Data Complexity of Answering Unions of Conjunctive Queries in $\mathcal{SHIQ}$

Technical Report
Faculty of Computer Science
Free University of Bolzano

Magdalena Ortiz, Diego Calvanese, Thomas Eiter

Abstract

The novel context of accessing and querying large data repositories through ontologies that are formalized in terms of expressive DLs, requires on the one hand to consider query answering as the primary inference technique, and on the other hand to optimize it with respect to the size of the data, which dominates the size of ontologies. While the complexity of DLs has been studied extensively, data complexity in expressive DLs has been characterized only for answering atomic queries, and was still open for more expressive query languages, such as unions of conjunctive queries (UCQs). In this paper we advocate the need for studying this problem, and provide a significant technical contribution in this direction. Specifically, we prove a tight coNP upper bound for answering UCQs over $\mathcal{SHIQ}$ knowledge bases. We thus establish that for a whole range of DLs from $\mathcal{AL}$ to $\mathcal{SHIQ}$, answering UCQs has coNP-complete data complexity. We obtain our result by a novel tableaux-based algorithm for checking query entailment, inspired by the one in [18], but which manages the technical challenges of simultaneous transitive roles and number restrictions (which leads to a DL lacking the finite model property).

1 Introduction

Description Logics (DLs) [2] provide the formal foundations for the standard Web ontology languages [12]. In fact, OWL-DL and OWL-Lite are syntactic variants of the DLs $\mathcal{SHIF}$ and $\mathcal{SHOIN}(D)$, respectively [13,19]. In the Semantic Web and domains such as Enterprise Application Integration and Data Integration [17], ontologies provide a high-level, conceptual view of the relevant information. However, they are increasingly seen also as a mechanism to access and query data repositories. This novel context poses an original combination of challenges both in DLs/ontologies and in related areas such as data modeling and querying in databases:

(i) On the one hand, data repositories can be very large and are usually much larger than the intensional level expressing constraints on the data. Therefore, the contribution of the extensional data to inference complexity must be singled out, and one must pay attention to optimizing inference techniques with respect to data size, as opposed to the overall size of the knowledge base. In databases, this is accounted for by *data complexity* of query answering [23], where the

relevant parameter is the size of the data, as opposed to *combined complexity*, which additionally considers the size of the query and of the schema.

(*ii*) On the other hand, the data underlying an ontology should be accessed using well established and flexible mechanisms such as those provided by database query languages. This goes well beyond the traditional inference tasks involving objects in DL-based systems, like *instance checking* [11,20]. Indeed, since explicit variables are missing, DL concepts have limited possibility for relating specific data items to each other. *Conjunctive queries (CQs)*, i.e., plain SQL queries, and unions of CQs (UCQs) provide a good tradeoff between expressive power and nice computational properties, and thus are adopted as core query language in several contexts, such as data integration [17].

(*iii*) Finally, the considered DLs should have sufficient expressive power to capture common constructs used in data modeling [4]. This calls for so called *expressive DLs* [5], in which a concept may denote the complement or union of others (to capture class disjointness and covering), may involve direct and inverse roles (to account for relationships that are traversed in both directions), and may contain number restrictions (to state existence and functionality dependencies and cardinality constraints on regarding the participation to relationships in general). A notable example of such an expressive DL is $\mathcal{SHIQ}$ [14], which moreover allows for transitivity of certain roles.

As for data complexity of DLs, [11,20] showed that instance checking is coNP-hard already in the rather weak DL $\mathcal{ALE}$, and [6] that CQ answering is coNP-hard in the yet weaker DL $\mathcal{AL}$. For suitably tailored DLs, answering UCQs is polynomial (actually LogSpace) in data complexity [7,6]; see [6] for an investigation of the NLogSpace, PTime, and coNP boundaries.

For expressive DLs (with the features above, notably inverse roles), TBox+ABox reasoning has been studied extensively using techniques ranging from reductions to Propositional Dynamic Logic (PDL) (see, e.g., [8,5]) over tableaux [3,14] to automata on infinite trees [5,22]. For many such DLs, the combined complexity of TBox+ABox reasoning is ExpTime-complete, including $\mathcal{ALCQI}$ [5,22], $\mathcal{DLR}$ [8], and $\mathcal{SHIQ}$ [22]. However, until recently, little attention has been explicitly devoted to data complexity in expressive DLs. An ExpTime upper bound for data complexity of UCQ answering in $\mathcal{DLR}$ follows from the results on UCQ containment and view-based query answering in [8,9]. They are based on a reduction to reasoning in PDL, which however prevents to single out the contribution to the complexity coming from the ABox. In [18] a tight coNP upper bound for CQ answering in $\mathcal{ALCNR}$ is shown. However, this DL lacks inverse roles and is thus not suited to capture semantic data models or UML. In [15,16] a technique based on a reduction to Disjunctive Datalog is used for $\mathcal{ALCHIQ}$. For instance checking, it provides a (tight) coNP upper bound for data complexity, since it allows to single out the ABox contribution. This is not the case for general CQs, resulting in a non-tight 2ExpTime upper bound (matching also combined complexity).

Summing up, a precise characterization of data complexity for UCQ answering in expressive DLs was still open, with a gap between a coNP lower-bound

and an ExpTime upper bound. We close this gap, thus simultaneously addressing the three challenges identified above. Specifically, we make the following contributions:

- Building on techniques of [18,14], we devise a novel tableaux-based algorithm for UCQ answering over $\mathcal{SHIQ}$ knowledge bases. Technically, to show its soundness and completeness, we have to deal both with a novel blocking condition (inspired by the one in [18], but taking into account inverse and transitive roles), and with the lack of the finite model property.
- This novel algorithm provides us with a characterization of data complexity for UCQ answering in expressive DLs. Specifically, we show that data complexity of UCQ answering over $\mathcal{SHIQ}$ knowledge bases is in coNP, and thus coNP-complete for all DLs ranging from $\mathcal{AL}$ to $\mathcal{SHIQ}$.

2 Preliminaries

2.1 $\mathcal{SHIQ}$ Knowledge Bases

The syntax and semantics of $\mathcal{SHIQ}$ are defined in the standard way.

Definition 1 ($\mathcal{SHIQ}$ knowledge base). *Let $\mathbf{C}$ be a set of concept names and $\mathbf{R}$ a set of role names with a subset $\mathbf{R}_+ \subseteq \mathbf{R}$ of transitive role names. The set of roles is $\mathbf{P} \cup \{P^- \mid P \in \mathbf{R}\}$. The function* Inv *and* Trans *are defined on roles.* Inv *is defined as $\mathsf{Inv}(R) = R^-$ and $\mathsf{Inv}(R^-) = R$ for any role name R.* Trans *is a boolean function, $\mathsf{Trans}(R) = true$ iff $R \in \mathbf{R}_+$ or $\mathsf{Inv}(R) \in \mathbf{R}_+$.*

A role inclusion *axiom is an expression of the form $R \sqsubseteq R'$ where R and R' are roles. A* role hierarchy *is a set of role inclusion axioms. The relation $\sqsubseteq^*$ denotes the reflexive transitive closure of $\sqsubseteq$ over a role hierarchy $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(R') \mid R \sqsubseteq R' \in \mathcal{R}\}$. If $R \sqsubseteq^* R'$, then we say that R is a* sub-role *of R' and R' is a* super-role *of R. We will assume that it is never the case that R is both a sub-role and a super-role of R'*[1]*. A role is* simple *if its neither transitive nor has transitive sub-roles.*

The set of $\mathcal{SHIQ}$ concepts is the smallest set such that:

- *Every concept name $B \in \mathbf{C}$ is a concept,*
- *If C and D are concepts, R is a role, S is a simple role and n is a non-negative integer, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $\geq n\, S.C$, $\leq n\, S.C$ are concepts.*

A concept inclusion axiom *is an expression of the form $C \sqsubseteq D$ for two concepts C and D. A* terminology *or T-Box is a set of concept inclusion axioms.*

Let $\mathbf{I}$ be a set of individual names. An assertion *is an expression that can have the form $B(a)$, $P(a, b)$ or $a \not\approx b$ where B is a concept name, P is a role name*

[1] This consideration is done for practical purposes, however it does not restrict the expressiveness of the language. It is clear that if R is at the same time a sub-role and a super-role of R' both roles will have the same extension and one of them can be eliminated and replaced by the other.

and $a, b \in \mathbf{I}$. An A-Box *is a set of assertions. Note that no complex concepts and roles are allowed to occur in the A-Box. Since this work is on data complexity, the A-Box must contain only extensional information.*

A $\mathcal{SHIQ}$ knowledge base is a triple $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a terminology, $\mathcal{R}$ is role hierarchy and $\mathcal{A}$ is an A-Box.

The semantics of $\mathcal{SHIQ}$ knowledge bases is given by interpretations.

Definition 2 (Interpretation). *An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is defined for a set of individual names $\mathbf{I}$, a set of concepts $\mathbf{C}$ and a set of roles $\mathbf{R}$. The set $\Delta^{\mathcal{I}}$ is called* domain *of $\mathcal{I}$. The valuation $\cdot^{\mathcal{I}}$ maps each individual name in $\mathbf{I}$ to an element in $\Delta^{\mathcal{I}}$, each concept in $\mathbf{C}$ to a subset of $\Delta^{\mathcal{I}}$, and each role in $\mathbf{R}$ to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Additionally, for any concepts C, D, any roles S, R and any non-negative integer n, the valuation $\cdot^{\mathcal{I}}$ must satisfy the following equations:*

$$
\begin{aligned}
R^{\mathcal{I}} &= (R^{\mathcal{I}})^{+} \quad \text{for each role } R \in \mathbf{R}_{+} \\
(R^{-})^{\mathcal{I}} &= \{\langle o', o \rangle \mid \langle o, o' \rangle \in R^{\mathcal{I}}\} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{o \mid \text{ for all } o', \langle o, o' \rangle \in R^{\mathcal{I}} \text{ implies } o' \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{o \mid \text{ for some } o', \langle o, o' \rangle \in R^{\mathcal{I}} \text{ and } o' \in C^{\mathcal{I}}\} \\
(\geq n\, S.C)^{\mathcal{I}} &= \{o \mid |\{o' \mid \langle o, o' \rangle \in S^{\mathcal{I}} \text{ and } o' \in C^{\mathcal{I}}\}| \geq n\} \\
(\leq n\, S.C)^{\mathcal{I}} &= \{o \mid |\{o' \mid \langle o, o' \rangle \in S^{\mathcal{I}} \text{ and } o' \in C^{\mathcal{I}}\}| \leq n\}
\end{aligned}
$$

Definition 3 (Model of a knowledge base). *An interpretation $\mathcal{I}$ satisfies an assertion A iff:*

$$
\begin{aligned}
a &\in B^{\mathcal{I}} \quad \text{if } A \text{ is of the form } B(a) \\
\langle a, b \rangle &\in P^{\mathcal{I}} \quad \text{if } A \text{ is of the form } P(a, b) \\
a^{\mathcal{I}} &\neq b^{\mathcal{I}} \quad \text{if } A \text{ is of the form } a \not\approx b
\end{aligned}
$$

An interpretation $\mathcal{I}$ satisfies an A-Box $\mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$. $\mathcal{I}$ satisfies a role hierarchy $\mathcal{R}$ if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ for every $R \sqsubseteq S$ in $\mathcal{R}$. $\mathcal{I}$ satisfies a terminology $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every $C \sqsubseteq D$ in $\mathcal{T}$. $\mathcal{I}$ is a model of $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ if it satisfies $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

A $\mathcal{SHIQ}$ concept is said to be in *negation normal form* (NNF) if negation occurs only in front of concept names. Since concepts can be translated into NNF in linear time [14], we will assume that all concepts are in NNF. We denote by $NNF(\neg C)$ the NNF of the concept $\neg C$. The closure of a concept $\mathsf{clos}(C)$ is the smallest set containing C that is closed under subconcepts and negation (in NNF). For a knowledge base K, $\mathsf{clos}(K)$ is defined as the union of $\mathsf{clos}(C)$ for all C occurring in K.

