for all $k_a \in K \setminus L$, some stable model of $SM[I]^L$ does not satisfy $a$.

The following formula characterizes world views:

$$ F[I] = F_K[I] \land F_5[I] \land F_P[I] $$

**Proposition 7.** If $W$ is a world view of $I$, then $k(W)$ satisfies $F[I]$. Conversely, if $L$ satisfies $F[I]$, then the set of classical models of $SM[I]^L$ is a world view of $I$.

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

Formula $F[I]$ can be transformed to a closed QBF in prefix normal form as follows, where $K$ and $A$, as used in $F_K[I]$, are existentially quantified. Since the existentially quantified variables $A$ in $F_P[I]$ depend on the variable $A$ 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 $F_K[I]$ has to be replaced by a fresh copy $a'$, resulting in a formula $F'_{k_1}$ over variables $K$ and $A'$ (universally quantified).

### 4 Decomposition-Guided Reductions for ELPs

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

The way a DG reduction is constructed for $I$, it yields a TD $T' = (T, \chi')$ of $G_{\varphi}$ of the resulting QBF $\varphi$. So, the idea of such a DG reduction is to construct $\varphi$ 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 $f_R$ that takes a TD node $t$, its bag $\chi(t)$ and a set $\chi'(\text{chldr}(t)) := \{\chi(t_i) | t_i \in \text{chldr}(t)\}$ of constructed bags for the child nodes of $t$. Figure 1 illustrates such a function $f_R$ taking node $t$, its original bag $\chi(t)$, as well as $\chi'(\text{chldr}(t))$, to construct each bag $\chi'(t_i)$, which then corresponds to $f_R(t, \chi(t), \chi'(\text{chldr}(t)))$.

Then, since width$(T)$ is bounded by $O(\max_{a \in T} |\chi(t)\|))$, also the treewidth of the resulting QBF is at most $O(\max_{a \in T} |f_R(t, \chi(t), \chi'(\text{chldr}(t)))\|))$. So, these DG reductions are TD$^3$-guided; their construction adheres to dynamic programming, thereby giving treewidth guarantees.

**Decomposition-Guided Program Completion.** As an example of such 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 tight programs this suffices to characterize stable models. So, let $I$ be a tight program and recall the completion $\text{COMP}[I]$.

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 $I$, Formulas (4) might be problematic. We resolve this issue by means of a DG reduction, it easily gives rise to a TD $T'$.

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

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

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 build-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 \(\prec\) among bag atoms [Fandinno and Hecher, 2021] for a TD node \(t\). Below we pursue a different approach.

Definition of SM[II, \(T\)]. If \(I\) is a tight program, we define SM[II, \(T\)] to be COMP[II, \(T\)]. Otherwise, i.e., for HCF programs II, we first create a tight program II’ 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 II be a tight program over atoms \(A = A(II)\) and \(K\) be the set of fresh atoms for II as defined in the previous section. Then, we solve the world view existence problem by \(\exists K.F[II]\) 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 \(F_{c}[II]\), \(F_{k}[II]\), \(F_{p}[II]\) a decomposition-guided variant. To this end, let \(T = (T, \chi)\) be a TD of \(\mathcal{G}_{II}\) in order to define three DG reductions constructing QBFs for solving WV existence on II with the help of \(T\). This will allow us to then to solve problem parts locally for each node of \(T\), thereby ensuring that the (tree)width is not increased arbitrarily. Precisely, we let \(F[II, T] := \exists K.F_{c}[II, T] \land F_{k}[II, T] \land F_{p}[II, T]\) and we define the three conjuncts in the following.

For the consistency part, i.e., in order to encode \(F_{c}[II]\) in a way that is decomposition-guided, we define \(F_{c}[II, T] := \exists \mathcal{A}.\text{SM}[II, T]\). Indeed, \(F_{c}[II, T]\) linearly preserves treewidth.

Lemma 9. Given a tight program II and a TD \(T = (T, \chi)\) of \(\mathcal{G}_{II}\). Then, the treewidth of \(\mathcal{G}_{F_c[II, T]}\) is in \(\mathcal{O}(\text{width}(T))\).

Proof. We define a TD \(T' = (T', \chi')\) of \(\mathcal{G}_{F_c[II, T]}\), where for every t in T we let \(\chi'(t) := \chi(t) \lor \{a_{r} | r \in \Pi_{a}, a \in H_{r} \lor \{a_{c} | a \in \chi(t) \lor \{a_{l} | t' \in \text{chldr}(t), a \in \chi(t) \land \chi(t')\} \lor \{b_{a} | b_{a} \in k(B_{a}) \land r \in \Pi_{a}\}. Observe that \(T'\) is well-defined, indeed all rules of SM[II, T] are covered, i.e., the variables of every clause of \(\text{COMP}[k(II), T]\) appear in at least one common bag of \(T'\).

