Extending CARIN to the Description Logics of the $\mathcal{SH}$ Family

Magdalena Ortiz

Institute of Information Systems, Vienna University of Technology
ortiz@kr.tuwien.ac.at

Abstract. This work studies the extension of the *existential entailment algorithm* of CARIN to DLs of the $\mathcal{SH}$ family. The CARIN family of knowledge representation languages was one of the first hybrid languages combining DATALOG rules and Description Logics. For reasoning in one of its prominent variants, which combines $\mathcal{ALCNR}$ with non-recursive DATALOG, the blocking conditions of the standard tableaux procedure for $\mathcal{ALCNR}$ were modified. Here we discuss a similar adaptation to the $\mathcal{SHOIQ}$ tableaux, which provides some new decidability results and tight data complexity bounds for reasoning in non-recursive CARIN, as well as for query answering over Description Logic knowledge bases.

1 Introduction

Description Logics (DLs) are specifically designed for representing structured knowledge in terms of concepts (i.e., classes of objects) and roles (i.e., binary relationships between classes). In the last years, they have evolved into a standard formalism for ontologies which describe a domain of interest in different applications areas. In the context of the Semantic Web, DL-based ontologies have been designated via the Web Ontology Language (OWL) as a standard for describing the semantics of complex Web resources, in order to facilitate access by automated agents. Driven by the need to overcome limitations of DLs and to integrate them into applications, recent research focuses on combining DLs with other declarative knowledge representation formalisms, and in particular with rule-based languages, which play a dominant role in Databases (as query languages) and in Artificial Intelligence [3,8,15,19].

One of the first such *hybrid languages*, CARIN [15], integrates DATALOG programs with some DLs of the $\mathcal{ALC}$ family, being $\mathcal{ALCNR}$ (the basic DL $\mathcal{ALC}$ with number restrictions and role intersection) the most expressive. The limited decidability of hybrid languages was recognised already with the introduction of CARIN, as even very weak DLs yield an undecidable formalism when combined with recursive DATALOG. Three alternatives were proposed to regain decidability: (*i*) the DL constructors causing undecidability are disallowed; (*ii*) only non-recursive rules are allowed; or (*iii*) the variable occurrences in the DL atoms appearing in rules are restricted according to some *safety conditions* that limit their ability to relate unnamed individuals.

In this work, we enhance CARIN with a more expressive DL component and focus on its non-recursive variant (safe rules are briefly discussed in Section 4). We consider the popular DLs of the $\mathcal{SH}$ family, which extend $\mathcal{ALC}$ with role transitivity and containment. The most expressive DL here considered, $\mathcal{SHOIQ}$ (which essentially corresponds to OWL-DL), also supports concepts denoting a single individual called *nominals* ($\mathcal{O}$), inverse roles ($\mathcal{I}$), and qualified number restrictions ($\mathcal{Q}$). By disallowing one of these three constructs, we obtain the expressive and mutually incomparable sublogics known as $\mathcal{SHIQ}$ (corresponding to OWL-Lite), $\mathcal{SHOQ}$, and $\mathcal{SHOI}$ respectively.

For reasoning in non-recursive CARIN, the authors of [15] identified the *existential entailment problem* as a key task and proposed an algorithm for it, based on a tableau (there named *constraint system*) algorithm for satisfiability of $\mathcal{ALCNR}$ knowledge bases with modified blocking conditions. In this way, they also obtained the first algorithm for answering Conjunctive Queries (CQs) and Union of Conjunctive Queries (UCQs) in DLs and for deciding their containment, problems that have become a central topic of interest in recent years. Another central contribution of CARIN was to show a tight CONP upper bound for the aforementioned tasks under *data complexity*, i.e., w.r.t. to the size of the data, assuming that the query/rule component and the terminological part of the knowledge base are fixed. This setting is of major importance, as data repositories can be very large and are usually much larger than the terminology expressing constraints on the data.

In [17] the tableaux algorithm for deciding $\mathcal{SHOIQ}$ knowledge base satisfiability of [11] was adapted following the ideas introduced in [15], to provide an algorithm for the entailment and containment of positive queries in the $\mathcal{SH}$ family of DLs. In this paper we show how this algorithm, analogous to CARIN's existential entailment one, can be exploited for reasoning in non-recursive CARIN and in other hybrid languages. Like [17], the results have two limitations: transitive roles are not allowed in the rule component, and the interaction between number restrictions, inverses and nominals in $\mathcal{SHOIQ}$ may lead to non-termination. However, reasoning is sound and complete if the DL component of the hybrid knowledge base is written in $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ or $\mathcal{SHOI}$, and sound if it is in $\mathcal{SHOIQ}$. We obtain a precise characterisation of the data complexity of reasoning whenever the DATALOG component is non-recursive, and for some cases where it is recursive, e.g., if it satisfies the weak safety conditions of $\mathcal{DL}$+log.