Global Concepts. A knowledge base K has an associated set of concepts that we will call the *global concepts* of K. This set contains two kinds of concepts:

– For each concept inclusion axiom $C \sqsubseteq D$ in the TBox, there is a global concept of the form $\neg C \sqcup D$. This way, if we assure that all individuals in a model belong to the extension of global concepts, the model will satisfy the T-Box of K[2].
– We will consider that K, additionally to the A-Box, T-Box, R-Box, might have a set of *distinguished concepts* names that we will denote $\mathcal{C}_q$. In order not to make the notation too cumbersome, we will not denote it explicitly as a part of K. For all concept names B in $\mathcal{C}_q$, the concept $B \sqcup \neg B$ belongs to the global concepts of K. In the algorithm we present in the following sections, we will use partial representations of models of a knowledge base to verify whether some formula Q is entailed in them. In these partial representations it may remain undecided whether some individuals belong to the extension of a concept or of its negation. However, for the concepts that appear in Q, we want to assure that the decision is taken. We will later see that the set $\mathcal{C}_q$ will be used as the alphabet of concept names that may appear in the queries to be answered [3].

Definition 4 (Global Concepts). *Given a knowledge base $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ and a set of distinguished concept names $\mathcal{C}_q$, the set of* global concepts for K and $\mathcal{C}_q$ *is defined as* $\mathsf{gcon}(K, \mathcal{C}_q) = \{\neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T}\} \cup \{A \sqcup \neg A \mid A \in \mathcal{C}_q\}$.

If not stated otherwise, in the following K will denote a $\mathcal{SHIQ}$ knowledge base $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, $\mathbf{R}_K$ the roles occurring in K together with their inverses, and $\mathsf{simple}(\mathbf{R}_K)$ the roles in $\mathbf{R}_K$ that are simple. $\mathcal{C}_q$ a distinguished set of concept names, and $\mathbf{I}_K$ the individual names occurring in $\mathcal{A}$.

Example 1. As a running example, we use the knowledge base $K = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\neg A\},\ \{\},\ \{A(a)\} \rangle$. We consider $\mathcal{C}_q = \{A\}$, so $\mathsf{gcon}(K, \mathcal{C}_q) = \{\neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A,\ A \sqcup \neg A, \}$

2.2 Answering Conjunctive Queries over Knowledge Bases

Definition 5 (conjunctive query). *A* conjunctive query *(CQ) Q over a knowledge base K is a set of atoms of the form*

$$\{p_1(\overline{Y_1}), \ldots, p_n(\overline{Y_n})\}$$

where each p_i in $p_1, \ldots, p_n$ is either a role name in $\mathsf{simple}(\mathbf{R}_K)$ or a concept name in $\mathcal{C}_q$; and each $\overline{Y_i}$ in $\overline{Y_1}, \ldots, \overline{Y_n}$ is is a tuple of variables or individuals in $\mathbf{I}_K$ matching its arity.

Note that we do not allow for transitive or super-roles of transitive roles in a CQ.

[2] In [14] the authors consider an internalized T-Box. We do not make this assumption.
[3] If $\mathcal{C}_q = \mathbf{C}$, the algorithm can be used to check entailment w.r.t. any concept in the knowledge base, however this may be inconvenient from an implementation perspective.

Definition 6 (union of conjunctive queries). *A* union of conjunctive queries *(UCQ) U over a knowledge base K is an expression of the form $Q_1 \vee \cdots \vee Q_m$ where Q_i is a CQ for each $0 \leq i \leq m$.*

To say that Q is either a CQ or an UCQ, we simply say that Q is a *query*. We denote by $\mathsf{varsIndivs}(Q)$ the set of variables and individuals in a query Q.

Queries are interpreted in the standard way. For a CQ Q_i, an interpretation $\mathcal{I}$ is a model of Q_i, denoted $\mathcal{I} \models Q_i$, if there is a substitution $\sigma : \mathsf{varsIndivs}(Q_i) \rightarrow \Delta^{\mathcal{I}}$ such that $\sigma(a) = a^{\mathcal{I}}$ for each individual $a \in \mathsf{varsIndivs}(Q_i)$ and $\mathcal{I} \models p(\sigma(\overline{Y}))$, for each $p(\overline{Y})$ in Q_i. For an UCQ U, $\mathcal{I} \models U$ is defined as $\mathcal{I} \models Q_i$ for some $0 \leq i \leq m$. For a knowledge base K and a query Q, we say that K *entails* Q, denoted $K \models Q$, if $\mathcal{I} \models Q$ for each model $\mathcal{I}$ of K.

Example 2 (cont'd). We consider the CQs $Q_1 = \{P_1(x, y), P_2(x, z), A(y)\}$ and $Q_2 = \{P_2(x, y), P_2(y, z)\}$ and the UCQ $U = Q_1 \vee Q_2$. Note that $K \models Q_1$. Indeed, for an arbitrary model $\mathcal{I}$ of K, we can map x to $a^{\mathcal{I}}$, y to an object connected to $a^{\mathcal{I}}$ via role P_1 (which by the inclusion axiom $A \sqsubseteq \exists P_1.A$ exists and is an instance of A), and z to an object connected to $a^{\mathcal{I}}$ via role P_2 (which exists by the inclusion $A \sqsubseteq \exists P_2.\neg A$). Also, $K \not\models Q_2$. A model $\mathcal{I}$ of K that is not a model of Q_2 is the one with $\Delta^{\mathcal{I}} = \{o_1, o_2\}$, $a^{\mathcal{I}} = o_1$, $A^{\mathcal{I}} = \{o_1\}$, $P_1^{\mathcal{I}} = \{(o_1, o_1)\}$, and $P_2^{\mathcal{I}} = \{(o_1, o_2)\}$. Finally, since $K \models Q_1$, then also $K \models U$.

Definition 7 (Query Entailment). *Let K be a knowledge base and let Q be a query. The* query entailment *problem is to decide whether $K \models Q$.*

In the traditional database setting, free variables in a query are called distinguished variables. For a query Q that has $\overline{X}$ as distinguished variables, the query answering problem over K consists on finding all the possible tuples of individuals $\overline{T}$ of the same arity as $\overline{X}$ such that when $\overline{X}$ is substituted by $\overline{T}$ in Q, it holds that $K \models Q$. The set of such tuples $\overline{T}$ is the answer of the query. Query answering has an associated recognition problem: given a tuple $\overline{T}$, the problem is to verify whether $\overline{T}$ belongs to the answer of Q[4]. We say that query answering for a certain description logic is in a class C w.r.t. data complexity when the corresponding recognition problem is in C. Since we will only focus on the recognition problem, we allow conjunctive queries to contain individuals and we are assuming that all variables in the query are existentially quantified.

It is important to notice that the query entailment problem is not reducible to satisfiability of knowledge bases, since the negation of the query can not be expressed as a part of a knowledge base. For this reason, the known algorithms for reasoning over knowledge bases do not suffice. A knowledge base K has an infinite number possibly infinite models, and we have to verify whether the query Q is entailed in all of them. In general, we want to provide an entailment algorithm, i.e. an algorithm for checking whether a sentence Q with a particular syntax (namely, a conjunctive query) is entailed by a $\mathcal{SHIQ}$ knowledge base K. Our technique builds on the $\mathcal{SHIQ}$ algorithm in [14]. Informally, the difference is

[4] This problem is usually known as the *query output problem*.

that they only focus on problems that can be reduced to checking satisfiability, and therefore they only need to ensure that if the knowledge base has some model then their algorithm will obtain a model. In our case, however, this is not enough. We need to make sure that the algorithm obtains a set of models that suffices to check query entailment. This adaption to query answering is inspired by [18], yet we deal with DLs that have no finite model property.

3 Completion Forests

In this section, Q will denote a CQ and U an UCQ. We will first describe our method for deciding $K \models Q$, and then how it is extended to $K \models U$.

Like the algorithm in [14], we will use *completion forests*. A completion forest is a relational structure that captures sets of models of a knowledge base. Roughly, K is represented as a completion forest $\mathcal{F}_K$. Then, by applying *expansion rules* repeatedly, new completion forests are generated. The application of the rules is non-deterministic, and sometimes new individuals are introduced. Modulo these new individuals, every model of the knowledge base is preserved in some forest that results from the expansion. Therefore checking $K \models Q$ is equivalent to checking whether the query is entailed in each completion forest $\mathcal{F}$ that cannot be further expanded. Then, for each such forest $\mathcal{F}$ we will construct a single *canonical model*. Semantically, these canonical models suffice for answering all queries Q of bounded size. Furthermore, it is proved that entailment in the canonical model can be checked effectively via a syntactic mapping of the variables in Q to the nodes in $\mathcal{F}$.

As customary with tableau-style algorithms, we give blocking conditions on the rules will ensure termination of forest expansion. They are more involved than those in [14], which serve for satisfiability checking but not for query answering, and they involve a parameter n which depends on Q. This parameter will be crucial in ensuring that the canonical models of the set of forests we obtain suffice to check query entailment.

3.1 $\mathcal{SHIQ}$ Completion Forests

A forest will be defined as a set of variable trees. A *variable tree* T is a tree all whose nodes are variables excepting the root, which may be an individual, and where each node v and arc $v \to w$ is labeled with a set of concepts $\mathcal{L}(v) \subseteq \mathsf{clos}(K)$ and a set of roles $\mathcal{L}(v \to w) \subseteq \mathbf{R}_K$, respectively.

Definition 8 (n-tree equivalence).
For any integer $n \geq 0$, the n-tree *of a node v in T, denoted T_v^n, is the subtree of T rooted at v that contains all descendants of v within distance n. We denote by $\mathsf{vars}(T_v^n)$ the set of nodes in the n-tree of v.*

Variables v and v' in T are n-tree equivalent *in T, if T_v^n and $T_{v'}^n$ are isomorphic, i.e., there is a bijection $\psi : \mathsf{vars}(T_v^n) \to \mathsf{vars}(T_{v'}^n)$ such that:*

- $\psi(v) = v'$

– *for every node w in* $\mathsf{vars}(T_v^n)$, $\mathcal{L}(w) = \mathcal{L}(\psi(w))$
– *for every arc connecting two nodes w and w' in* $\mathsf{vars}(T_v^n)$, $\mathcal{L}(w \to w') = \mathcal{L}(\psi(w) \to \psi(w'))$.

Definition 9 (n-Witness). *If nodes v and v' in T are n-tree equivalent, v' is an ancestor of v in T and v is not in $T_{v'}^n$, then v' is a* n-witness of v in T. *Furthermore, $T_{v'}^n$* tree-blocks *T_v^n and each variable w in $T_{v'}^n$* tree-blocks *variable $\psi^{-1}(w)$ in T_v^n.*

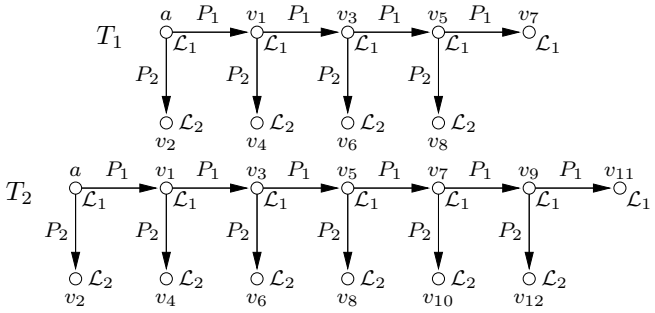

Figure1. Trees and completion forests for example knowledge base

Example 3 (cont'd). Consider the variable tree T_1 in Figure 1, with a as root, where $\mathcal{L}_1 = \{A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A,\ A \sqcup \neg A,\ \exists P_1.A,\ \exists P_2.\neg A\}$, and $\mathcal{L}_2 = \{\neg A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A,\ A \sqcup \neg A\}$. Then, v_1 and v_5 are 1-tree equivalent in T_1; v_1 is a witness of v_5 (but not vice versa); ${T_1}_{v_1}^1$ tree-blocks ${T_1}_{v_5}^1$; and v_1 (resp., v_3, v_4) tree-blocks v_5 (resp., v_7, v_8).

