I N F S Y S

R E S E A R C H

R E P O R T

INSTITUT FÜR INFORMATIONSSYSTEME

ARBEITSBEREICH WISSENSBASIERTE SYSTEME

DATA COMPLEXITY OF QUERY ANSWERING IN EXPRESSIVE DESCRIPTION LOGICS WITH NOMINALS

M. Magdalena Ortiz Diego Calvanese Thomas Eiter

INFSYS RESEARCH REPORT 1843-06-03

MAY 2006

Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstrassße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TU WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-06-03, MAY 2006

DATA COMPLEXITY OF QUERY ANSWERING IN EXPRESSIVE DESCRIPTION LOGICS WITH NOMINALS

M. Magdalena Ortiz,[1] Diego Calvanese,[2] and Thomas Eiter[3]

Abstract. The formal foundations of the standard web ontology languages, OWL-Lite and OWL-DL, are provided by expressive Description Logics (DLs), such as $\mathcal{SHIF}$ and $\mathcal{SHOIQ}$. In the Semantic Web and other domains, ontologies are increasingly seen also as a mechanism to access and query data repositories. This novel context poses an original combination of challenges that has not been addressed before: (*i*) sufficient expressive power of the DL to capture common data modeling constructs; (*ii*) well established and flexible query mechanisms such as those inspired by database technology; (*iii*) optimization of inference techniques with respect to data size, which typically dominates the size of ontologies. This calls for investigating data complexity of query answering in expressive DLs. While the complexity of DLs has been studied extensively, data complexity of query answering in expressive DLs has been characterized only for restricted forms of queries, and was still open for the standard query languages mutuated from databases, such a conjunctive queries (CQs) and unions of CQs. We tackle this issue and prove a tight CONP upper bound for the problem in $\mathcal{SHOIQ}$, for the case where the query does not contain transitive roles. We thus establish that for a whole range of DLs from $\mathcal{AL}$ to $\mathcal{SHOIQ}$, answering such CQs has CONP-complete data complexity. We obtain our result by a novel tableaux-based algorithm for checking query entailment, inspired by work on hybrid knowledge bases, but which manages the technical challenges of simultaneous presence of inverse roles, number restrictions (which already lead to a DL lacking the finite model property), and nominals.

Keywords: expressive description logics, query answering, data complexity, conjunctive queries, unions of conjunctive queries, tableaux algorithms.

---

[1]Faculty of Computer Science, Free University of Bozen-Bolzano, and Institute of Information Systems, Vienna University of Technology. E-mail: magdalena.ortiz@stud-inf.unibz.it.

[2]Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, I-39010 Bolzano, Italy. E-mail: calvanese@inf.unibz.it.

[3]Institute of Information Systems, Knowledge-Based Systems Group, Vienna University of Technology, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: eiter@kr.tuwien.ac.at.

Acknowledgements: This work was partially supported by the Austrian Science Funds (FWF) project P17212 and the European Commission project REWERSE (IST-2003-506779).

Some results in this paper appear in preliminary form in the Proceedings of the 21th National Conference on Artificial Intelligence *(AAAI '06)* [34] and in the Informal Proceedings of the International Workshop on Description Logics *(DL 2006)* [35].

Contents

1 Introduction

Description logics (DLs) [2] are logics specifically designed for representing structured knowledge by concepts (i.e., classes of objects) and roles (i.e., binary relationships between classes). They have been initially developed to provide a formalization of frame-based systems and semantic networks, and expressive variants of DLs were shown to be in tight correspondence with representation formalisms used in databases and software engineering [13, 5]. More recently, DLs gained increasing attention as the formal foundation for the standard Web ontology languages [20]. In fact, the most significant representatives of such languages, OWL-Lite and OWL-DL, are syntactic variants of two DLs of the well known $\mathcal{SH}$ family, namely the DLs $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$, respectively [22, 36]. In the Semantic Web and in other application domains such as Enterprise Application Integration and Data Integration [31], ontologies provide a high-level, conceptual view of the information relevant in a specific domain or managed by an organization. However, they are increasingly seen also as a mechanism to access and query data repositories, while taking into account the constraints that are inherent in the common conceptualization.

This novel context poses an original combination of challenges unmet before, both in DLs/ontologies and in related areas such as data modeling and querying in databases:

1. On the one hand, a DL should have sufficient expressive power to capture common constructs typically used in data modeling [6]. This calls for *expressive DLs* [7, 4], 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 the participation to relationships in general). Such concepts are then used in the intentional component of a knowledge base (the TBox), which contains inclusion assertions between concepts and roles, while the extensional component (the ABox) contains assertions about the membership of individuals to concepts and roles.

2. 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 that have been considered and implemented in DL-based systems, like *instance checking* [15, 37]. Indeed, since explicit variables are missing, DL concepts have limited possibility for relating specific data items to each other. *Conjunctive queries* (CQs), i.e., select-project-join SQL queries, and unions of CQs (UCQs) provide a good tradeoff between expressive power and nice computational properties, (e.g., decidability of containment), and thus are adopted as core query language in several contexts, such as data integration [31].

3. Finally, one has to take into account that data repositories can be very large and are usually much larger than the intentional level used to express constraints on the data. Therefore, the contribution of the extensional level (i.e., the data) to the complexity of inference should 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 [41], 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.

Notable examples of the expressive DLs equipped with the features discussed above are $\mathcal{SHIQ}$, which also allows for asserting the transitivity of certain roles, and $\mathcal{SHOIQ}$, which in addition provides the ability

to talk and assert properties about specific individuals in the TBox. Note that these two DLs essentially[1] correspond to the web ontology languages OWL-Lite and OWL-DL [22, 36], which have been promoted as standards by the World Wide Web Consortium within the Semantic Web effort[2].

A distinguishing feature of the web ontology language OWL-DL, and its corresponding counterpart $\mathcal{SHOIQ}$, is the inclusion of *nominals*, which are concepts denoting a single individual. In their presence, the crisp separation between TBox and ABox mentioned in item 1 above may become blurred. Indeed, the use of nominals allows one to generalize ABox assertions, by combining them within the TBox in more complex forms than simply conjunctively. Also, nominals allow for the modeling of individuals that play a prominent role at the conceptual level. The added expressive power coming from nominals, especially when combined with the other features requested from expressive DLs, unfortunately results in an increased computational complexity of inference. Indeed, while for most expressive DLs, TBox+ABox reasoning is EXPTIME-complete [7], the addition of nominals makes the problem NEXPTIME-complete [39].

As for data complexity of DLs, [15, 37] showed that instance checking is CONP-hard already in the rather weak DL $\mathcal{ALE}$, and [8] that CQ answering is CONP-hard in the yet weaker DL $\mathcal{AL}$. For suitably tailored DLs, answering UCQs over DL knowledge base is polynomial (actually LOGSPACE) in data complexity [9, 10]. In [10], the problem is studied for the *DL-Lite* family of DLs, and two DLs are identified for which the problem is LOGSPACE, and can effectively be reduced to evaluating a UCQ (i.e., a union of select-project-join SQL queries) over a database, using standard relational technology. Also, an analysis is carried out on which additions to the DL make the problem NLOGSPACE-hard, PTIME-hard, and CONP-hard. The analysis essentially shows that the two identified DLs are the maximal ones enjoying so called *FOL-reducibility* (and hence LOGSPACE data complexity) of query answering. An further interesting consequence is that seemingly minor additions to the DL (such as universal quantification, one of the construct considered as basic in DLs) make the problem already CONP-hard, and hence, as shown in our work, as hard as for the very expressive DLs that we are considering.

For expressive DLs (with the features we have mentioned above, notably inverse roles), TBox+ABox reasoning has been studied extensively using a variety of techniques ranging from reductions to reasoning in Propositional Dynamic Logic (PDL) [16] (see, e.g., [11, 7]) over tableaux [4, 26] to automata on infinite trees [7, 40]. For many such DLs, the combined complexity of TBox+ABox reasoning is EXPTIME-complete, including $\mathcal{ALCQI}$ [7, 40], $\mathcal{DLR}$ [11], and $\mathcal{SHIQ}$ [40]. However, until recently, little attention has been explicitly devoted to data complexity in expressive DLs. An EXPTIME upper bound for data complexity of CQ answering in $\mathcal{DLR}$ follows from the results on CQ containment and view-based query answering in [11, 12]. 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. Similar considerations hold for the techniques in [27], which refine and extend the ideas introduced in [11], making the resulting algorithms better suited for implementation on top of tableaux-based algorithms. In [32] 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 [28, 30] 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. The result can immediately be extended to tree shaped conjunctively queries, since these admit a representation as a description logic concept (e.g., by making use of the notion of tuple-graph of [11], or via rolling up [27]). However, this is not the case for general CQs, resulting in a non-tight 2EXPTIME upper bound (matching also combined complexity).

[1] In fact, the OWL family of languages provides also the ability to deal with datatypes, which are an important feature for applications, and whose theoretical counterpart in DLs are concrete domains [3, 33].

[2] `http://www.w3.org/2001/sw/`

Summing up, a precise characterization of data complexity for CQ 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 tableaux-based techniques of [32, 26], we devise a novel tableaux-based algorithm for CQ 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 [32], 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 CQ answering in expressive DLs. Specifically, we show that data complexity of answering CQs with no transitive roles over $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ knowledge bases is in coNP, and thus coNP-complete for all DLs ranging from $\mathcal{AL}$ to $\mathcal{SHOIQ}$.

The rest of the paper is organized as follows. After the necessary technical preliminaries in Section 2, we present in Section 3 our algorithm for answering UCQs over $\mathcal{SHIQ}$ knowledge bases. We then show, in Section 4, how the algorithm can be extended to $\mathcal{SHOIQ}$. In Section 5, we discuss the resulting complexity bounds, and in Section 6 we draw final conclusions. The proofs of two technical lemmas are included in an appendix.

2 Preliminaries

In this section we introduce the technical preliminaries used in the rest of the paper. Specifically, we first introduce syntax and semantics of the two description logics $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$, on which we base our results, and then we define the query answering problem addressed in this work.

2.1 The description logics $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$

Description logics [2] are logics specifically well-suited for the representation of structured knowledge. The basic elements of description logics are *concepts*, denoting sets of objects of the domain of interest, and *roles*, denoting binary relations between the instances of concepts. Arbitrary concept and role expressions (in the following simply called concepts and roles) are formed by starting from a set of concept names and a set of role names, and applying concept and role *constructors*. The domain of interest is then modeled through a knowledge base, which is constituted by logical assertions both at the intensional level (specifying the properties of concepts and roles), and at the extensional level (specifying the properties of individuals and the relationships among individuals).

We start with the definition of roles, which is identical for $\mathcal{SHIQ}$ and for $\mathcal{SHOIQ}$.

Definition 2.1 [$\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ roles] Let $\mathbf{R}$ be a countable set of *role names*, which we denote with P, and let $\mathbf{R}_+$ be a subset of $\mathbf{R}$ of *transitive role names*. A *role expression* R (or simply *role*) is either a role name $P \in \mathbf{R}$ or the inverse P^- of a role name P. A *role inclusion* axiom is an expression of the form $R \sqsubseteq R'$ where R and R' are roles. A *role hierarchy* $\mathcal{R}$ is a set of role inclusion axioms.

The semantics of description logics, and specifically of $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$, is defined in terms of first-order interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is constituted by a non-empty set $\Delta^{\mathcal{I}}$, the *domain*

of $\mathcal{I}$, and an *interpretation function* $\cdot^{\mathcal{I}}$ that maps each role R in $\mathbf{R}$ to a subset $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, such that the following conditions are satisfied:

$$\begin{array}{rcl} R^{\mathcal{I}} & = & (R^{\mathcal{I}})^{+} \quad \text{for each transitive role } R \in \mathbf{R}_{+} \\ (R^{-})^{\mathcal{I}} & = & \{\langle o', o\rangle \mid \langle o, o'\rangle \in R^{\mathcal{I}}\} \end{array}$$

An interpretation $\mathcal{I}$ satisfies a role inclusion axiom $R \sqsubseteq R'$ if $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$.

To avoid the need of treating differently (direct) and inverse roles, we introduce the function Inv as follows:

$$\mathsf{Inv}(R) = \begin{cases} P^{-}, & \text{if } R = P \text{ is a role-name} \\ P, & \text{if } R = P^{-} \text{ for some role name } P \end{cases}$$

The relation $\sqsubseteq^{*}_{\mathcal{R}}$ 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 $R \sqsubseteq^{*}_{\mathcal{R}} R'$, then we say that R is a *sub-role* of R' and R' is a *super-role* of R relative to $\mathcal{R}$.

We need to characterize whether a role is transitive, either because it is a role name belonging to $\mathbf{R}_{+}$, or because it is the inverse of such a role, or because the role hierarchy implies that it is both a sub-role and a super-role of a transitive role. To this aim, we define the boolean function $\mathsf{Trans}(R, \mathcal{R})$ as follows:

$$\mathsf{Trans}(R, \mathcal{R}) = \begin{cases} \mathit{true}, & \text{if } R' \in \mathbf{R}_{+} \text{ or } \mathsf{Inv}(R') \in \mathbf{R}_{+}, \\ & \quad \text{for some } R' \text{ with } R \sqsubseteq^{*}_{\mathcal{R}} R' \text{ and } R' \sqsubseteq^{*}_{\mathcal{R}} R \\ \mathit{false}, & \text{otherwise} \end{cases}$$

Finally, a role S is *simple* w.r.t. a role hierarchy $\mathcal{R}$ if it is neither transitive nor has transitive sub-roles, i.e., for no role R with $\mathsf{Trans}(R, \mathcal{R})$ we have that $R \sqsubseteq^{*}_{\mathcal{R}} S$.