2 Preliminaries

In this section, we define CARIN knowledge bases. The languages that are used in the two components are defined first: DL knowledge bases and DATALOG programs.

Throughout the paper, we consider a fixed alphabet containing the following pairwise disjoint countably infinite sets: a set $\mathbf{C}$ of *DL predicates of arity* 1, called *concept names*; a set $\mathbf{R}$ of *DL predicates of arity* 2, called *role names*, with a subset $\mathbf{R}_+ \subseteq \mathbf{R}$ of *transitive role names*; an alphabet $\mathbf{P}$ of *rule predicates*, where each $p \in \mathbf{P}$ has an associated arity $m \geq 0$; a set $\mathbf{I}$ of *individuals*; and a set $\mathbf{V}$ of *variables*. This alphabet is used for defining knowledge bases, whose semantics is given by (first-order) interpretations.

Definition 1 (Interpretation). *An* interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *is given by a non-empty* domain $\Delta^\mathcal{I}$ *and an* interpretation function $\cdot^\mathcal{I}$ *that maps each predicate* $p \in \mathbf{P} \cup \mathbf{C} \cup \mathbf{R}$ *of arity* n *to a subset of* $(\Delta^\mathcal{I})^n$*, and each individual in* $\mathbf{I}$ *to an element of* $\Delta^\mathcal{I}$*.*

2.1 Description Logics

The DL $\mathcal{SHOIQ}$ and its sublogics $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ are defined as usual.[1]

Definition 2 ($\mathcal{SHOIQ}$ Knowledge Bases). *A* role expression R *(or simply* role*) is a role name* $P \in \mathbf{R}$ *or its* inverse P^-*. A* role inclusion axiom *is an expression* $R \sqsubseteq R'$*, where* R *and* R' *are roles. A* role hierarchy $\mathcal{R}$ *is a set of role inclusion axioms.*

[1] For the sake of uniformity, we use the name $\mathcal{SHOI}$ instead of the also common $\mathcal{SHIO}$.

As usual, $\mathsf{Inv}(R) = P^-$ *if* $R = P$ *for some* $P \in \mathbf{R}$ *and* $\mathsf{Inv}(R) = P$ *if* $R = P^-$. *For a role hierarchy* $\mathcal{R}$*, the relation* $\sqsubseteq^*_{\mathcal{R}}$ *denotes the reflexive, transitive closure of* $\sqsubseteq$ *over* $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(R') \mid R \sqsubseteq R' \in \mathcal{R}\}$. *We write* $\mathsf{Trans}(R, \mathcal{R})$ *if* $R \sqsubseteq^*_{\mathcal{R}} R'$ *and* $R' \sqsubseteq^*_{\mathcal{R}} R$ *for some* $R' \in \mathbf{R}_+ \cup \{R^- \mid R \in \mathbf{R}_+\}$. *A role* S *is* simple *w.r.t.* $\mathcal{R}$ *if for no role* R *with* $\mathsf{Trans}(R, \mathcal{R})$ *we have that* $R \sqsubseteq^*_{\mathcal{R}} S$.

Let $a, b \in \mathbf{I}$ *be individuals,* $A \in \mathbf{C}$ *a concept name,* C *and* C' *concepts,* $P \in \mathbf{R}$ *a role name,* R *a role,* S *a simple role, and* $n \geq 0$ *an integer. Concepts are defined inductively according to the following syntax:*

$$C, C' \longrightarrow A \mid \{a\} \mid C \sqcap C' \mid C \sqcup C' \mid \neg C \mid \forall R.C \mid \exists R.C \mid \geq n\, S.C \mid \leq n\, S.C$$

Concepts of the form $\{a\}$ *are called* nominals. *A* concept inclusion axiom *is an expression* $C \sqsubseteq D$. *An* assertion *is an expression* $A(a)$, $P(a, b)$ *or* $a \not\approx b$. *A* TBox *is a finite set of concept inclusion axioms, and an* ABox *is a finite set of assertions. A* $(\mathcal{SHOIQ})$ knowledge base *(KB) is a triple* $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$*, where* $\mathcal{T}$ *is a TBox,* $\mathcal{R}$ *is a role hierarchy, and* $\mathcal{A}$ *is an ABox.*[2]

