Characterizing Data Complexity for Conjunctive Query Answering in Expressive Description Logics*

Magdalena Ortiz[1,2], **Diego Calvanese**[1], **Thomas Eiter**[2]

[1] Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano, Italy
calvanese@inf.unibz.it,
magdalena.ortiz@stud-inf.unibz.it

[2] Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, Vienna, Austria
eiter@kr.tuwien.ac.at

Abstract

Description Logics (DLs) are the formal foundations of the standard web ontology languages OWL-DL and OWL-Lite. 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 Conjunctive Queries (CQs); (*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 has been characterized only for answering atomic queries, and was still open for answering CQs in expressive DLs. We tackle this issue and prove a tight CONP upper bound for the problem in $\mathcal{SHIQ}$, as long as no transitive roles occur in the query. We thus establish that for a whole range of DLs from $\mathcal{AL}$ to $\mathcal{SHIQ}$, answering CQs with no transitive roles has CONP-complete data complexity. We obtain our result by a novel tableaux-based algorithm for checking query entailment, inspired by the one in [19], but which manages the technical challenges of simultaneous inverse roles and number restrictions (which leads to a DL lacking the finite model property).

Introduction

Description Logics (DLs) [2] are specifically designed for representing structured knowledge by concepts (i.e., classes of objects) and roles (i.e., binary relationships between classes). They gained increasing attention recently as the formal foundation for the standard Web ontology languages [11]. In fact, OWL-Lite and OWL-DL are syntactic variants of the DLs $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$, respectively [12, 21]. In the Semantic Web and domains such as Enterprise Application Integration and Data Integration, ontologies provide a high-level, conceptual view of the relevant information. However, they are increasingly seen also as a mechanism to access and query data repositories.

*This work was partially supported by the Austrian Science Funds (FWF) project P17212 and by the European Commission under project REWERSE (IST-2003-506779) and FET project TONES (FP6-7603).

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:

(*i*) On the one hand, a DL should have sufficient expressive power to capture common constructs used in data modeling [4]. This calls for *expressive DLs* [5, 3], 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 relationships in general). Such concepts are then used in the intentional component (called TBox) of a knowledge base, which contains inclusion assertions between concepts and roles, while the extensional component (called ABox) contains assertions about the membership of individuals to concepts and roles. A notable example of such an expressive DL is $\mathcal{SHIQ}$, which in addition allows for asserting the transitivity of certain roles.

(*ii*) On the other hand, the data underlying an ontology should be accessed using well established and flexible mechanisms such as those provided by database query languages. This goes well beyond the traditional inference tasks involving objects in DL-based systems, like *instance checking* [10]. 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 queries, provide a good tradeoff between expressive power and nice computational properties, and thus are adopted as core query language in several contexts, such as data integration [18].

(*iii*) Finally, data repositories can be very large and are usually much larger than the intentional level. Therefore, the contribution of the extensional level (i.e., the data) to the complexity of inference must be singled out, and one must pay attention to optimizing inference techniques with respect to data size, as opposed to the overall size of the knowledge base. In databases, this is accounted for by *data complexity* of query answering [23], where the relevant parameter is the size of the data, as opposed to *combined complexity*, which additionally considers the size of the query and of the schema.

As for data complexity of DLs, [10] showed that instance checking is CONP-hard already in the rather weak DL $\mathcal{ALE}$, and [7] that CQ answering is CONP-hard in the yet weaker

DL $\mathcal{AL}$. For suitably tailored DLs, CQ answering is polynomial (actually LOGSPACE) in data complexity [6, 7]; see [7] for an investigation of the NLOGSPACE, PTIME, and CONP boundaries.