Definition 10 (completion forest [14]). *A* completion forest *for a knowledge base K is given by a forest of trees and an inequality relation $\not\approx$, implicitly assumed to be symmetric. The forest is a set of variable trees whose roots are the individuals in $\mathbf{I}_K$ and can be arbitrarily connected by arcs.*

For a completion forest $\mathcal{F}$, we denote $\mathsf{nodes}(\mathcal{F})$ the set of individuals and variables in $\mathcal{F}$, and $\mathsf{vars}(\mathcal{F})$ the nodes in $\mathcal{F}$ which are variables. For every arc $v \to w$ and role R, if the label $\mathcal{L}(v \to w)$ contains some role R' with $R' \sqsubseteq^* R$, then w is an *R-successor* and an $\mathsf{Inv}(R)$*-predecessor* of v. We call w an *R-neighbor* of v, if w is an *R-successor* or an $\mathsf{Inv}(R)$*-predecessor* of v. The *ancestor* relation is the transitive closure of the union of the R-predecessor relations for all roles R.

In order to provide a method for verifying entailment of a conjunctive query Q in a knowledge base K, we will first associate to K an initial completion forest and then we will generate new completion forests by applying *expansion rules* until no more expansions can be obtained. As we will see in the following, the set forests obtained by this method will be sufficient to check entailment of Q.

We associate an initial completion forest $\mathcal{F}_K$ with knowledge base K as follows:

- The nodes are the individuals $a \in \mathbf{I}_K$, and $\mathcal{L}(a) = \{B \mid B(a) \in \mathcal{A}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$.
- The arc $a \to b$ is present iff $P(a,b) \in \mathcal{A}$ for some role name P, and $\mathcal{L}(a \to b) = \{P \mid P(a,b) \in \mathcal{A}\}$.
- $a \not\approx b$ iff $a \neq b \in \mathcal{A}$.

Example 4. In our running example, $\mathcal{F}_K$ contains only the node a which has the label $\mathcal{L}(a) := \{A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A,\ A \sqcup \neg A\}$.

Next, before giving the expansion rules, we define a notion of blocking which depends on a depth parameter $n \geq 0$. This notion generalizes blocking in [14], where the n parameter is not present.

Definition 11 (n-blocking). *For integer $n \geq 0$, a node v in a completion forest $\mathcal{F}$ is* n-blocked*, if v is not a root and either directly or indirectly n-blocked. Node v is* indirectly n-blocked*, if one of its ancestors is n-blocked or $\mathcal{L}(w \to v) = \emptyset$ for some arc $w \to v$ in $\mathcal{F}$. Node v is* directly n-blocked *iff none of its ancestors is n-blocked and v is a leaf of a tree-blocked n-tree in $\mathcal{F}$.*

Note that x is m-blocked for each $m \leq n$ if it is n-blocked. When $n \geq 1$, then n-blocking implies *pairwise blocking*, which is the blocking used in [14]. When n=0, then n-blocking corresponds to blocking by equal node labels, which is a sufficient blocking condition in some DLs weaker than $\mathcal{SHIQ}$.

Example 5. Consider $\mathcal{F}_1$ with the variable tree T_1 from Example 3 and with an empty $\not\approx$ relation. $\mathcal{F}_1$ is 1-blocked. Analogously, consider the completion forest $\mathcal{F}_2$ that has the variable tree T_2 in Figure 1. In $\mathcal{F}_2$ the $\not\approx$ relation is also empty. $\mathcal{F}_2$ is 2-blocked.

Now we can give our expansion rules. From $\mathcal{F}_K$, new completion forests for K can be obtained by applying the rules in Table 1. We denote by $\mathbb{F}_K$ the set of all $\mathcal{F}$ obtained in this way. Note that the application of the rules is non-deterministic. Different choices for E in the $\sqcup$-rule and the *choose*-rule generate different forests. The $\exists$-rule and the $\geq$-rule are called *generating rules* since they add new nodes to the forest. Note that our rules are very similar to the ones in [14]. The main differences are that "blocked" is uniformly replaced by "n-blocked" and the $\exists$-rule and $\geq$-rule in [14] are slightly different, since now the labels of the nodes they generate must contain $\mathsf{gcon}(K, \mathcal{C}_q)$.

Definition 12 (Clash free completion forest). *A node v in a completion forest $\mathcal{F}$ contains a* clash *iff for some concept C, $\{C, \neg C\} \subseteq \mathcal{L}(v)$ or if $\leq n\, S.C. \in \mathcal{L}(v)$ and v has $n+1$ S-successors $w_0, \ldots, w_n$ such that $C \in \mathcal{L}(w_i)$ for all w_i and $w_i \not\approx w_j \in \mathcal{F}$ for all $0 \leq i \lneq j \leq n$. A completion forest $\mathcal{F}$ is* clash free *if none of its nodes contains a clash.*

$\sqcap$-rule:	if	$C_1 \sqcap C_2 \in \mathcal{L}(v)$, v is not indirectly n-blocked and $\{C_1, C_2\} \not\subseteq \mathcal{L}(v)$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_1, C_2\}$
$\sqcup$-rule:	if	$C_1 \sqcup C_2 \in \mathcal{L}(v)$, v is not indirectly n-blocked and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{E\}$ for some $E \in \{C_1, C_2\}$
$\exists$-rule:	if	$\exists R.C \in \mathcal{L}(v)$, v is not n-blocked and v has no R-neighbor w with $C \in \mathcal{L}(w)$
	then	create new node w with $\mathcal{L}(v \to w) := \{R\}$ and $\mathcal{L}(w) := \{C\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$
$\forall$-rule:	if	$\forall R.C \in \mathcal{L}(v)$, v is not indirectly n-blocked and there is an R-neighbor w of v with $C \notin \mathcal{L}(w)$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{C\}$
$\forall_+$-rule:	if	$\forall R.C \in \mathcal{L}(v)$, v is not indirectly n-blocked, there is some R' with $\mathsf{Trans}(R')$ and $R' \sqsubseteq^* R$ and there is an R'-neighbor w of v with $\forall R'.C \notin \mathcal{L}(w)$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{\forall R'.C\}$
choose-rule:	if	$\leq n\, S.C \in \mathcal{L}(v)$ or $\geq n\, S.C \in \mathcal{L}(v)$, v is not indirectly n-blocked and there is an S-neighbor w of v with $\{C, NNF(\neg C)\} \cap \mathcal{L}(w) = \emptyset$
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{E\}$ for some $E \in \{C, NNF(\neg C)\}$
$\geq$-rule:	if	$\geq n\, S.C \in \mathcal{L}(v)$, v is not n-blocked and there are not S-neighbors $w_1, \ldots, w_n$ of v such that $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j$ for $1 \leq i < j \leq n$
	then	create new nodes $w_1, \ldots, w_n$ with $\mathcal{L}(v \to w_i) := \{S\}$, $\mathcal{L}(w_i) := \{C\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$ and $w_i \not\approx w_j$ for $1 \leq i < j \leq n$
$\leq$-rule:	if	$\leq n\, S.C \in \mathcal{L}(v)$, v is not indirectly n-blocked, $\lvert\{w \mid w$ is an S-neighbor of v and $C \in \mathcal{L}(w)\}\rvert > n$ and there are S-neighbors w, w' of v with not $w \not\approx w'$, w is neither a root node nor an ancestor of w' and $C \in \mathcal{L}(w) \cap \mathcal{L}(w')$
	then	$\mathcal{L}(w') := \mathcal{L}(w') \cup \mathcal{L}(w)$, if z is an ancestor of v, then $\mathcal{L}(w' \to w) := \mathcal{L}(w' \to v) \cup \mathsf{Inv}(\mathcal{L}(v \to w'))$, else $\mathcal{L}(v \to w') := \mathcal{L}(v \to w') \cup \mathcal{L}(v \to w)$, $\mathcal{L}(v \to w) := \emptyset$, set $w'' \not\approx w'$ for all w'' with $w'' \not\approx w$
$\leq_r$-rule:	if	$\leq n\, S.C \in \mathcal{L}(v)$, $\lvert\{w \mid w$ is an S-neighbor of v and $C \in \mathcal{L}(w)\}\rvert > n$ and there are S-neighbors w, w' of v with not $w \not\approx w'$, both w and w' are root nodes and $C \in \mathcal{L}(w) \cap \mathcal{L}(w')$
	then	$\mathcal{L}(w') := \mathcal{L}(w') \cup \mathcal{L}(w)$, $\mathcal{L}(w) := \emptyset$ For all edges $w \to w''$ (i) if $w' \to w''$ does not exist, create it with $\mathcal{L}(w' \to w'') = \emptyset$ (ii) set $\mathcal{L}(w' \to w'') := \mathcal{L}(w' \to w'') \cup \mathcal{L}(w \to w'')$; For all edges $w'' \to w$ (i) if $w'' \to w'$ does not exist, create it with $\mathcal{L}(w'' \to w') = \emptyset$ (ii) set $\mathcal{L}(w'' \to w') := \mathcal{L}(w'' \to w') \cup \mathcal{L}(w'' \to w)$; Remove all edges to and from w; Set $w \approx w'$ and $w'' \not\approx w'$ for all w'' with $w'' \not\approx w$.

Table1. Expansion Rules

Definition 13 (n-complete completion forest). *A completion $\mathcal{F}$ is n-complete if none of the rules in Table 1 can be applied to it (under n-blocking).*

Finally, for a set of completion forests $\mathbb{F}$, we will denote by $\mathsf{ccf}(\mathbb{F}^n)$ the set of forests in $\mathbb{F}$ that are n-complete and clash free.

Example 6. Both $\mathcal{F}_1$ and $\mathcal{F}_2$ can be obtained from F_K by applying the expansion rules. They are both complete and clash-free, so $\mathcal{F}_1 \in \mathsf{ccf}(\mathbb{F}^1_K)$ and $\mathcal{F}_2 \in \mathsf{ccf}(\mathbb{F}^2_K)$.

3.2 Models of a Completion Forest.

Semantically, we can interpret a completion forest in the way we interpret a knowledge base. Viewing variables in a completion forest $\mathcal{F}$ for K as individuals, an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ of the individual names, concepts an roles in $\mathcal{F}$ is an extended interpretation of K. We thus define models of $\mathcal{F}$ in terms of extended models of K. We will see completion forests as a representation of a set of models of the knowledge base.

Definition 14 (Model of a completion forest). *For a completion forest $\mathcal{F}$ for K, $\mathcal{F} \in \mathbb{F}^n_K$, an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is a model of $\mathcal{F}$, represented $\mathcal{I} \models \mathcal{F}$ if $\mathcal{I} \models K$ and for all nodes $v, w \in \mathcal{F}$ the following hold:*

- *if $C \in \mathcal{L}(v)$, then $v^\mathcal{I} \in C^\mathcal{I}$*
- *if $R \in \mathcal{L}(v \to w)$ then $\langle v^\mathcal{I}, w^\mathcal{I} \rangle \in R^\mathcal{I}$*
- *if $v \not\approx w \in \mathcal{F}$, then $v^\mathcal{I} \neq w^\mathcal{I}$*

We want to emphasize that in order to be a model of a completion forest for K, an interpretation must be a model of K. The initial completion forest is just an alternative representation of the knowledge base, and it has exactly the same models. When we expand the forest, we will make choices and obtain new forests that capture a subset of the models of the knowledge base.