The definition of the remaining formulas \(F_{k}[II, T]\) and \(F_{p}[II, T]\) is more involved. Intuitively, in order to locally evaluate both formulas for each node of \(T\), we require additional information that is guided along \(T\), which we develop next.

5.1 Encoding Knowledge

Observe that in \(F_{k}[II]\), the formula uses the result of SM[II], which, when solving parts locally along the tree decomposition \(T\) (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 \(usat_{t'}\) for every node \(t\) of \(T\), that intuitively holds the information whether up to \(t\), those parts of \(II\) that have been encountered so far, are inconsistent. We define \(U := \{usat_{t} | t \in T\}\) and we construct \(F_{k}[II, T] := \forall (A \cup U).\mathcal{F}_{ap}[II, T] \land \mathcal{F}_{k}[II, T]\), where \(\mathcal{F}_{ap}[II, T]\) is defined by means of Formulas (8)–(11), resulting in a DNF.

\[
usat_{t} \land \neg \usat_{t'} \quad \text{for every } t \in T, \Pi_{t} = 0 \quad (8)
\]

\[
usat_{t} \land \neg \usat_{t'} \land \{l \land \usat_{t} \quad \text{for every } t \in T, \{r\} = \Pi_{t}, \quad (9)
\]

\[
usat_{t} \land \{l \land \usat_{t} \quad \text{for every } t \in T, \{r\} = \Pi_{t} \quad (10)
\]

Intuitively, Formulas (8) ignore cases for nodes \(t\) with \(\Pi_{t}\), where the inconsistency is not properly propagated from a node to its child nodes. Then, Formulas (9) do the same for nodes \(t\) with \(\Pi_{t} \neq 0\), 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 \(F_{k}[II, T]\) in CNF, comprising a conjunction of Formulas (12).

\[
usat_{t_{\text{root}}(T)} \lor \neg a_{a} \quad \text{for every } a \in A(II) \quad (12)
\]

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

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

Proof (Sketch). \(\Longrightarrow\): Assume that \(F_{k}[II][\alpha]\) is valid. Then for any assignment \(\beta : A \to \{0, 1\}\), we have that \(\mathcal{F}_{k}[II, T][\alpha][\beta]\) evaluates to true. From this we show that for any assignment \(\beta' : U \to \{0, 1\}\) we have \(\mathcal{F}_{k}[II, T][\alpha][\beta'][\beta'']\) evaluates to true as well. Assume towards a contradiction that this is not the case. However, Formulas (8) and (9) ensure that assignments \(\beta'\) are valid, where \(\beta'(usat_{t}) = 1\) for a node \(t\) but neither \(\beta'(usat_{t'}) = 1\) for any node \(t' \in \text{chldr}(t)\), nor a rule in \(I\), is dissatisfied. Further, Formulas (10) and (11) ensure that whenever despite inconsistency up to a node \(t\), \(\beta'(usat_{t}) = 0\). Consequently, we only need to consider remaining cases, where \(usat_{t'}\) is maintained properly for every node \(t\) of \(T\). Then, assuming that any Formula (12) is dissatisfied due to \(\beta'\), we have \(\mathcal{F}_{k}[II][\beta']\) 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 II be a tight ELP and \(T = (T, \chi)\) be a TD of \(\mathcal{G}_{II}\). Then, the treewidth of \(\mathcal{G}_{F_{k}[II, T]}\) is in \(\mathcal{O}(\text{width}(T))\).
Proof. We construct a TD \( T' = (T, \chi') \), where we define \( \chi' \) as follows. For every node \( t \) of \( T \), let \( \chi'(t) := \chi(t) \cup k(\chi(t)) \cup \{ \text{usat} \leq t, \text{usat} \leq \text{root}(T) \} \cup \{ \text{usat} \leq t' \mid t' \in \text{chldr}(t) \} \). Observe that \( T' \) is well-defined and that since \( |\text{chldr}(t)| \leq 2 \), we have \( \text{width}(T') \in O(\text{width}(T)) \).

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 \( F_p[I] \). 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 \( K \) is tight. Towards defining \( F_{p}[I][T] \), we use auxiliary variables of the form \( I_t \) for every node \( t \) of \( T \) and every interpretation \( I \subseteq 2^{(t)} \) restricted to \( \chi(t) \), which we address by \( I_t := \{ I_t \mid t \in T, I \subseteq 2^{(t)} \} \). Observe that therefore the number of variables in \( I_t \) 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 \( F_{p}[I][T] := \exists I_t. \mathcal{F}_s[I, T] \), where \( \mathcal{F}_s[I, T] \) is defined by means of Formulas (13)–(17) below.