For expressive DLs (with the features above, notably inverse roles), TBox+ABox reasoning has been studied extensively using a variety of techniques ranging from reductions to Propositional Dynamic Logic (PDL) (see, e.g., [8, 5]) over tableaux [3, 14] to automata on infinite trees [5, 22]. For many such DLs, the combined complexity of TBox+ABox reasoning is EXPTIME-complete, including $\mathcal{ALCQI}$ [5, 22], $\mathcal{DLR}$ [8], and $\mathcal{SHIQ}$ [22]. However, until recently, little attention has been explicitly devoted to data complexity in expressive DLs. An EXPTIME upper bound for data complexity of CQ answering in $\mathcal{DLR}$ follows from the results on CQ containment and view-based query answering in [8, 9]. They are based on a reduction to reasoning in PDL, which however prevents to single out the contribution to the complexity coming from the ABox. In [19] a tight CONP upper bound for CQ answering in $\mathcal{ALCN}\mathcal{R}$ is shown. However, this DL lacks inverse roles and is thus not suited to capture semantic data models or UML. In [16, 17] a technique based on a reduction to Disjunctive Datalog is used for $\mathcal{ALCHIQ}$. For instance checking, and (by making use of the notion of tuple-graph [8] or via rolling-up [15]) also for tree-shaped CQs, it provides a (tight) CONP upper bound for data complexity, since it allows to single out the ABox contribution. This is not the case for general CQs, resulting in a non-tight 2EXPTIME upper bound (matching also combined complexity).

Summing up, a precise characterization of data complexity for 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 techniques of [19, 14], we devise a novel tableaux-based algorithm for CQ answering over $\mathcal{SHIQ}$ knowledge bases, that works under the assumption that the CQ does not contain transitive roles. Technically, to show soundness and completeness of the algorithm, we have to deal both with a novel blocking condition (inspired by the one in [19], 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 CQ answering over $\mathcal{SHIQ}$ knowledge bases is in CONP, and thus CONP-complete for all DLs ranging from $\mathcal{AL}$ to $\mathcal{SHIQ}$.

For lack of space, proofs are omitted here; they can be found in an accompanying technical report [20].

Preliminaries

We only briefly recall $\mathcal{SHIQ}$ and refer to the literature (e.g., [2]) for further details and background. We denote by $\mathbf{C}$, $\mathbf{R}$, $\mathbf{R}_+$ (where $\mathbf{R}_+ \subseteq \mathbf{R}$), and $\mathbf{I}$ the sets of *concept names*, *role names*, *transitive role names*, and *individuals* respectively. The function Inv is defined on $\mathbf{R} \cup \{P^- \mid P \in \mathbf{R}\}$ by $\mathsf{Inv}(R) = R^-$, and $\mathsf{Inv}(R^-) = R$; $\mathsf{Trans}(R)$ holds true if either $R \in \mathbf{R}_+$ or $\mathsf{Inv}(R) \in \mathbf{R}_+$.

A *role expression* R (or simply *role*) is either an atomic role name P or the inverse P^- of a role P. A *concept expression* (or simply *concept*) C is either an atomic concept name A or one of $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$, $\geq n\, S.C$, or $\leq n\, S.C$, where C and D denote concepts, R a role, S a *simple* role (see later), and $n \geq 0$ an integer. A *knowledge base* is a triple $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where

- $\mathcal{T}$, the *TBox*, is a set of *concept inclusion axioms* $C_1 \sqsubseteq C_2$;
- $\mathcal{R}$, the *role hierarchy*, is a set of *role inclusion axioms* $R_1 \sqsubseteq R_2$; and
- $\mathcal{A}$, the *ABox*, is a set of *assertions* $A(a)$, $P(a, b)$, and $a \neq b$, where A (resp., P) is an atomic concept (resp., role) and a and b are individuals.

Let $\sqsubseteq^*$ denote the reflexive and transitive closure of $\sqsubseteq$ (i.e., the *subrole relation*) over $\mathcal{R} \cup \{\mathsf{Inv}(R_1) \sqsubseteq \mathsf{Inv}(R_2) \mid R_1 \sqsubseteq R_2 \in \mathcal{R}\}$. A role S is *simple*, if for no role $R \in \mathbf{R}_+$ we have that $R \sqsubseteq^* S$. 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.

Unless stated otherwise, K will denote a knowledge base $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, $\mathbf{R}_K$ the roles occurring in K and their inverses, and $\mathbf{I}_K$ the individuals occurring in $\mathcal{A}$.

Example 1 *As a running example, we use the knowledge base* $K = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\neg A\},\ \{\},\ \{A(a)\} \rangle$.

The semantics of K is defined in terms of first-order *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the domain and $\cdot^{\mathcal{I}}$ the valuation function, as usual (without unique names assumption;[1] see [2]). $\mathcal{I}$ is a *model* of K, denoted $\mathcal{I} \models K$, if it satisfies $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$.