Lemma 1. *An interpretation $\mathcal{I}$ is a model of $\mathcal{F}_K$ iff $\mathcal{I}$ is a model of K.*

Proof. The if direction follows from Definition 14. To prove the other direction, it suffices to consider an arbitrary model $\mathcal{I}$ of K and verify that for for all nodes $a, b \in \mathcal{F}_K$ the following hold:

(i) if $C \in \mathcal{L}(a)$, then $a^\mathcal{I} \in C^\mathcal{I}$
(ii) if $R \in \mathcal{L}(a \to b)$ then $\langle a^\mathcal{I}, b^\mathcal{I} \rangle \in R^\mathcal{I}$
(iii) if $a \not\approx b \in \mathcal{F}$, then $a^\mathcal{I} \neq b^\mathcal{I}$

By definition, the nodes in $\mathcal{F}_K$ correspond exactly to the individuals in $\mathbf{I}_K$. For each such individual a, the label of a in $\mathcal{F}_K$ is given as $\mathcal{L}(a) = \{B \mid B(a) \in \mathcal{A}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, if $B(a) \in \mathcal{A}$ then $a^\mathcal{I} \in B^\mathcal{I}$. For any concept $C \in \mathsf{gcon}(K, \mathcal{C}_q)$, either C is of the form $\neg D \sqcup E$ for some $D \sqsubseteq E$ in $\mathcal{T}$ or C is of the form $B \sqcup \neg B$ for a concept name B. In the first case, $a^\mathcal{I} \in (\neg D \sqcup E)^\mathcal{I}$ must hold because $\mathcal{I}$ is a model of $\mathcal{T}$. In the other case, $o^\mathcal{I} \in (B \sqcup \neg B)^\mathcal{I}$ holds for any individual o in $\Delta^\mathcal{I}$ and any concept B by the definition of interpretation. So

we have that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every $C \in \mathcal{L}(a)$ and item *(i)* holds. The label of a pair of nodes a, b in $\mathcal{F}_K$ is given by $\mathcal{L}(a \to b) = \{P \mid P(a,b) \in \mathcal{A}\}$. Since $\mathcal{I}$ is a model of $\mathcal{A}$, $\langle a^{\mathcal{I}} b^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$ for every $P(a,b)$ in $\mathcal{A}$, hence item *(ii)* holds. Analogously, the $\not\approx$ relation was initialized with $a \neq b$ for every $a \not\approx b$ in $\mathcal{A}$, so item *(iii)* will also hold for any $\mathcal{I}$ model of $\mathcal{A}$.

As we prove in Proposition 1, the union of all the models of the forests in $\mathsf{ccf}(\mathbb{F}_K^n)$ captures all the models of a knowledge base K, independently of the value of n. This result is crucial, since it allows us to ensure that checking the forests in $\mathsf{ccf}(\mathbb{F}_K^n)$ suffices to check all models of K. In order to prove this result, we will first prove the following lemma. It states that when applying any of the rules in Table 1, all models are preserved.

Lemma 2. *Let $\mathcal{F}$ be a completion forests in $\mathbb{F}_K^n$, let r be a rule in Table 1 and let $\mathbf{F}$ be the set of completion forests that can be obtained from $\mathcal{F}$ by applying r. Then for every $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{F}$ there is some $\mathcal{F}' \in \mathbf{F}$ and some $\mathcal{I}'$ that is an extension of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{F}'$.*

Proof. We will do the proof for each rule r in Table 1.

First we will consider the deterministic, non-generating rules. There is only one $\mathcal{F}'$ in $\mathbf{F}$ and the models of $\mathcal{F}$ are exactly the models of $\mathcal{F}'$. For the case of the $\sqcap$-rule, there is some node v in $\mathcal{F}$ s.t. $C_1 \sqcap C_2 \in \mathcal{L}(v)$. Since $\mathcal{I}$ is a model of $\mathcal{F}$, then $v^{\mathcal{I}} \in (C_1 \sqcap C_2)^{\mathcal{I}}$, and since $\mathcal{I}$ is a model of K, then both $v^{\mathcal{I}} \in C_1^{\mathcal{I}}$ and $v^{\mathcal{I}} \in C_2^{\mathcal{I}}$ hold. The inequality relation and all labels in $\mathcal{F}'$ are exactly as in $\mathcal{F}$, the only change is that $\{C_1, C_2\} \subset \mathcal{L}(v)$ in $\mathcal{F}'$, so $\mathcal{I} \models \mathcal{F}'$.

The cases of the the $\forall$-rule and the $\forall_+$-rule, are similar to the $\sqcap$-rule. All labels of $\mathcal{F}$ are preserved in $\mathcal{F}'$. Only the label of the node w to which the rule was applied is modified in $\mathcal{F}'$, having $C \subset \mathcal{L}(w)$ or $\forall R'.C \subset \mathcal{L}(w)$ respectively. Since $\mathcal{I}$ is a model of K, $v^{\mathcal{I}} \in (\forall R.C)^{\mathcal{I}}$ and w and R-neighbor of v imply $y^{\mathcal{I}} \in C^{\mathcal{I}}$, and $v^{\mathcal{I}} \in (\forall R.C)^{\mathcal{I}}$ and w and R'-neighbor of v for some transitive sub-role of R imply $w^{\mathcal{I}} \in (\forall R'.C)^{\mathcal{I}}$, then trivially $\mathcal{I} \models \mathcal{F}'$ in both cases.

Let us analyze the non-deterministic rules. For the case of the $\sqcup$-rule, there is some node v in $\mathcal{F}$ s.t. $C_1 \sqcup C_2 \in \mathcal{L}(v)$. After applying the $\sqcup$-rule, we will have two forests $\mathcal{F}_1'$, $\mathcal{F}_2'$ with $\{C_1\} \subset \mathcal{L}(v)$ in $\mathcal{F}_1'$ and $\{C_2\} \subset \mathcal{L}(v)$ in $\mathcal{F}_2'$ respectively. For every $\mathcal{I}$ such that $\mathcal{I}$ is a model of $\mathcal{F}$ we have $v^{\mathcal{I}} \in (C_1 \sqcup C_2)^{\mathcal{I}}$, and since $\mathcal{I}$ is a model of K, then either $v^{\mathcal{I}} \in C_1^{\mathcal{I}}$ or $v^{\mathcal{I}} \in C_2^{\mathcal{I}}$ hold. If it is the case that $v^{\mathcal{I}} \in C_1^{\mathcal{I}}$, then $\mathcal{I} \models \mathcal{F}_1'$, and otherwise $\mathcal{I} \models \mathcal{F}_2'$, so the claim holds.

The proof for the choose rule is trivial, since after its application we will have two forests $\mathcal{F}_1'$, $\mathcal{F}_2'$ with $\{C\} \subset \mathcal{L}(v)$ in $\mathcal{F}_1'$ and $\{NNF(\neg C)\} \subset \mathcal{L}(v)$ in $\mathcal{F}_2'$ respectively, but since trivially $v^{\mathcal{I}} \in (C \sqcup \neg C)^{\mathcal{I}}$ holds for any v, any C and any $\mathcal{I}$ model of K, then for every $\mathcal{I}$ either $\mathcal{I} \models \mathcal{F}_1'$ or $\mathcal{I} \models \mathcal{F}_2'$ holds.

When the $\leq$-rule or the $\leq_r$-rule are applied to a variable v in $\mathcal{F}$, there are some variables w, w' neighbors of v s.t. w is identified with w' in $\mathcal{F}'$. This can only be done if we do not have that $w^{\mathcal{I}} \neq w'^{\mathcal{I}}$ in $\mathcal{I}$, hence it must be the case that $w^{\mathcal{I}} = w'^{\mathcal{I}}$. In $\mathcal{F}'$, we will add the pair $\langle w, w' \rangle$ to the extension of $\approx$. Due to $w^{\mathcal{I}} = w'^{\mathcal{I}}$ the extensions of all labels of $\mathcal{F}$ will be preserved in $\mathcal{F}'$ and so $\mathcal{I} \models \mathcal{F}'$ holds.

Finally we consider the two generating rules. For the case of the $\exists$-rule, since the propagation rule was applied, there is some v in $\mathcal{F}$ such that $\exists R.C \in \mathcal{L}(v)$, which implies the existence of some $o \in \Delta^{\mathcal{I}}$ with $\langle v^{\mathcal{I}}, o\rangle \in R^{\mathcal{I}}$ and $o \in C^{\mathcal{I}}$. $\mathcal{F}'$ was obtained by adding to $\mathcal{F}$ a new node which we denote w. $\mathcal{I}$ will be extended to $\mathcal{I}'$ by setting $w^{\mathcal{I}'} = o$, and thus $\mathcal{I}' \models \mathcal{F}'$.

The case of the $\geq$-rule is analogous to the $\exists$-rule, since in models of $\mathcal{F}'$ we have that $w_i^{\mathcal{I}} = o_i$ for $1 \leq i \leq n$, where $\{w_1, \ldots, w_n\}$ are the variables added to $\mathcal{F}$ and $o_1, \ldots, o_n$ denote the elements in $\Delta^{\mathcal{I}}$ s.t. $\langle v^{\mathcal{I}}, o_i\rangle \in R^{\mathcal{I}}$ and $o_i \in C^{\mathcal{I}}$ for the variable v in $\mathcal{F}$ to which the rule was applied.

Finally, we can prove that the union of models of the forests in $\mathsf{ccf}(\mathbb{F}_K^n)$ is exactly the set of all models of K modulo new individuals.

Proposition 1. *Let $n \geq 0$. For every $\mathcal{I}$ such that $\mathcal{I} \models K$, there is some $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ and some $\mathcal{I}'$ that is an extension of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{F}$.*

Proof. From Lemmas 1 and 2, we have that for every $\mathcal{I}$ such that $\mathcal{I} \models K$, there is some $\mathcal{F} \in \mathbb{F}_K^n$ with $n \geq 0$ such that $\mathcal{I}'$ is a model of $\mathcal{F}$ for some $\mathcal{I}'$ extension of $\mathcal{I}$. Now we want to prove that there is some $\mathcal{F}_c \in \mathsf{ccf}(\mathbb{F}_K^n)$ such that $\mathcal{I}' \models \mathcal{F}_c$ for some $\mathcal{I}'$ extension of $\mathcal{I}$. Suppose there is an $\mathcal{I}'$ extension of $\mathcal{I}'$ such that $\mathcal{I}$ is a model of some completion forest $\mathcal{F}$ that is not complete. Then either it is possible to obtain a new forest $\mathcal{F}'$ such that $\mathcal{I}'' \models \mathcal{F}'$ for some $\mathcal{I}''$ that extends $\mathcal{I}'$ (and hence $\mathcal{I}$), or none of the propagation rules can be applied. The latest would imply that either $\mathcal{F}$ was complete, which is a contradiction, or that $\mathcal{F}$ had a clash, which is also a contradiction since $\mathcal{F}$ has a model. Hence, while applying the propagation rules, the model will be preserved, and maybe extended, until some complete forest $\mathcal{F}_c$ is reached.

4 Answering Conjunctive Queries

Recall, that for a knowledge base K and a query U, we say that $K \models U$ iff for every interpretation $\mathcal{I}$, $\mathcal{I} \models K$ implies $\mathcal{I} \models U$. Analogously, we define a semantical notion of query entailment in a completion forest: for a completion forest $\mathcal{F}$ and a query U, we say that $\mathcal{F} \models U$ iff for every interpretation $\mathcal{I}$, $\mathcal{I} \models \mathcal{F}$ implies $\mathcal{I} \models U$. We are interested in checking whether $K \models U$, but this means that entailment of U has to be verified in every model of K. However, we know that is suffices to check entailment in each forest $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ for any n, since semantically, they capture all the models of the knowledge base. This is stated in the following proposition:

Proposition 2. *Let $n \geq 0$ be arbitrary. Then $K \models U$ iff $\mathcal{F} \models U$ for each $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$.*

Proof. The only if direction is trivial. Consider any $\mathcal{F} \in \mathbb{F}_K^n$. Since any model $\mathcal{I}$ of $\mathcal{F}$ is a model of K by definition, then $K \models U$ implies $\mathcal{F} \models U$. The if direction can be done by contraposition. If $K \not\models U$, then there is some model $\mathcal{I}$ of K such that $\mathcal{I} \not\models U$. By Proposition 1, there is some $\mathcal{I}'$ extension of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{F}$ for some $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$. $\mathcal{I} \not\models U$ implies $\mathcal{I}' \not\models U$, and thus $\mathcal{F} \not\models U$.

This result is crucial for our query answering method, since it ensures that to check query entailment we must only consider $\mathsf{ccf}(\mathbb{F}_K^n)$, a finite set of finite structures. Now, we will see that for a suitable n, $\mathcal{F} \models U$ for an $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$, we can be verified by finding a syntactical mapping of the query into $\mathcal{F}$.