\[
\begin{align*}
\neg k_a & \iff \text{for every } t \in T, I \in 2^{(t)}, I \notin k(\Pi_t) \quad (13) \\
\bigvee_{k_a \in k(B_c)} \neg k_a & \iff \text{for every } t \in T, I \in 2^{(t)}, I \notin k(\Pi_t) \quad (14) \\
\bigvee_{I_t'} I_t' & \iff \text{for every } t, t' \in \text{chldr}(t), I \in 2^{(t)}, I \cap \chi(t') = I_t' \cap \chi(t') \quad (15) \\
\bigvee_{I_t} & \text{ for every } t, t' \in \text{chldr}(t), I \in 2^{(t')}, I_t' = I_t' \cap \chi(t') \quad (16) \\
\bigvee_{I_t \cap k_a} & \text{ for every } t, a \in \chi(t) \quad (17)
\end{align*}
\]

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 \( I_t \), 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 \( \Pi_t \) 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 \( \Pi \) be a tight ELP and \( T = (T, \chi) \) be a TD of \( G_{\Pi} \). Then, given any assignment \( \alpha : K \rightarrow \{0, 1\} \) we have that \( F_{p}[I][\alpha] \) is valid iff \( F_{p}[I, T][\alpha] \) is valid.

Proof (Sketch). \( \rightarrow \): Assume that \( F_{p}[I, T][\alpha] \) 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 \( \beta_a' \) such that \( F_{p}[\alpha][\beta_a'] \) evaluates to true. For every node \( t \) and \( a \in A(\Pi), \) we set \( \beta_a'(I_t) := 1 \), whenever \( I \in \chi(t) \cap \beta_a^{-1}(1) \) and \( \beta_a'(I_t) := 0 \) otherwise. Then, by construction of \( \beta_a' \), we have that \( F_{p}[I, T][\alpha][\beta_a'] \) evaluates to true. \( \leftarrow \): The other direction works as follows. We assume an assignment \( \beta_a' \) such that \( F_{p}[I, T][\alpha][\beta_a'] \) evaluates to true and for every \( k_a \in K \) with \( \alpha(k_a) = 0 \), we construct an assignment \( \beta_a \) such that \( \text{SM}[\Pi][\alpha][\beta_a] = \text{COMP}[\Pi][\alpha][\beta_a] \) evaluates to true. To this end, we construct an assignment \( I_\alpha \), consisting of the union over one assignment \( I_t \) for every node \( t \) of \( T \) such that \( \beta_a'(I_t) = 1 \) and \( I \cap \chi(t') \cap I_t \cap \chi(t) = \emptyset \) for every child node \( t' \in \text{chldr}(t) \). Since \( F_{p}[I, T][\alpha][\beta_a'] \) 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.

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

Lemma 13 (TW-Awareness (single exponential)). Let \( \Pi \) be a tight ELP and \( T = (T, \chi) \) be a TD of \( G_{\Pi} \). Then, the treewidth of \( G_{F_{p}[I, T]} \) is in \( 2^{O(\text{width}(T))} \).

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

5.3 Merging Consistency, Knowledge & Possibility

Overall, we obtain the following runtime result.

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

Proof. The runtime is due to the \( 2^w \) many different variables of the form \( I_t \) in the construction of formula \( F_{p}[I, T] \).

The formula \( F[I, T] \) 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 \( \Pi \) be any tight program, whose treewidth of primal graph \( G_{\Pi} \) is \( w \). Then, WV existence for \( \Pi \) can be decided in time \( 2^{O(w)} \cdot \text{poly}(|A(\Pi)|) \).

Proof. We construct both a TD \( T \) of \( G_{\Pi} \) of width \( 5 \cdot w \) [Bodlaender et al., 2016] as well as the formula \( F[I, T] \), in time \( 2^{O(w)} \cdot \text{poly}(|A(\Pi)|) \). The constructed formula is an instance of 2-QSAT can be solved in the desired runtime.
Then, unless ETH fails, WV existence for II cannot be decided in time $2^{2^{O(w)}} \cdot \text{poly}(|A(II)|)$. See Proposition 1, since by Lemma 11 the treewidth over the inner-most (universally) quantified variables is linear in $w$. □

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 II be any tight program, whose treewidth of the primal graph $G_{II}$ is $w$. Then, unless ETH fails, WV existence for II cannot be decided in time $2^{2^{O(w)}} \cdot \text{poly}(|A(II)|)$.