In the following, when $\mathcal{R}$ is clear from the context, we may omit it, and use $\sqsubseteq^{*}$ and $\mathsf{Trans}(R)$ instead of $\sqsubseteq^{*}_{\mathcal{R}}$ and $\mathsf{Trans}(R, \mathcal{R})$, respectively. For the same reason, we also may omit the specification "w.r.t. $\mathcal{R}$".

We introduce now $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ concepts, and then knowledge bases.

Definition 2.2 [$\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ concepts] Let $\mathbf{C}$ be a countable set of *concept names*, disjoint from the set $\mathbf{R}$ of role names. $\mathcal{SHIQ}$-concepts are defined inductively according to the following syntax:

C	$\longrightarrow$	A	atomic concept
		$C \sqcap D$	conjunction
		$C \sqcup D$	disjunction
		$\neg D$	negation
		$\forall R.C$	universal quantification
		$\exists R.C$	existential quantification
		$\geq n\, S.C, \leq n\, S.C$	(qualified) number restriction

where A denotes a concept name, C and D denote concepts, R a role, S a simple role, and $n \geq 0$ an integer.

For $\mathcal{SHOIQ}$ concepts we additionally consider a set $\mathbf{N}$ of *individuals* to be used in *nominals*, i.e., in concepts denoting a single object, and we augment the above syntax rules with the following one:

$$C \longrightarrow \{o\} \qquad \text{nominal}$$

An *atomic $\mathcal{SHOIQ}$ concept* is either a nominal $\{o\} \in \mathbf{N}$ or a concept name $B \in \mathbf{C}$.

For an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, we first extend the interpretation function $\cdot^{\mathcal{I}}$ to individuals in $\mathbf{N}$, in such a way that it assigns to each individual $o \in \mathbf{N}$ an element $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ under the *unique name assumption*, i.e., if $o_1 \neq o_2$, then $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$. We then extend $\cdot^{\mathcal{I}}$ to $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ concepts by assigning to each concept (including nominals, for $\mathcal{SHOIQ}$) a subset of $\Delta^{\mathcal{I}}$ in such a way that the following conditions are satisfied:

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

For $\mathcal{SHOIQ}$, note that the interpretation of each nominal $\{o\}$ is a singleton. Notice that the term *nominal* typically refers to concepts that have to be interpreted as singletons, which we denote $\{o\}$. However, with some abuse of terminology, we use the term nominal also to denote the individual o, which may appear outside of concept expressions (in an ABox, see below).

In description logics, the knowledge about the domain of interest is encoded in a knowledge base, which is constituted by an intensional component, called TBox, representing general knowledge about the domain, and an extensional component, called ABox, representing knowledge about specific objects. Additionally, in $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ (as in the other description logics of the $\mathcal{SH}$ family), a role hierarchy might be present.

Definition 2.3 [$\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ knowledge base] A ($\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$) *concept inclusion axiom* is an expression of the form $C \sqsubseteq D$ for two ($\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$) concepts C and D. A ($\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$) *TBox*, or terminology, $\mathcal{T}$, is a finite set of ($\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$) concept inclusion axioms.

Let $\mathbf{I}$ be a set of *individuals*, disjoint from the set $\mathbf{C}$ of concept names and from the set of $\mathbf{R}$ role names. Instead, we consider nominals as individuals, i.e., $\mathbf{N} \subseteq \mathbf{I}$. An *assertion* α is an expression of the form $A(a)$, $P(a, b)$ or $a \not\approx b$, where A is a concept name, P is a role name and a, b are individuals in $\mathbf{I}$. An *ABox* $\mathcal{A}$ is a set of assertions.

A ($\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$) knowledge base is a triple $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a ($\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$) terminology, $\mathcal{R}$ is a role hierarchy, and $\mathcal{A}$ is an ABox.

Without loss of expressivity, we assume that all concepts in K are in *negation normal form* (NNF), i.e., negation appears only in front of atomic concepts. The *closure* of a concept C, $\mathsf{clos}(C)$, is the smallest set of concept expressions containing C that is closed under subconcepts and their negation (expressed in NNF). The *closure of* K is denoted $\mathsf{clos}(K)$ and defined as the union of all $\mathsf{clos}(C)$ for each C occurring in K. We will denote by $\mathbf{R}_K$ the roles occurring in K and their inverses. The individuals occurring in $\mathcal{A}$ are denoted $\mathbf{I}_{\mathcal{A}}$, $\mathbf{I}_K$ denotes all the individuals occurring in K, and $\mathbf{N}_K$ denotes the nominals in K, i.e., $\mathbf{I}_K \cap \mathbf{N}$. Note that, if K is a $\mathcal{SHIQ}$ knowledge base, then $\mathbf{I}_K = \mathbf{I}_{\mathcal{A}}$. For a $\mathcal{SHOIQ}$ knowledge base K we have that $\mathbf{I}_{\mathcal{A}} \subseteq \mathbf{I}_K$ and $\mathbf{I}_K \setminus \mathbf{I}_{\mathcal{A}} \subseteq \mathbf{N}_K$.

Example 2.4 As a running example, we use the $\mathcal{SHIQ}$ knowledge base

$$K_1 = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\neg A\},\ \{\},\ \{A(a)\} \rangle$$

and the $\mathcal{SHOIQ}$ knowledge base

$$K_2 = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\{o\}\},\ \{\},\ \{A(a)\}\rangle.$$ ∎

We now define the semantics of knowledge bases. To do so, the interpretation function $\cdot^{\mathcal{I}}$ is extended to all individuals in $\mathbf{I}$, again under the unique name assumption.

Definition 2.5 [Model of a knowledge base] An interpretation $\mathcal{I}$ satisfies an assertion α if and only if:

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

An interpretation $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ 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 $\mathcal{A}$.

We note that no complex concepts and roles are allowed to occur in an ABox. However, this is not a limitation, since an assertion $C(a)$ involving a complex concept can always be replaced by an assertion $A(a)$ in the ABox, together with a pair of inclusion assertions $A \sqsubseteq C$ and $C \sqsubseteq A$, where A is a new concept name. Such a transformation is satisfiability preserving.

Finally, we observe that in $\mathcal{SHOIQ}$, due to the presence of nominals, an ABox $\mathcal{A}$ in a knowledge base $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$ can be *internalized* in the TBox, obtaining a knowledge base $K' = \langle \mathcal{T}', \mathcal{R}, \{\}\rangle$ with an empty ABox. Indeed, $\mathcal{T}'$ is obtained from $\mathcal{T}$ by adding, for each ABox assertion α in $\mathcal{A}$ a TBox inclusion axiom as follows:

- if α is of the form $A(a)$, then add $\{a\} \sqsubseteq A$ to $\mathcal{T}'$;
- if α is of the form $P(a,b)$, then add $\{a\} \sqsubseteq \exists P.\{b\}$ to $\mathcal{T}'$;
- if α is of the form $a \not\approx b$, then add $\{a\} \sqsubseteq \neg\{b\}$ to $\mathcal{T}'$.

It is easy to see that K and K' have exactly the same models, so all reasoning services are preserved [38].

2.2 Conjunctive queries and unions of conjunctive queries

We introduce now conjunctive queries, which can be considered as the logical counterparts of select-project-join SQL queries, and unions of conjunctive queries.

We assume that K has an associated set of *distinguished concept names*, denoted $\mathcal{C}_q$, which are the concepts that can occur in queries.

Definition 2.6 [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 simple role name in $\mathbf{R}_K$ 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.

Definition 2.7 [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 $U = Q_1 \vee \cdots \vee Q_m$, $\mathcal{I} \models U$ 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 2.8 Let $\mathcal{C}_q = \{A\}$. We consider the CQs

$$\begin{aligned} Q_1 &= \{\ P_1(x,y),\ P_2(x,z),\ A(y)\ \}, \\ Q_2 &= \{\ P_2(x,y),\ P_2(y,z)\ \}, \\ Q_3 &= \{\ P_2(x,y),\ P_2(y,o)\}. \end{aligned}$$

Note that $K_1 \models Q_1$. Indeed, for an arbitrary model $\mathcal{I}$ of K_1, 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 axiom $A \sqsubseteq \exists P_2.\neg A$). Also, $K_1 \not\models Q_2$. A model $\mathcal{I}$ of K_1 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 $P_2^{\mathcal{I}} = \{(o_1, o_2)\}$.

We have that $K_2 \models Q_1$, since for an arbitrary model $\mathcal{I}$ of K_2, 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 $o^{\mathcal{I}}$, which is connected to $a^{\mathcal{I}}$ via role P_2 by the axiom $A \sqsubseteq \exists P_2.o$. To see that $K_2 \not\models Q_2$, simply extend the interpretation $\mathcal{I}$ given above by setting $o^{\mathcal{I}} = \{o_2\}$. This extended interpretation is a model of K_2 and not a model of Q_2. Finally, $K_2 \models Q_3$. In any model $\mathcal{I}$ of K_2, o must be mapped to the only element of $o^{\mathcal{I}}$ and x can be mapped to $a^{\mathcal{I}}$. Then, by the inclusion axiom $A \sqsubseteq \exists P_1.A$, $a^{\mathcal{I}}$ must be connected via P_1 to some instance of A. The variable y can be mapped to this object, since the axiom $A \sqsubseteq \exists P_2.o$ ensures that it is connected to the element of $o^{\mathcal{I}}$ via role P_2. ∎

Definition 2.9 [Query Entailment] Let K be a knowledge base and let Q be a query. The *query entailment* problem is to decide whether $K \models Q$.

Note that, according to Definitions 2.6 and 2.7, CQs (and hence also UCQs) have no free (i.e., distinguished) variables, so they are Boolean queries. In the traditional database setting, free variables in a query are called distinguished variables. For a query Q that has $\vec{x}$ as distinguished variables, the query answering problem over K consists on finding all the possible tuples of individuals $\vec{t}$ of the same arity as $\vec{x}$ such that when $\vec{x}$ is substituted by $\vec{t}$ in Q, it holds that $K \models Q$. The set of such tuples $\vec{t}$ is the answer of the query. Query answering has an associated recognition problem: given a tuple $\vec{t}$, the problem is to verify whether $\vec{t}$ belongs to the answer of Q[3].

Query answering for a certain DL $\mathcal{L}$ is in a complexity class $\mathcal{C}$, if given any knowledge base K in $\mathcal{L}$ and query Q, deciding $K \models Q$ is in $\mathcal{C}$; this is also called *combined complexity*. The *data complexity* of query answering is the complexity of deciding $K \models Q$ where Q and all of K except $\mathcal{A}$ is fixed.

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

3 A Tableaux Algorithm for Query Entailment

In this section, we describe our method for solving the query entailment problem for UCQs in $\mathcal{SHIQ}$.

It is important to notice that the query entailment problem is not reducible to satisfiability of knowledge bases, since the negation of a query in general 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. In general, a knowledge base has an infinite number of possibly infinite models, and in principle we have to verify whether the query 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 (in our case, a conjunctive query or a union of conjunctive queries) is entailed by a $\mathcal{SHIQ}$ knowledge base. Our technique builds on the tableaux algorithm for knowledge base satisfiability $\mathcal{SHIQ}$ in [26]. Informally, the difference to that work is that it only focuses on problems that can be reduced to checking satisfiability, and therefore the satisfiability algorithm only needs to ensure that if the knowledge base has some model then the 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 [32], yet we deal with Description Logics that lack the finite model property.

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

Like the algorithm in [26], 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 [26], 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 $v \rightarrow w$ is labeled with a set of concepts $\mathcal{L}(v) \subseteq \mathsf{clos}(K)$ and a set of roles $\mathcal{L}(v \rightarrow w) \subseteq \mathbf{R}_K$, respectively. We denote by $\mathsf{nodes}(T)$ the nodes of the variable tree T, by $\mathsf{vars}(T)$ the nodes in $\mathsf{nodes}(T)$ which are variables, and by $\mathsf{arcs}(T)$ the arcs in T.

Definition 3.1 [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. 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 $\psi : \mathsf{nodes}(T_v^n) \rightarrow \mathsf{nodes}(T_{v'}^n)$ such that:

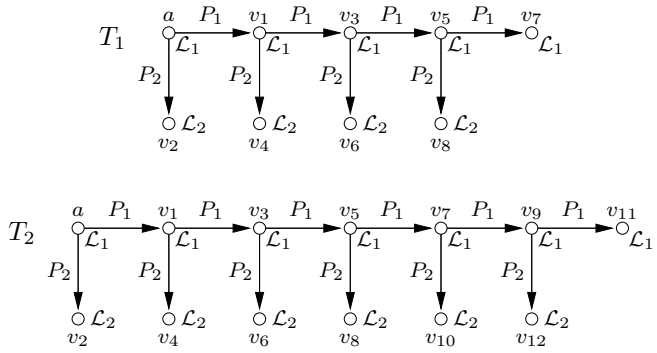

Figure 1: Trees and completion forests for the example knowledge base

- $\psi(v) = v'$
- for every node w in $\mathsf{nodes}(T^n_v)$, $\mathcal{L}(w) = \mathcal{L}(\psi(w))$
- for every arc connecting two nodes w and w' in $\mathsf{nodes}(T^n_v)$, $\mathcal{L}(w \to w') = \mathcal{L}(\psi(w) \to \psi(w'))$.

Definition 3.2 [n-Witness] If variables v and v' in T are n-tree equivalent, v' is an ancestor of v in T and v is not in $T^n_{v'}$, then v' is a *n-witness of v in T*. Furthermore, $T^n_{v'}$ *tree-blocks* T^n_v and each variable w in $T^n_{v'}$ *tree-blocks* variable $\psi^{-1}(w)$ in T^n_v.

Example 3.3 [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); ${T_1}^1_{v_1}$ tree-blocks ${T_1}^1_{v_5}$; and v_1 (resp., v_3, v_4) tree-blocks v_5 (resp., v_7, v_8). ∎

Definition 3.4 [completion forest [26]] 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. The set of arcs in $\mathcal{F}$ is denoted $\mathsf{arcs}(\mathcal{F})$. 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.