We will first do it for a CQ Q, and in Section 4.3 we will extend it to an UCQ U. We say that Q can be mapped into a completion forest $\mathcal{F}$, denoted $Q \hookrightarrow \mathcal{F}$, if there is a mapping $\mu : \mathsf{varsIndivs}(Q) \to \mathsf{nodes}(\mathcal{F})$ that is the identity mapping for all individuals in $\mathsf{varsIndivs}(Q)$ and that satisfies the following:

1. For all $C(x)$ in Q, $C \in \mathcal{L}(\mu(x))$.
2. For all $R(x, y)$ in Q, $\mu(y)$ is an R-neighbor of $\mu(x)$.

Example 7. $Q_1 \hookrightarrow \mathcal{F}_2$ holds, as witnessed by the mapping $\mu(x) = a$, $\mu(y) = v_2$ and $\mu(z) = v_1$. Note that there is no mapping of Q_2 into $\mathcal{F}_2$ satisfying the above conditions.

We will relate the semantical notion $\mathcal{F} \models Q$, with the syntactical notion of map we will show that the query can be mapped into a forest iff every model of the forest is a model of the query. The only if direction is easy, if a mapping μ exists, then Q is satisfied in any model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{F}$.

Lemma 3. *If $Q \hookrightarrow \mathcal{F}$, then $\mathcal{F} \models Q$.*

Proof. Since $Q \hookrightarrow \mathcal{F}$, there is a mapping $\mu : \mathsf{varsIndivs}(Q) \to \mathsf{nodes}(\mathcal{F})$ satisfying conditions 1 and 2. Take any arbitrary model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{F}$. By definition, it satisfies the following:

- if $C \in \mathcal{L}(x)$, then $x^{\mathcal{I}} \in C^{\mathcal{I}}$
- if x is an R-neighbor of y, then $\langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$.
- if $x \not\approx y \in \mathcal{F}$, then $x^{\mathcal{I}} \neq y^{\mathcal{I}}$

We can define a substitution σ from the variables and individuals in $\mathsf{varsIndivs}(Q)$ to objects in $\Delta^{\mathcal{I}}$ as $\sigma(x) = \mu(x)^{\mathcal{I}}$, and it satisfies $\sigma(\overline{Y}) \in p^{\mathcal{I}}$ for all $p(\overline{Y})$ in Q.

The if direction is more challenging. Now the blocking conditions come into play and the mapping will only be feasible if n is sufficiently large. We show that provided $\mathcal{F}$ has been expanded far enough, a suitable mapping μ can be constructed from some model of $\mathcal{F}$. In particular, we construct for each $\mathcal{F}$ a *single* model $\mathcal{I}_{\mathcal{F}}$, called the *canonical model of* $\mathcal{F}$. This canonical model suffices to check entailment in the forest for *all* queries Q of bounded size. As we will see, the canonical model can be used to prove that if Q is satisfied in this model, then we can construct the mapping μ from it.

4.1 Tableaux and Canonical Models

The (possibly infinite) canonical model for $\mathcal{F}$ is model is obtained by unraveling the forest $\mathcal{F}$, where the blocked nodes act like 'loops'. Its domain is given by the set of all paths that start in a root node and finish in some node of $\mathcal{F}$.

In order to build the canonical model for $\mathcal{F}$, we will proceed in two steps. First, we will unravel the forest into a tableau, and then induce a model from this tableau.

From any forest $\mathcal{F} \in \mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ for $n \geq 1$, we can construct a tableau for K. If $\mathcal{F}$ contains blocked nodes, then its tableau will be an infinite structure. The tableau T of a forest $\mathcal{F}$ will correspond to the *unraveling* of $\mathcal{F}$. i.e. the structure obtained by considering each path to a node in $\mathcal{F}$ as a node of T. Following [14], we will give a rather complex definition of a tableau. Defining a model of K from a tableau will be straightforward with this definition.

Definition 15 (Tableau). *$T = \langle \mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{I} \rangle$ is a tableau for a knowledge base $K = \langle \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle$ iff*

- *$\mathbf{S}$ is a non-empty set,*
- *$\mathcal{L} : \mathbf{S} \to 2^{\mathsf{clos}(K)}$ maps each element in $\mathbf{S}$ to a set of concepts,*
- *$\mathcal{E} : \mathbf{R}_K \to 2^{\mathbf{S} \times \mathbf{S}}$ maps each role to a set of pairs of elements in $\mathbf{S}$, and*
- *$\mathcal{I} : \mathbf{I}_K \to \mathbf{S}$ maps each individual occurring in $\mathcal{A}$ to an element in $\mathbf{S}$.*

Furthermore, for all $s, t \in \mathbf{S}$; $C, C_1, C_2 \in \mathsf{clos}(K)$ and $R, R', S \in \mathbf{R}_K$, T satisfies:

(P1) *if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$,*
(P2) *if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$,*
(P3) *if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$,*
(P4) *if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$,*
(P5) *if $\exists R.C \in \mathcal{L}(s)$, then there is some $t \in \mathbf{R}$ such that $\langle s, t \rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$,*
(P6) *if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(R')$ for some $R' \sqsubseteq^* R$ with $\mathsf{Trans}(R') = true$ then $\forall R.C \in \mathcal{L}(t)$,*
(P7) *$\langle s, t \rangle \in \mathcal{E}(R)$ iff $\langle t, s \rangle \in \mathcal{E}(\mathsf{Inv}(R))$,*
(P8) *if $\langle s, t \rangle \in \mathcal{E}(R)$ and $R \sqsubseteq^* S$ then $\langle s, t \rangle \in \mathcal{E}(S)$,*
(P9) *if $\leq n\, S.C \in \mathcal{L}(s)$, then $|\{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(t)\}| \leq n$,*
(P10) *if $\geq n\, S.C \in \mathcal{L}(s)$, then $|\{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(t)\}| \geq n$,*
(P11) *if $\langle s, t \rangle \in \mathcal{E}(R)$ and either $\leq n\, S.C \in \mathcal{L}(s)$ or $\geq n\, S.C \in \mathcal{L}(s)$, then $C \in \mathcal{L}(t)$ or $NNF(\neg C) \in \mathcal{L}(t)$,*
(P12) *if $C(a) \in \mathcal{A}$ then $C \in \mathcal{L}(\mathcal{I}(a))$,*
(P13) *if $R(a, b) \in \mathcal{A}$ then $\langle \mathcal{I}(a), \mathcal{I}(a) \rangle \in \mathcal{E}(R)$,*
(P14) *if $a \neq b \in \mathcal{A}$ then $\mathcal{I}(a) \neq \mathcal{I}(a)$,*
(P15) *if $C \in \mathsf{gcon}(K, \mathcal{C})$, then for all $s \in \mathbf{S}$ $C \in \mathcal{L}(s)$.*

Trivially, we can obtain a canonical model of a knowledge base from a tableau for it.

Definition 16 (Canonical Model of a Tableau). *Let T be a tableau. The* canonical model of *T, $\mathcal{I}_T = (\Delta^{\mathcal{I}_T}, \cdot^{\mathcal{I}_T})$ is defined as follows:*

$$\Delta^{\mathcal{I}_T} := \mathbf{S}$$

for all concept names A in $\mathsf{clos}(K)$,

$$A^{\mathcal{I}_T} := \{s \mid A \in \mathcal{L}(s)\}$$

for all individual names a in $\mathbf{I}_K$,

$$a^{\mathcal{I}} := a$$

for all role names R in $\mathcal{R}$,

$$R^{\mathcal{I}_T} := \mathcal{E}(R)^{\oplus}$$

where $\mathcal{E}(R)^{\oplus}$ the *closure of the extension* of R under $\mathcal{R}$, which is defined as:

$$\mathcal{E}(R)^{\oplus} := \begin{cases} (\mathcal{E}(R))^{+} & \text{if } \mathsf{Trans}(R) \\ \mathcal{E}(R) \cup \mathsf{sub}(\mathcal{E}(R)^{\oplus}) & \textit{otherwise} \end{cases}$$

where $(\mathcal{E}(R))^{+}$ denotes the transitive closure of $\mathcal{E}(R)$ and

$$\mathsf{sub}(\mathcal{E}(R)^{\oplus}) = \bigcup_{S \sqsubseteq^{*} R, P \neq R} \mathcal{E}(S)^{\oplus}.$$

Lemma 4. *Let T be a tableau for K. The canonical model of T is a model of K.*

Proof. That $\mathcal{I}_T$ is a model of $\mathcal{R}$ and $\mathcal{A}$ can be proved exactly as in the proof of Lemma 2 in [14]. Due to $(P15)$, it can be easily verified that $\mathcal{I}_T$ is also a model of $\mathcal{T}$.

4.2 Canonical Interpretation of a Completion Forest.

Each $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ induces a tableau $T_{\mathcal{F}}$, and this tableau gives us a *canonical model* for $\mathcal{F}$, which we will denote $\mathcal{I}_{\mathcal{F}}$.

Definition 17 (Tableau induced by a completion forest). *A* path *in a completion forest $\mathcal{F}$ is a sequence of pairs of nodes of the form $p = [\frac{v_0}{v'_0}, \ldots, \frac{v_n}{v'_n}]$. In such a path, we define $\mathsf{tail}(p) = v_n$ and $\mathsf{tail}'(p) = v'_n$; and $[p \mid \frac{v_{n+1}}{v'_{n+1}}]$ denotes the path $[\frac{v_0}{v'_0}, \ldots, \frac{v_n}{v'_n}, \frac{v_{n+1}}{v'_{n+1}}]$. For any path p and variable $w \in \mathsf{vars}(\mathcal{F})$, if w is not blocked and w is an R-successor of $\mathsf{tail}(p)$, then $[p \mid \frac{w}{w}]$ is an R-*step *of p. If w' is blocked by w and w' is an R-successor of $\mathsf{tail}(p)$, then $[p \mid \frac{w}{w'}]$ is an R-step of p.*

Given a completion forest $\mathcal{F}$, the set $\mathsf{paths}(\mathcal{F})$ is defined inductively as follows:

- *If a is a root in $\mathcal{F}$, $[\frac{a}{a}] \in \mathsf{paths}(\mathcal{F})$.*
- *If $p \in \mathsf{paths}(\mathcal{F})$ and q is a step of p, then $q \in \mathsf{paths}(\mathcal{F})$.*

The tableau $T_{\mathcal{F}} = (\mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{I})$ induced by the completion forest $\mathcal{F}$ is defined as follows:

$$\begin{aligned} \mathbf{S} &= \mathsf{paths}(\mathcal{F}) \setminus \{p \mid p \in \mathsf{paths}(\mathcal{F}) \textit{ and } p = [\tfrac{v}{v}] \textit{ for some } v \textit{ with } \mathcal{L}(v) = \emptyset\} \\ \mathcal{L}(p) &= \mathcal{L}(\mathsf{tail}(p)) \\ \mathcal{E}(R) &= \{\langle p, q\rangle \in \mathbf{S} \times \mathbf{S} \mid q \textit{ is an } R\textit{-step of } p\} \cup \\ &\quad \{\langle p, q\rangle \in \mathbf{S} \times \mathbf{S} \mid p \textit{ is an } \mathsf{Inv}(R)\textit{-step of } q\} \cup \\ &\quad \{\langle [\tfrac{a}{a}], [\tfrac{b}{b}]\rangle \in \mathbf{S} \times \mathbf{S} \mid a, b \textit{ are root nodes and } a \textit{ is an } R\textit{-neighbor of } b\} \end{aligned}$$

Lemma 5. *Every $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ for $n \geq 1$ induces a canonical model $\mathcal{I}_\mathcal{F}$.*

Proof. First, it is proved as in [14] that every $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ for $n \geq 1$ induces a tableau $T_\mathcal{F}$ for K. For the last item of the proof of $(P9)$, note that since $n \geq 1$, pairwise blocking is subsumed and the existence the u predecessor can be ensured. $(P15)$ also holds due to the following facts:

- All nodes v are initialized with $\mathsf{gcon}(K, \mathcal{C}_q) \subseteq \mathcal{L}(v)$.
- The concept names in $\mathsf{gcon}(K, \mathcal{C}_q)$ are never removed from the label of a node unless the label is set to $\emptyset$ by the $\leq_r$-rule. In this case, the label of the node is never modified again.

Since $T_\mathcal{F}$ is a tableau for K, it has a canonical model $\mathcal{I}_\mathcal{F}$ that is a model of K.