Proof. Let $\exists X, Y, \varphi$ be any QBF with $\varphi$ being in DNF and $\text{var}(\varphi) = X \cup Y$. From this we construct an ELP II over the set $\{\text{sat}, x, \bar{x} \mid x \in \text{var}(\varphi)\}$ of atoms as follows. For every variable $x \in X$, we construct the rules $x \leftarrow K x, x \leftarrow K \bar{x}, \bar{x} \leftarrow K x,$ and $\bar{x} \leftarrow K \bar{x}$. For every $y \in Y$, we construct rules $y \leftarrow \neg \bar{y}$ and $\bar{y} \leftarrow \neg y$. Further, we build the rule $\bot \leftarrow \neg K \text{sat}$, which can be easily transformed to our narrow syntax using $K$ 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: $\text{sat} \leftarrow l_1, \ldots, l_i$, where for a literal $l$ we let $\hat{l} = \bar{l}$ if $l = \neg a$ and $\hat{l} := l$ otherwise. Observe that II 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 $\varphi$ and the views of II. Let $I \in 2^X$ be any assignment with $Q[I]$ being valid. Then, we construct a WV $\tilde{W}$ by defining $\tilde{W} := \{\text{sat} \mid I(x) = 0\} \cup \{x \mid I(x) = 1\} \cup \{J \mid J \in 2^Y\}$. Assume towards a contradiction that $W$ is not a WV of II. But then, $\tilde{W}$ does not coincide with set $W$ of stable models of $\Pi^W$. Take any $J \in W$. Observe that by construction of II and since $Q[I]$ is valid, we have that $J$ is also an answer set of $\Pi^W$. Further, $W' \subseteq W$ since $W$ contains every subset over variables $Y$, which yields $W = W'$.

Take any WV $W$ of II. Then, by construction of II, we have that $\text{sat} \in W'$ for every $W' \in W$ and consequently, for any stable model $N$ of $\Pi^W, \text{sat} \in N$. From this we construct an assignment $I : X \rightarrow \{0, 1\}$ 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 $x \not\in W'$. Assume towards a contradiction that $Q[I]$ is invalid. Then, there is an assignment $J : Y \rightarrow \{0, 1\}$ with $Q[I][J] = \emptyset$. As a consequence, we can define a stable model $N'$ of $\Pi^W$ with $N' := (W' \setminus \{\text{sat}\}) \cup \{y \mid J(y) = 1\} \cup \{\bar{y} \mid J(y) = 0\}$. 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 $G_{II}$. From this we construct a TD $\mathcal{T}' = (T, \chi')$ of $G_{\Pi}$ as follows. For every node $t$ of $T$, we define $\chi'(t) := \chi(t) \cup \{\text{sat}\} \cup \{\bar{x} \mid x \in \chi(t)\}$. Observe that indeed $\mathcal{T}'$ is a well-defined TD of $G_{\Pi}$. Obviously, we have that $|\chi'(t)| \leq 2|\chi(t)| + 1$ and therefore we follow that $\text{tw}(\Pi) \in \mathcal{O}(\text{tw}(Q))$. Then, assuming WV existence on $\Pi$ can be decided in time $2^{2^{O(w)}} \cdot \text{poly}(|A(II)|)$ with $w = \text{tw}(G_{II})$ contradicts Proposition 2. □

**5.4 Normal (HCF) and $\iota$-Tight Programs**

For HCF programs II, we first construct a tight program $\Pi'$, using a treewidth-aware translation $\mathcal{R}$ from $k(\Pi)$ to $\Pi' = R(k(\Pi))$ [Fandinno and Hecher, 2021]. Then, the treewidth of $G_{\Pi'}$ is in $\mathcal{O}(\text{tw}(G_{II}) \cdot \log(\text{tw}(G_{II})))$, 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 $\Pi'$ and obtaining an adapted TD $T'$ of $G_{\Pi'}$, we apply $\mathcal{F}(T', T)$, as defined in the previous subsections, but where $K = k(A(II))$.

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

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. □

**Theorem 18 ( Tight Lower Bound for normal ELPs, $\iota^3$).** Let II be any normal program, whose treewidth of the primal graph $G_{II}$ is $w$. Then, unless ETH fails, WV existence for II cannot be decided in time $2^{2^{O(w \cdot \log(w))}} \cdot \text{poly}(|A(II)|)$.

For $\iota$-tight programs II, the construction works analogously, but instead of $\mathcal{R}$, we use an adapted reduction $\mathcal{R}'$ [Hecher, 2023] to construct a tight program $\Pi'$. There, the treewidth of $G_{\Pi'}$ is in $\mathcal{O}(\text{tw}(G_{II}) \cdot \log(\log(w)))$. 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 $K a$ by conjunction $a \land K a$.

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, $a \leftarrow K a$ is objectively tight, but its corresponding program $a \leftarrow a \land K a$ is not. Conversely, an ELP under G94 semantics can be translated into a new program under the K15 semantics where expressions $K a$ are replaced by $not.K.not.K.a$ and the rules $not.K.not.K.a \leftarrow not.K.not.K.a \land not.K.a \leftarrow K a, not.K.not.K.a \leftarrow K a \land not.K.a, not.K.a \leftarrow K a$ with $not.K.not.K.a, not.K.not.K.a, not.K.a$ 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.

3Proofs of statements marked with “$\ast$” are given in the appendix.
References