Now we introduce the completion forests for K. In them we use a set of *global concepts* $\mathsf{gcon}(K, \mathcal{C}_q) = \{\neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T}\} \cup \{C \sqcup \neg C \mid C \in \mathcal{C}_q\}$. Informally, by requiring that each individual belongs to all global concepts, satisfaction of the TBox is enforced and, by case splitting, each individual can be classified with respect to the distinguished concepts (i.e., those appearing in queries).

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 3.5 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 [26], where the n parameter is not present.

Definition 3.6 [n-blocking] For integer $n \geq 0$, a variable 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 [26]. 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 3.7 Consider $\mathcal{F}_1$ with the variable tree T_1 from Example 3.3 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. 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. The $\leq$-rule is a *shrinking rule*, since it removes a node of the foret by merging it into another. Note that our rules are very similar to the ones in [26]. The main differences are that "blocked" is uniformly replaced by "n-blocked" and the $\exists$-rule and $\geq$-rule in [26] are slightly different, since now the labels of the nodes they generate must contain $\mathsf{gcon}(K, \mathcal{C}_q)$.

In the rules we use two operations on completion forests called prune and merge. To illustrate the use of this operations, consider the rule $\leq$-rule. Suppose some node v is labeled by the concept $\leq 2\, S.C$, and v has three successors v_1, v_2, v_3 all labeled with C, but $v_2 \not\approx v_3$ does not hold. Then we can make v satisfy $\leq 2\, S.C$, by merging the nodes v_2 and v_3 into one. For this purpose, we use merge and prune. Intuitively, $\mathsf{merge}(y, x)$ merges the node y into x: the label of y is added to the label of x, all incoming arcs to x are copied to y, and the outgoing arcs of x to an individual node are also copied to y. After the merging, $\mathsf{prune}(y)$ removes y from $\mathcal{F}$ and, recursively, all its variables successors. Note that when we apply $\mathsf{merge}(y, x)$ and x is a variable node, we do not need to copy any outgoing label, since variable nodes only have variable nodes as successors, and these will be removed by prune.

Formally, for a completion forest $\mathcal{F}$ and $x, y \in \mathsf{vars}(\mathcal{F})$, the operation $\mathsf{prune}(y)$ yields a forest that is obtained from $\mathcal{F}$ as follows:

1. For each $z \in \mathsf{nodes}(\mathcal{F})$ successor of y, remove $y \to z$ from $\mathsf{arcs}(\mathcal{F})$, and if $z \in \mathsf{vars}(\mathcal{F})$ then $\mathsf{prune}(z)$.
2. Remove y from $\mathsf{nodes}(\mathcal{F})$.

The operation $\mathsf{merge}(y, x)$ yields a forest obtained from $\mathcal{F}$ as follows:

1. For each $z \in \mathsf{nodes}(\mathcal{F})$ such that $z \rightarrow y \in \mathsf{arcs}(\mathcal{F})$
   (a) if neither $x \rightarrow z$ nor $z \rightarrow x$ are in $\mathsf{arcs}(\mathcal{F})$, then add $z \rightarrow x$ to $\mathsf{arcs}(\mathcal{F})$ and set $\mathcal{L}(z \rightarrow x) = \mathcal{L}(z \rightarrow y)$;
   (b) if $z \rightarrow x$ is in $\mathsf{arcs}(\mathcal{F})$, then set $\mathcal{L}(z \rightarrow x) = \mathcal{L}(z \rightarrow x) \cup \mathcal{L}(z \rightarrow y)$;
   (c) if $x \rightarrow z$ is in $\mathsf{arcs}(\mathcal{F})$, then set $\mathcal{L}(x \rightarrow z) = \mathcal{L}(x \rightarrow z) \cup \{\mathsf{Inv}(R) \mid R \in \mathcal{L}(z \rightarrow y)\}$;
   (d) remove $z \rightarrow y$ from $\mathsf{arcs}(\mathcal{F})$.
2. For each $z \in \mathsf{nodes}(\mathcal{F}) \setminus \mathsf{vars}(\mathcal{F})$ such that $y \rightarrow z \in \mathsf{arcs}(\mathcal{F})$
   (a) if neither $x \rightarrow z$ nor $z \rightarrow x$ are in $\mathsf{arcs}(\mathcal{F})$, then add $x \rightarrow z$ to $\mathsf{arcs}(\mathcal{F})$ and set $\mathcal{L}(x \rightarrow z) = \mathcal{L}(y \rightarrow z)$;
   (b) if $x \rightarrow z$ is in $\mathsf{arcs}(\mathcal{F})$, then set $\mathcal{L}(x \rightarrow z) = \mathcal{L}(x \rightarrow z) \cup \mathcal{L}(y \rightarrow z)$;
   (c) if $z \rightarrow x$ is in $\mathsf{arcs}(\mathcal{F})$, then set $\mathcal{L}(z \rightarrow x) = \mathcal{L}(z \rightarrow x) \cup \{\mathsf{Inv}(R) \mid R \in \mathcal{L}(y \rightarrow z)\}$;
   (d) remove $y \rightarrow z$ from $\mathsf{arcs}(\mathcal{F})$.
3. Set $\mathcal{L}(x) = \mathcal{L}(x) \cup \mathcal{L}(y)$.
4. Add $x \not\approx z$ for each z with $y \not\approx z$.
5. $\mathsf{prune}(y)$.

Definition 3.8 [Clash free completion forest] A node v in a completion forest $\mathcal{F}$ contains a *clash* iff

1. for some concept C, $\{C, \neg C\} \subseteq \mathcal{L}(v)$
2. $\leq n\, S.C. \in \mathcal{L}(v)$ 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.

Definition 3.9 [n-complete completion forest] A completion forest $\mathcal{F}$ is n-complete if none of the rules in Table 1 can be applied to it (under n-blocking).

We will denote as $\mathbb{F}_K$ the set of all completion forest that can be obtained from the initial $\mathcal{F}_K$ by applying the expansion rules, and by $\mathsf{ccf}_n(\mathbb{F}_K)$ we denote the set of forests in $\mathbb{F}_K$ that are n-complete and clash free.

Example 3.10 Both $\mathcal{F}_1$ and $\mathcal{F}_2$ can be obtained from $\mathcal{F}_K$ by applying the expansion rules. They are both complete and clash-free, so $\mathcal{F}_1 \in \mathsf{ccf}_1(\mathbb{F}_K)$ and $\mathcal{F}_2 \in \mathsf{ccf}_2(\mathbb{F}_K)$. ∎

$\sqcap$-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 $\mathcal{L}(v \rightarrow w) := \{R\}$ 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 $\mathcal{L}(v \rightarrow w_i) := \{S\}$, $\mathcal{L}(w_i) := \{C\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$ 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 $w \not\approx w'$, and $C \in \mathcal{L}(w) \cap \mathcal{L}(w')$
	then	(*i*) if w is an individual node, then $\mathsf{merge}(w', w)$ else (*ii*) if w' is an individual node or an ancestor of w, then $\mathsf{merge}(w, w')$ else (*iii*) $\mathsf{merge}(w', w)$

Table 1: Expansion Rules

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 3.11 [Model of a completion forest] For a completion forest $\mathcal{F} \in \mathbb{F}_K$, 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 3.12 *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 3.11. To prove the other direction, it suffices to consider an arbitrary model $\mathcal{I}$ of K and verify that for for all nodes $a, b \in \mathsf{nodes}(\mathcal{F}_K)$ 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 $a \not\approx b \in \mathcal{F}_K$, 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 3.14, the union of all the models of the forests in $\mathsf{ccf}_n(\mathbb{F}_K)$ 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 $\mathsf{ccf}_n(\mathbb{F}_K)$ suffices to check all models of K. In order to prove this result, we will use following lemma. It states that when applying any of the rules in Table 1, all models are preserved. The proof is straightforward yet long, so it is given in the Appendix.

Lemma 3.13 *Let $\mathcal{F}$ be a completion forests in $\mathbb{F}_K$, 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 $\mathcal{I}' \models \mathcal{F}'$.*

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

Proposition 3.14 *Let $n \geq 0$. For every $\mathcal{I}$ such that $\mathcal{I} \models K$, there is some $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ and some $\mathcal{I}'$ that is an extension of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{F}$.*

Proof. First observe that by the definition of interpretation and by Definition 3.11, if a forest $\mathcal{F}$ has a clash, then there is no $\mathcal{I}$ with $\mathcal{I} \models K$. Let $\mathbf{F}^n$ denote the set of completion forests obtained from $\mathcal{F}_K$ by n applications of the expansion rules, and $\mathsf{cf}(\mathbf{F}^n)$ the set there forests that are clash free. Consider any $\mathcal{I}$ such that $\mathcal{I} \models K$. The proposition follows from the fact that while applying the propagation rules, $\mathcal{I}$ will be preserved, and maybe extended, until some complete forest $\mathcal{F}_c$ is reached. This is captured in the following claim: There is some $\mathcal{I}'$ extension of $\mathcal{I}$ and some $\mathcal{F} \in \mathsf{cf}(\mathbf{F}^n)$ with $\mathcal{I}' \models \mathcal{F}$. The claim can be verified by a simple induction on n. If $n = 0$, then either $\mathsf{cf}(\mathbf{F}^n) = \{\mathcal{F}_K\}$ and the claim holds by Lemma 3.12, or $\mathcal{F}_K$ contains a clash. In the later case, K has no models and the claim holds by antecedent failure. For the inductive step, consider any $\mathcal{F} \in \mathsf{cf}(\mathbf{F}^n)$. If $\mathcal{I} \models \mathcal{F}$, then by Lemma 3.13, there is some $\mathcal{I}'$ extension of $\mathcal{I}$ and some $\mathcal{F}' \in \mathbf{F}^{n+1}$ such that $\mathcal{I}' \models \mathcal{F}'$. Since $\mathcal{F}'$ has a model, then $\mathcal{F}' \in \mathsf{cf}(\mathbf{F}^{n+1})$. □

3.3 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 it suffices to check entailment in each forest $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ for any n, since semantically, they capture all the models of the knowledge base. This is stated in the following proposition:

Proposition 3.15 *Let $n \geq 0$ be arbitrary. Then $K \models U$ iff $\mathcal{F} \models U$ for each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$.*

Proof. The only if direction is easy. Consider any $\mathcal{F} \in \mathbb{F}_K$. 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 3.14, there is some $\mathcal{I}'$ extension of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{F}$ for some $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$. $\mathcal{I} \not\models U$ 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 $\mathsf{ccf}_n(\mathbb{F}_K)$, a finite set of finite structures. Now, we will see that for a suitable n, $\mathcal{F} \models U$ for an $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, 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 3.4 we will extend it to an UCQ U.

Definition 3.16 We say that Q can be mapped into a completion forest $\mathcal{F}$, denoted $Q \hookrightarrow \mathcal{F}$, if there is a mapping $\mu : \mathsf{varsIndivs}(Q) \rightarrow \mathsf{nodes}(\mathcal{F})$ 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 3.17 $Q_1 \hookrightarrow \mathcal{F}_2$ 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 mappability $\mathcal{F} \hookrightarrow Q$, i.e., 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 3.18 *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.

3.3.1 Tableaux and canonical models

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 $\mathcal{F} \in \mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ 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, where the blocked nodes act like 'loops'. Following [26], we will give a rather complex definition of a tableau. Defining a model of K from it will then be straightforward.

Definition 3.19 [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

- $\mathcal{I} : \mathbf{I}_K \rightarrow \mathbf{S}$ 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 $s \in \mathbf{S}$ $C \in \mathcal{L}(s)$.

We can easily obtain a canonical model of a knowledge base from a tableau for it.

Definition 3.20 [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:

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

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 3.21 *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 [26]. Due to $(P15)$, it can be easily verified that $\mathcal{I}_T$ is also a model of $\mathcal{T}$. □

Each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ 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 3.22 [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:

$$\begin{array}{rcl} \mathbf{S} & = & \mathsf{paths}(\mathcal{F}) \\ \mathcal{L}(p) & = & \mathcal{L}(\mathsf{tail}(p)) \\ \mathcal{E}(R) & = & \{\langle p, q\rangle \in \mathbf{S} \times \mathbf{S} \mid q \text{ is an } R\text{-step of } p\} \cup \\ & & \{\langle p, q\rangle \in \mathbf{S} \times \mathbf{S} \mid p \text{ is an } \mathsf{Inv}(R)\text{-step of } q\} \cup \\ & & \{\langle p, q\rangle \in \mathbf{S} \times \mathbf{S} \mid \mathsf{tail}(q) \text{ is an individual node and} \\ & & \qquad \text{an } R\text{-successor of } \mathsf{tail}(p)\} \cup \\ & & \{\langle p, q\rangle \in \mathbf{S} \times \mathbf{S} \mid \mathsf{tail}(p) \text{ is an individual node and} \\ & & \qquad \text{an } \mathsf{Inv}(R)\text{-successor of } \mathsf{tail}(q)\} \\ \mathcal{I}(a) & = & [\frac{a}{a}] \end{array}$$

Note that the definition of R-steps requires w to be a variable node. Every path starts with a node $\frac{a}{a}$ for some individual a, and a node of such a form never occurs after the first position in such a path. The last two cases in the definition of $\mathcal{E}(R)$ are necessary in order to consider the arcs arbitrarily connecting individual nodes, which are not unraveled.