Definition 3 ($\mathcal{SHOQ}$**,** $\mathcal{SHIQ}$**, and** $\mathcal{SHOI}$ **Knowledge Bases).** *Roles and concepts in* $\mathcal{SHOQ}$*,* $\mathcal{SHIQ}$*, and* $\mathcal{SHOI}$ *are defined as in* $\mathcal{SHOIQ}$*, except that*

- *in* $\mathcal{SHOQ}$*, the inverse role constructor* P^- *is not available;*
- *in* $\mathcal{SHIQ}$*, nominals* $\{a\}$ *are not available;*
- *in* $\mathcal{SHOI}$*, number restrictions* $\geq n\, S.C$*,* $\leq n\, S.C$ *are not available,*

For $\mathcal{L}$ *one of* $\mathcal{SHOQ}$*,* $\mathcal{SHIQ}$*, or* $\mathcal{SHOI}$*, an* $\mathcal{L}$ knowledge base *is a* $\mathcal{SHOIQ}$ *knowledge base* $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ *such that all roles and concepts occurring in it are in* $\mathcal{L}$.

Definition 4 (Semantics of DL KBs). *Let* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *be an interpretation such that* $R^{\mathcal{I}} = (R^{\mathcal{I}})^+$ *for each* $R \in \mathbf{R}_+$. *To interpret* K*, the interpretation function is inductively extended to complex concepts and roles as follows:*

$$\begin{array}{ll}
(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} & (\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\} \\
(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} & (\leq n\, R.C)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \leq n\} \\
(P^-)^{\mathcal{I}} = \{(y, x) \mid (x, y) \in P^{\mathcal{I}}\} & (\geq n\, R.C)^{\mathcal{I}} = \{x \mid |\{y \mid (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \geq n\}
\end{array}$$

$\mathcal{I}$ satisfies *an assertion* α*, denoted* $\mathcal{I} \models \alpha$*, if* $\alpha = A(a)$ *implies* $a^{\mathcal{I}} \in A^{\mathcal{I}}$*,* $\alpha = P(a, b)$ *implies* $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$ *and* $\alpha = a \not\approx b$ *implies* $a^{\mathcal{I}} \neq b^{\mathcal{I}}$*;* $\mathcal{I}$ *satisfies a role inclusion axiom* $R \sqsubseteq R'$ *if* $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$*, and a concept inclusion axiom* $C \sqsubseteq C'$*, if* $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$. $\mathcal{I}$ satisfies a role hierarchy $\mathcal{R}$ *and* a terminology $\mathcal{T}$*, if it satisfies every axiom of* $\mathcal{R}$ *and* $\mathcal{T}$ *respectively. Furthermore,* $\mathcal{I}$ satisfies an ABox $\mathcal{A}$*, if it satisfies every assertion in* $\mathcal{A}$. *Finally,* $\mathcal{I}$ *is a* model *of* $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$*, denoted* $\mathcal{I} \models K$*, if it satisfies* $\mathcal{T}$*,* $\mathcal{R}$*, and* $\mathcal{A}$.

2.2 DATALOG

We now define DATALOG programs and their semantics, also given by interpretations.[3]

Definition 5 (DATALOG rules and DATALOG programs). *A* (rule/DL) atom *is an expression* $p(\overline{x})$*, where* p *is a (rule/DL) predicate, and* $\overline{x}$ *is a tuple from* $\mathbf{V} \cup \mathbf{I}$ *of the same arity as* p. *If* $\overline{x} \subseteq \mathbf{I}$*, then* $p(\overline{x})$ *is* ground.

[2] Note that only concepts and role names may occur in $\mathcal{A}$, but this is no limitation. Indeed, for a complex C, an assertion $C(a)$ can be expressed by $A_c(a)$ and an axiom $A_c \sqsubseteq C$ in $\mathcal{T}$, while an assertion $R^-(x, y)$ can be replaced by $\mathsf{Inv}(R)(a, b)$.

[3] Note that we consider first-order semantics, without the minimality requirement.

A DATALOG rule *is an expression of the form* $q(\overline{x}) :- p_1(\overline{y_1}), \ldots, p_n(\overline{y_n})$ *where* $n \geq 0$, $q(\overline{x})$ *is a rule atom, each* $p_i(\overline{y_i})$ *is an atom, and* $\overline{x} \cap \mathbf{V} \subseteq \overline{y_1} \cup \ldots \cup \overline{y_n}$. *As usual,* $q(\overline{x})$ *is called the* head *of the rule, and* $p_1(\overline{y_1}), \ldots, p_n(\overline{y_n})$ *is called the* body. *A rule with* $n = 0$ *is called a* fact *and can be written simply* $q(\overline{x})$.

