Query Answering in the Description Logic

Horn-$SHIQ^*$

Thomas Eiter$^1$, Georg Gottlob$^2$, Magdalena Ortiz$^1$, and Mantas Šimkus$^1$

$^1$ Institute of Information Systems, TU Vienna, Austria
(eiter|ortiz|simkus)@kr.tuwien.ac.at
$^2$ Computing Laboratory, Oxford University, United Kingdom
georg.gottlob@comlab.ox.ac.uk

Abstract. We provide an ExpTime algorithm for answering conjunctive queries (CQs) in Horn-$SHIQ$, a Horn fragment of the well-known Description Logic $SHIQ$ underlying the OWL-Lite standard. The algorithm employs a domino system for model representation, which is constructed via a worst-case optimal tableau algorithm for Horn-$SHIQ$; the queries are answered by reasoning over the domino system. Our algorithm not only shows that CQ answering in Horn-$SHIQ$ is not harder than satisfiability testing, but also that it is polynomial in data complexity, making Horn-$SHIQ$ an attractive expressive Description Logic.

1 Introduction

Driven by the development of semantically enhanced systems, as in the context of the Semantic Web and of Enterprise Application Integration, query answering in Description Logics (DLs) has emerged as an important topic. A variety of algorithms have been proposed for this problem and, aiming at different applications, aspects like combined and data complexity have been guiding their development. The former characterizes the cost of query answering in the general case, while the latter in the case when the query and the knowledge base except the factual part are fixed. Data complexity is especially important for applications in which DLs are used to formalize rich data models for data repositories, as in such context the model is static as compared to the data contents, and typical user queries are known. For querying DLs, conjunctive queries (CQs) have been most widely considered and three major settings have been addressed:

- very expressive DLs for which standard reasoning tasks, like satisfiability testing or instance checking, are intractable both in data and combined complexity. As query answering is at least as hard, the problem is trivially intractable in general. For example, CQs in $SHIQ$ have 2ExpTime-complete combined complexity [13] and coNP-complete data complexity [6].
- tailored DLs, like DL-Lite [5], which aim at lower complexity at the price of limited expressiveness. In DL-Lite, CQ answering is coNP-complete in combined complexity, but polynomial if the query is fixed, and has very low data complexity (reducible to FOL, thus inside logarithmic space).

* This work was partially supported by the Austrian Science Fund (FWF) grant P20840, Wolfgang Pauli Institute (WPI), and the Mexican National Council for Science and Technology (CONACYT) grant 187697.
• weak DLs, like $\mathcal{EL}$ [1], for which standard reasoning and CQ answering under data complexity are P-complete, while CQ answering under combined complexity is intractable [17]. Several extensions of $\mathcal{EL}$ sharing this property (e.g., $\mathcal{ELH}$, $\mathcal{ELL}^I$, $\mathcal{EL}^+$, $\mathcal{EL}^{++}$) can be found in [17, 12, 10].

Since CQ answering is intractable under combined complexity already over very simple knowledge bases, fragments of expressive DLs with tractable data complexity are of particular interest. Ideally, such fragments should also allow for CQ answering with combined complexity not higher than of standard reasoning.

In this paper, we identify Horn-$\mathcal{SHIQ}$ as a DL with this property. It was introduced in [9] as a Horn fragment of $\mathcal{SHIQ}$, in which the syntax is restricted in a way that disjunction is not expressible. While standard reasoning in Horn-$\mathcal{SHIQ}$ is ExpTime-complete in general [11], it is polynomial if the taxonomy is fixed [9].

Our main contributions and results are briefly summarized as follows:

• We provide an ExpTime algorithm for answering CQs in Horn-$\mathcal{SHIQ}$. The algorithm is based on answering tree-shaped queries over particular trees (that capture a universal model of the KB) finitely represented by domino systems. The latter are relatives of saturated mosaic sets known in other branches of logic, and the recent knot sets [16] and domino sets [18] in DLs.
• For constructing domino systems, we exploit a dedicated tableaux-based algorithm for consistency checking in Horn-$\mathcal{SHIQ}$, which is of independent interest. It adapts the standard $\mathcal{SHIQ}$ tableaux [8] (using, e.g., anywhere blocking [15] and a kind of lazy unfolding [7]) to terminate in deterministic single exponential time, yielding a representation of a universal model of $\mathcal{K}$ such that each CQ over $\mathcal{K}$ can be answered on it. This may also be exploited to precompile $\mathcal{K}$ into a (query-independent) domino system for on-line query answering.
• Based on our algorithm, we show that CQs in Horn-$\mathcal{SHIQ}$ have ExpTime-complete combined complexity and P-complete data complexity. We also present a fragment of Horn-$\mathcal{SHIQ}$ for which CQs are easier. In Horn-$\mathcal{SHIQ}^-$, which forbids inverse roles and existential projection on the left hand side of containment axioms, the combined complexity of CQs is lowered to PSpace-completeness.

As Horn-$\mathcal{SHIQ}$ is an expressive fragment of OWL-Lite, our results are relevant for the Semantic Web context. They extend in fact from CQs to the class of positive (existential) queries. Our result on the combined complexity of CQs in Horn-$\mathcal{SHIQ}$ is of particular interest. Firstly, it reveals another expressive DL for which CQ answering is not harder than standard reasoning (cf. [16, 13, 14]). Secondly, it suggests that the exponential jump in combined complexity of CQs by adding inverse roles to $\mathcal{ALC}$, which was found by Lutz [13], relies on their interaction with disjunction. In other words, if disjunction is eliminated, then inverse roles do not make CQ answering harder.