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 1 (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 is either a role in $\mathbf{R}_K$ or a concept in $\mathcal{C}_q$, and $\overline{Y_i}$ is a tuple of variables or individuals in $\mathbf{I}_K$ matching its arity.*

$\mathsf{VC}(Q)$ denotes the set of variables and individuals in Q.

CQs are interpreted in the standard way. An interpretation $\mathcal{I}$ is a model of Q, denoted $\mathcal{I} \models Q$, if there is a substitution $\sigma : \mathsf{VC}(Q) \to \Delta^{\mathcal{I}}$ such that $\sigma(a) = a^{\mathcal{I}}$ for each individual $a \in \mathsf{VC}(Q)$ and $\mathcal{I} \models p(\sigma(\overline{Y}))$, for each $p(\overline{Y})$ in Q. A knowledge base K *entails* Q, denoted $K \models Q$, if $\mathcal{I} \models Q$ for each model $\mathcal{I}$ of K.

[1] The unique names assumption can be easily emulated using $\not\approx$.

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

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 $\mathcal{A}$ are fixed.

An important note is that CQ answering is not reducible to knowledge base satisfiability, since the negated query is not expressible within the knowledge base. For this reason, the known algorithms for reasoning over $\mathcal{SHIQ}$ knowledge bases are insufficient.

Note that CQs have no free (i.e., distinguished) variables, so they are Boolean queries. However, this is not a limitation, since as usual we can reduce answering a CQ $Q(\overline{X})$ with distinguished variables $\overline{X}$ to answering all its ground instances $Q(\overline{c})$, where $\overline{c}$ is a tuple of individuals.

The Query Answering Algorithm

We present now a method for deciding $K \models Q$ that builds on the results of [14]. Our method works under the assumption that the CQ Q does not contain transitive roles or their superroles, and in the following we make this assumption.

Like in [14], we use *completion forests*, which are finite relational structures capturing sets of models of K. Roughly speaking, the models of K are represented by an initial completion forest $\mathcal{F}_K$. By applying tableaux-style *expansion rules* repeatedly, new completion forests are generated nondeterministically where also new individuals might be introduced. Each model of K is preserved in some of the resulting forests. Therefore, checking $K \models Q$ equals checking $\mathcal{F} \models Q$ for each completion forest $\mathcal{F}$. We will show that, for largely enough expanded $\mathcal{F}$, we can check $\mathcal{F} \models Q$ effectively via a syntactic mapping of the variables in Q to the nodes in $\mathcal{F}$. Thus, to witness that $K \not\models Q$, it is sufficient to (nondeterministically) construct a large enough forest $\mathcal{F}$ to which Q cannot be mapped. This is in effect what our algorithm does.

As customary with tableaux-style algorithms, we define suitable blocking conditions on the rules to ensure termination of forest expansion. They are inspired by those in [19], yet must handle logics that have no finite model property. They are also more involved than those in [14], which serve for satisfiability checking but not for query answering, and involve a depth parameter n which depends on Q.

Completion forests and n-blocking

A *variable tree* T is a tree whose nodes are variables except the root, which might be also an individual, and where each node v is labeled with a set of concepts $\mathcal{L}(v)$ and each arc $v \to w$ is labeled with a set of roles $\mathcal{L}(v \to w)$. 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 ψ from the nodes of T_v^n to those of $T_{v'}^n$ which preserves all labels). If, for such v and v', v' is an ancestor of v in T and v is not in $T_{v'}^n$, then we say that $T_{v'}^n$ *tree-blocks* T_v^n, that v' is a *n-witness of v in T*, and that each variable w in $T_{v'}^n$ *tree-blocks* the corresponding variable $\psi^{-1}(w)$ in T_v^n.

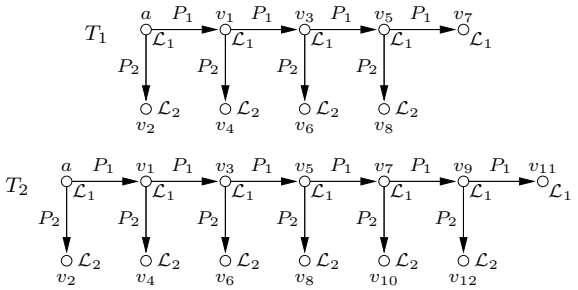

Figure 1: Trees and completion forests for the example knowledge base.