Example 3.23 By unraveling $\mathcal{F}_1$, we obtain a model $\mathcal{I}_{\mathcal{F}_1}$ that has as a domain the infinite set of paths from a to each v_i. Note that a path actually comprises a sequence of pairs of nodes, in order to witness the loops introduced by blocked variables. When a node is not blocked, like v_1, the pair $\frac{v_1}{v_1}$ is added to the path. Since

$T^1_{v_1}$ tree-blocks $T^1_{v_5}$, every time a path reaches v_7, which is a leaf of a blocked tree, we add $\frac{v_3}{v_7}$ to the path and 'loop' back to the successors of v_3. In this way, we obtain the following infinite set of paths:

$$
\begin{array}{ll}
p_0 = [\frac{a}{a}], & p_6 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_6}{v_6}], \\
p_1 = [\frac{a}{a}, \frac{v_1}{v_1}], & p_7 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_3}{v_7}], \\
p_2 = [\frac{a}{a}, \frac{v_2}{v_2}], & p_8 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_4}{v_8}], \\
p_3 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}], & p_9 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_3}{v_7}, \frac{v_5}{v_5}], \\
p_4 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_4}{v_4}], & p_{10} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_3}{v_7}, \frac{v_6}{v_6}], \\
p_5 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}], & p_{11} = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_3}{v_3}, \frac{v_5}{v_5}, \frac{v_3}{v_7}, \frac{v_5}{v_5}, \frac{v_3}{v_7}], \\
 & \vdots
\end{array}
$$

This set of paths is the domain of $\mathcal{I}_\mathcal{F}$. The extension of each concept C is determined by the set all p_i such that C occurs in the label of the last node in p_i. For the extension of each role R, we consider the pairs $\langle p_i, p_j \rangle$ such that the last node in p_j is an R-successor of p_i. If $R \in \mathbf{R}_+$, its extension is transitively expanded. Therefore p_0, p_1, p_3, ... are in $A^{\mathcal{I}_{\mathcal{F}_1}}$, and $\langle p_0, p_1 \rangle$, $\langle p_1, p_3 \rangle$, $\langle p_3, p_5 \rangle$, $\langle p_5, p_7 \rangle$, ... are all in $P_1^{\mathcal{I}_{\mathcal{F}_1}}$. ■

Lemma 3.24 *Let $n \geq 1$. Every $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ induces a model $\mathcal{I}_\mathcal{F}$ of K.*

Proof. First, it is proved as in [26] that every $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ 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 $\leq$-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. For a given forest $\mathcal{F}$ in $\mathsf{ccf}_n(\mathbb{F}_K)$ 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 image of Q under σ in $T_\mathcal{F}$. 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. Informally, 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.

Definition 3.25 Let $\mathcal{F}$ be a forest in $\mathsf{ccf}_n(\mathbb{F}_K)$ (for some n) such that $\mathcal{I}_{\mathcal{F}} \models Q$. Consider a mapping σ verifying the conditions in Definition 3.16. Let G denote the subgraph of the image of Q under σ in $T_{\mathcal{F}}$ obtained by removing each node of the form $[\frac{a}{a}]$ for some individual a. For any x, y in $\mathsf{varsIndivs}(Q)$, if $\sigma(x)$ and $\sigma(y)$ are nodes of G and there is a path between them, then $d^{\sigma}(x, y)$ denotes the length of the shortest such path. Otherwise $d^{\sigma}(x, y) = 0$. Finally, the *maximal Q-distance of σ*, denoted d^{σ}_{Q}, is the maximal $d^{\sigma}(x, y)$ for all x, y in $\mathsf{varsIndivs}(Q)$.

Example 3.26 The canonical model $\mathcal{I}_{\mathcal{F}_1}$ models Q_1. We can consider the following substitution σ: $\sigma(x) = p_7$, $\sigma(y) = p_9$ and $\sigma(z) = p_1 0$. The image of Q_1 under σ has no nodes of the form $[\frac{a}{a}]$, so G is the graph with nodes p_7, p_9 and $p_1 0$ and arcs $p_7 \rightarrow p_9$ and $p_7 \rightarrow p_1 0$. Moreover, $d^{\sigma}(x, y) = 1$, $d^{\sigma}(x, z) = 1$ and $d^{\sigma}(y, z) = 0$, so $d^{\sigma}_{Q} = 1$. ∎

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 $\mu : \mathsf{varsIndivs}(Q) \rightarrow \mathsf{nodes}(\mathcal{F})$ that witnesses the entailment of Q.

Proposition 3.27 *Consider any $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ 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 $\sigma : \mathsf{varsIndivs}(Q) \rightarrow \Delta^{\mathcal{I}_{\mathcal{F}}}$ 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 $\Delta^{\mathcal{I}_{\mathcal{F}}} = \mathbf{S}$, $\sigma(x)$ and $\sigma(y)$ 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 $\mu : \mathsf{varsIndivs}(Q) \rightarrow \mathsf{nodes}(\mathcal{F})$. 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 $\sigma(x)$ is in $\mathsf{afterblocked}(G_i)$ for some G_i, $\mathsf{tail}(\sigma(x)) = \mathsf{tail}'(\sigma(x))$.
If $\sigma(x) \in \mathsf{afterblocked}(G_i)$ then, by definition, $\sigma(x)$ is a successor of $\sigma(y)$ for 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 $\sigma(x) \notin \mathsf{afterblocked}(G_i)$, 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 $\mu : \mathsf{varsIndivs}(Q) \to \mathsf{nodes}(\mathcal{F})$ 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 which satisfies that $\mathsf{afterblocked}(G_i) = \emptyset$, $\mu(x) = \mathsf{tail}(\sigma(x))$.
- For each variable x in $\mathsf{varsIndivs}(Q)$ such that $\sigma(x) \in \mathsf{nodes}(G_i)$ for some G_i which satisfies that $\mathsf{afterblocked}(G_i) \neq \emptyset$, the mapping μ is given by:

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

Now we will show that it 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 $\sigma(x)$ is in $\mathsf{afterblocked}(G_i)$ and $\sigma(y)$ is not, thus we have the following cases:

(a) Both $\sigma(x)$ and $\sigma(y)$ 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 $\sigma(x)$ nor $\sigma(y)$ is in $\mathsf{afterblocked}(G_i)$.
Note that in this case $\mathsf{tail}(\sigma(x)) = \mathsf{tail}'(\sigma(x))$, otherwise $\sigma(y)$ 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) $\sigma(x)$ is not in $\mathsf{afterblocked}(Q)$, but $\sigma(y)$ 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 is the identity for all individuals and it has properties 1 and 2, $Q \hookrightarrow \mathcal{F}$.

□

Example 3.28 The graph G has only one connected component, namely G itself. In G we have that $\mathsf{blockedLeaves}(G) = \{p_7\}$, and $\mathsf{afterblocked}(G) = \{p_9, p_{10}\}$. For the σ given in Example 3.26, we get the mapping μ defined as: $\mu(x) = \psi(\mathsf{tail}'(p_7)) = v_3$; $\mu(y) = \mathsf{tail}'(p_9) = v_5$; $\mu(z) = \mathsf{tail}'(p_{10})) = v_6$. It satisfies the conditions of Definition 3.16, so it proves that $Q_1 \hookrightarrow \mathcal{F}_1$. ∎

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 3.29 *Let Q be a CQ and K a $\mathcal{SHIQ}$ knowledge base. $K \models Q$ iff $Q \hookrightarrow \mathcal{F}$ for every completion forest $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, $n \geq n_Q$.*

Proof. First we prove that if $K \models Q$ then $Q \hookrightarrow \mathcal{F}$. Take any arbitrary $\mathcal{F} \in \mathsf{ccf}_{n_Q}(\mathbb{F}_K)$. Since $K \models Q$, then $\mathcal{F} \models Q$ (Proposition 3.15). 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 3.27, $Q \hookrightarrow \mathcal{F}$.

To prove the other direction, observe that from $Q \hookrightarrow \mathcal{F}$ and Lemma 3.18, we have that $\mathcal{F} \models Q$ for every $\mathcal{F} \in \mathsf{ccf}_{n_Q}(\mathbb{F}_K)$. Finally, by Proposition 3.15, $K \models Q$. □

Example 3.30 $K \models Q_1$, so $\mathcal{F}_1 \models Q_1$ must hold. This is witnessed by the mapping μ in Example 3.28. 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 $n_{Q'} > 1$. ∎

3.4 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 $\mathsf{ccf}_n(\mathbb{F}_K)$ for any arbitrary $n \geq 0$ (in fact, Proposition 3.15 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 3.31 *Consider any $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ 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 $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, 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 3.27 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 3.32 By the mapping $Q_1 \hookrightarrow \mathcal{F}_1$ in Example 3.17, 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 $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ for any $n \geq n_U$.

Theorem 3.33 *Let U be an UCQ and K a $\mathcal{SHIQ}$ knowledge base. $K \models U$ iff $U \hookrightarrow \mathcal{F}$ for every $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, $n \geq n_U$.*

Proof. As in the proof of Theorem 3.29, it follows Proposition 3.15 and Proposition 3.31. □

4 Extending the Algorithm to $\mathcal{SHOIQ}$

When describing the algorithm for $\mathcal{SHIQ}$, we used completion forests to represent models. The ABox individuals form a graph that might be arbitrarily interconnected, which we call a cloud, and each individual in the cloud is the root of a tree. Since the cloud has a fixed size, we can talk about some kind of forest model property.

In $\mathcal{SHOIQ}$ however, this property is almost completely lost. Initially the nominals forms a cloud of arbitrarily connected nodes. Each of this nodes is the root of a variable tree, which might be blocked representing an infinite structure. However, due to the interaction between nominals, inverse roles and number restrictions, we have to consider arbitrary relational structures between some nodes that are not named individuals. For example, consider a knowledge base containing the axioms $\top \sqsubseteq \exists R^-.\{o\}$ and $\{o\} \sqsubseteq \leq 3\, R.B$. If there was a blocked tree structure whose unraveling generates instances of B, they would all have to have an R^- arc to o possibly violating the number restriction $\leq 3\, R.B$. It is then necessary to identify these nodes as individuals, and avoid their multiplication by unraveling. Thus when applying the expansion rules, we will generate not only variable nodes, but also individual nodes which might be arbitrarily connected to the other individuals, and that may also be descendants of some variable in a tree. Therefore, instead of completion forests, we now refer to completion graphs. In completion graphs the initial cloud may grow, and some variable nodes may contain edges connecting them to an individual in the cloud.

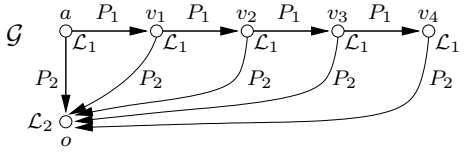

Figure 2: Completion graph for the example knowledge base

Definition 4.1 [completion graph [23]] A *completion graph* $\mathcal{G}$ for a knowledge base K is a given by a directed graph and an inequality relation $\not\approx$, implicitly assumed to be symmetric. In the directed graph, each node v is labeled with a set of concepts $\mathcal{L}(v) \subseteq \mathsf{clos}(K)$ and each arc $v \rightarrow w$ with a set of roles $\mathcal{L}(v \rightarrow w) \subseteq \mathbf{R}_K$, respectively. The nodes and arcs of $\mathcal{G}$ are denoted by $\mathsf{nodes}(\mathcal{G})$ and $\mathsf{arcs}(\mathcal{G})$. The set of nodes v in $\mathsf{nodes}(\mathcal{G})$ such that $\{o\} \notin \mathcal{L}(v)$ for each nominal $o \in \mathbf{N}$, is denoted $\mathsf{vars}(\mathcal{G})$, these nodes are called *variable nodes*. The *individual nodes* are the nodes in the set $\mathsf{nodes}(\mathcal{G}) \setminus \mathsf{vars}(\mathcal{G})$, i.e., $\{o\} \in \mathcal{L}(v)$ for some nominal $o \in \mathbf{N}$ [4].

The notions of R-successor, $\mathsf{Inv}(R)$-predecessor, R-neighbor and ancestor are defined as usual (see Definition 3.4). As in the algorithm for $\mathcal{SHIQ}$, we will associate to K an initial completion graph, and then we will apply *expansion rules* to obtain new completion graphs, until no more rules can be applied. The initial completion graph $\mathcal{G}_K$ associated with K has a node a for each individual $a \in \mathbf{I}_K$, labeled with $\mathcal{L}(a)$ = $\{\{a\}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$. The relation $\not\approx$ is initialized as $a \not\approx b$ iff $a \neq b \in \mathcal{A}$.

Example 4.2 In our running example, $\mathcal{G}_{K_2}$ contains two nodes, a and o with the labels $\mathcal{L}(a) := \{\{a\}, A, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\{o\}, A \sqcup \neg A\}$ and $\mathcal{L}(\{o\}) := \{\{o\}, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\{o\}, A \sqcup \neg A\}$. The $\not\approx$ relation is empty. ∎

Note that if K is a $\mathcal{SHIQ}$ knowledge base, $\mathcal{F}_K$ is isomorphic to $\mathcal{G}_{K'}$, where K' is obtained from K by internalizing the ABox in the TBox (see Section 2).

From this initial $\mathcal{G}_K$ we will obtain new graphs by applying expansion rules. Initially, $\mathcal{G}_K$ contains only individual nodes which can be arbitrarily interconnected. When we apply the expansion rules, we might introduce new nodes. Like in the $\mathcal{SHIQ}$ case, variable nodes will only be introduced in such a way, that they will always occur in a variable tree (see Section 3.1). However, now we also have a rule that introduces individual nodes which may be successors of a variable node ($o?$-rule, see later). Informally, a branch of a tree may end with an arc leading from a variable to an individual in the cloud. If we remove from $\mathcal{G}$ all such arcs, we will obtain a forest of variable trees rooted at individual nodes, and arbitrary arcs connecting these individuals, i.e., a completion forest (see Definition 3.4). Therefore we can talk about the *forest part* of $\mathcal{G}$.

Definition 4.3 The *forest part* of a completion graph $\mathcal{G}$, denoted $\mathcal{G}_f$, is obtained from $\mathcal{G}$ by removing from $\mathsf{arcs}(\mathcal{G})$ all arcs $v \rightarrow w$ with v a variable node and w an individual node.