2 Preliminaries

The Description Logics $\mathcal{SHIQ}$ and Horn-$\mathcal{SHIQ}$ We assume countably infinite sets $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{I}$ of concept names, role names, and individuals respectively, where $\mathbf{C}$ contains special concepts names $\top$ and $\bot$. Roles are expressions
$R$ and $R^{-}$, where $R \in \mathbb{R}$, and their inverses are $\text{Inv}(R) = R^{-}$ and $\text{Inv}(R^{-}) = R$, respectively. For any roles $R$ and $S$, $R \subseteq S$ is a role inclusion axiom (RI), and $\text{Trans}(R)$ is a transitivity axiom (TA). For any set $\mathcal{R}$ of RIs and TAs, $\mathcal{R}^*$ denotes the reflexive transitive closure of $\{(R, S) \mid R \subseteq S \in \mathcal{R} \text{ or } \text{Inv}(R) \subseteq \text{Inv}(S) \in \mathcal{R}\}$; we write $\text{Trans}_R(R')$ if $R' \subseteq_R R$ and $R \subseteq_R R'$ for some $R$ s.t. $\text{Trans}(R) \in \mathcal{R}$ or $\text{Trans}(R^{-}) \in \mathcal{R}$. A role $R$ is simple w.r.t. $\mathcal{R}$, if there is no $S \subseteq_R R$ with $\text{Trans}_R(S)$.

Concepts are inductively defined as follows: (a) each $A \in \mathcal{C}$ is a concept, and (b) if $C, D$ are concepts, $R$ is a role, and $S$ is a simple role, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, and $\leq n S.C$ and $\geq n S.C$, for $n \geq 1$, are concepts.

An expression $C \sqsubseteq D$, where $C, D$ are concepts, is a general concept inclusion axiom (GCI), and expressions $a \sqsubseteq A$ and $(a, b) : R$, where $A \in \mathcal{C}$, and $a$ and $b$ are individuals, and $R$ is a role, are concept and role assertions, respectively. A $\text{SHIQ}$ knowledge base (KB) is a tuple $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where the TBox $\mathcal{T}$ is a finite set of GCIs, the RBox $\mathcal{R}$ is a finite set of RIs and TAs, and the ABox $\mathcal{A}$ is a finite nonempty set of assertions. We denote by $\mathcal{C}(\mathcal{K})$, $\mathcal{R}(\mathcal{K})$ and $\mathcal{I}(\mathcal{K})$ the sets of concept names, role names, and individuals occurring in $\mathcal{K}$.

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \mathcal{I}^\mathcal{T})$ for a KB $\mathcal{K}$ consists of a nonempty domain $\Delta^\mathcal{I}$ and a valuation function $\mathcal{I}^\mathcal{T}$ that maps each individual $c \in \mathcal{I}(\mathcal{K})$ to an element $c^\mathcal{I} \in \Delta^\mathcal{I}$, each concept name $C \in \mathcal{C}(\mathcal{K})$ to a subset $C^\mathcal{I}$ of $\Delta^\mathcal{I}$, and each role name $R \in \mathcal{R}(\mathcal{K})$ to a subset $R^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$, in such a way that $\top = \Delta^\mathcal{I}$ and $\bot = \emptyset$. The function $\mathcal{I}^\mathcal{T}$ is extended to all concepts and roles in the standard way (see, e.g., [11]), and satisfaction of $\mathcal{K}$ by $\mathcal{I}$ ($\mathcal{I} \models \mathcal{K}$), i.e. modelhood, is also standard.

The DL Horn-$\text{SHIQ}$ was introduced [9] as a fragment of $\text{SHIQ}$. The main idea is to restrict the syntax in a way that $\sqcup$ is not expressible, establishing a correspondence to a Horn fragment of first-order logic with equality. Without loss of generality, we focus here on a normal form of Horn-$\text{SHIQ}$ in [11], to which each Horn-$\text{SHIQ}$ KB can be efficiently rewritten while preserving the answers to arbitrary CQs (as follows from [11]).

**Definition 1.** (Normal) Horn-$\text{SHIQ}$ KBs contain only GCIs of the forms
\[
A \sqcap B \sqsubseteq C \quad A \sqsubseteq \forall R.B \quad A \sqsubseteq \exists m S.B
\]
\[
\exists R.A \sqsubseteq B \quad A \sqsubseteq \exists R.B \quad A \sqsubseteq \ldots S.B
\]
where $A, B, C$ are concept names, $R$ is a role, $S$ is a simple role, and $m \geq 1$.

**Example 1.** Assume two Horn-$\text{SHIQ}$ KBs $\mathcal{K}_1 = (\mathcal{T}, \emptyset, \mathcal{A})$ and $\mathcal{K}_2 = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where $\mathcal{T} = \{A \sqsubseteq \forall R.A, B \sqsubseteq \exists P.C\}$, $\mathcal{R} = \{P \sqsubseteq R\}$, and $\mathcal{A} = \{a : A, a : B\}$. Note that both are consistent, however adding $A \sqsubseteq \forall R.B \to T$ makes them inconsistent.

**Conjunctive Queries** Let $\mathbf{V}$ be a countably infinite set of variables. A (Boolean) conjunctive query (CQ, or query) over a KB $\mathcal{K}$ is a finite set $q$ of atoms of the form $A(x)$ or $R(x, y)$, where $A$ is a concept name, $R$ is a role and $x, y \in \mathbf{V}$. By $\mathbf{V}(q)$ we denote the variables occurring in the atoms of $q$. The query graph of $q$ is the directed graph $G^q$ over nodes $\mathbf{V}(q)$ with an edge between nodes $x$ and $y$ iff $R(x, y) \in q$ for some role $R$. The query $q$ is tree-shaped if $G^q$ is a tree.

---

3 W.l.o.g., no individuals occur in $q$; we can replace each $a$ in $q$ by a new variable $y$, add $C_a(y) \to q$ and $a : C_a$ to $\mathcal{A}$, where $C_a$ is a new concept name. Non-Boolean queries (i.e., with answer variables) can be reduced to Boolean queries as usual.
A match for \( q \) in an interpretation \( \mathcal{I} \) is a mapping \( \theta : \mathbb{V}(q) \to \Delta^\mathcal{I} \) s.t. (i) \( \theta(x) \in A^\mathcal{I} \) for each \( A(x) \in q \), and (ii) \( (\theta(x), \theta(y)) \in R^\mathcal{I} \) for each \( R(x,y) \in q \). We say that \( \mathcal{I} \) satisfies \( q(\mathcal{I} \models q) \), if \( q \) has a match in \( \mathcal{I} \), and that \( \mathcal{K} \) entails \( q(\mathcal{K} \models q) \), if \( q \) has a match in each model \( \mathcal{I} \) of \( \mathcal{K} \).