Example 3 *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); $T_{1v_1}^1$ tree-blocks $T_{1v_5}^1$; and v_1 (resp., v_3, v_4) tree-blocks v_5 (resp., v_7, v_8).*

Definition 2 *A* completion forest *(cf. [14]) for K is constituted by* (i) *a set of variable trees whose roots are the individuals in $\mathbf{I}_K$ and can be arbitrarily connected by arcs; and* (ii) *a symmetric binary relation $\not\approx$ over the nodes of the variable trees.*

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.

We next introduce the initial completion forest of a knowledge base K. For that, 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).

Definition 3 *With any knowledge base K, we associate an* initial completion forest *$\mathcal{F}_K$ as follows:*

- *The nodes are the individuals $a \in \mathbf{I}_K$, and $\mathcal{L}(a) = \{C \mid C(a) \in \mathcal{A}\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$.*

$\exists$-rule:	**if** $\exists R.C \in \mathcal{L}(v)$, v is not n-blocked and has no R-neighbor w with $C \in \mathcal{L}(w)$ **then** create a new node w with $\mathcal{L}(w) := \{C\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$ and an arc $v \to w$ with label $\mathcal{L}(v \to w) := \{R\}$
$\geq$-rule:	**if** $\geq n\, S.C \in \mathcal{L}(v)$, v is not n-blocked and has no S-neighbors $w_1, \ldots, w_n$ such that $C \notin \mathcal{L}(w_i)$ and $w_i \not\approx w_j$, for $1 \leq i < j \leq n$ **then** create new nodes $w_1, \ldots, w_n$ with $\mathcal{L}(w_i) := \{C\} \cup \mathsf{gcon}(K, \mathcal{C}_q)$, arcs $v \to w_i$ with labels $\mathcal{L}(v \to w_i) := \{S\}$, and $w_i \not\approx w_j$ for $1 \leq i < j \leq n$

Table 1: Two expansion rules from the algorithm.

- *The arc $a \to a'$ is present iff $\mathcal{A}$ contains some assertion $R(a, a')$, and $\mathcal{L}(a \to a') = \{R \mid R(a, a') \in \mathcal{A}\}$.*
- *$a \not\approx a'$ iff $a \neq a' \in \mathcal{A}$.*

If $\mathcal{A} = \emptyset$, then $\mathcal{F}_K$ contains a single node a with $\mathcal{L}(a) = \mathsf{gcon}(K, \mathcal{C}_q)$.

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

We now define a notion of blocking which depends on a depth parameter $n \geq 0$.

Definition 4 *For an integer $n \geq 0$, a node v in a completion forest $\mathcal{F}$ is* n-blocked*, if v is not a root and is either directly or indirectly n-blocked. Node v is* directly n-blocked*, if none of its ancestors is n-blocked and v is a leaf of a tree-blocked n-tree in $\mathcal{F}$. Node v is* indirectly n-blocked*, if one of its ancestors is n-blocked or some arc $v' \to v$ in $\mathcal{F}$ has empty label.*

Note that if v is n-blocked, then it is m-blocked for each $m \leq n$. Furthermore, n-blocking implies blocking as in [14] for $n = 1$, and for $n = 0$ amounts to blocking by equal node labels.

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

Starting from $\mathcal{F}_K$, we can generate new completion forests $\mathcal{F}$ by applying expansion rules in the style of tableau algorithms. We denote by $\mathbb{F}_K$ the set of all forests obtained this way. The rules make use of "n-blocking" to ensure termination. The $\exists$-rule and $\geq$-rule, which replace analogous rules in [14], are shown in Table 1. The complete set of expansion rules is given in the extended report [20]. Note that in our case newly introduced nodes must be initialized with a label containing $\mathsf{gcon}(K, \mathcal{C}_q)$.

A node v in $\mathcal{F}$ contains a *clash* if either $\{C, \neg C\} \subseteq \mathcal{L}(v)$ or $\leq n\, S.C \in \mathcal{L}(v)$ and v has 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 $i \neq j$. $\mathcal{F}$ is *clash free*, if none of its nodes contains a clash.