Example 4.4 Figure 2 shows the completion graph $\mathcal{G}$, where $\mathcal{L}_1 = \{A, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\{o\}, A \sqcup \neg A, \exists P_1.A, \exists P_2.\{o\}\}$,
$\mathcal{L}_2 = \{\{o\}, \neg A, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\{o\}, A \sqcup \neg A\}$
$\mathcal{G}_f$, the forest part of $\mathcal{G}$, is obtained by removing the arcs: $v_1 \rightarrow o$, $v_2 \rightarrow o$, $v_3 \rightarrow o$, $v_4 \rightarrow o$. The $\not\approx$ relation is empty in both the graph and the forest. ∎

[4] Our individual nodes correspond to *nominal nodes* in [23], and our variable nodes to *blockable nodes*.

Before given the expansion rules, we will discuss the blocking conditions. For the forest part of a completion graph, we can use the same blocking conditions as for a $\mathcal{SHIQ}$ completion forest, given in Definition 3.6.

Definition 4.5 [n-blocking] For an integer $n \geq 0$, a variable node v in a completion graph $\mathcal{G}$ is *n-blocked*, if v is *n-blocked* in the completion forest $\mathcal{G}_f$. Node v is *(in)directly n-blocked* in $\mathcal{G}$ if it is (in)directly n-blocked in $\mathcal{G}_f$.

Note that, as usual, only variable nodes can be blocked. When the blocking conditions for $\mathcal{SHIQ}$ are extended to $\mathcal{SHOIQ}$ in [23], the authors impose as an additional condition that no individual node occurs between the blocking and the blocked tree. We do not need to state this requirement explicitly: any path in $\mathcal{G}$ between two variable nodes that contains an individual node must contain some arc $v \rightarrow w$ with v a variable and w an individual, and thus it does not exist in $\mathcal{G}_f$.

Example 4.6 In the completion forest $\mathcal{G}_f$, the 1-tree rooted at v_1 tree-blocks the 1-tree rooted at v_3. Since v_4 is a leaf of a tree-blocked 1-tree, v_4 is 1-blocked in $\mathcal{G}_f$, so it is also 1-blocked in the completion graph $\mathcal{G}$. ■

To obtain the new completion graphs from the initial $\mathcal{G}_K$, we apply the rules in Table 1, plus two new rules handling the creation and merging of nominal nodes, given in Table 2. The $o?$-rule is a generating rule, and the o-rule a shrinking one. Note that the new rules do not require blocking. The only difference between the rules in Table 2 and the ones in [26] is that in the $o?$-rule, the labels of the newly introduced nodes must contain $\mathsf{gcon}(K, \mathcal{C}_q)$.

The following strategy is used to apply the expansion rules:

1. The o-rule is applied with highest priority.
2. Next come the $\leq$-rule and $o?$-rule. They are applied first to individual nodes with lower level. The level of an individual node v is 0 if $\mathcal{L}(v) \cap \mathbf{N}_K \neq \emptyset$; it is i if v has a neighbor of level $i-1$ and it is not of level j for any $0 \leq j \leq i$. If both the $\leq$-rule and $o?$-rule are applicable to the same node, the $o?$-rule is applied first.
3. All the remaining rules are applied with a lower priority.

We will see later that this strategy is necessary to ensure termination of the algorithm.

Definition 4.7 [Clash free completion graph, n-complete completion graph] A completion graph $\mathcal{G}$ contains a *clash* if some node $v \in \mathsf{nodes}(\mathcal{G})$ satisfies conditions 1 or 2 in Definition 3.8, or if there is some nominal $o \in \mathbf{N}$ such that $\{o\} \in \mathcal{L}(v) \cap \mathcal{L}(v')$ for some $v, v' \in \mathsf{nodes}(\mathcal{G})$ with $v \not\approx v'$. A completion graph $\mathcal{G}$ is *clash free* if it contains no clash. A completion graph $\mathcal{G}$ is n-complete if none of the rules in Tables 1 and 2 can be applied to it (under n-blocking).

We denote by $\mathbb{G}_K$ the set of all $\mathcal{G}$ obtained from the initial $\mathcal{G}_K$ by means of the expansion rules, and by $\mathsf{ccf}_n(\mathbb{G}_K)$ we denote the set of graphs in $\mathbb{G}_K$ that are n-complete and clash free.

Example 4.8 Consider the completion graph $\mathcal{G}'$ in Figure 3. Both $\mathcal{G}$ and $\mathcal{G}'$ can be obtained from $\mathcal{G}_K$ by means of the expansion rules. They are both clash free completion graphs, and they are 1-complete and 2-complete respectively, so $\mathcal{G} \in \mathsf{ccf}_1(\mathbb{G}_K)$ and $\mathcal{G}' \in \mathsf{ccf}_2(\mathbb{G}_K)$. ■

o-rule:	if	there are v, v' in $\mathsf{nodes}(\mathcal{G})$ with not $v \not\approx v'$ and $\{o\} \in \mathcal{L}(v) \cap \mathcal{L}(v')$ for some $o \in \mathbf{N}$
	then	$\mathsf{merge}(v, v')$
$o?$-rule:	if	$\leq n\, S.C \in \mathcal{L}(v)$, v is an individual node there is a v' in $\mathsf{vars}(\mathcal{G})$ such that v' is an S-neighbour of v, $C \in \mathcal{L}(v')$ and v is a successor of v'; and there is no m with $1 \leq m \leq n$, $\leq m\, S.C \in \mathcal{L}(v)$, v has m S-neighbours $w_1, \ldots, w_m$ such that for all $1 \leq j \leq i \leq m$ w_i is an individual node, $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j$
	then	guess $m \leq n$, set $\mathcal{L}(v) := \mathcal{L}(v) \cup \leq m\, S.C$, create m new nodes $w_1, \ldots, w_m$ with $\mathcal{L}(v \to w_i) := \{S\}$, $\mathcal{L}(w_i) := \{C, \{o_i\}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$ for some $o_i \in \mathbf{N}$ new in $\mathcal{G}$, and $w_i \not\approx w_j$ for all $1 \leq j \leq i \leq m$.

Table 2: New Expansion Rules for $\mathcal{SHOIQ}$

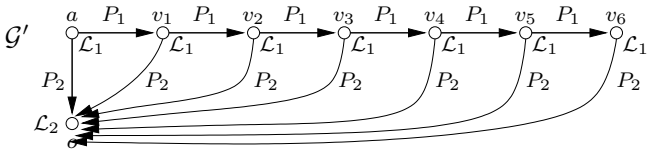

Figure 3: 2-complete completion graph for the example knowledge base

Each completion graph $\mathcal{G}$ in $\mathbb{G}_K$ represents a set of possibly extended models of K in the natural way: $\mathcal{I} \models \mathcal{G}$ if $\mathcal{I} \models K$ and for all nodes $v, w \in \mathcal{G}$ the three conditions given in Definition 3.11 hold, i.e., each node corresponds to an individual in the interpretation in such a way that the node and arc labels, as well as the inequality relation, are contained in the concept, role and inequality extensions of $\mathcal{I}$. The following extension of Lemma 3.12 shows that $\mathcal{G}_K$ represents all models of K:

Lemma 4.9 *An interpretation $\mathcal{I}$ is a model of $\mathcal{G}_K$ iff $\mathcal{I}$ is a model of K.*

Proof. The proof is similar to the one of Lemma 3.12. The if direction follows from the definition of model of a completion graph. To prove the only if direction, it suffices to verify that for an arbitrary model $\mathcal{I}$ of K and for all nodes $a, b \in \mathsf{nodes}(\mathcal{G}_K)$, conditions (i) to (iii) of Lemma 3.12 hold. □

The semantical notion of query entailment in a completion graph is defined in the natural way: for a completion graph $\mathcal{G}$ and a query U, we say that $\mathcal{G} \models U$ iff for every interpretation $\mathcal{I}$, $\mathcal{I} \models \mathcal{G}$ implies $\mathcal{I} \models U$.

Lemma 3.13 and Proposition 3.14 can be easily extended to completion graphs (see Lemma A.1 and Proposition A.2 in the Appendix). The union of all the models of the graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ captures all the models of a knowledge base K, independently of the value of n. Thus we need to consider only the set of graphs $\mathsf{ccf}_n(\mathbb{G}_K)$ when verifying all models of K for query entailment.

Proposition 4.10 *Let $n \geq 0$ be arbitrary. Then $K \models U$ iff $\mathcal{G} \models U$ for each $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.*

Now it is only left to prove that semantical entailment in a completion graph can be reduced to syntactical mappability, if a suitable n-blocking is used. Mappability of Q into a completion graph $\mathcal{G}$, is defined exactly as for completion forests:

Definition 4.11 $Q \hookrightarrow \mathcal{G}$ if there is a mapping $\mu : \mathsf{varsIndivs}(Q) \rightarrow \mathsf{nodes}(\mathcal{G})$ 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 4.12 The mapping $\mu(x) = a$, $\mu(y) = v_1$ and $\mu(o) = o$ shows that $Q_3 \hookrightarrow \mathcal{G}$ and $Q_3 \hookrightarrow \mathcal{G}'$. ∎

Clearly mappability implies entailment (see Lemma 3.18):

Lemma 4.13 *If $Q \hookrightarrow \mathcal{G}$, then $\mathcal{G} \models Q$.*

The other direction is also easy, since it follows directly from the results in Section 3.3. If $\mathcal{G} \models Q$ and $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ for a suitable n, then we can obtain a canonical model of $\mathcal{G}$ and obtain a mapping of Q into $\mathcal{G}$ from it.

In order to obtain the canonical model of $\mathcal{G}$, we follow the same steps as before. The completion graph $\mathcal{G}$ is first unraveled into a tableau $T_{\mathcal{G}}$, and this tableau induces a canonical model $\mathcal{I}_{\mathcal{G}}$. We will shortly describe how the definitions and results given in Section 3.3.1 are extended to $\mathcal{SHOIQ}$.

To obtain the canonical model of a completion graph $\mathcal{G}$, $\mathcal{G}$ is first unraveled into a tableau $T_{\mathcal{G}}$. The definition of $\mathcal{SHOIQ}$ tableau is extended to $\mathcal{SHOIQ}$ by adding to the conditions (P1) to (P15), given in Definition 3.19, an additional one:

(P16) if $\{o\} \in \mathcal{L}(s) \cap \mathcal{L}(s')$ for some $o \in \mathbf{N}$, then $s = s'$.

This condition ensures that nominals are interpreted as singletons.

Every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ can be unraveled into a possibly infinite a tableau $T_{\mathcal{G}}$. Since only the forest-shaped part of $\mathcal{G}$ is unraveled into possibly infinite paths, the unraveling is defined in the same way as for $\mathcal{SHIQ}$. Note that in Definition 3.22, we defined $[p \mid \frac{w}{w}]$ to be an R-step of p only if w is a variable, so only nodes in $\mathsf{vars}(\mathcal{G})$ will occur after the first position in a path. The definition of the tableau induced by a completion forest is extended to completion graphs as follows:

Definition 4.14 [Tableau induced by a completion graph] A path q is an *R-step* of a path p if q is an *R-step* of p in the completion forest $\mathcal{G}_f$, and the set $\mathsf{paths}(\mathcal{G})$ of *paths in a completion graph* is defined as $\mathsf{paths}(\mathcal{G}) = \mathsf{paths}(\mathcal{G}_f)$. The *tableau* $T_{\mathcal{G}} = (\mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{I})$ *induced by a completion graph* $\mathcal{G}$ is obtained by setting $\mathbf{S} = \mathsf{paths}(\mathcal{G})$, and $\mathcal{L}$, $\mathcal{E}$, $\mathcal{I}$ are defined as in Definition 3.22.

The tableau $T_{\mathcal{G}}$ is already very close to a model of K. To obtain its canonical model $\mathcal{I}_{\mathcal{G}}$, it suffices add the missing transitive edges as we did with the $\mathcal{SHIQ}$ tableau. The canonical model of a $\mathcal{SHOIQ}$ tableau is defined exactly as for a $\mathcal{SHOIQ}$ tableau, i.e., as in Definition 3.20.

Lemma 4.15 *Let $n \geq 1$. Every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ induces a model $\mathcal{I}_{\mathcal{G}}$ of K.*

Proof. First, observe that every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ for $n \geq 1$ can be unraveled into a tableau $T_{\mathcal{G}}$ for K. In the proof of Lemma 3.24, we already proved that every $\mathcal{SHIQ}$ completion forest can be unraveled into a $\mathcal{SHIQ}$ tableau. The only difference between a $\mathcal{SHIQ}$ completion forest and a $\mathcal{SHOIQ}$ completion graph is that there might be additional arcs connecting some variable node to an individual in the cloud. In [23, 24] the authors prove that each complete and clash free completion graph can be unraveled into a tableau $T_{\mathcal{G}}$.

Their blocking conditions are different from ours, but since 1-tree blocking implies their blocking, their proof applies to every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ for $n \geq 1$. It is verified that conditions (P1) to (P14) hold, the details are almost exactly as in the proof for $\mathcal{SHIQ}$, but taking into account these new arcs. Additionally, to verify that $T_\mathcal{G}$ is indeed a $\mathcal{SHOIQ}$ tableau, we must prove that (P16) also holds, which is clear since $\mathcal{G}$ is complete and clash free. Condition (P15), which is not considered in [24], holds because all nodes v are initialized with $\mathsf{gcon}(K, \mathcal{C}_q) \subseteq \mathcal{L}(v)$ (see Lemma 3.24).