A DATALOG program $\mathcal{P}$ *is a set of* DATALOG *rules. Its* dependency graph *is the directed graph whose nodes are the predicates* p *occurring in* $\mathcal{P}$ *with an edge* $p \rightarrow p'$ *if* p' *occurs in the head and* p *in the body of a rule in* $\mathcal{P}$. $\mathcal{P}$ *is* recursive *if its dependency graph contains some cycle, and* non-recursive *otherwise.*

Definition 6 (Semantics of DATALOG Programs). *An interpretation* $\mathcal{I}$ satisfies a ground atom $p(\overline{a})$, *written* $\mathcal{I} \models p(\overline{a})$, *if* $(\overline{a})^{\mathcal{I}} \in p^{\mathcal{I}}$. *A* substitution *is a mapping* $\sigma : \mathbf{V} \cup \mathbf{I} \rightarrow \Delta^{\mathcal{I}}$ *with* $\sigma(a) = a^{\mathcal{I}}$ *for every* $a \in \mathbf{I}$. *For an atom* $p(\overline{x})$ *and a substitution* σ, *we say that* σ makes $p(\overline{x})$ true in $\mathcal{I}$, *in symbols* $\mathcal{I}, \sigma \models p(\overline{x})$, *if* $\mathcal{I} \models p(\sigma(\overline{x}))$. *We say that* $\mathcal{I}$ satisfies a rule r, *denoted* $\mathcal{I} \models r$, *if every substitution that makes all the atoms in the body true also makes the atom in the head true. If* $\mathcal{I} \models r$ *for each* $r \in \mathcal{P}$, *then* $\mathcal{I}$ is a model of $\mathcal{P}$, *in symbols* $\mathcal{I} \models \mathcal{P}$.

2.3 CARIN Knowledge Bases

Now we define the CARIN language. In what follows, $\mathcal{L}$ denotes a DL of the $\mathcal{SH}$ family.

Definition 7 (CARIN knowledge bases). *A* CARIN-$\mathcal{L}$ knowledge base *is a tuple* $\langle K, \mathcal{P} \rangle$ *where* K *is an* $\mathcal{L}$ *knowledge base, called the* DL component *of* K, *and* $\mathcal{P}$ *is a* DATALOG *program, called its* rule (or DATALOG) component. *A* CARIN-$\mathcal{L}$ *knowledge base is (non-)recursive if its rule component* $\mathcal{P}$ *is (non-)recursive.*

Note that only rule predicates can occur in the head of rules of $\mathcal{P}$. This is a common feature of many hybrid languages that assume that the DL knowledge base provides a commonly shared conceptualisation of a domain, while the rule component defines application-specific relations that can not change the structure of this conceptual model.

The semantics of CARIN KBs arises naturally from the semantics of its components. As in the original CARIN, we define as main reasoning task the entailment of a ground atom, which may be either a DL assertion or a DATALOG ground fact.

Definition 8 (CARIN-$\mathcal{L}$ entailment problem). *An interpretation* $\mathcal{I}$ *is a* model *of a* CARIN-$\mathcal{L}$ *knowledge base* $\mathcal{K} = \langle K, \mathcal{P} \rangle$, *in symbols* $\mathcal{I} \models \mathcal{K}$, *if* $\mathcal{I} \models K$ *and* $\mathcal{I} \models \mathcal{P}$. *For a ground atom* α, $\mathcal{K} \models \alpha$ *denotes that* $\mathcal{I} \models \mathcal{K}$ *implies* $\mathcal{I} \models \alpha$ *for every* $\mathcal{I}$. *The* CARIN-$\mathcal{L}$ entailment problem *is to decide, given* $\mathcal{K}$ *and* α, *whether* $\mathcal{K} \models \alpha$.

We note that the standard DL reasoning tasks (e.g., KB consistency and subsumption) are reducible to entailment in CARIN, as the latter generalises instance checking.

3 Reasoning in non-recursive CARIN

In this section, we provide an algorithm for reasoning in non-recursive CARIN. The key to the decidability in this variant of CARIN is the limited interaction between the DL and rule predicates. Indeed, if we have a non-recursive DATALOG program $\mathcal{P}$ and we want to verify entailment of an atom $p(\overline{a})$, it is sufficient to consider the rules in $\mathcal{P}$ whose head predicate is p and unfold them into a set of rules where only $p(\overline{a})$ occurs in the head, and

the bodies contain only DL atoms and ground facts. The CARIN-$\mathcal{L}$ entailment problem with such a restricted rule component is then reducible to the *entailment of UCQs.*

The *query entailment* (or informally, query answering) problem is DLs has gained much attention in recent times. Many papers have studied the problem of answering CQs and UCQs over DL knowledge bases, e.g., [1,5,6,14,20]. We consider the more expressive language of positive existential queries.