Example 2. Assume the queries \( tq_1 = \{ A(x), R(x,y), A(y), R(x,z), C(z) \} \) and \( tq_2 = \{ B(x), R(x,y), A(y), P(x,z), C(z) \} \). As easily seen, \( \mathcal{K}_1 \models tq_1 \) and \( \mathcal{K}_1 \models tq_2 \), while \( \mathcal{K}_2 \models tq_1 \) and \( \mathcal{K}_2 \models tq_2 \). Note that both queries are tree-shaped.

3 Conjunctive Queries Over Domino Trees

This section describes an algorithm for answering CQs over trees induced by domino systems, which is exploited in the next sections for deciding CQ entailment in Horn-SHQC KBs. A domino system finitely represents a possibly infinite tree-shaped interpretation that can be built by connecting matching dominoes.

Definition 2. A domino is a tuple \( \langle c, r, c' \rangle \) where \( c, c' \) are sets of concepts and \( r \) is a set of roles (w.r.t. an underlying alphabet). A domino system is a tuple \( \langle D, \triangleright, R \rangle \), where \( D \) is a set of dominoes, \( \triangleright \subseteq D \times D \) is a direct successor relation with \( c'_1 = c_2 \) whenever \( \langle c_1, r_1, c'_1 \rangle \triangleright \langle c_2, r_2, c'_2 \rangle \), and \( R \) is an RBox. We also require that for each \( \langle c, r, c' \rangle \in D \), the set \( r \) is closed under role inclusions in \( R \), i.e., \( r \in r \) and \( R \subseteq R' \subseteq R \) imply \( r \in R' \). Furthermore, \( D \) contains one designated initial domino of the form \( \langle \emptyset, \emptyset, c' \rangle \).

Following the terminology in [14], we define next the tree-shaped interpretation induced by a domino system. Its domain is represented by a prefix-closed set of words; for a word \( w = e_1 \cdots e_n \), let \( \langle w | e_{n+1} \rangle \) denote the word \( e_1 \cdots e_n e_{n+1} \).

Definition 3. The tree base of a domino system \( \mathcal{D} = \langle D, \triangleright, R \rangle \) is the interpretation \( \mathcal{I} = \langle \Delta^\mathcal{I}, \mathcal{T} \rangle \) (w.r.t. the alphabet underlying \( \mathcal{D} \)) defined as follows:

1. The domain \( \Delta^\mathcal{I} \) is the smallest set of words over dominoes such that:
   - if \( t \in D \) is the initial domino, then \( t \in \Delta^\mathcal{I} \);
   - if \( t_1 \cdots t_n \in \Delta^\mathcal{I} \) and \( t_n \triangleright t_{n+1} \), then \( t_1 \cdots t_n \cdot t_{n+1} \in \Delta^\mathcal{I} \).

2. The valuation function \( \mathcal{T} \) is defined as follows:
   - For each atomic concept \( A \), \( A^\mathcal{I} = \{ \langle s | t \rangle \in \Delta^\mathcal{I} \mid t = \langle c, r, c' \rangle \land A \in c' \} \).
   - For each role name \( R \),
     \[
     R^\mathcal{I} = \{ \langle s, \langle s | t \rangle \rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid t = \langle c, r, c' \rangle \land R \in r \} \cup \\
     \{ \langle \langle s | t \rangle, s \rangle \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid t = \langle c, r, c' \rangle \land \text{Inv}(R) \in r \}. 
     \]

The domino tree \( \mathcal{T}_D = \langle \Delta^\mathcal{I}, \mathcal{T} \rangle \) of \( \mathcal{D} \) is the interpretation identical to \( \mathcal{I} \) except that, for each role \( R \), we have \( R^\mathcal{I} = R^\mathcal{T} = R^\mathcal{T} \cup \bigcup_{\mathcal{S} \in \mathcal{T}} R \cap \text{Trans}_{\mathcal{S}}(S^\mathcal{T})^+ \).

Query entailment in a domino system is naturally defined via the existence of matches in the represented domino tree. We first provide a procedure to verify the existence of special ordered matches for tree-shaped queries, and we then extend the result to all CQs via the standard method of query treification.

Definition 4. A domino system \( \mathcal{D} \) entails a CQ \( q(\mathcal{D} \models q) \), if there is a match for \( q \) in \( \mathcal{T}_D \). A match \( \pi \) for a tree-shaped CQ \( tq \) in \( \mathcal{T}_D \) is ordered if, for each \( x,y \in \mathbb{V}(tq) \), \( \pi(x) \) is a proper prefix of \( \pi(y) \) whenever \( R(x,y) \in tq \) for some \( R \). We write \( \mathcal{D} \models \_o \) \( tq \), if there is some ordered match for \( tq \) in \( \mathcal{T}_D \).
function checkRoleSucc(D = ⟨D, ∗, R⟩; dominoes t1 = ⟨c1, r1, c′1⟩, t3 = ⟨c3, r3, c′3⟩ from D; role set r ≠ ∅)
output: true if t3 is an r-successor of t1
if t1 has no direct successor in D, then return false;
Initialize t with some direct successor ⟨c2, r2, c′2⟩ of t1;
s := r2; i := 0
repeat
  if t = t3 and r ⊆ s then return true;
  if r ⊈ s or t has no direct successor in D, then return false;
  Reassign t to some direct successor ⟨c2, r2, c′2⟩ of t;
  Set s′ to the smallest set closed under the following rule:
  if S ⊆ R, S ∈ s, S ∈ r2, and TransR(S), then R ∈ s′;
s := s′
until i > |D|; return false

Fig. 1. Verifying Successor Dominoes.