Finally, it only remains to verify that the canonical model induced by $T_\mathcal{G}$ is indeed a model of K. Again the proof is a straightforward extension of the one of Lemma 3.21. The only additional consideration is that in $\mathcal{I}_\mathcal{G}$ the nominals must be interpreted as singletons, which is ensured by condition (P16). This is also the way the proof of Lemma 4 in [23] extends the corresponding proof for $\mathcal{SHIQ}$. □

Example 4.16 By unraveling $\mathcal{G}'$, we obtain a model $\mathcal{I}_{\mathcal{G}'}$ that has as domain the infinite set of paths from a to each v_i, since there are no paths from o to any other node, i.e., the domain is:

$$
\begin{array}{ll}
p_0 = [\frac{o}{o}], & \\
p_1 = [\frac{a}{a}], & p_5 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}], \\
p_2 = [\frac{a}{a}, \frac{v_1}{v_1}], & p_6 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}, \frac{v_5}{v_5}] \\
p_3 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}], & p_7 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}, \frac{v_5}{v_5}, \frac{v_4}{v_6}] \\
p_4 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}] & p_8 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}, \frac{v_5}{v_5}, \frac{v_4}{v_6}, \frac{v_5}{v_5}] \\
 & \vdots
\end{array}
$$

The extension of the concepts are $\{o\}^{\mathcal{I}_\mathcal{G}} = \{p_0\}$ and $A^{\mathcal{I}_\mathcal{G}} = \{p_i \mid i \geq 1\}$, and the extentions of the roles are $P_1^{\mathcal{I}_\mathcal{G}} = \{\langle p_i, p_{i+1}\rangle \mid i \geq 1\}$ and $P_2^{\mathcal{I}_\mathcal{G}} = \{\langle p_i, p_0\rangle \mid i \geq 1\}$. ∎

Now we are ready to prove that, for a sufficiently large n, the canonical model of an n-complete and clash free completion graph semantically suffices to check query entailment. It is not hard to see that for the $\mathcal{SHOIQ}$ case, a suitable lower bound for n is also given by the maximal distance between any two paths ending in variable nodes in the graph defined by the image of Q in the tableau $T_\mathcal{G}$ (under any satisfying mapping, which must exist since $\mathcal{I}_\mathcal{G}$ entails Q)

Given a completion graph $\mathcal{G}$ in $\mathsf{ccf}_n(\mathbb{G}_K)$ for some n, let $T_\mathcal{G} = \langle \mathbf{S}, \mathcal{L}, \mathcal{E}, \mathcal{I}\rangle$ denote its tableau and $\mathcal{I}_\mathcal{G}$ its canonical model. If $\mathcal{I}_\mathcal{G} \models Q$, 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 image of Q under σ in the tableau $T_\mathcal{G}$. Remove from this graph every node of the form $[\frac{a}{a}]$ for some individual node $a \in \mathsf{nodes}(\mathcal{G})$, with its corresponding incoming and outgoing edges, to obtain the graph G. Note that G comprises a set of tree-shaped components exactly as in the $\mathcal{SHIQ}$ case. On this graph G the *maximal Q-distance of σ*, d_Q^σ, is defined exactly as in Section 3. Recall that d_Q^σ is bounded by n_Q, the number of role atoms in Q.

Example 4.17 As we saw before, $K_2 \models Q_3$, thus $\mathcal{I}_{\mathcal{G}'} \models Q_3$. Consider the substitution $\sigma(x) = p_7$, $\sigma(y) = p_8$ and $\sigma(o) = p_0$. It defines the graph with nodes p_0, p_7 and p_8, and arcs $p_7 \to p_8$ and $p_8 \to p_0$. When we remove from the graph the node p_0 (it is of the form $[\frac{o}{o}]$ and o is an individual) with the corresponding arc $p_8 \to p_0$, we obtain the graph G with two nodes p_7 and p_8 and an arc $p_7 \to p_8$. Obviously G is tree shaped and $d_{Q_3}^\sigma = 1$. ∎

Now Proposition 3.27 can be extended to completion graphs as follows:

Proposition 4.18 *Consider any $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ with $n \geq n_Q$, and let $\mathcal{I}_\mathcal{G}$ be the canonical model of $\mathcal{G}$. If $\mathcal{I}_\mathcal{G} \models Q$ then $Q \hookrightarrow \mathcal{G}$.*

Proof. Since $\mathcal{I}_\mathcal{G} \models Q$, there is a substitution $\sigma : \mathsf{varsIndivs}(Q) \to \Delta^{\mathcal{I}_\mathcal{G}}$ which witnesses this entailment. As before, we can use this σ to define a the mapping $\mu : \mathsf{varsIndivs}(Q) \to \mathsf{nodes}(\mathcal{G})$. This μ maps each individual a in $\mathsf{varsIndivs}(Q)$ to $\mathsf{tail}(\sigma(a)) = a$. For each variable x in $\mathsf{varsIndivs}(Q)$ such that $\sigma(x) \in \mathsf{nodes}(G_i)$ for some G_i with $\mathsf{afterblocked}(G_i) = \emptyset$, μ maps x to $\mathsf{tail}(\sigma(x))$; and for all other variables μ is be defined by:

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

The mapping μ is the identity for all individuals. As showed in the proof of Proposition 3.27, it satisfies the conditions 1 and 2, hence $Q \hookrightarrow \mathcal{G}$ holds. □

Example 4.19 The graph G has only one tree shaped component, namely G itself, and $\mathsf{afterblocked}(G) = p_8$. Thus μ is defined as $\mu(x) = \psi(\mathsf{tail}'(p_7)) = v_4$, $\mu(y) = \mathsf{tail}'(p_8) = v_5$ and $\mu(o) = \mathsf{tail}(p_0) = o$. Note that μ satisfies the necessary conditions to prove that $Q_3 \hookrightarrow \mathcal{G}'$. ∎

With this we have finished giving a method to answer CQs in $\mathcal{SHOIQ}$.

Theorem 4.20 *Let Q be a CQ and K a $\mathcal{SHOIQ}$ knowledge base. $K \models Q$ iff $Q \hookrightarrow \mathcal{G}$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq n_Q$.*

Proof. In Proposition 4.10 we proved that $K \models Q$ iff $\mathcal{G} \models Q$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. From Lemma 4.13 and Proposition 4.18 we have that $\mathcal{G} \models Q$ iff $Q \hookrightarrow \mathcal{G}$ for each $\mathcal{G} \in \mathsf{ccf}_{n_Q}(\mathbb{G}_K)$ (since $\mathcal{G} \models Q$ implies $\mathcal{I}_\mathcal{G} \models Q$), and then the Theorem holds. □

The extension of the method to an UCQ U is analogous to the case of $\mathcal{SHIQ}$. By uniformly replacing $\mathcal{F}$ by $\mathcal{G}$ and $\mathsf{ccf}_n(\mathbb{F}_K)$ by $\mathsf{ccf}_n(\mathbb{G}_K)$ in the statement and proof of Proposition 3.31, the result applies to any $\mathcal{SHOIQ}$ knowledge base K. From this and Proposition 4.10 we obtain the main result of this section:

Theorem 4.21 *Let U be an UCQ and K a $\mathcal{SHOIQ}$ knowledge base. $K \models U$ iff $U \hookrightarrow \mathcal{G}$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq n_U$.*

Example 4.22 Let U be the UCQ $Q_1 \vee Q_2 \vee Q_3$. Since $K_2 \models Q_3$, then $K_2 \models U$. To verify this, it would be necessary to check whether $U \hookrightarrow \mathcal{G}$ for each $\mathcal{G}$ in $\mathsf{ccf}_2(\mathbb{G}_{K_2})$. In particular, since $\mathcal{G}' \in \mathsf{ccf}_2(\mathbb{G}_{K_2})$, it must be the case that $U \hookrightarrow \mathcal{G}'$. In Example 4.19 we saw that $Q_3 \hookrightarrow \mathcal{G}'$, thus we also have that $U \hookrightarrow \mathcal{G}'$. ∎

5 Termination and Complexity

In this section, we consider the complexity of the query answering method which we have developed in the previous sections, and we shall prove our main result concerning the data complexity of query answering in $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$.

To this end, we shall first derive a bound on the size of the completion forests in $\mathsf{ccf}_n(\mathbb{F}_K)$ for a $\mathcal{SHIQ}$ knowledge base K, and then of the completion graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ for a $\mathcal{SHOIQ}$ knowledge base K. Since the forest part of a $\mathcal{SHOIQ}$ completion graph corresponds to $\mathcal{SHIQ}$ a completion forest, we can

treat both structures in a similar way. First we will obtain a bound on the possible size of an n-blocked variable tree, which applies both to the trees in a completion forest $\mathcal{F}$ and in the forest-part of a completion graph $\mathcal{G}_f$.

We point out, however, that conjunctive query answering is intractable with respect to combined complexity already on very small completion graphs and completion forest, and in fact even for a fixed completion graph which consists of few nodes. This is shown in the proof of the next proposition.

Proposition 5.1 *Let $\mathcal{G}$ be a (fixed) completion graph in $\mathbb{G}_K$ (or a forest in $\mathbb{F}_K$), and let Q be a given CQ. Deciding $Q \hookrightarrow \mathcal{G}$ is* NP*-hard.*

Proof. Finding a mapping $Q \hookrightarrow \mathcal{G}$ has the same query complexity as evaluating a conjunctive query over a database (given by the ABox), which is an NP-hard problem. To verify this, it suffices to consider the completion graph $\mathcal{G}_{col}$ 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: the nodes in G are the variables in Q and for each arc $\langle x, y\rangle$ in G there is a literal $E(x, y)$ in Q. Then Q can be mapped into $\mathcal{G}_{col}$ iff G is 3-colorable. □

Note that when Q is fixed, the test $Q \hookrightarrow \mathcal{G}$ can be done in time polynomial in the size of $\mathcal{G}$ by simple methods, since only a polynomial number of candidate mappings needs to be checked, This will be relevant to prove a tight upper bound in data complexity.

5.1 Bounding the size of completion forests and graphs

In what follows, for a knowledge base K, we will denote by $\mathbf{c}$ the cardinality of $\mathsf{clos}(K) \cup \mathcal{C}_q$, by $\mathbf{r}$ the cardinality of $\mathbf{R}_K$, and by $\mathbf{m}$ the maximum number n occurring in a concept of the form $\leq n\, R.C$ or $\geq n\, R.C$ in K. Furthermore, $|\mathcal{A}|$ denotes the number of assertions in $\mathcal{A}$.

Claim 5.2 Let T_n be the maximal number of non-isomorphic n-trees in a completion forest or in the forest part of a completion graph for K. Then, $T_n = O((2^{2\cdot\mathbf{c}}(\mathbf{c}\cdot\mathbf{m})^{\mathbf{r}})^{(\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r})^n})$.

Proof. There are only variable nodes in the tree-shaped part, so the label of each such node is a subset of $\mathsf{clos}(K) \cup \mathcal{C}_q$ and there are at most $2^{\mathbf{c}}$ different such labels. 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 $(T_{n-1})^x$ (ordered) combinations of trees of depth $(n-1)$ rooted at the successors of v.

There are three generating rules: the $\exists$-rule, the $\geq$-rule and the $o?$-rule. Since the nodes introduced by the $o?$-rule are individual nodes (they initially contain a nominal label), variable nodes are only introduced by applying the $\exists$-rule and the $\geq$-rule. Only concepts of the form $\exists R.S$ or $\geq n\, R.C$ trigger the application these rules, and there are at most $\mathbf{c}$ such concepts. Each time one such rule it is applied, it generates at most $\mathbf{m}$ variable R-successors for each role R. Note that if a node v is identified with another by a shrinking rule, then the rule application which led to the generation of v will never be repeated [24], so a generating rule can be applied to each node at most $\mathbf{c}$ times. This gives a bound of $\mathbf{c}\cdot\mathbf{m}$ variable R-successors for each role.

The number of R-successors of a node, x, might range from 0 to $\mathbf{c}\cdot\mathbf{m}$, and for each x, we have at most $(T_{n-1})^{(\mathbf{c}\cdot\mathbf{m})}$ combinations of trees of depth $(n-1)$. So, each node can have at most $(\mathbf{c}\cdot\mathbf{m})(T_{n-1})^{(\mathbf{c}\cdot\mathbf{m})}$

combinations of trees of depth $(n-1)$ as successors, if we consider one single role. Since for every role in $\mathbf{R}_K$ at most this number of trees can be generated, there is a bound of $((\mathbf{c}\cdot\mathbf{m})(T_{n-1})^{(\mathbf{c}\cdot\mathbf{m})})^{\mathbf{r}}$ combinations of trees of depth $(n-1)$ for the successors of each node. The number of different roots of an n-tree is bounded by $2^{\mathbf{c}}$. We thus obtain as an upper bound on the number of non isomorphic n-trees

$$T_n = O(2^{\mathbf{c}}((\mathbf{c}\cdot\mathbf{m})(T_{n-1})^{(\mathbf{c}\cdot\mathbf{m})})^{\mathbf{r}}).$$

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

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

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

$$T_n = O((2^{2\mathbf{c}}(\mathbf{c}\cdot\mathbf{m})^{\mathbf{r}})^{(\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r})^n})$$

□

Claim 5.3 Let T be a variable tree in a completion forest $\mathcal{F} \in \mathbb{F}_K$ or in the forest part $\mathcal{G}_f$ of a completion graph $\mathcal{G} \in \mathbb{G}_K$. The number of nodes in T is bounded by

$$O((\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r})^{1+n\cdot(2^{2\cdot\mathbf{c}}\cdot(\mathbf{c}\cdot\mathbf{m})^{\mathbf{r}})^{(\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r})^n}})$$

Proof. The claim follows from the following properties:

i) The outdegree of T is bounded by $\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r}$.
As shown above, there are at most $\mathbf{c}\cdot\mathbf{m}$ variable R-successors for each role R, and there are $\mathbf{r}$ roles.

ii) The depth of T is bounded by $d = (T_n + 1)\cdot n$.
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 $(T_n + 1)\cdot n$ to a node v in T, 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 T is bounded by $O((\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r})^{d+1})$.

□

There can be one such tree rooted at each individual node. Since for a $\mathcal{SHIQ}$ completion forest there is at most one individual node for each ABox individual, then we easily get a bound on the size of a completion forest for a $\mathcal{SHIQ}$ knowledge base.