Now we will prove that, for a sufficiently large n, if Q is entailed by the canonical model of an n-complete and clash free forest, then a mapping of the variables in Q into the forest itself can be achieved. This reveals that, semantically, this canonical model suffices to check query entailment.

In this proof, the blocking parameter n will be crucial. As we mentioned, it depends on Q. More specifically, it depends in what we call *maximal Q-distance*. If the canonical model of a forest $\mathcal{F}$ entails Q, then there is a mapping σ of the variables in Q into the nodes of the tableau induced by $\mathcal{F}$. Intuitively, the *maximal Q-distance* is the length of the longest path between two connected nodes of the graph G defined by the image of the query under σ. For a maximal Q-distance of d, a d-complete completion forest will be large enough to find a mapping whose image is isomorphic to G, since G has no paths longer than d.

We show that from any mapping σ of the variables and constants in Q into $\mathcal{I}_\mathcal{F}$ satisfying Q, a mapping μ of Q into $\mathcal{F}$ can be obtained. Formally, for a given forest $\mathcal{F}$ in $\mathsf{ccf}(\mathbb{F}_K^n)$ for some n, let $T_\mathcal{F} = \langle \mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{I} \rangle$ denote its tableau and $\mathcal{I}_\mathcal{F}$ the canonical interpretation of $\mathcal{F}$. If $\mathcal{I}_\mathcal{F} \models Q$, then there is a mapping $\sigma : \mathsf{varsIndivs}(Q) \to \mathbf{S}$ such that for every $R(x, y)$ in Q, $\langle \sigma(x), \sigma(y) \rangle \in \mathcal{E}(R')$ for some $R' \sqsubseteq^* R$. Consider the graph that has as nodes the set $\mathbf{S}$ of nodes of $T_\mathcal{F}$, and as edges $\mathcal{E}(R')$ for each role R' such that $R' \sqsubseteq^* R$ and R occurs in Q. Consider the image of Q under σ in this graph. We restrict it to the subgraph G obtained by removing each node of the form $[\frac{a}{a}]$ for some individual a(with its corresponding incoming and outgoing edges). Note that G comprises a set of tree-shaped components. The reason to consider only the subgraph G will be clear later. We want prove that there is, in the completion graph $\mathcal{F}$, a subgraph isomorphic to the image of Q into $T_\mathcal{F}$. For the arcs in the query graph involving nodes like $[\frac{a}{a}]$ for some individual a, the existence of an isomorphic arc in $\mathcal{F}$ is trivial, since the non-tree shaped part of $T_\mathcal{F}$ is isomorphic to $\mathcal{F}$. It is only in the tree-shaped parts of $T_\mathcal{F}$ that the structure was unraveled, and the mapping of the query into $T_\mathcal{F}$ may use nodes that do not explicitly exist in $\mathcal{F}$. The possibility of finding a mapping of Q into $\mathcal{F}$ from the mapping of Q into $T_\mathcal{F}$ will depend on the size and structure of the tree-shaped components of the query image.

For any x, y in $\mathsf{varsIndivs}(Q)$, $d^\sigma(x, y)$ is the length of the shortest path between $\sigma(x)$ and $\sigma(y)$ in G. If $\sigma(x)$ and $\sigma(y)$ are in two non-connected components

of G, then $d^\sigma(x, y) = 0$. Finally, the *maximal Q-distance of σ*, denoted d^σ_Q, is the maximal $d^\sigma(x, y)$ for all x, y in $\mathsf{varsIndivs}(Q)$.

In the following, let n_Q denote the number of role atoms in Q. Since only simple roles occur in the query, every pair of variables x, y in $\mathsf{varsIndivs}(Q)$ that occur in some $R(x, y) \in Q$ has $d^Q(x, y) = 1$, and thus d^σ_Q is bounded by n_Q. Due to this fact, when expanding the completion forest it is sufficient to consider n-blocking as a termination condition for any $n \geq n_Q$. Now we prove that for any such n and for any complete and clash free n-completion forest $\mathcal{F}$, if $\mathcal{I}_\mathcal{F} \models Q$, then there is a mapping $\mu : \mathsf{varsIndivs}(Q) \to \mathsf{nodes}(\mathcal{F})$ that witnesses the entailment of Q.

Proposition 3. *Consider any $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K)$ with $n \geq n_Q$, and let $\mathcal{I}_\mathcal{F}$ be the canonical model of $\mathcal{F}$. If $\mathcal{I}_\mathcal{F} \models Q$ then $Q \hookrightarrow \mathcal{F}$.*

Proof. Since $\mathcal{I}_\mathcal{F} \models Q$, then there is a $\sigma : \mathsf{varsIndivs}(Q) \to \Delta^{\mathcal{I}_\mathcal{F}}$ s.t.

- For all $C(x)$ in Q, $\sigma(x) \in C^{\mathcal{I}_\mathcal{F}}$.
- For all $R(x, y)$ in Q, $\langle \sigma(x), \sigma(y) \rangle \in R^{\mathcal{I}_\mathcal{F}}$.

Since $\Delta^{\mathcal{I}_\mathcal{F}} = V_{T_\mathcal{F}}$, $\sigma(x)$ and $\sigma(y)$ are nodes in $T_\mathcal{F}$ and correspond to paths in $\mathcal{F}$. By the definition of $\mathcal{I}_\mathcal{F}$, the mapping σ satisfies that for all $C(x)$ in Q, $C \in \mathcal{L}(\sigma(x))$ and for all $R(x, y)$ in Q, $\langle \sigma(x), \sigma(y) \rangle \in \mathcal{E}(R')$ for some $R' \sqsubseteq^* R$.

We will define a new mapping $\mu : \mathsf{varsIndivs}(Q) \to \mathsf{nodes}(\mathcal{F})$. In order to define μ, we will use again the graph G, given by the image of Q under σ (on the graph $T_\mathcal{F}$ restricted to the roles occurring in the query), and we will restrict it to have only the images of the variables in $\mathsf{vars}(Q)$ as nodes. This graph consists of a set of tree-shaped components $G_1, \ldots, G_k$. For each connected component G_i, let $\mathsf{nodes}(G_i)$ denote the set of nodes of G_i. We define the set $\mathsf{blockedLeaves}(G_i)$ as the set containing each node p of G_i such that $\mathsf{tail}(p) \neq \mathsf{tail}'(p)$, and for every ancestor p' of p in G_i, $\mathsf{tail}(p') = \mathsf{tail}'(p')$. The set $\mathsf{afterblocked}(G_i)$ contains all the nodes in $\mathsf{nodes}(G_i)$ that are descendants of some node in $\mathsf{blockedLeaves}(G_i)$.

Recalling the definition of $\mathsf{paths}(\mathcal{F})$, since $\mathcal{F}$ is n-blocked, it is easy to see that if a path p contains two nodes $\frac{v}{v'}$ and $\frac{w}{w'}$ such that $v \neq v'$ and $w \neq w'$, then the distance between these two nodes in p is strictly greater than n. Also, if p contains first a node $\frac{v}{v}$ that is not tree blocked and further on p there is a node $\frac{w}{w'}$ such that $w \neq w'$, then the distance between $\frac{v}{v}$ and $\frac{w}{w'}$ is also greater than n. Thus the following also hold:

(*) If x is in $\mathsf{afterblocked}(G_i)$ for some G_i, $\mathsf{tail}(\sigma(x)) = \mathsf{tail}'(\sigma(x))$.
If $x \in \mathsf{afterblocked}(G_i)$ then, by definition, $\sigma(x)$ is mapped to a successor of some $y \in \mathsf{vars}(Q)$ such that $\mathsf{tail}(\sigma(y)) \neq \mathsf{tail}'(\sigma(y))$, i.e., $\sigma(x)$ is of the form $[p \mid \frac{v_0}{v_0'}, \ldots, \frac{v_m}{v_m'}]$, with $\mathsf{tail}(p) \neq \mathsf{tail}'(p)$. Therefore, if $v_m \neq v_m{}'$ then the sequence of nodes $\frac{\mathsf{tail}(p)}{\mathsf{tail}'(p)}, \frac{v_0}{v_0'}, \ldots, \frac{v_m}{v_m'}$ has a length strictly greater than n, and thus $d^\sigma(x, y) > n_Q$, which is a contradiction.

(**) If $\sigma(x) \in \mathsf{nodes}(G_i)$ for some G_i with $\mathsf{afterblocked}(G_i) \neq \emptyset$ and $x \notin \mathsf{afterblocked}(G_i)$, then $\mathsf{tail}'(\sigma(x))$ is tree-blocked by $\psi(\mathsf{tail}'(\sigma(x)))$.
If $\mathsf{afterblocked}(G_i) \neq \emptyset$, then there is some $y \in \mathsf{varsIndivs}(Q)$ such that

$\sigma(y) \in \mathsf{nodes}(G_i)$ has as a proper subpath some p such that $\mathsf{tail}(p) = \mathsf{tail}'(p)$. Since $\sigma(x)$ and $\sigma(y)$ are in the same tree component G_i, then either $\sigma(x)$ is an ancestor of $\sigma(y)$ or there is some $z \in \mathsf{varsIndivs}(Q)$ such that $\sigma(z)$ is a common ancestor of $\sigma(x)$ and $\sigma(y)$ in $\mathsf{nodes}(G_i)$. In the first case, if $\mathsf{tail}'(\sigma(x))$ was not tree-blocked, we would have that $d^{\sigma}(x, y) > n \geq n_Q$, which is a contradiction. In the second case, if $\mathsf{tail}'(\sigma(x))$ was not tree-blocked, then $\mathsf{tail}'(\sigma(z))$ would not be tree-blocked either, and thus we also derive a contradiction since $d^{\sigma}(z, y) > n \geq n_Q$.

Therefore, we can define the mapping $\mu : \mathsf{varsIndivs}(Q) \rightarrow \mathsf{nodes}(\mathcal{F})$ as follows:

- For each individual a in $\mathsf{varsIndivs}(Q)$, $\mu(a) = \mathsf{tail}(\sigma(a)) = a$.
- For each variable x in $\mathsf{varsIndivs}(Q)$ such that $\sigma(x) \in \mathsf{nodes}(G_i)$ for some G_i with $\mathsf{afterblocked}(G_i) = \emptyset$, $\mu(x) = \mathsf{tail}(\sigma(x))$.
- For each variable x in $\mathsf{varsIndivs}(Q)$ such that $\sigma(x) \in \mathsf{nodes}(G_i)$ for some G_i with $\mathsf{afterblocked}(G_i) \neq \emptyset$, the mapping μ is given by:

$$\mu(x) = \begin{cases} \mathsf{tail}'(\sigma(x)) & \text{if } x \in \mathsf{afterblocked}(G_i) \\ \psi(\mathsf{tail}'(\sigma(x))) & \text{otherwise} \end{cases}$$

Now we will show that the mapping μ has the following properties:

1. If $C \in \mathcal{L}(\sigma(x))$, then $C \in \mathcal{L}((\mu(x)))$.
2. If $\langle \sigma(x), \sigma(y) \rangle \in \mathcal{E}(R')$, then $\mu(y)$ is an R'-neighbor of $\mu(x)$.

The proof of 1 is trivial, since $\mathcal{L}(\sigma(x)) = \mathcal{L}(\mathsf{tail}'(\sigma(x))) = \mathcal{L}(\psi(\mathsf{tail}'(\sigma(x))))$, so $\mathcal{L}(\sigma(x)) = \mathcal{L}(\mu(x))$.

The proof of 2 is slightly more involved. By the definition of $\mathcal{E}(R')$ and of R'-step, if $\langle \sigma(x), \sigma(y) \rangle \in \mathcal{E}(R')$ then either:
(i) $\mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mathsf{tail}(\sigma(x))$ or
(ii) $\mathsf{tail}'(\sigma(x))$ is an $\mathsf{Inv}(R')$-successor of $\mathsf{tail}(\sigma(y))$.

Now we will prove that (i) implies that $\mu(y)$ is an R'-successor of $\mu(x)$. The same proof shows that (ii) implies that $\mu(x)$ is an $\mathsf{Inv}(R')$-successor of $\mu(y)$. Together, this two facts complete the proof of 2.