Let tq be a fixed tree-shaped query and D = ⟨D, ∗, R⟩ a domino system with tree TD. To obtain a procedure for deciding D |=o tq, we provide some necessary and sufficient conditions for the existence of ordered matches that can be verified without building TD explicitly. Roughly, we search for an association of dominoes from D with the variables of tq. As the association must witness an ordered match, the domino t2 associated with variable x must encode the concept names needed to satisfy each unary atom A(x) ∈ tq, while for each role atom R(x, y) ∈ tq, the domino t4 must ‘reach’ the domino t3 via an R-path.

Definition 5. For two dominoes t1 = ⟨c1, r1, c′1⟩ and t3 = ⟨c3, r3, c′3⟩ in D and a set of roles r ≠ ∅, we say t3 is a r-successor of t1 if one of the following holds:
(a) t1 ⊳ t3 and r ⊆ r3, or
(b) for some role set r′, D contains an r′-successor t2 of t1 such that t2 ⊳ t3 and for each R ∈ r there exists S ∈ r′ with TransR(S), S ⊆ R and S ∈ r3.

We are ready to define domino associations, which characterize the |=o relation.

Definition 6. A domino association for tq is a mapping µ that assigns to each z ∈ V(tq) a domino µ(z) ∈ D in a way such that, for each pair x, y ∈ V(tq):
(a) if A(x) ∈ tq, then A ∈ c′, where µ(x) = ⟨c, r, c′⟩; and
(b) if r = {R | R(x, y) ∈ tq} is not empty, then µ(y) is an r-successor of µ(x).

Example 3. Consider the domino system D = ⟨D, ∗, R⟩ in Figure 2, where the black arrows show the dominoes of D and the dashed arrows the ⊳ relation; R = {TransR} and the initial domino is t1. A possible domino association for tq3 in Example 2 is the mapping µ1 with µ1(t1) = t2, µ1(y) = t2, µ1(z) = t4. Note that t2 and t4 are {R}-successors of t2 as R is transitive, and that tq3 has no domino association for R = ∅. The query tq2 in Example 2 has a domino association even in this case, witnessed by µ2 with µ2(t1) = t3, µ2(z) = t2, and µ2(z) = t4.

The following is immediate from the definition of |=o and Definition 6.
Theorem 1. \( \mathcal{D} \models_{\omega} tq \) iff there exists a domino association for \( tq \).

By Theorem 1, we can decide \( \mathcal{D} \models_{\omega} tq \) by deciding existence of a domino association. We exploit for the latter the procedure \textsf{checkRoleSucc} in Figure 1, which nondeterministically checks whether a domino \( t_2 \) is an \( r \)-successor of a domino \( t_1 \).

Proposition 1. Let \( t_1, t_2 \) be dominos of \( \mathcal{D} \), and \( r \) a role set. Then \( t_2 \) is an \( r \)-successor of \( t_1 \) iff some run of \textsf{checkRoleSucc}(\( D \), \( t_1 \), \( t_2 \), \( r \)) returns true.

Now the following simple procedure \textsf{assocDominoes}(\( D \), \( tq \)) non-deterministically decides the existence of a domino association for \( tq \) w.r.t. \( \mathcal{D} \): (1) guess a mapping \( \mu \) from \( V(tq) \) to dominos of \( \mathcal{D} \), and (2) check satisfaction of the conditions (a) and (b) in Definition 6; to check (b), call \textsf{checkRoleSucc} for each arc in \( tq \).

Theorem 2. \( \mathcal{D} \models_{\omega} tq \) iff some run of \textsf{assocDominoes}(\( D \), \( tq \)) returns true.

Having a procedure to decide \( \mathcal{D} \models_{\omega} tq \) for tree-shaped queries \( tq \), we now settle deciding \( \mathcal{D} \models q \) for arbitrary CQs \( q \). Following [4, 6, 14], we consider query treecifications, i.e., tree-shaped queries whose matches induce matches for \( q \).

Definition 7. For every CQ \( q \), let \( q^R \) be the smallest query such that: (a) \( q \subseteq q^R \), (b) \( R(x, y) \in q^R \) and \( R \subseteq P \in \mathcal{R} \) implies \( P(x, y) \in q^R \), (c) \( R(x, y) \in q^R \), \( R(y, z) \in q^R \) and \( \text{Trans}_R(R) \) imply \( R(x, z) \in q^R \), and (d) \( R(x, y) \in q^R \) implies \( \text{Inv}_R(R)(y, x) \in q^R \).

A treecification of \( q \) is a tree-shaped query \( q' \) such that \( |q'| \leq 2|q| \) and there exists a mapping \( \theta \) from \( V(q) \) to \( V(q') \) fulfilling:
a) \( A(x) \in q \) implies \( A(\theta(x)) \in q' \), and
b) \( R(x, y) \in q \) implies \( R(\theta(x), \theta(y)) \in (q')^R \).

As easily shown, each match for a treecification \( q' \) of \( q \) in \( T_\mathcal{D} \) is also a match for \( q \). On the other hand, from each match for \( q \) in \( T_\mathcal{D} \), we can obtain a treecification \( q' \) that is mappable into \( T_\mathcal{D} \) via an ordered match.\(^5\) Hence, we obtain:

Theorem 3. For any CQ \( q \), \( \mathcal{D} \models q \) iff \( \mathcal{D} \models_{\omega} q' \), for some treecification \( q' \) of \( q \).

It follows that we can decide \( \mathcal{D} \models q \) by listing all treecifications of \( q \) and posing them separately over \( \mathcal{D} \). Note that there are finitely many treecifications of \( q \).