3.1 Non-recursive CARIN and Query Entailment

We introduce *positive (existential) queries* (PQs), which generalise CQs and UCQs.[4]

Definition 9 (Positive Queries, Query Entailment). *A* positive (existential) query *(PQ) over a KB K is a formula $\exists\overline{x}.\varphi(\overline{x})$, where $\overline{x}$ is a vector of variables from $\mathbf{V}$ and $\varphi(\overline{x})$ is built using $\wedge$ and $\vee$ from DL atoms whose variables are in $\overline{x}$. If $\varphi(\overline{x})$ is a conjunction of atoms then $\exists\overline{x}.\varphi(\overline{x})$ is a* conjunctive query *(CQ); if $\varphi(\overline{x})$ is in disjunctive normal form then it is a* union of conjunctive queries *(UCQ).*

Let $Q = \exists\overline{x}.\varphi(\overline{x})$ be a PQ over K and let $\mathcal{I}$ be an interpretation. For a substitution σ, let Q^σ be the Boolean expression obtained from φ by replacing each atom α with $\top$ if $\mathcal{I}, \sigma \models \alpha$, and with $\bot$ otherwise. We call σ a match for $\mathcal{I}$ and Q*, denoted $\mathcal{I}, \sigma \models Q$, if Q^σ evaluates to $\top$. $\mathcal{I}$* is a model of Q*, written $\mathcal{I} \models Q$, if $\mathcal{I}, \sigma \models Q$ for some σ.*

We say that K entails *Q, denoted $K \models Q$, if $\mathcal{I} \models Q$ for each model $\mathcal{I}$ of K. The* query entailment problem *is to decide, given K and Q, whether $K \models Q$.*

Note that a PQ can be rewritten into an equivalent, possibly exponentially larger, UCQ.

The UCQ (and thus PQ) entailment problem and CARIN entailment problem are closely related. In fact, we can reduce the former to the latter as follows:

Proposition 1. *Let K be a $\mathcal{SHOIQ}$ knowledge base and let $Q = \exists\overline{x}.\varphi_1(\overline{x_1}) \vee \ldots \vee \varphi_n(\overline{x_n})$ be a UCQ over K. Then $\mathcal{K} \models Q$ iff $\langle K, \mathcal{P}\rangle \models q$, where $q \in \mathbf{P}$ is fresh, $\mathcal{P}$ is the* DATALOG *program containing the rules $q :- \varphi'_i(\overline{x_i})$ for each $1 \leq i \leq n$, and each $\varphi'_i(\overline{x_i})$ is obtained from $\varphi_i(\overline{x_i})$ by replacing each connective $\wedge$ by a comma.*

We show next that the converse also holds, i.e., the CARIN-$\mathcal{SHOIQ}$ entailment problem can be reduced to query entailment over the DL component. As a consequence, whenever we have a procedure for deciding query entailment, we obtain a sound and complete algorithm for reasoning in non-recursive CARIN.

Definition 10 (Rule unfolding and program depth). *Given two* DATALOG *rules:*
$r_1 = q_1(\overline{x_1}) :- p_1(\overline{y_1}), \ldots, p_n(\overline{y_n})$, *and* $r_2 = q_2(\overline{x_2}) :- p'_1(\overline{y'_1}), \ldots, p'_m(\overline{y'_m})$, *where $q_2 = p_i$ for some $1 \leq i \leq n$, let θ be the most general unifier of $\overline{x_2}$ and $\overline{y_i}$. Then the following rule r' is an* unfolding of r_2 in r_1*:*
$q_1(\theta\overline{x_1}) :- p_1(\theta\overline{y_1}), \ldots, p_{i-1}(\theta\overline{y_{i-1}}), p'_1(\theta\overline{y'_1}), \ldots, p'_m(\theta\overline{y'_m}), p_{i+1}(\theta\overline{y_{i+1}}), \ldots p_n(\theta\overline{y_n})$.
The width *of a rule r, denoted* $\mathsf{width}(r)$*, is the number of atoms in its body. The* depth *of a non-recursive* DATALOG *program $\mathcal{P}$, written* $\mathsf{depth}(\mathcal{P})$*, is $w+1$, where w is the width of the longest rule that can be obtained from some rule in $\mathcal{P}$ by repeatedly unfolding in it other rules of $\mathcal{P}$, until no more unfoldings can be applied. If $\mathcal{P} = \emptyset$,* $\mathsf{width}(r) = 1$.

[4] We consider Boolean queries, to which non-Boolean ones can be reduced as usual, and disregard the difference between the equivalent query entailment and query answering problems.