We will consider each connected component G_i. The case when $\mathsf{afterblocked}(G_i) = \emptyset$ is trivial. In this case, for each x in $\mathsf{varsIndivs}(Q)$ such that $\sigma(x) \in G_i$, $\mathsf{tail}(\sigma(x)) = \mathsf{tail}'(\sigma(x))$ (in fact, $\sigma(x)$ does not contain any $\frac{v}{v'}$ with $v \neq v'$), so if $\mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mathsf{tail}(\sigma(x))$, then $\mu(y) = \mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mu(x) = \mathsf{tail}'(\sigma(x)) = \mathsf{tail}(\sigma(x))$. To do the proof for any G_i with $\mathsf{afterblocked}(G_i) \neq \emptyset$, we will proceed by cases. Note that since $\sigma(y)$ is an R'-step of $\sigma(x)$, it can not be the case that x is in $\mathsf{afterblocked}(G_i)$ and y is not, thus we have the following cases:

(a) Both x and y are in $\mathsf{afterblocked}(G_i)$.
In this case we have that $\mu(x) = \mathsf{tail}'(\sigma(x))$ and $\mu(y) = \mathsf{tail}'(\sigma(y))$, and by $(*)$, $\mathsf{tail}(\sigma(x)) = \mathsf{tail}'(\sigma(x))$. If $\mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mathsf{tail}(\sigma(x))$, then $\mu(y) = \mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mu(x) = \mathsf{tail}'(\sigma(x)) = \mathsf{tail}(\sigma(x))$ as desired.

(b) Neither x nor y is in $\mathsf{afterblocked}(G_i)$.
Note that in this case $\mathsf{tail}(\sigma(x)) = \mathsf{tail}'(\sigma(x))$, otherwise y would be in $\mathsf{afterblocked}(G_i)$. By **(**)**, we know that $\mathsf{tail}'(\sigma(x))$ is tree-blocked by $\psi(\mathsf{tail}'(\sigma(x)))$ and $\mathsf{tail}'(\sigma(y))$ is tree-blocked by $\psi(\mathsf{tail}'(\sigma(y)))$. Thus, if $\mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mathsf{tail}(\sigma(x)) = \mathsf{tail}'(\sigma(x))$, then $\mu(y) = \psi(\mathsf{tail}'(\sigma(y)))$ is an R'-successor of $\mu(x) = \psi(\mathsf{tail}'(\sigma(x)))$ as desired.

(c) x is not in $\mathsf{afterblocked}(Q)$, but y is.
In this case we have that $\mathsf{tail}(\sigma(x)) \neq \mathsf{tail}'(\sigma(x))$ and $\mathsf{tail}(\sigma(x)) = \psi(\mathsf{tail}'(\sigma(x)))$(i.e., $\sigma(x)$ ends in a blocked leaf), so if $\mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mathsf{tail}(\sigma(x))$ then $\mu(y) = \mathsf{tail}'(\sigma(y))$ is an R'-successor of $\mu(x) = \psi(\mathsf{tail}'(\sigma(x)))$.

Since the mapping μ has properties 1 and 2, $Q \hookrightarrow \mathcal{F}$.

Summing up, to solve the conjunctive query entailment problem, it suffices to check for entailment the set of complete and clash free completion forests for K, no matter the n that is used as a termination condition. However, if we choose a suitable n-blocking, checking for entailment in all the models of a completion forest can be done through one single canonical model, and this is achieved by mapping the query into the completion forest itself.

Theorem 1. *Let Q be a CQ. $K \models Q$ iff $Q \hookrightarrow \mathcal{F}$ for every $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}^n_K)$, $n \geq n_Q$.*

Proof. First we prove that if $K \models Q$ then $Q \hookrightarrow \mathcal{F}$. Take any arbitrary $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}^{n_Q}_K)$. Since $K \models Q$, then $\mathcal{F} \models Q$ (Proposition 2). In particular, we have that $\mathcal{I}_\mathcal{F} \models Q$, where $\mathcal{I}_\mathcal{F}$ is the canonical model of the tableau induced by $\mathcal{F}$. Thus, by Proposition 3, $Q \hookrightarrow \mathcal{F}$.

To prove the other direction, observe that from $Q \hookrightarrow \mathcal{F}$ and Lemma 3, we have that $\mathcal{F} \models Q$ for every $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}^{n_Q}_K)$. Finally, by Proposition 2, $K \models Q$.

Example 8. $K \models Q_1$, so $\mathcal{F}_1 \models Q_1$ must hold. This is witnessed by the mapping $Q_1 \hookrightarrow \mathcal{F}_1$ in Example 7. Note that there are longer queries, like $Q' = \{P_1(a, x_0)$, $P_1(x_0, x_1)$, $P_1(x_1, x_2)$, $P_1(x_2, x_3)$, $P_1(x_3, x_4)\}$ such that $K \models Q'$ holds, but the entailment $\mathcal{F}_1 \models Q'$ cannot be verified by mapping Q' into $\mathcal{F}_1$ since $\mathcal{F}_1$ is 1-blocked and $n_{Q'} > 1$.

4.3 Answering Unions of Conjunctive Queries

The results given above can be extended straightforwardly to an UCQ U. As usual, we will use $\mathcal{F} \models U$ to denote that $\mathcal{F}$ semantically entails U (i.e., every model of $\mathcal{F}$ is a model of U), and $U \hookrightarrow \mathcal{F}$ to denote syntactical mappability, which is defined as $Q_i \hookrightarrow \mathcal{F}$ for some Q_i in U. We already know that to decide $K \models U$ it suffices to verify whether $\mathcal{F} \models U$ for every $\mathcal{F}$ in $\mathsf{ccf}(\mathbb{F}^n_K)$ for any arbitrary $n \geq 0$ (in fact, Proposition 2 holds for any kind of query). It is only left to prove that for a suitable n, $\mathcal{F} \models U$ can be effectively reduced to $U \hookrightarrow \mathcal{F}$. For an UCQ U, we will denote by n_U the maximal n_{Q_i} for all Q_i in U. We can then prove the following result:

Proposition 4. *Consider any $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K)$ with $n \geq n_U$. Then $\mathcal{F} \models U$ iff $U \hookrightarrow \mathcal{F}$.*

Proof. Again, one direction is trivial. If $U \hookrightarrow \mathcal{F}$, then by definition, there is some Q_i in U such that $Q_i \hookrightarrow \mathcal{F}$, and as this implies $\mathcal{F} \models Q_i$, then we also have that $\mathcal{F} \models U$. The other direction is also quite straightforward. For each $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$, with $n \geq n_U$, if $\mathcal{I}_\mathcal{F} \models U$, then $\mathcal{I}_\mathcal{F} \models Q_i$ for some Q_i in U. As $n \geq n_U \geq n_{Q_i}$, by Proposition 3 we know that $Q_i \hookrightarrow \mathcal{F}$ and then $U \hookrightarrow \mathcal{F}$. Thus $\mathcal{F} \models U$ implies $U \hookrightarrow \mathcal{F}$ as well.

Example 9. By the mapping $Q_1 \hookrightarrow \mathcal{F}_1$ in Example 7, we have $U \hookrightarrow \mathcal{F}_1$. $\mathcal{F}_1 \models Q_1$ implies $\mathcal{F}_1 \models U$.

Finally, we establish our key result: answering $K \models U$ for an UCQ U reduces to finding a mapping of U into every $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ for any $n \geq n_U$.

Theorem 2. *Let U be an UCQ. $K \models U$ iff $U \hookrightarrow \mathcal{F}$ for every $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$, $n \geq n_U$.*

Proof. As in the proof of Theorem 1, it follows Proposition 2 and Proposition 4.

5 Complexity

In this section, for a knowledge base K, we will use $\mathbf{c}$ to denote the cardinality of $\mathsf{clos}(K) \cup \mathcal{C}_q$, $\mathbf{r}$ the cardinality of $\mathbf{R}_K$ and $m_\mathbf{C}$ the maximum m occurring in a concept of the form $\leq m\, R.C$ or $\geq m\, R.C$ in K. $|\mathcal{A}|$ denotes the number of assertions in $\mathcal{A}$. By $||K, Q||$ we will denote the total size of (the string encoding) the knowledge base K and the query Q.

Note that $m_\mathbf{C}$ is linear in $||K, Q||$ if we assume unary coding of numbers in number restrictions, and single exponential if binary coding is used. In any case, if we consider fixed Q and all of K except for $\mathcal{A}$, then $m_\mathbf{C}$ is a constant. Furthermore, $\mathbf{c}$ and $\mathbf{r}$ are linear in $||K, Q||$, but also constant in $|\mathcal{A}|$. Finally, $|\mathbf{I}_K|$ is linear in both.

Lemma 6. *The maximal number T_n of non-isomorphic n-trees in a completion forest for K is given by $T_n = \mathbf{O}((2^{2\mathbf{c}}(\mathbf{c}m_\mathbf{C})^\mathbf{r})^{(\mathbf{c}m_\mathbf{C}\mathbf{r})^n})$.*

Proof. Since $\mathcal{L}(x) \subseteq \mathsf{clos}(K) \cup \mathcal{C}_q$, there are at most $2^\mathbf{c}$ different node labels in a completion forest. Each successor of a node can be the root of a tree of depth $(n-1)$. considering a single role R, if a node v has x R-successors, then there is a maximum number of $(T_{n-1})^x$ trees of depth $(n-1)$ rooted at v. A generating rule can be applied to each node at most $\mathbf{c}$ times. Each time it is applied, it generates at most $m_\mathbf{C}$ R-successors for each role R. This gives a bound of $\mathbf{c}m_\mathbf{C}$ R-successors for each role. The number of R-successors of a node might range from 0 to $\mathbf{c}m_\mathbf{C}$, and for each number of R-successors, we have at most $(T_{n-1})^{(\mathbf{c}m_\mathbf{C})}$ trees of depth $(n-1)$. So, each node can be the root of at most $(\mathbf{c}m_\mathbf{C})(T_{n-1})^{(\mathbf{c}m_\mathbf{C})}$ trees of depth $(n-1)$ if we consider one single role.

Since at most the same number of trees can be generated for every role in $\mathbf{R}_K$, there is a bound of $((\mathbf{c}m_{\mathbf{C}})(T_{n-1})^{(\mathbf{c}m_{\mathbf{C}})})^{\mathbf{r}}$ trees of depth $(n-1)$ rooted at each node. The number of different roots of an n-tree is bounded by $2^{\mathbf{c}}$. We now give an upper bound on the number of non isomorphic n-trees as

$$T_n = \mathbf{O}(2^{\mathbf{c}}((\mathbf{c}m_{\mathbf{C}})(T_{n-1})^{(\mathbf{c}m_{\mathbf{C}})})^{\mathbf{r}})$$

To simplify the notation, let's consider $x = 2^{\mathbf{c}}(\mathbf{c}m_{\mathbf{C}})^{\mathbf{r}}$ and $a = \mathbf{c}m_{\mathbf{C}}\mathbf{r}$. Then we have

$$T_n = \mathbf{O}(x(T_{n-1})^a) = \mathbf{O}(x^{1+a+\ldots+a^{n-1}}(T_0)^{a^n}) = \mathbf{O}((xT_0)^{a^n})$$

The maximal number of trees of depth 0 is also bounded by $2^{\mathbf{c}}$. Returning to the original notation we get

$$T_n = \mathbf{O}((2^{2\mathbf{c}}(\mathbf{c}m_{\mathbf{C}})^{\mathbf{r}})^{(\mathbf{c}m_{\mathbf{C}}\mathbf{r})^n})$$

Lemma 7. *The number of nodes in a completion forest $\mathcal{F} \in \mathbb{F}_K^n$ is bounded by*

$$\mathbf{O}(|\mathbf{I}_K|(\mathbf{c}m_{\mathbf{C}}\mathbf{r})^{n(2^{2\mathbf{c}}(\mathbf{c}m_{\mathbf{C}})^{\mathbf{r}})^{(\mathbf{c}m_{\mathbf{C}}\mathbf{r})^n}})$$

Proof. The claim follows from the following properties:

i) The outdegree of $\mathcal{F}$ is bounded by $\mathbf{c}m_{\mathbf{C}}\mathbf{r}$.
Nodes are only added to the forest by applying a generating rule. Only concepts of the form $\exists R.S$ or $\geq n\, R.C$ trigger the application of a generating rule, and there are at most $\mathbf{c}$ such concepts. Each such rule generates at most $m_{\mathbf{C}}$ successors for each role, and there are $\mathbf{r}$ roles. Note that if a node v is identified with another by the $\forall$-rule or the $\forall_r$-rule, then the rule application which led to the generation of v will never be repeated [14].

ii) The depth of $\mathcal{F}$ is bounded by $d = (T_n + 1)n$.
This is due to the fact that there is a maximum of T_n non-isomorphic n-trees. If there was a path of length greater than $(T_n+1)n$ to a node v in $\mathcal{F}$, this would imply that v occurred after a sequence of $T_n + 1$ non overlapping n-trees, and then one of them would have been blocked and v would not have been generated.

iii) The number of variables in a variable tree in $\mathcal{F}$ is bounded by $\mathbf{O}((\mathbf{c}m_{\mathbf{C}}\mathbf{r})^{d+1})$.

iv) The number of variables in $\mathcal{F}$ is bounded by $\mathbf{O}(|\mathbf{I}_K|(\mathbf{c}m_{\mathbf{C}}\mathbf{r})^{d+1})$.