4 A Tableaux Algorithm for \textbf{Horn-SHIQ}

In order to exploit the results of Section 3 for query answering in \textbf{Horn-SHIQ}, we first provide a tableaux algorithm for KB satisfiability. Like the standard \textbf{SHIQ} tableau [8], it uses a completion forest to represent a model; in the next section we extract from it a domino system that can be used for query answering.

In what follows, \( \mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A}) \) is a \textbf{Horn-SHIQ} KB. We use \( A, B, C \) to denote concepts names; \( D, E \) to denote (arbitrary) concepts; \( R, R' \) to denote a role; \( S \) a simple role; and \( a, b \) to denote individuals.

\(^4\) Unlike [4, 6, 14], we do not use treecifications to reduce CQ entailment to concept satisfiability, as this would require the use of role conjunction and the decidability of this extension of \textbf{Horn-SHIQ} in \textsc{ExpTime} is not apparent.

\(^5\) As implicit in [6], such a \( q' \) with \( |q'| \leq 2|q| \) exists: to obtain a treecification from a match, one replaces each atom \( R(x, y) \in q \) with a pair of atoms in case \( x, y \) are mapped (i) to the same node, or (ii) to nodes in different branches of the domino tree.
Most of the following definitions are based on [8], while 9 and 14 follow [15] and are related to anywhere blocking. Definition 8 is simplified since only normalized KBs are considered, and the ≈ relation from [8] is omitted in Definition 10.6

**Definition 8.** (concept closure) We define $\text{Cl}(K)$ as the smallest set of concepts closed under subconcepts such that (i) $D, E \in \text{Cl}(K)$ for every $D \subseteq E \in T$; and (ii) if $\forall R. A \in \text{Cl}(K)$, then $\forall R'. A \in \text{Cl}(K)$.

**Definition 9.** (named/unnamed) nodes We assume a countably infinite set $N$ of nodes and a strict total order $\prec$ on $N$. Each $a \in I(K)$ is associated with one fixed node $n_a \in N$; the nodes $n_a$ are named, all other nodes are unnamed.

**Definition 10.** (completion forest) A completion forest for a KB $K$ is a tuple $F = (N, E, L, \not\approx)$ where $N \subseteq N$ and $E \subseteq N \times N$ define a directed graph; $L$ is a labeling function assigning each $n \in N$ a subset of $\text{Cl}(K)$ and each pair $u, u' \in N \times N$ to a set of roles (over $K$), in such a way that $\mathcal{L}(u, u') = \emptyset$ for all $(u, u') \not\in E$; and $\not\approx \subseteq N \times N$ is a binary relation, tacitly assumed to be symmetric.

**Definition 11.** (successor, neighbor) For a completion forest $F = (N, E, L, \not\approx)$ and a pair $u, u' \in N$, $u'$ is a successor of $u$ if $(u, u') \in E$. The inverse of successor is called predecessor; the transitive closures of successor and predecessor are ancestor and descendant respectively. For all $R$, $u'$ is an $R$-successor of $u$ if $R' \subseteq L(u, u')$ for some $R'$ with $R' \subseteq R$. We call $u'$ an $R$-neighbor of $u$, if $u'$ is an $R$-successor of $u$, or if $u$ is an $\text{Inv}(R)$-successor of $u'$.

**Definition 12.** (clash-free completion forest) A completion forest $F$ contains a clash, if (i) for some $u \in N$, $\bot \in \mathcal{L}(u)$; or (ii) for some $u \in N$ with $\leq n.S.C \in \mathcal{L}(u)$, $u$ has $n+1$ $S$-neighbors $w_0, \ldots, w_n$ such that, for all $0 \leq i < j \leq n$, $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j \in F$. If $F$ contains no clash, then $F$ is clash-free.

**Definition 13.** (initial completion forest) The initial completion forest $F_A$ for $K$ has the named node $n_a$ labeled with $\mathcal{L}(n_a) = \{A \in \text{Cl}(K) \mid a : A \in A\}$ for each individual $a \in I(K)$, and an edge $(n_a, n_b) \in E$ labeled $\mathcal{L}(n_a, n_b) = \{R \mid (a, b) : R \in A\}$ for each pair $a, b \in I(K)$; the relation $\not\approx$ is empty.

**Definition 14.** (blocking) For a completion forest $F = (N, E, L, \not\approx)$, a node $u \in N$ is blocked if $u$ is unnamed and $u$ is either directly or indirectly blocked; $u$ is indirectly blocked if one of its ancestors is blocked; $u$ is directly blocked if none of its ancestors is blocked and there is some $u' \prec u$ such that $u, u'$ are unnamed nodes, $\mathcal{L}(u) = \mathcal{L}(u')$, $\mathcal{L}(v) = \mathcal{L}(v')$, and $\mathcal{L}(v, u) = \mathcal{L}(v', u')$, where $v$ and $v'$ are the predecessors of $u$ and $u'$ respectively.

The expansion rules are given in Figure 3, where a node $u \in N$ is new in $F$ if $u' \prec u$ for every $u' \in N$. The $\leq$-rule calls the operation $\text{merge}(u, N)$ described in Figure 4. The rules are similar to those in [8], except for the first three, which (lazily) ensure the satisfaction of the TBox axioms. Also, the restricted form of at-most number restrictions allows us to have just one $\leq$-rule and a deterministic $\text{merge}(u, N)$ that simultaneously merges all nodes in $N$ into one.7

---

6 It is irrelevant for query answering, but could be emulated e.g. using node labels $C_a$.

7 Note that the TBox internalization of [8] is not adequate for Horn-SHIQ, and that the other rules of [8] are not necessary due to the normal form of the KB.
The initial $F_A$ is expanded by exhaustively applying the rules in Figure 3. The expansion stops, if a clash is reached; otherwise, it continues until the forest is complete, i.e., no rule is applicable. It can be shown similarly as in [8, 15] that this algorithm is a decision procedure for KB satisfiability in Horn-SHIQ.

**Theorem 4.** Let $K$ be a Horn-SHIQ KB. Then $K$ is satisfiable if and only if a complete and clash-free completion forest for $K$ can be obtained.