Lemma 5.4 *Let K be a $\mathcal{SHIQ}$ knowledge base. The number of nodes in a completion forest $\mathcal{F} \in \mathbb{F}_K$ is bounded by*

$$O(|\mathbf{I}_K|\cdot(\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r})^{1+n\cdot(2^{2\cdot\mathbf{c}}\cdot(\mathbf{c}\cdot\mathbf{m})^{\mathbf{r}})^{(\mathbf{c}\cdot\mathbf{m}\cdot\mathbf{r})^n}})$$

For the $\mathcal{SHOIQ}$ case, we first need to derive a bound on the number of individual nodes. The arguments are essentially the same as in [23]. The bounds we derive are not the same, since they depend on the maximal depth of a variable node.

Claim 5.5 Let K be a $\mathcal{SHOIQ}$ knowledge base. The number of individual nodes in a completion graph $\mathcal{G} \in \mathbb{G}_K$ is bounded by

$$O(|\mathbf{I}_K| \cdot (\mathbf{c} \cdot \mathbf{m})^{1+n \cdot (2^{2 \cdot \mathbf{c}} \cdot (\mathbf{c} \cdot \mathbf{m})^{\mathbf{r}})^{(\mathbf{c} \cdot \mathbf{m} \cdot \mathbf{r})^n}})$$

Proof (sketch). The argument is exactly as in [23] (Proof of Lemma 6, item 4). Simply replace λ (maximal depth of a variable node) by d. Due to the the strategy for the rule application and the preconditions of the $o?$-rule, this implies that the $o?$-rule can only be applied to nominal nodes of level below d. This and some counting gives a bound of $O(|\mathbf{I}_K|(\mathbf{cm})^d)$. For more details, we refer to [24]. □

Lemma 5.6 *Let K be a $\mathcal{SHOIQ}$ knowledge base. The number of nodes in a completion graph $\mathcal{G} \in \mathbb{G}_K$ is bounded by*

$$O(|\mathbf{I}_K|^2 \cdot (\mathbf{c} \cdot \mathbf{m} \cdot \mathbf{r})^{2+2 \cdot n \cdot (2^{2 \cdot \mathbf{c}} \cdot (\mathbf{c} \cdot \mathbf{m})^{\mathbf{r}})^{(\mathbf{c} \cdot \mathbf{m} \cdot \mathbf{r})^n}})$$

Proof. Follows from Claims 5.3 and 5.5. □

Note that a $\mathcal{SHOIQ}$ completion forest may be quadratically larger than a $\mathcal{SHOIQ}$ completion graph.

5.2 Complexity of the algorithm for $\mathcal{SHIQ}$

Let K be a $\mathcal{SHIQ}$ knowledge base. We will determine the complexity of deciding $K \models Q$ and $K \models U$ for a CQ Q and an UCQ U respectively.

By $||K, Q||$ we will denote the total size of (the string encoding) the knowledge base K and the query Q. Note that $\mathbf{m}$ 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 Q and all of K except for $\mathcal{A}$ is fixed, then $\mathbf{m}$ 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.

From Lemma 5.4, we obtain the following corollaries.

Corollary 5.7 *(i) If n is polynomial in $||K, Q||$, then the maximum number of nodes in a completion forest $\mathcal{F} \in \mathbb{F}_K$ is triple exponential in $||K, Q||$.*
(ii) If Q and all of K except for $\mathcal{A}$ is fixed and n is a constant, then the maximum number of nodes in a completion forest $\mathcal{F} \in \mathbb{F}_K$ is linear in $|\mathcal{A}|$.

Proposition 5.8 *The expansion of $\mathcal{F}_K$ into some $\mathcal{F} \in \mathbb{F}_K$ terminates in time triple exponential in $||K, Q||$ if n is polynomial in $||K, Q||$. If Q and all of K except for $\mathcal{A}$ is fixed, and n is a constant, then the expansion of $\mathcal{F}_K$ into some $\mathcal{F} \in \mathbb{F}_K$ terminates in time polynomial in $|\mathcal{A}|$.*

Proof. We have given a bound on the size of $\mathcal{F}$ in Lemma 5.4. The proposition follows from this and a polynomial bound on the number of times that each expansion rule can be applied to a node while expanding the forest. The $\leq$-rule is the only shrinking rule. If this rule is not applied, then it is clear that all rules extend the node labels and possibly add new nodes to the forest. They are applied at most once for each concept in $\mathcal{L}(v)$ for each node v, thus their application is bounded by $\mathbf{c}$. If the $\leq$-rule is applied to a node v, then a neighbor w of v is merged into a neighbor w' of v. Since w' inherits the labels and inequalities of w, the rule application that led to the generation of w' will never be repeated. Clearly, the times the $\leq$-rule can be applied is bounded by the branching degree of $\mathcal{F}$. Thus the application of all rules to a node will be polynomially bounded also under the merging of nodes caused by the $\leq$-rule. □

Checking whether $Q \hookrightarrow \mathcal{F}$ can be done by naive methods in time single exponential in the size of Q. For an $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K)$ with M nodes and a query Q with n_Q literals, the naive search space has $M^{2 \cdot n_Q}$ candidate assignments, and each one can be polynomially checked. So, if M is triple exponentially bounded in $||K, Q||$, then also $M^{2 \cdot n_Q}$ is triple exponentially bounded in $||K, Q||$. On the other hand, the test $Q \hookrightarrow \mathcal{F}$ can be done in time polynomial in the size of $\mathcal{F}$ when Q is fixed.

Therefore, we obtain the following result:

Theorem 5.9 *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:*

1. *in* CO-3NEXPTIME *w.r.t. combined complexity, for both unary and binary encoding of number restrictions in K.*
2. *in* CO-2NEXPTIME *w.r.t. combined complexity for a fixed U if number restrictions are encoded in unary.*
3. *in* CONP *w.r.t. data complexity.*

Proof. If $K \not\models U$, then there is a completion forest $\mathcal{F} \in \mathsf{ccf}_{n_U}(\mathbb{F}_K)$ such that $U \not\hookrightarrow \mathcal{F}$. If this $\mathcal{F}$ is guessed non-deterministically then, by Proposition 5.8, it can be obtained in time 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. Therefore, non-entailment of U can be checked in 3NEXPTIME, entailment in CO-3NEXPTIME and item 1 holds.

Item 2 follows from the above arguments, modified according to the observation that $\mathbf{m}$ does not occur in the uppermost exponent of the bound of the forest size, and thus any $\mathcal{F}$ in $\mathsf{ccf}_{n_U}(\mathbb{F}_K)$ can be obtained in double exponential time.

As for Item 3, 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. By Proposition 5.8, we know that every completion forest $\mathcal{F} \in \mathsf{ccf}(\mathbb{F}_K)$ can be nondeterministically generated in polynomial time. Since deciding whether $U \hookrightarrow \mathcal{F}$ is polynomial in the size of $\mathcal{F}$, $K \models U$ is in CONP. □

We note that the test $Q \hookrightarrow \mathcal{F}$ might also be done in time polynomial in the size of $\mathcal{G}$ when Q 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 and there are no arcs connecting variable nodes to individuals in the cloud), then the complexity of the mapping corresponds to evaluating a conjunctive query over a tree-shaped database, which is known to be polynomial in the size of the database in certain cases [18].

5.3 Complexity of the algorithm for $\mathcal{SHOIQ}$

In this section, we will discuss the data and combined complexity of deciding entailment of conjunctive queries and unions of conjunctive queries on $\mathcal{SHOIQ}$. In contrast to $\mathcal{SHIQ}$, the distinction between extensional and intensional data in $\mathcal{SHOIQ}$ is not so clear. The presence of nominals naturally expresses extensional information in the TBox axioms. Moreover, the ABox of a $\mathcal{SHIQ}$ knowledge base can be expressed as new TBox axioms, and ABox+TBox reasoning is polynomially reducible to reasoning w.r.t. a TBox only [38]. Recall that for any concept assertion $A(a)$ in the ABox, we can equivalently add to the TBox the axiom $\{a\} \sqsubseteq A$, and for any role assertion $P(a, b)$ the axiom $\{a\} \sqsubseteq \exists P.\{b\}$.

For determining data complexity, the TBox, RBox, and the query are fixed. If in addition we would assume that $\mathcal{SHOIQ}$ ABoxes are always empty (as we did when describing the algorithm), then query entailment can be done trivially in constant time. Clearly, this setting seems not appropriate.

To obtain a more interesting result, we will assume that initially an arbitrary $\mathcal{SHOIQ}$ knowledge base K is given. This K, as defined in Definition 2.3, comprises a TBox, an RBox, and an ABox. To decide query entailment w.r.t. to this K, we will replace the ABox by new TBox axioms to obtain a knowledge base K' with an empty ABox as described above. The complexity bounds will then be given w.r.t. to the initial K, i.e., for data complexity we will consider the size of the original $\mathcal{A}$ as variable.

In this section, K will denote a $\mathcal{SHOIQ}$ knowledge base $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, and K' will denote the knowledge base $\langle \mathcal{T}', \mathcal{R}, \emptyset \rangle$ obtained by expressing $\mathcal{A}$ within $\mathcal{T}'$. We will denote by $||K, Q||$ the total size of the string encodings the knowledge bases K and the query Q, and by $||K', Q||$ the size of the string encoding K' and Q.

Because of the axioms added to express the ABox assertions, $\mathcal{T}'$ may be larger than $\mathcal{T}$. One axiom is added to $\mathcal{T}$ for each assertion in $\mathcal{A}$, and there will be at most one new nominal in $\mathbf{N}_{K'}$ for each individual in $\mathcal{A}$. Hence, $||K', Q||$ is polynomially (in fact, linearly) bounded in $||K, Q||$, and $|\mathbf{I}_{K'}|$ polynomially in $|\mathbf{I}_K|$.

By Lemma 5.6, we know that the maximum number of nodes in a completion graph $\mathcal{G} \in \mathbb{G}_{K'}$ is triple exponential in $||K', Q||$ if n is polynomial on $||K', Q||$, and that it is polynomial (actually, quadratic) in $|\mathcal{A}|$ if n is a constant. From this, together with the polynomial bound on the size of $||K', Q||$ and $|\mathbf{I}_{K'}|$, we easily obtain:

Corollary 5.10 *(i) If n is polynomial on $||K, Q||$, then the maximum number of nodes in a completion graph $\mathcal{G} \in \mathbb{G}_{K'}$ is triple exponential in $||K, Q||$.*
(ii) 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 $\mathcal{G} \in \mathbb{G}_{K'}$ is polynomial in $|\mathcal{A}|$.

Proposition 5.11 *The expansion of $\mathcal{G}'_K$ into some $\mathcal{G} \in \mathbb{G}_{K'}$ 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 $\mathcal{G}'_K$ into some $\mathcal{G} \in \mathbb{G}_{K'}$ terminates in time polynomial in $|\mathcal{A}|$.*

Proof. The argument is similar to the one in the proof of Proposition 5.8 for the nodes in the forest part of the completion graph. Additionally, we must observe that :

1. The bound in the rule applications holds also in the presence of the new shrinking rule. Again, if a node w is merged into a node w', then w' will inherits the labels and inequalities of w, as well as all its neighbors that are not variable successors (which are removed by prune). This will ensure that the rule application that led to the generation of w' will never be repeated.

2. For each nominal node v, the $o?$-rule can only be applied once for each concept of the form $\leq n\, S.C$ in the label of v.

Thus the number of rule applications is polynomially bounded in the number of nodes of $\mathcal{G}$.

Note that, in total, the maximal number of applications of the $o?$-rule is $|\mathbf{I}_K| \cdot \mathbf{c} \cdot (\mathbf{c} \cdot \mathbf{m})^d$ (i.e., triple exponential in $||K, Q||$ if n is polynomial on $||K, Q||$). The arguments are similar to those of the proof of Claim 5.5. The bound is obtained from the one given in [23] by replacing the maximal depth λ by d. □

Theorem 5.12 *Given a $\mathcal{SHOIQ}$ knowledge base K and a union of conjunctive queries U in which all roles are simple, deciding whether $K \models U$ is:*

1. *in* CO-3NEXPTIME *w.r.t. combined complexity, for both unary and binary encoding of number restrictions in* K.

2. *in* CO-2NEXPTIME *w.r.t. combined complexity for a fixed* U *if number restrictions are encoded in unary.*

3. *in* CONP *w.r.t. data complexity.*

Proof (sketch). The argument is analogous to that in the proof of Theorem 5.9. By Proposition 5.11, $K \not\models U$ can be decided in 3NEXPTIME, and in NP w.r.t. data complexity. For fixed Q, unary encoding yields an exponential drop in the size of the completion graph (and thus in the number of rule applications to obtain it) w.r.t. combined complexity. □

5.4 Data complexity

The upper bound for data complexity given in Theorems 5.9 and 5.12 are worst-case optimal. In [15], CONP-hardness was proved for instance checking over $\mathcal{ALE}$ knowledge bases, and recently this result has been extended to description logics which are even less expressive than $\mathcal{ALE}$ [10]. This allows us to state the following main result.