We call a completion forest *n-complete*, if (under n-blocking) no rule can be applied to it. We denote by $\mathsf{ccf}_n(\mathbb{F}_K)$ the set of n-complete and clash free completion forests in $\mathbb{F}_K$.

Example 6 *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 $\mathcal{F}_2 \in \mathsf{ccf}_2(\mathbb{F}_K)$.*

Models of a completion forest. If we view variables in a completion forest $\mathcal{F}$ as individuals, we can define models of $\mathcal{F}$ in terms of models of K over an extended vocabulary.

An interpretation $\mathcal{I}$ (over the extended vocabulary) is a *model* of a completion forest $\mathcal{F}$ for K, denoted $\mathcal{I} \models \mathcal{F}$, if $\mathcal{I} \models K$ and for all nodes v, w in $\mathcal{F}$ it holds that (*i*) $v^{\mathcal{I}} \in C^{\mathcal{I}}$ if $C \in \mathcal{L}(v)$, (*ii*) $\langle v^{\mathcal{I}}, w^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ if $\mathcal{F}$ has an arc $v \to w$ and $R \in \mathcal{L}(\langle v, w \rangle)$, and (*iii*) $v^{\mathcal{I}} \neq w^{\mathcal{I}}$ if $v \not\approx w \in \mathcal{F}$.

Clearly, the models of the initial completion forest $\mathcal{F}_K$ and of K coincide, and thus $\mathcal{F}_K$ semantically represents K. The following result states that the n-complete and clash-free forests for K semantically capture K (modulo new individuals). The proof shows that each expansion rule preserves the models of a forest to which it is applied.

Theorem 1 *Let $n \geq 0$. Then for each model $\mathcal{I}$ of K, there exists some $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ and a model of $\mathcal{F}$ extending $\mathcal{I}$.*

By virtue of this result, we can transfer query entailment $K \models Q$ to logical consequence of Q from completion forests as follows. For any completion forest $\mathcal{F}$ and CQ Q, let $\mathcal{F} \models Q$ denote that $\mathcal{I} \models Q$ for every model $\mathcal{I}$ of $\mathcal{F}$.

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

Checking query entailment from completion forests

We can decide $\mathcal{F} \models Q$ for an $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, by syntactically "mapping" the query Q into $\mathcal{F}$, if n is sufficiently large.

We say that Q can be *mapped into* $\mathcal{F}$, denoted $Q \hookrightarrow \mathcal{F}$, if there is a mapping μ from the variables and individuals in $\mathsf{VC}(Q)$ into the nodes of $\mathcal{F}$, such that

- $\mu(a) = a$ for each individual a,
- for each atom $C(x)$ in Q, $C \in \mathcal{L}(\mu(x))$, and
- for each atom $R(x, y)$ in Q, $\mu(y)$ is an R-neighbor of $\mu(x)$.

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

We establish now our key result, which directly leads to our query answering algorithm. In the following, let n_Q denote the number of role atoms in Q.

Theorem 3 *Let $n \geq n_Q$. Then $K \models Q$ iff for each $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, it holds that $Q \hookrightarrow \mathcal{F}$.*

The if direction is easy. If Q can be mapped to $\mathcal{F}$ via μ, then Q is satisfied in each model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{F}$ by assigning to each variable x in Q the value of its image $\mu(x)^{\mathcal{I}}$. Proposition 2 then implies $K \models Q$.

The only if direction is more involved. We show that if n is large enough, a mapping of Q into $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ can be constructed from a distinguished *canonical* model of $\mathcal{F}$.

The canonical model $\mathcal{I}_{\mathcal{F}}$ of $\mathcal{F}$ has as domain the set of all paths from some root in $\mathcal{F}$ to some node of $\mathcal{F}$ (thus, it can be infinite). It is constructed by unraveling the forest $\mathcal{F}$ in the standard way, where the blocked nodes act like 'loops' (cf. [14]). Note that in order to obtain a model of $\mathcal{F}$ by unraveling, $\mathcal{F}$ must be in $\mathsf{ccf}_n(\mathbb{F}_K)$ for some $n \geq 1$. The

formal definition of $\mathcal{I}_\mathcal{F}$ is then straightforward yet complex, and we must refer to the extended report [20] for the details. Instead, we provide an example.