Note that $\mathsf{depth}(\mathcal{P})$ is finite and can be effectively computed, as $\mathcal{P}$ is non-recursive.

Definition 11 (Unfolding). *The* unfolding *of a non-recursive* DATALOG *program $\mathcal{P}$ for a ground rule atom $p(\overline{a})$ is the program $\mathcal{P}_{p(\overline{a})}$ obtained as follows:*
(1) Let $\mathcal{P}_1$ denote the set of rules in $\mathcal{P}$ where the head is of the form $p(\overline{x})$ and there is a unifier of of $\overline{a}$ and $\overline{x}$. $\mathcal{P}_2$ is the set of rules $p(\theta\overline{x}) :- q_1(\theta\overline{y_1}), \ldots, q_n(\theta\overline{y_n})$ where $p(\overline{x}) :- q_1(\overline{y_1}), \ldots, q_n(\overline{y_n}) \in \mathcal{P}_1$ and θ is the most general unifier of $\overline{a}$ and $\overline{x}$.
(2) For a rule r, let $r_{\mathcal{P}}$ denote the set of unfoldings in r of a rule from $\mathcal{P}$ (note that it may be empty). Apply exhaustively the following rule: if $r \in \mathcal{P}_2$ and the body of r contains a rule atom α such that $\alpha \notin \mathcal{P}$, replace r by $r_{\mathcal{P}}$ in $\mathcal{P}_2$. The resulting program is $\mathcal{P}_{p(\overline{a})}$.

Every model of $\mathcal{P}$ is also a model of $\mathcal{P}_{p(\overline{a})}$. Intuitively, $\mathcal{P}_{p(\overline{a})}$ captures the part of $\mathcal{P}$ that is relevant for the entailment of $p(\overline{a})$. Each rule in $\mathcal{P}_{p(\overline{a})}$ has $p(\overline{a})$ as head, and its body contains only DL atoms and ground facts from $\mathcal{P}$, which are true in every model of $\mathcal{P}$. Due to this restricted form, $\mathcal{P}_{p(\overline{a})}$ can easily be transformed into an equivalent UCQ.

Definition 12 (Query for a ground atom). *The* query for a ground atom α w.r.t. a non-recursive DATALOG program $\mathcal{P}$, *denoted $U_{\mathcal{P},\alpha}$, is the UCQ defined as follows:*
- If α is a DL atom, then $U_{\mathcal{P},\alpha} = \alpha$.
- Otherwise $U_{\mathcal{P},\alpha} = \exists\overline{x}.Q_1 \vee \ldots \vee Q_m$, where $r_1 \ldots r_m$ are the rules of $\mathcal{P}_\alpha$, each Q_i is the conjunction of the DL atoms in the body of r_i, and $\overline{x}$ contains the variables of each Q_i.

Note that if a rule atom α occurs as a fact in $\mathcal{P}$, it also occurs as a fact in $\mathcal{P}_\alpha$, and $U_{\mathcal{P},\alpha}$ is trivially true (since it has an empty disjunct which is always true). If $\mathcal{P}_\alpha = \emptyset$ then $U_{\mathcal{P},\alpha}$ is always false; this is the case, e.g., if α does not unify with the head of any rule.

Proposition 2. *Let $\mathcal{K} = \langle K, \mathcal{P} \rangle$ be a non-recursive* CARIN-$\mathcal{SHOIQ}$ *knowledge base and let α be a ground atom. Then $\mathcal{K} \models \alpha$ iff $K \models U_{\mathcal{P},\alpha}$.*

3.2 A Tableaux Algorithm for Query Entailment

We have shown that the non-recursive CARIN-$\mathcal{SHOIQ}$ entailment problem can be reduced to the entailment of a PQ (in fact, a UCQ suffices). In this section, we describe the algorithm given in [17] to solve the latter for the $\mathcal{SH}$ family DLs. Provided that the query contains only simple roles, it is sound and complete for $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$; for $\mathcal{SHOIQ}$ it is sound, but termination remains open.

The algorithm is an extension of the one in [15] for the *existential entailment problem*, which informally speaking, simultaneously captures UCQ entailment and CQ/UCQ containment (i.e., given a CQ Q_1 and a UCQ Q_2, decide whether $K \models Q_1$ implies $K \models Q_2$). We present it as a query entailment algorithm: this suffices for reasoning in non-recursive CARIN and the generalisation to containment is trivial. A first extension to CQs in $\mathcal{SHIQ}$ was presented in [18]. Here we recall the extension to PQs in $\mathcal{SHOIQ}$ of [17], where the reader may find detailed definitions, proofs and examples.