Note that after applying any rule from Figure 3, the resulting forest is uniquely determined up to renaming of nodes. The only source of differences in the resulting forests lies in possibly different orderings of rule applications (this could be eliminated, e.g., using $<$ and any fixed ordering on $CL(K)$ and on the rules).

However, these differences are not relevant: each $F$ represents a universal model $T_F$ (defined as its standard unravelling [8]) that is embeddable into every model of $K$, and can be used for query answering. The following is shown by a straightforward induction on the construction of $T_F$:
Theorem 5. Let $I$ be a model of $K$, let $\mathcal{F}$ be a complete and clash-free completion forest for $K$, and let $\mathcal{I}_F$ be the model of $K$ represented by $\mathcal{F}$. Then there is a homomorphic embedding of $\mathcal{I}_F$ into $I$. Hence, for any CQ $q$, $K \models q$ iff $\mathcal{I}_F \models q$.

5 Conjunctive Queries over Horn-SHIQ

To answer CQs over Horn-SHIQ KBs, we exploit the method for answering tree-shaped queries over domino systems. For this section, we assume that $K = \langle T, R, A \rangle$ is a consistent Horn-SHIQ KB, and $q$ is an arbitrary CQ. From a complete and clash-free completion forest $\mathcal{F}_K$ for $K$, we extract a domino system $\mathcal{D}_{\mathcal{F}_K}$ that encodes a forest-shaped universal model of $K$ for query answering. We then rewrite $q$ into a set of tree-shaped queries which can be posed separately over $\mathcal{D}_{\mathcal{F}_K}$, such that $K \models q$ iff one of the generated queries is entailed by $\mathcal{D}_{\mathcal{F}_K}$.

The transformation of the completion forest into $\mathcal{D}_{\mathcal{F}_K}$, which we now present, eliminates the ‘graph part’ of the forest by encoding it into the initial domino.

Definition 15. Let $\mathcal{F} = \langle N, E, L, \neq \rangle$ be a complete and clash-free completion forest for $K$. For every $u \in N$, let $L'(u) = L(u) \cap C(K)$.

Let $t_0 = \langle \emptyset, \emptyset, c \rangle$ be the domino where $c$ is the smallest set of fresh concept names such that $Rcd \in c$ and, for each pair $na, nb$ of named nodes in $N$, (i) $A \in L'(na)$ implies $A_n \in c$, (ii) if $nb$ is an $R$-neighbour of $na$, then $R_{a,b} \in c$, (iii) $R_{a,b} \in c$ and $R \subseteq R' \subseteq R$ implies $R_{a,b} \in c$, (iv) $R_{a,b} \in c$, $R_{b,d} \in c$ and $\text{Trans}_R(R)$ implies $R_{a,d} \in c$, and (v) $R_{a,b} \in c$ implies $\text{Inst}(R)_{a,b} \in c$.

For each named node $na \in N$, let $t_a$ denote the domino $t_a = \langle c, \{Q_a\}, c', \rangle$, where $Q_a$ is fresh and $c' = L'(na)$. For a pair $(u, u') \in E$, let $t(u, u')$ denote the domino $t(u, u')$. Then $\mathcal{D}_F = \langle D, \triangleright, R \rangle$ is the domino system with initial domino $t_0$, where

- $D$ is the smallest domino set containing (i) $t_0$, (ii) $t_a$ for each named $na \in N$, and (iii) each $t(u, u')$ such that $(u, u') \in E$ and $u'$ is unnamed and not blocked.
- $\triangleright$ is the smallest relation s.t. (i) for all named $na \in N$, $t_0 \triangleright t_a$ and $t_0 \triangleright t(na, u)$ for every $t(na, u) \in D$; and (ii) if $t(u, u'), t(u', v) \in D$ for some $(u, u'), (u', u'') \in E$ such that either $u'' = v$ is not blocked or $u''$ is blocked by $v$, then $t(u, u') \triangleright t(u', v)$.

Note that in case of inconsistent KBs, query entailment is trivial.
Definition 16. A link rewriting of \( q \) w.r.t. \( K \) is a CQ obtained from \( q \) as follows:

1. Exhaustively replace, one by one, \( R(y, x) \) by \( \text{Inv}(R)(x, y) \) whenever there are atoms of the form \( R(y, x) \) and \( S(x, y) \) in \( q \).

2. Let \( \mu : V(q) \rightarrow 1(K) \) be a partial function, and let \( \nu(x) \in \{ r \text{ (root)}, i \text{ (inside)} \} \) be a choice for each \( x \in \delta(\mu) \). Let \( \{ z \} \cup \{ x' \mid x \in V(q) \} \) be fresh variables.

Then, for each \( R(x, y) \in q \) with \( \{ x, y \} \subseteq \delta(\mu) \), let \( S \subseteq R \) be arbitrary such that \( \text{Trans}_{\mathcal{R}}(S) \) holds if either \( \nu(x) = i \) or \( \nu(y) = i \), and (i) replace \( R(x, y) \) in \( q \) by \( \text{Root}(z) \), \( S_{a,b}(z) \), where \( \mu(x) = a \) and \( \mu(y) = b \), and (ii) add in \( q \), depending on the choice \( |\nu(x), \nu(y)| \), the following atoms:

\[
\begin{align*}
[r, r]: & \quad Q_a(z, x), Q_b(z, y); \\
[i, i]: & \quad Q_a(z, x'), Q_b(z, y'), S(x, x'), S(y', y); \\
[i, r]: & \quad Q_a(z, x'), Q_b(z, y), S(x, x'); \\
[r, i]: & \quad Q_a(z, x), Q_b(z, y'), S(y', y).
\end{align*}
\]