Example 8 *By unraveling $\mathcal{F}_1$, we obtain a model $\mathcal{I}_\mathcal{F}$ 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 $A^{\mathcal{I}_\mathcal{F}}$, 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 $P_1^{\mathcal{I}_\mathcal{F}}$.

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. Consider the graph that has as nodes the domain (paths) of $\mathcal{I}_\mathcal{F}$, and as arcs the R-successor edges of $\mathcal{I}_\mathcal{F}$ for each role R occurring in Q. For any two nodes v_1, v_2 in $\mathcal{I}_\mathcal{F}$, let $d(v_1, v_2)$ denote the distance between v_1 and v_2 in this graph, and let G denote the image of Q under σ. Let d^σ_Q be the maximum distance $d(\sigma(x), \sigma(y))$ for any two variables x and y appearing in Q. G does not contain any paths longer than d^σ_Q. If $\mathcal{F}$ is d^σ_Q-complete, then for each path in G there is an isomorphic one in $\mathcal{F}$. Therefore, a d^σ_Q-complete completion forest will be large enough to find a mapping whose image is isomorphic to G. As all roles in Q are simple, it is immediate to see that d^σ_Q is bounded by the number of atoms in Q.

Proposition 4 *Let $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, with $n \geq n_Q$, and let $\mathcal{I}_\mathcal{F} \models Q$. Then $Q \hookrightarrow \mathcal{F}$.*

Example 9 *$K \models Q_1$, so $\mathcal{F}_1 \models Q_1$ must hold. This is witnessed by the mapping $Q_1 \hookrightarrow \mathcal{F}_1$ in Example 7. Note that there are longer queries, like $Q' = \{P_1(a, x_0), P_1(x_0, x_1), P_1(x_1, x_2), P_1(x_2, x_3), P_1(x_3, x_4)\}$ such that $K \models Q'$ holds, but the entailment $\mathcal{F}_1 \models Q'$ cannot be verified by mapping Q' into $\mathcal{F}_1$ since $\mathcal{F}_1$ is 1-blocked and $n_{Q'} > 1$.*

Proposition 4 establishes the only if direction of Theorem 3. Thus, query answering $K \models Q$ reduces to finding a mapping of Q into every $\mathcal{F} \in \mathsf{ccf}_{n_Q}(\mathbb{F}_K)$.

Complexity of Query Answering

We are now ready to prove the result on data complexity of conjunctive query answering that we are aiming at.

In the sequel, we use $\|K, Q\|$ to denote the total size of the string encoding a given knowledge base K and query Q. As follows from [20], branching in each variable tree in a completion forest $\mathcal{F} \in \mathbb{F}_K$ is polynomially bounded in $\|K, Q\|$, and the maximal depth of a variable is double exponential in $\|K, Q\|$ if n is polynomial in $\|K, Q\|$. Therefore, $\mathcal{F}$ has at most triple exponentially many nodes. Since each rule can be applied only polynomially often to a node, the expansion of the initial completion forest $\mathcal{F}_K$ into some $\mathcal{F} \in \mathbb{F}_K$ terminates in nondeterministic triple exponential time in $\|K, Q\|$ in this case, in particular for $n = n_Q$, which suffices for our purposes.

Theorem 5 *Given a $\mathcal{SHIQ}$ knowledge base K and a conjunctive query Q all of whose roles are simple, deciding whether $K \models Q$ is in* CO-3NEXPTIME.

Proof (*Sketch*). It is sufficient to check for every $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$, $n = n_Q$, whether $Q \hookrightarrow \mathcal{F}$. As argued above, $\mathcal{F}$ has size at most triple exponential in $\|K, Q\|$. Furthermore, we can check whether $Q \hookrightarrow \mathcal{F}$ holds by naive methods in triple exponential time in $\|K, Q\|$ time as well. (We stress that this test is NP-hard even for fixed $\mathcal{F}$.) □

Notice that the result holds for binary encoding of number restrictions in K. An exponential drop results for unary encoding if Q is fixed.