We build on [11] and use *completion graphs*, finite relational structures that represent models of a $\mathcal{SHIQ}$ knowledge base K. After an initial completion graph $\mathcal{G}_K$ for K is built, new completion graphs are generated by repeatedly applying *expansion rules*. Every model of K is represented in some completion graph that results from the expansion, thus $K \models Q$ can be decided by considering a suitable set of such graphs.

In what follows, $K{=}\langle\mathcal{T},\mathcal{R},\mathcal{A}\rangle$ denotes a $\mathcal{SHOIQ}$ knowledge base; the set of roles occurring in K and their inverses is denoted $\mathbf{R}_K$. A denotes a concept name; D, E denote concepts; R, R' denote roles; and a, b denote individuals.

A *completion graph* $\mathcal{G}$ for K comprises a finite labelled directed graph whose nodes $\mathsf{nodes}(\mathcal{G})$ are labelled by concepts and whose arcs $\mathsf{arcs}(\mathcal{G})$ are labelled by roles. The nodes in $\mathsf{nodes}(\mathcal{G})$ are of two kinds: *individual nodes* and *variable nodes*. The label of each individual node contains some nominal $\{a\}$ indicating that the node stands for the individual $a \in \mathbf{I}$. A variable node contains no nominal concepts and represents one or more unnamed individuals whose existence is implied by the knowledge base. An additional binary relation is used to store explicit inequalities between the nodes of $\mathcal{G}$.

In a completion graph $\mathcal{G}$, each arc $v \to w$ is labelled with a set $\mathcal{L}(v \to w)$ of roles from $\mathbf{R}_K$ and each node v is labelled with a set $\mathcal{L}(v)$ of 'relevant' concepts. The set of all the relevant concepts is denoted by $\mathsf{clos}(K)$ and contains the standard concept closure of $\neg C \sqcup D$ for each axiom $C \sqsubseteq D$ in the knowledge base K (closed under subconcepts and their negations) and some additional concepts that may be introduced by the rules (e.g., to correctly ensure the propagation of the universal restrictions, concepts of the form $\forall R'.D$ for some $\forall R.D \in \mathsf{clos}(K)$ and R' a transitive subroles of R are used, so they are also included in the closure).

The usual relations between the nodes in a completion graph $\mathcal{G}$ are defined as in [11,17]: if $v \to w \in \mathsf{arcs}(\mathcal{G})$, then w is a *successor* of v and v a *predecessor* of w. The transitive closures of successor and predecessor are *ancestor* and *descendant* respectively. If $R' \in \mathcal{L}(v \to w)$ for some role R' with $R' \sqsubseteq^* R$, then w is an R-*successor* of v. We call w an R-*neighbour* of v, if w is an R-*successor* of v, or if v is an $\mathsf{Inv}(R)$-*successor* of w. The *distance* between two nodes in $\mathcal{G}$ is defined in the natural way.

The *initial completion graph* $\mathcal{G}_K$ for K contains a node a labelled $\mathcal{L}(a) = \{\{a\}\} \cup \{\neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T}\} \cup \{\neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T}_\mathcal{A}\}$ for each individual a in K, where $\mathcal{T}_\mathcal{A} = \{\{a\} \sqsubseteq A \mid A(a) \in \mathcal{A}\} \cup \{\{a\} \sqsubseteq \exists P.\{b\} \mid P(a,b) \in \mathcal{A}\} \cup \{\{a\} \sqsubseteq \neg\{b\} \mid a \neq b \in \mathcal{A}\}$ is a set of concept inclusion axioms representing the assertions in $\mathcal{A}$.

We apply *expansion rules* to the initial $\mathcal{G}_K$ and obtain new completion graphs. The rules may introduce new variable nodes, but they are always successors of exactly one existing node. Hence the variable nodes form a set of trees that have individual nodes as roots. Some of these variable nodes may have an individual node as a successor, thus a tree can have a path ending with an arc to an individual node.

Blocking conditions are given to ensure that the expansion stops after sufficiently many steps. They are inspired by [15], but adapted to these more expressive logics, and depend on a depth parameter $n \geq 0$, generalising the non-parametrised blocking of [11]. This blocking is the crucial difference between our algorithm and [11]. According to the blocking conditions of [11], the expansion of a completion graph $\mathcal{G}$ terminates when a node v with a predecessor u is reached such that there is some ancestor u' of u that has in turn a successor v' such that the pairs (u', v') and (u, v) have the same node-arc-node labels, i.e., when a pair of nodes that is isomorphic to a previously existing one appears in $\mathcal{G}$. This *pairwise blocking* condition ensures that the expansion stops when $\mathcal{G}$ already represents a model of $\mathcal{K}$. If the knowledge base is satisfiable, then there is a way to non-deterministically apply the expansion rules until this blocking occurs, and a completion graph that represents a model of the knowledge base is obtained.