Roughly speaking, possible \( R \)-connections between matches for \( x \) and \( y \) in \( \mathcal{I}_{F_K} \) via two individuals \( a, b \) are reflected in the query by the atoms \( \text{Root}(z), S_{a,b}(z) \) and \( |\nu(x), \nu(y)| \). For example, if \( \mathcal{R} = \{ \text{Trans}(S), S \subseteq R \} \), the CQ \( q' = \{ \text{Root}(z), S_{a,b}(z), Q_a(z, x'), S(x, x'), A(x), Q_b(z, y), B(y) \} \) is a link rewriting of \( q = \{ A(x), R(x, y), B(y) \} \), obtained by choosing \( \mu(x) = a, \mu(y) = b, \nu(x) = i, \nu(y) = r \) and \( S \). A match for \( q' \) in \( \mathcal{T}_{D_K} \) corresponds to a match for \( q \) in \( \mathcal{I}_{F_K} \) mapping \( x \) to a descendant of \( a \) (i.e., inside the tree rooted at \( a \)) and \( y \) to \( b \) (i.e., the root of \( b \)'s tree), which are connected via an \( S \)-path and thus in the extension of \( R \). Note that, as \( \nu(x) = i \) was chosen, the non-transitive \( R \) can not link the matches of \( x \) and \( y \). Choosing \( \mu(x) = a, \mu(y) = b, \nu(x) = r, \nu(y) = r \) and \( R \) we obtain a rewriting \( q'' = \{ \text{Root}(z), R_{a,b}(z), Q_a(z, x), A(x), Q_b(z, y), B(y) \} \) that captures the matches for \( q \) which map \( x \) and \( y \) to \( a \) and \( b \) if they are \( R \)-neighbors in \( \mathcal{I}_{F_K} \).

Theorem 6. \( K \models q \) iff \( D_K \models q' \) for some link rewriting \( q' \) of \( q \). Hence, due to Theorem 3, \( K \models q \) iff \( D_K \models q \circ t_q \) for some treeification \( t_q \) of a link rewriting of \( q \).

Theorem 6 suggests a procedure for deciding \( K \models q \); it suffices to verify the existence of a treeification of a link rewriting of \( q \) that has an ordered match in the domino tree of \( D_K \). The latter can be verified using the method from Section 3.

6 Computational Complexity

We now show that CQ entailment in Horn-\( S H IQ \) is decidable in exponential time. The presented method relies on the extraction of a domino system from a complete and clash-free completion forest. Hence, the following is important.

Theorem 7. The tableau algorithm for Horn-\( S H IQ \) in Section 4 decides consistency of Horn-\( S H IQ \) KBs in single exponential time. For a consistent KB, it constructs a complete and clash-free completion forest of at most exponential size.
Proof (Sketch). Definition 14 ensures that if a completion forest \( F \) contains two pairs of nodes with the same node-arc-node label combination, one of them is blocked. The number of such combinations, and thus of nodes in a forest, is single exponential in the input KB \( K \) (in fact, it is bounded by \( 2^{2^{|Cl(K)|} \times 2^{|R(K)|}} \)). Using the usual arguments [8], it can be shown that the number of rule applications needed to generate \( F \) is polynomially bounded by the maximal number of nodes it can have, as the shrinking rules do not cause repeated rule applications.

We are ready to formulate the main complexity results of this paper.

**Theorem 8.** Conjunctive query entailment \( K \models q \) in Horn-SHIQ is ExpTime-complete in combined complexity, i.e., in the size of the KB \( K \) and the query \( q \).

Proof (Sketch). By Theorem 7, we can check the consistency of \( K \) using the tableau-based algorithm in exponential time. If \( K \) is found inconsistent, then \( K \models q \) trivially holds. Otherwise, we can extract the domino system \( D_K \) from the completion forest \( F_K \) that was constructed, in time polynomial in \( |F_K| \).

Each link rewriting \( q' \) of \( q \), as well as each treeification \( tq \) of \( q' \), has size polynomial w.r.t. \( |K| + |q| \). The are at most exponentially many \( q' \) and, for each \( q' \), at most exponentially many \( tq \); hence, there are at most exponentially many \( tq \) in total, and they can be traversed in polynomial space. To show ExpTime membership of \( K \models q \), it is thus sufficient to show that \( D_K \models_o tq \) is decidable in exponential time w.r.t. \( |K| + |q| \). Indeed, checkRoleSucc runs in NPSPACE w.r.t. \( |K| + |q| \) if \( D_K \) is precomputed (note that the counter \( i \) needs only polynomial space). The procedure assocDominos runs in NP (w.r.t. \( |K| + |q| \)) using checkRoleSucc as an oracle. Hence, \( D_K \models_o tq \) is in \( NPPSPACE = PSPACE \) w.r.t. \( |K| + |q| \), if \( D_K \) is precomputed. As computing \( D_K \) is feasible in exponential time, it follows that deciding \( K \models q \) is in ExpTime. The matching lower bound follows from the ExpTime-hardness of consistency checking in Horn-SHIQ [11].

The next result shows that CQs in Horn-SHIQ are tractable in data complexity.

**Theorem 9.** Conjunctive query entailment \( K \models q \) in Horn-SHIQ is P-complete in data complexity, i.e., in the size of the ABox \( A \) of the KB \( K = (T, R, A) \).