Under data complexity, Q and all components of $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}\rangle$ except for the ABox $\mathcal{A}$ are fixed. Therefore, n_Q is constant. Thus every completion forest $\mathcal{F} \in \mathsf{ccf}_n(\mathbb{F}_K)$ has *linearly* many nodes in $|\mathcal{A}|$, and any expansion of $\mathcal{F}_K$ terminates in polynomial time. Furthermore, deciding whether $Q \hookrightarrow \mathcal{F}$ is polynomial in the size of $\mathcal{F}$ by simple methods. As a consequence,

Theorem 6 *For a knowledge base K in $\mathcal{SHIQ}$ and a conjunctive query Q all of whose roles are simple, deciding whether $K \models Q$ is in* CONP *w.r.t. data complexity.*

Matching CONP-hardness follows from the respective result for $\mathcal{ALE}$ [10], which has been extended later to DLs even less expressive than $\mathcal{AL}$ [7]. Thus we obtain the following main result.

Theorem 7 *On knowledge bases in any DL from $\mathcal{AL}$ to $\mathcal{SHIQ}$, answering conjunctive queries all of whose roles are simple, is* CONP*-complete w.r.t. data complexity.*

This result not only exactly characterizes the data complexity of CQs for a range of DLs, but also extends two previous CONP-completeness results w.r.t. data complexity which are not obvious: the result on CQs over $\mathcal{ALCNR}$ given in [19] is now extended to $\mathcal{SHIQ}$, and the result in [17] for atomic queries in $\mathcal{SHIQ}$ is extended to *CQs*.

Conclusion

We studied conjunctive query (CQ) answering in the expressive DL $\mathcal{SHIQ}$, and generalizing a technique presented in [19] for a less expressive DL, we have developed a novel tableaux-based algorithm for CQ answering. It manages the technical challenges caused by the simultaneous presence of inverse roles, number restrictions, and general knowledge bases, leading to a DL lacking the finite model property. The

algorithm is worst case optimal in data complexity, and allows us to characterize the data complexity of the problem as CONP-complete for a wide range of DLs, including expressive ones. This closes the gap between the known CONP lower bound and the best known EXPTIME upper bound for even weaker DLs.

We point out that, by virtue of the correspondence between query containment and query answering [1], our algorithm can also be applied to decide containment of two conjunctive queries over a $\mathcal{SHIQ}$ knowledge base. Our results can be immediately extended to unions of CQs [20]. Also, the technique is applicable to the DL $\mathcal{SHOIQ}$, which extends $\mathcal{SHIQ}$ with *nominals*, i.e., concepts denoting single individuals, by tuning of the $\mathcal{SHOIQ}$ tableaux rules [13]. With little extra effort (by avoiding internalization of the ABox), we can obtain also a CONP upper-bound for the data complexity of $\mathcal{SHOIQ}$ in our setting. Finally, we are currently working on the case where the query may contain arbitrary roles, including transitive ones. However, it is still open whether CQ answering is decidable in this case

Combined complexity remains to be further investigated. It follows from [16] that the problem is in 2EXPTIME. Hence, the bound established above in Theorem 5 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.

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS'98*, 1998.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

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

[4] A. Borgida and R. J. Brachman. Conceptual modeling with description logics. In Baader et al. [2], chapter 10.

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

[6] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, 2005.

[7] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR 2006*, 2006.

[8] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, 1998.

[9] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of AAAI 2000*, 2000.

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

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

[12] 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.

[13] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of IJCAI 2005*, 2005.

[14] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. of CADE 2000*, volume 1831 of *LNCS*. Springer, 2000.

[15] I. Horrocks and S. Tessaris. A conjunctive query language for description logic ABoxes. In *Proc. of AAAI 2000*, 2000.

[16] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of LPAR 2004*, 2004.

[17] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI 2005*, 2005.

[18] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, 2002.

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

[20] M. M. Ortiz de la Fuente, D. Calvanese, and T. Eiter. Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. Technical report, Fac. of Computer Science, Free Univ. of Bozen-Bolzano, Mar. 2006. Available at `www.inf.unibz.it/~calvanese/papers/orti-calv-eite-TR-2006-03.pdf`. Also available as Technical Report INFSYS RR 1843-61-03, Inst. of Information Systems, Vienna Univ. of Technology.

[21] 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.

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

[23] M. Y. Vardi. The complexity of relational query languages. In *Proc. of STOC'82*, 1982.