Corollary 1. *If n is polynomial on $||K, Q||$, then the maximum number of nodes in a completion forest $\mathcal{F} \in \mathbb{F}_K^n$ is triple exponential in $||K, Q||$.*

Corollary 2. *If Q and all of K except for $\mathcal{A}$ are fixed and n is a constant, then the maximum number of nodes in a completion forest $\mathcal{F} \in \mathbb{F}_K^n$ is linear in $|\mathcal{A}|$.*

Proposition 5. *The expansion of $\mathcal{F}_K$ into some $\mathcal{F} \in \mathbb{F}_K^n$ terminates in time triple exponential in $||K, Q||$ if n is polynomial on $||K, Q||$. If Q and all of K except for $\mathcal{A}$ are fixed, and n is a constant, then the expansion of $\mathcal{F}_K$ into some $\mathcal{F} \in \mathbb{F}_K^n$ terminates in time polynomial in $|\mathcal{A}|$.*

Proof. The proposition is a consequence of the fact that the number of times that each expansion rule can be applied to a node is polynomially bounded. Let $M = \mathbf{O}(|\mathbf{I}_K|(\mathbf{c}m_{\mathbf{C}}\mathbf{r})^{n(2^{2\mathbf{c}}(\mathbf{c}m_{\mathbf{C}})^{\mathbf{r}})^{(\mathbf{c}m_{\mathbf{C}}\mathbf{r})^n}})$ denote the maximal number of nodes in $\mathcal{F}$. We will obtain an upper bound of the number of rules that are applied to expand $\mathcal{F}_K$ into $\mathcal{F}$.

i) For a single node v, the $\sqcap$-rule, the $\sqcup$-rule and the choose-rule can be applied $\mathbf{O}(\mathbf{c})$ times, since they are applied at most once for each concept in $\mathcal{L}(v)$.
ii) For the $\exists$-rule, $\forall$-rule, $\forall_+$-rule, $\geq$-rule and $\leq$-rule, the bound on the number of times it can be applied to v is given by the maximal number of successors of v, i.e. $\mathbf{O}(\mathbf{c}m_{\mathbf{C}}\mathbf{r})$.
iii) Rules 1 to 8 can be applied at most $\mathbf{O}(M\mathbf{c}m_{\mathbf{C}}\mathbf{r})$ times to obtain $\mathcal{F}$.
iv) The $\leq_r$-rule can be applied at most once to each root node in $\mathcal{F}_K$, hence it is bounded by $|\mathbf{I}_K|$.
v) The total rule applications required to expand $\mathcal{F}_K$ into $\mathcal{F}$ is $\mathbf{O}(|\mathbf{I}_K| + (M\mathbf{c}m_{\mathbf{C}}\mathbf{r}))$

5.1 Complexity of answering UCQs

For a CQ Q_i, the problem of finding a mapping $Q_i \hookrightarrow \mathcal{F}$ has the same query complexity as evaluating a conjunctive query over a database (given by the ABox), so it is an NP-hard problem, even for fixed $\mathcal{F}$. To verify this, it suffices to consider the forest $\mathcal{F}_{col}$ associated to the ABox:

$$\begin{array}{lll} E(red, green) & E(green, red) & E(red, blue) \\ E(blue, red) & E(green, blue) & E(blue, green) \end{array}$$

Any (directed) graph G can be encoded as a conjunctive query Q_i: nodes in G are variables in Q_i and for each arc $\langle x, y\rangle$ in G there is a literal $E(x, y)$ in Q_i. Then Q_i can be mapped into $\mathcal{F}_{col}$ iff G is 3-colorable.

However, the test $Q_i \hookrightarrow \mathcal{F}$ might be done in time polynomial in the size of $\mathcal{F}$ when Q_i is fixed, or when the expansion rules generate a big enough completion forest, such that its size exponentially dominates the size of the query. Other particular cases can be solved in polynomial time as well. For example, when $\mathcal{F}$ is tree shaped (i.e., the Abox is tree shaped), then the complexity of the mapping corresponds to evaluating a conjunctive query over a tree-shaped database, which is known proved to be polynomial in the size of the database.

For the general case, to check whether $Q_i \hookrightarrow \mathcal{F}$ can be done by naive methods in time single exponential in the size of Q_i. For an $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ with M nodes and a query Q_i with n_{Q_i} literals, the naive search space has $M^{2n_{Q_i}}$ candidate assignments, and each one can be polynomially checked. So, if M is triple exponentially bounded in $||K, Q_i||$, then also $M^{2n_{Q_i}}$ is triple exponentially bounded in $||K, Q_i||$.

Therefore, we obtain the following result:

Theorem 3. *Given a $\mathcal{SHIQ}$ knowledge base K and a union of conjunctive queries U in which all roles are simple, deciding whether $K \models U$ is in* co-3NExpTime *(w.r.t. combined complexity).*

Proof. It is sufficient to check for every $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^{n_U})$ whether $U \hookrightarrow \mathcal{F}$, i.e., $Q_i \hookrightarrow \mathcal{F}$ for some CQ Q_i in U. By Proposition 5, the size of $\mathcal{F}$ is at most triple exponential in $\|K, Q_{i^*}\|$, where Q_{i^*} is such that $n_{Q_{i^*}} = n_U$, thus also in $\|K, U\|$. Furthermore, $Q_i \hookrightarrow \mathcal{F}$ can be checked by naive methods in triple exponential time in $\|K, Q_{i^*}\|$ and thus in $\|K, U\|$ as well.

Notice that since $m_{\mathbf{C}}$ does not occur in the uppermost exponent of the bound of the forest size, these results hold both for unary and binary encoding of number restrictions in K. For a fixed U an exponential drop in the forest size can be achieved if unary encoding of numbers is used.

The complexity upper bounds given in [18] are exponentially lower than the ones we obtained. This is due to the fact that in [18] they do not require the witness of a blocked variable to be its ancestor, hence blocking occurs sooner and their constraints systems are exponentially smaller than our completion forests. In order to make the blocking conditions of our algorithm closer to the conventional ones in DL tableaux algorithms, we do require the blocking and the blocked variable to be on the same path. This restriction, however, could be eliminated, and blocking with any previous occurrence of an isomorphic n-tree could be used, without affecting the soundness and completeness of the algorithm. This would indeed bring an exponential drop in the complexity upperbound. It is worth mentioning that the same more relaxed blocking technique could be used in satisfiability tableaux algorithms(like in [14]) obtaining the same exponential drop, but it may not be convenient from an implementation perspective.

By the results in [16] we know that an 2ExpTime bound can be achieved. This bound coincides with the one given in [6] for containment of conjunctive queries over $\mathcal{DLR}$. As we mentioned above, our algorithm can be improved to get co-2NExpTime worst-case complexity in the combined case. Note that the the tableaux algorithm from [14], which we extended, is also not worse case optimal: it solves in 2NExpTime reasoning problems that can be solved in deterministic single exponential time.

We will see now give the main complexity results of this work, namely those concerning data complexity. Moreover, we will see that the proposed algorithm is worse case optimal. Under data complexity, U and all components of $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ except for the ABox $\mathcal{A}$ are fixed. Therefore, n_U is constant. As a consequence we get:

Theorem 4. *For a knowledge base K in $\mathcal{SHIQ}$ and a union of conjunctive queries U in which all roles are simple, deciding $K \models U$ is in* coNP *w.r.t. data complexity.*

Proof. By Proposition 5, every completion forest $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K^n)$ has *linearly* many nodes in $|\mathcal{A}|$, and any expansion of $\mathcal{F}_K$ terminates in polynomial time. Furthermore, deciding whether $U \hookrightarrow \mathcal{F}$ is polynomial in the size of $\mathcal{F}$ by simple methods.

The bound given by Theorem 4 is worst-case optimal in data complexity. In [21], coNP-hardness was proved for $\mathcal{ALC}$, and later this result has been extended to description logics which are even less expressive than $\mathcal{ALE}$ [6]. This allows us to state the following main result.

Theorem 5. *On knowledge bases in any DL from $\mathcal{AL}$ to $\mathcal{SHIQ}$, answering unions of conjunctive queries in which all roles are simple is* coNP*-complete w.r.t. data complexity.*

This result not only provides an exact characterization of the data complexity of UCQs for a wide range of description logics, some of them considered highly expressive. It is interesting to see that once simple constructors like negation are allowed in the DL, many more constructors can be added without having an strong impact in the data complexity. Additionally, this result also extends two previous coNP-completeness results w.r.t. data complexity which are not obvious. In [18] the authors proved the same bound for answering UCQs over $\mathcal{ALCNR}$. Now we extend this result to role hierarchies, as well as transitive and inverse roles. There was also a previous result for ground atomic queries in $\mathcal{SHIQ}$ given in [16], and we extend it to UCQs.

6 Conclusions

In this paper we have studied conjunctive query answering in the expressive DL $\mathcal{SHIQ}$, and generalizing a technique presented in [18] for a less expressive DL, we have developed a novel tableaux-based algorithm for UCQ answering. In the algorithm, we had to manage the technical challenges caused by the simultaneous presence of inverse roles and number restrictions, leading to a DL lacking the finite model property. Our algorithm is worse case optimal in data complexity, and thus allowed us to characterize data complexity of the problem as coNP-complete for a wide range of DLs, including expressive ones. This closed the gap between the known coNP lower bound, and the best known ExpTime upper bound for even weaker DLs.

We point out that, by virtue of the correspondence between query containment and query answering [1], our algorithm can also be applied to decide containment of two conjunctive queries over a DL knowledge base. Currently, we are working on extending our results to the case where the query may contain arbitrary roles, including transitive ones. It remains open whether the proposed technique can be applied to even more expressive logics, for example, containing reflexive-transitive closure in the TBox (in the style of PDL), or to more expressive query languages, notably allowing for atoms that are regular expressions over roles. It also remains to investigate tighter bounds in combined complexity. Optimization of the algorithm following the ideas in [10] may be feasible.

References

1. S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS'98*, 1998.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

3. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1), 2001.
4. A. Borgida and R. J. Brachman. Conceptual modeling with description logics. In Baader et al. [2], chapter 10.
5. D. Calvanese and G. De Giacomo. Expressive description logics. In Baader et al. [2], chapter 5.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of DL'05*, 2005.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI'05*, 2005.
8. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, 1998.
9. D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of AAAI'00*, 2000.
10. G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Information and Computation*, 160(1–2), 2000.
11. F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Log. and Comp.*, 4(4), 1994.
12. J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2), 2001.
13. I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1), 2003.
14. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. of CADE'00*, 2000.
15. U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of LPAR'04*, 2004.
16. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI'05*, 2005.
17. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS'02*, 2002.
18. A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2), 1998.
19. P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004.
20. A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intelligent Information Systems*, 2, 1993.
21. A. Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues*. PhD thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 1994.
22. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.
23. M. Y. Vardi. The complexity of relational query languages. In *Proc. of STOC'82*, 1982.