Proof (Sketch). As in Theorem 8, we can check the consistency of \( K \) when we construct the completion forest \( F_K \). As \( T \) and \( R \) are fixed, \( |F_K| \) is polynomial w.r.t. \( A \), so \( F_K \) and \( D_K \) can be constructed in time polynomial w.r.t. \( A \). Next, for fixed \( q \), \( T \) and \( R \), and they are polynomially many treeifications \( tq \) of link rewritings \( q' \) of \( q \) w.r.t. \( A \), and they can be traversed in polynomial time. By Theorem 1, it remains to show that the existence of a domino association \( \mu: V(tq) \rightarrow D \), where \( D \) is the domino set of \( D_K \), is decidable in polynomial time w.r.t. \( A \). Since \( |V(tq)| \) is bounded by a constant w.r.t. \( A \) and \( |D| \) is polynomial w.r.t. \( A \), there are polynomially many candidate \( \mu \) w.r.t. \( A \). We can check \( r \)-successorship between dominos \( t_1, t_2 \) of \( D_K \) in time polynomial in \( |D_K| \), i.e., polynomial w.r.t. \( A \). Hence, we can check whether \( \mu \) satisfies Definition 6 in polynomial time w.r.t. \( A \). The resulting P membership bound is tight, as consistency checking in any DL allowing for conjunction on the left hand side and quantified universal restrictions on the right hand side of GCIs is P-hard in data complexity [3].
The source of \( \text{ExpTime}\)-hardness of consistency testing in \( \text{Horn-SHiQ}^- \), and hence of query entailment, are inverse roles and concepts of the form \( \exists R.A \) on the left hand side of the GCIs. Intuitively, both constructs allow to propagate information from a node to its ancestors in a completion forest; any one of them allows for an encoding of a generic Alternating \( \text{PSPACE} \) Turing machine. If we disallow both, obtaining the DL \( \text{Horn-SHQ}^- \), consistency testing and CQ entailment drop to \( \text{PSPACE}\)-completeness. Roughly, the direct successors of a node in a completion forest can be inferred in polynomial time from its label. Hence, the existence of a complete and clash-free completion forest \( F \) is refutable in \( \text{PSPACE} \) without building it, by non-deterministically following a path in \( F \) (of at most exponential length) that leads to a clash. CQ entailment is decidable in \( \text{PSPACE} \) by supplying \( \text{checkRoleSucc} \) with a \( \text{PSPACE} \) oracle for navigating the domino system \( D_K \) (note that the explicit construction of \( D_K \) may require exponential space). This procedure is worst-case optimal, as consistency checking in \( \text{Horn-SHQ}^- \) is \( \text{PSPACE}\)-hard (provable, e.g., by a generic Turing machine encoding). Finally, also our \( \text{P}\)-completeness result for data complexity of CQs carries over from \( \text{Horn-SHiQ} \) to \( \text{Horn-SHQ}^- \). Details will be given in the full paper.

7 Related Work and Conclusion

As found recently, answering CQs in some expressive DLs, including \( \text{ALCH} \) [16] and \( \text{ALCHQ} \) [14], is not harder than standard reasoning, and in fact \( \text{ExpTime} \)-complete in combined complexity. \( \text{Horn-SHiQ} \) is another such DL but orthogonal to those mentioned, as it offers transitive and inverse roles but excludes disjunction (we note that one can infer from [2] the \( \text{ExpTime}\)-completeness result for the DL \( \text{Horn-ALCHI} \), i.e., \( \text{Horn-SHiQ} \) without transitive roles and number restrictions). Moreover, the data complexity of CQs is polynomial in \( \text{Horn-SHiQ} \) but intractable in \( \text{ALCH} \) and \( \text{ALCHQ} \) (in fact, it is \( \text{coNP}\)-hard already for \( \text{AL} \) [3]).

Different approaches have been recently used to show that CQs have tractable data complexity in some DLs. A large class of such DLs are extensions of \( \text{EL} \) [1], considered e.g. in [17, 12, 10], of which \( \text{ELH}, \text{ELI}^f, \) and \( \text{EL}^{++} \) are particularly noticeable. For \( \text{EL} \) and \( \text{ELH} \), which are subsumed by \( \text{Horn-SHiQ} \) but not by \( \text{Horn-SHQ}^- \) (due to the absence of existential restrictions on the LHS of the GCIs), a reduction to Datalog has been given in [17]. In both \( \text{EL} \) and \( \text{ELH} \), CQs have \( \text{coNP}\)-complete combined complexity and \( \text{P}\)-complete data complexity. For \( \text{ELI}^f \), which is also strictly subsumed by \( \text{Horn-SHiQ} \) (as the latter offers qualified universal quantification on the RHS of axioms and more general number restrictions) an explicit (partial) construction of a universal model was used in [10]. Like in \( \text{Horn-SHiQ} \) and \( \text{Horn-SHQ}^- \), CQs have \( \text{P}\)-complete data complexity in \( \text{ELI}^f \). Finally, for \( \text{EL}^{++} \), which is orthogonal to both \( \text{Horn-SHiQ} \) and \( \text{Horn-SHQ}^- \) (\( \text{EL}^{++} \) has nominals and regular role hierarchies, but lacks universal quantification), special proof-graphs with automata were used in [12]. Noticeably, CQs in \( \text{EL}^{++} \) have \( \text{PSPACE}\)-complete combined and \( \text{P}\)-complete data complexity respectively, and thus the same complexity as in \( \text{Horn-SHQ}^- \).

Another prominent family for which data complexity has been deeply investigated is DL-Lite [5]. For the core DL-Lite and its extension with functionality and conjunction, which is subsumed by \( \text{Horn-SHiQ} \) but not by \( \text{Horn-SHQ}^- \), query
rewriting into first-order logic over relational databases has been employed. CQ answering has very low data complexity (inside logarithmic space), and its coNP-complete combined complexity is also much lower than for $\text{Horn-SHQ}^\dagger$.

Our ongoing and future work is devoted to the following issues. The first concerns richer query syntax. As the normal form and the universal model property of $\text{Horn-SHIQ}$ carry over to unions of CQs and the more general positive existential queries (PQs), our results can be immediately extended to them. In fact, answering PQs in $\text{Horn-SHIQ}$ is reducible to answering at most exponentially many (in the size of the PQ) CQs. Further, since the universal model property allows us to precompile a knowledge base $\mathcal{K}$ into a (query-independent) domino system $D_\mathcal{K}$ for on-line query answering, the identification of cases in which $D_\mathcal{K}$ is small would be beneficial. Finally, an obvious issue is a detailed study of other fragments of $\text{Horn-SHIQ}$ besides $\text{Horn-SHQ}^\dagger$. The effect of syntactic restrictions similarly as in [3] on data complexity is here of particular interest.

References