Theorem 5.13 *On knowledge bases in any description logic from* $\mathcal{AL}$ *to* $\mathcal{SHOIQ}$*, answering unions of conjunctive queries in which all roles are simple is* CONP*-complete w.r.t. data complexity.*

This result provides an exact characterization of the data complexity of UCQs for a wide range of description logics. Note that for the most expressive one, $\mathcal{SHOIQ}$, the result is highly significant, since such a logic is an extension of $\mathcal{SHOIN}$, the description logic counterpart of the standard ontology language OWL-DL [22, 36]. It is interesting to see that once we include in the description logic universal quantification, one of the basic constructs of description logics, many more constructs can be added without affecting worst-case data complexity. Also, this result extends two previous CONP-completeness results w.r.t. data complexity, which are not obvious: On the one hand, in [32], the same bound was proved for answering UCQs over $\mathcal{ALCNR}$ knowledge bases. Now we extend this result to a description logic including role hierarchies, as well as inverse (and transitive) roles. On the other hand, [30] showed a CONP-upper bound for data complexity of query answering in the quite expressive description logic $\mathcal{SHIQ}$, already a logic lacking the finite model property. The result was established for atomic queries only, but can be immediately extended to tree shaped queries, since these admit a representation as a description logic concept (e.g., by making use of the notion of tuple-graph of [11], or via rolling up [27]). Instead, its extension to arbitrary conjunctive queries, which is what we have done in the present work, proved to be surprisingly involved from a technical point of view.

5.5 Combined complexity

Theorems 5.9 and 5.12 do not provide optimal upper bounds with respect to the combined complexity of query answering. The main reason is that the tableaux algorithms in [26] and [23], which we extended, are also not worse case optimal. They are both nondeterministically double exponential, while satisfiability of a knowledge base is an EXPTIME-complete problem for $\mathcal{SHIQ}$ [40] and NEXPTIME-complete for $\mathcal{SHOIQ}$ [39]. It is well known that often tableaux algorithms for expressive DLs do not provide optimal complexity

upper bounds. However, they are easy to implement and amenable for optimizations [4]. Moreover, there are efficient reasoners available that implement these algorithms [21, 19].

We want to point out that, in our algorithm, the witness of a blocked variable must be its ancestor. 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. We use the stricter conditions for blocking in order to make them closer to the conventional ones in DL tableaux, where it is usually required that the blocking and the blocked variable are on the same path. Despite the fact that this condition actually increases the overall complexity of the algorithm, it is imposed for practical reasons, since it is consider better for implementation. If this condition is relaxed, blocking may occur sooner and the resulting completion graph/forest may be exponentially smaller than the one we have described. This exponential drop applies also to the satisfiability tableaux algorithms like in [26] and in [23]. With this relaxed condition, we would obtain the same complexity upper bounds as those given in [32]. In fact, the absence of this additional condition of 'blocking on the same path' is the actual reason why the bounds in [32] are exponentially lower than the ones we obtained.

Finally, from the results in [30] we know that a 2EXPTIME bound for $\mathcal{SHIQ}$ can be achieved. This bound coincides with the one given in [11] for containment of conjunctive queries over $\mathcal{DLR}$. It remains an open question whether this bound is tight. Our algorithm, even if we relax the blocking conditions as described above, would yield a non-optimal CO-2NEXPTIME worst-case complexity in the combined case. As for $\mathcal{SHOIQ}$, to the best of our knowledge, this is the only existing result concerning complexity of answering conjunctive query answering, even without transitive roles in queries.

6 Conclusion

In this paper, we have studied answering conjunctive queries (CQs) and union of conjunctive Queries (UCQs) over knowledge bases in the expressive Description Logics (DLs) $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$, where we have focused on the issue of data complexity, i.e., measuring the complexity of query answering with respect to the size of the ABox of the knowledge base while the other parts are fixed. This setting is gaining importance since DL knowledge bases are more and more used also for representing data repositories, especially in the context of the Semantic Web and in Enterprise Application Integration.

Generalizing a technique presented in [32] for a DL which is far less expressive than $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$, we have developed novel tableaux-based algorithms for CQ answering in these DLs. These algorithms manage the technical challenges caused by the simultaneous presence of inverse roles, number restrictions, and general knowledge bases, leading to DLs which are lacking the finite model property. To this end, we have developed suitable blocking conditions which ensure termination of the algorithm. They are more involved than previous blocking conditions in [26], and parameterized with the depth of trees which must be considered in blocking. Query answering itself is then accomplished by a technique which maps the query to completion graphs, which are constructed using the tableaux-style rules, of bounded depth, provided that queries have only simple roles.

The algorithms which we have developed are worst case optimal in data complexity, and allows us to characterize the data complexity of answering CQs and UCQs for a wide range of DLs, including very expressive ones. Namely, for each description logic ranging from $\mathcal{AL}$ to $\mathcal{SHOIQ}$, both answering CQs and UCQs having only simple roles is CONP-complete w.r.t. data complexity. This closes the gap between the known CONP lower bound and the best known EXPTIME upper bound for even weaker DLs, providing a negative answer to the open issue whether the data complexity of expressive description logics will similarly increase as their combined complexity.

Some comments on our results in this paper are in order. We first point out that our method for query answering can also be exploited to the problem of deciding query containment, i.e., given two queries Q_1 and Q_2, is it true that $K \models Q_2$ if $K \models Q_1$, for each knowledge base K? By virtue of the correspondence between query containment and query answering [1], one can adapt the algorithms which we have presented to decide containment of CQs, and furthermore also the containment of UCQs (which follows from easy relationships between CQs and UCQs), for both $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$. As a simple consequence, we thus obtain that the equivalence of queries in $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ is decidable. This result may be exploited for query optimization, and to the best of our knowledge is the first result in this direction for (union of) conjunctive queries in expressive DLs.

Several issues remain for further work. The query languages considered in this paper do not allow arbitrary roles, but supposed that roles are simple. This constraint, however, is also adopted in [29]. To our knowledge, only the recent [17] deals with transitive roles in CQs, but on the far less expressive DL $\mathcal{SHQ}$, which does not allow inverse roles and nominals and, differently from $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$, has the finite model property. A natural question is whether the results in this paper extend to a query language in which arbitrary roles may occur in queries. Currently, this is open, and to our knowledge it is yet unknown whether conjunctive query answering in $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ remains decidable in this setting. It remains unclear whether the algorithm which we have presented here can be exploited, since the presence of transitive roles imposes difficulties in establishing a bound on the depth of completion forests which need to be considered for answering a given query. It also remains to explore whether the proposed technique can be applied to even more expressive description logics, for example, containing reflexive-transitive closure in the TBox (in the style of PDL), or to more expressive query languages. We note, however, that including inequality atoms in CQs is not feasible; as follows from results in [11], answering CQs with inequalities over $\mathcal{SHIQ}$ knowledge bases is undecidable.

Apart from the issue of data complexity for expressive DLs, also the combined complexity remains for further investigation. It follows from [28] that the problem is in 2ExpTime for $\mathcal{SHIQ}$. Hence, the bound established above in Theorem 5.9 is not tight, since we build on tableaux algorithms that are not optimal in the worst case. Indeed, a more relaxed blocking condition can be used, where the witness of the root of a blocked tree need not necessarily be its ancestor. This optimization yields an exponential drop in the worst-case size of the forest, thus obtaining a co-2NExpTime upper bound. Note that this can also be done in the standard tableau algorithms for satisfiability checking, but might not be convenient from an implementation perspective. Further optimization of the algorithms which we have presented here may be considered following the ideas in [14].

Finally, it remains as an interesting issue whether other techniques may be applied to derive similar results as those in this paper. For instance, whether resolution-based techniques as in [28, 30] or techniques based on tree automata can be fruitfully applied. While the latter may look conceptually appealing, in particular in the light of the completion forests employed by our algorithms, it remains less clear how with respect to the data complexity the contribution of the ABox may be singled out, which was easy in the tableaux-style algorithms which we have presented.

Acknowledgments

The authors thank Ian Horrocks and Birte Glimm for many fruitful and stimulating discussions. They are very grateful to them for pointing out errors in preliminary work to this paper, and for helpful suggestions for improvement.

A Appendix

Proof of Lemma 3.13 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 clearly $\mathcal{I} \models \mathcal{F}'$ 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 is applied to a variable v in $\mathcal{F}$, then there is some concept $\leq n\, S.C$ in $\mathcal{L}(v)$ and v has more than n S-neighbors $w_1, \ldots, w_n, w_{n+1}$ that are labeled with C. Since $\mathcal{I} \models \mathcal{F}$, then $v^{\mathcal{I}} \in (\leq n\, S.C)^{\mathcal{I}}$, which implies that in there are at most $o_1, \ldots, o_n$ elements such that $\langle v^{\mathcal{I}}, o_i \rangle \in S^{\mathcal{I}}$ and $o_i \in C^{\mathcal{I}}$. Thus there are w_i and w_j, S-neighbors of v and instances of C, with $i \neq j$ and $w_i^{\mathcal{I}} = w_j^{\mathcal{I}}$. This implies that $w_i \not\approx w_j \notin \mathcal{F}$, and the nodes can be merged as a result of the rule application. Hence it holds that $\mathcal{I} \models \mathcal{F}'$, where $\mathcal{F}$ is obtained from $\mathcal{F}$ by merging node w_i into w_j.

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. □

Note that this proof is very similar to the proof for soundness for $\mathcal{SHIQ}$, given in detail in [25]. Intuitively, the difference is that they define a mapping π from the forest nodes into the elements of the tableau. Since we are doing the 'steering' of the algorithm directly with the model, the interpretation function itself maps nodes to elements of the domain. In the same way as the proof of soundness from [25] is extended to $\mathcal{SHOIQ}$, this proof can be extended to the following Lemma.

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

Proof. For the $\mathcal{SHIQ}$ rules, the proof of Lemma 3.13 holds, just replace $\mathcal{F}$ by $\mathcal{G}$. For the o-rule, it is applicable when $\{a\} \in \mathcal{L}(v) \cap \mathcal{L}(v')$ for some nominal $\{a\}$ and two nodes v and v'. Since $\mathcal{I} \models \mathcal{F}$, it must

be the case that $v^{\mathcal{I}} = v'^{\mathcal{I}} = a$, therefore v can be merged into v' to obtain $\mathcal{F}'$ and $\mathcal{I} \models \mathcal{F}'$. Finally, for the $o?$-rule, it is only applicable to v when $\leq n\, S.C \in \mathcal{L}(v)$ and there is a v' S-neighbor of v with $C \in \mathcal{L}(v')$. If $m = 1$ is guessed, then one new node w will be generated in $\mathcal{F}'$ with $\mathcal{L}(w) := \{C, \{w\}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$. Since $\{C\} \cup \mathsf{gcon}(K, \mathcal{C}_q) \subseteq \mathcal{L}(v')$ and v' is an S-neighbor of v, we can extend $\mathcal{I}$ to $\mathcal{I}'$ by setting $w^{\mathcal{I}} = v'$, and then $\mathcal{I}' \models \mathcal{F}'$ holds. □

Using Lemma 4.9 and Lemma A.1 a proof of the next proposition is obtained from the proof of Proposition 3.14 (by replacing 'forests' by 'graphs' and $\mathcal{F}$ by $\mathcal{G}$).

Proposition A.2 *Let $n \geq 0$. For every $\mathcal{I}$ such that $\mathcal{I} \models K$, there is some $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ and some $\mathcal{I}'$ that is an extension of $\mathcal{I}$ such that $\mathcal{I}' \models \mathcal{G}$.*

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pp. 254–265, 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 P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pp. 452–457, 1991.

[4] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

[5] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.

[6] A. Borgida and R. J. Brachman. Conceptual modeling with description logics. In Baader et al. [2], chapter 10, pp. 349–372.

[7] D. Calvanese and G. De Giacomo. Expressive description logics. In Baader et al. [2], chapter 5, pp. 178–218.

[8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-147/`, 2005.

[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pp. 602–607, 2005.

[10] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, 2006.

[11] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pp. 149–158, 1998.

[12] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pp. 386–391, 2000.

[13] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[14] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Information and Computation*, 160(1–2):117–137, 2000.

[15] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[16] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and System Sciences*, 18:194–211, 1979.

[17] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for description logics with transitive roles. In *Proc. 2006 Description Logic Workshop (DL 2006)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/`, 2006.

[18] G. Gottlob, C. Koch, and K. U. Schulz. Conjunctive queries over trees. In *Proc. 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, pp. 189–200, 2004.

[19] V. Haarslev and R. Möller. RACER system description. In *Proc. Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pp. 701–705. Springer, 2001.

[20] J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[21] I. Horrocks. The FaCT system. In H. de Swart, editor, *Proc. 7th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pp. 307–312. Springer, 1998.

[22] 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):7–26, 2003.

[23] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pp. 448–453, 2005.

[24] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. Technical report, Department of Computer Science, University of Manchester, 2005. Available at `http://www.cs.man.ac.uk/~sattler/publications/shoiq-tr.pdf`.

[25] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pp. 161–180. Springer, 1999.

[26] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In D. McAllester, editor, *Proc. 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pp. 482–496. Springer, 2000.

[27] I. Horrocks and S. Tessaris. A conjunctive query language for description logic ABoxes. In *Proc. 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pp. 399–404, 2000.

[28] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. 11th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004)*, pp. 21–35, 2004.

[29] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}$-description logic to disjunctive datalog programs. In *Proc. 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pp. 152–162, 2004.

[30] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pp. 466–471, 2005.

[31] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pp. 233–246, 2002.

[32] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[33] C. Lutz. Description logics with concrete domains: A survey. In P. Balbiani, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logics*, volume 4. King's College Publications, 2003.

[34] M. Ortiz de la Fuente, D. Calvanese, T. Eiter, and E. Franconi. Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics. In *Proceedings 21th National Conference on Artificial Intelligence (AAAI '06), July 16-23, 2006, Boston*. AAAI Press, 2006.

[35] M. Ortiz de la Fuente, D. Calvanese, T. Eiter, and E. Franconi. Data Complexity of Answering Unions of Conjunctive Queries in SHIQ. In *Proceedings 2006 International Workshop on Description Logics (DL2006), The Lake District of the UK, May 30-June 1, 2006*, 2006.

[36] 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. Available at `http://www.w3.org/TR/owl-semantics/`.

[37] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intelligent Information Systems*, 2:265–278, 1993.

[38] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

[39] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. of Artificial Intelligence Research*, 12:199–217, 2000.

[40] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[41] M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pp. 137–146, 1982.