Since we want to decide query entailment, this is not enough: we need to obtain a set of models that suffices to check query entailment. Our modified blocking ensures that a completion graph is blocked only if it represents a set of models that are indistinguishable by the query. Instead of halting the expansion when a previously occurred *pair* of nodes appears, we stop when a repeated instance of an n-graph occurs, where the n-graph of a node v is a tree of variable nodes of depth at most n rooted at v, plus arcs to the individual nodes that are direct successors of a node in this tree. We now define formally this modified blocking. The next definition is technically quite involved. It is taken from [17], where more explanations and some examples can be found.

Definition 13 (n-graph blocking). *Given an integer $n \geq 0$ and a completion graph $\mathcal{G}$, let $\mathsf{vn}(\mathcal{G})$ denote the set of variable nodes in $\mathcal{G}$. The* blockable n-graph of node $v \in \mathsf{vn}(\mathcal{G})$ *is the subgraph $\mathcal{G}^{n,v}$ of $\mathcal{G}$ that contains v and (i) every descendant $w \in \mathsf{vn}(\mathcal{G})$ of v within distance n, and (ii) every successor $w' \in \mathsf{in}(\mathcal{G})$ of each such w. If w has in $\mathcal{G}^{n,v}$ no successors from $\mathsf{vn}(\mathcal{G})$, we call w a* leaf of $\mathcal{G}^{n,v}$. *Nodes v, v' of $\mathcal{G}$ are* n-graph equivalent via a bijection ψ from $\mathsf{nodes}(\mathcal{G}^{n,v})$ to $\mathsf{nodes}(\mathcal{G}^{n,v'})$ *if (1) $\psi(v) = v'$; (2) for every $w \in \mathsf{nodes}(\mathcal{G}^{n,v})$, $\mathcal{L}(w) = \mathcal{L}(\psi(w))$; (3) $\mathsf{arcs}(\mathcal{G}^{n,v'}) = \{\psi(w) \rightarrow \psi(w') \mid w \rightarrow w' \in \mathsf{arcs}(\mathcal{G}^{n,v})\}$; and (4) for every $w \rightarrow w' \in \mathsf{arcs}(\mathcal{G}^{n,v})$ $\mathcal{L}(w \rightarrow w') = \mathcal{L}(\psi(w) \rightarrow \psi(w'))$.*

Let $v, v' \in \mathsf{vn}(\mathcal{G})$ be n-graph equivalent via ψ, where both v and v' have predecessors in $\mathsf{vn}(\mathcal{G})$, v' is an ancestor of v in $\mathcal{G}$, and v is not in $\mathcal{G}^{n,v'}$. If v' reaches v on a path containing only nodes in $\mathsf{vn}(\mathcal{G})$, then v' is a n-witness of v in $\mathcal{G}$ via ψ. *Moreover, $\mathcal{G}^{n,v'}$* graph-blocks *$\mathcal{G}^{n,v}$* via *ψ, and each $w \in \mathsf{nodes}(\mathcal{G}^{n,v'})$* graph-blocks via *$\psi$ the node $\psi^{-1}(w)$ in $\mathcal{G}^{n,v}$.*

Let ψ be a bijection between two subgraphs G', G of $\mathcal{G}$ such that G' graph-blocks G via ψ. A node $v \in \mathsf{nodes}(\mathcal{G})$ is n-blocked, *if $v \in \mathsf{vn}(\mathcal{G})$ and v is either directly or indirectly n-blocked; v is* indirectly n-blocked, *if one of its ancestors is n-blocked; v is* directly n-blocked *iff none of its ancestors is n-blocked and v is a leaf of G; in this case we say that v is (directly) n-blocked by $\psi(v)$. An R-neighbour w of a node v in $\mathcal{G}$ is* n-safe *if $v \in \mathsf{vn}(\mathcal{G})$ or if w is not n-blocked.*

Note that v is m-blocked for each $m \leq n$ if it is n-blocked. When $n \geq 1$, then n-blocking implies pairwise blocking.

The expansion rules are analogous to the ones in [11], where 'blocked' is replaced by 'n-blocked' and'safe' is replaced by 'n-safe'. Due to space restrictions, we can not present the expansion rules here, but they can be found in [17].

A *clash* in a completion graph $\mathcal{G}$ is an explicit contradiction (e.g.,$\{A, \neg A\} \subseteq \mathcal{L}(v)$ for some node v), and it indicates that $\mathcal{G}$ represents an empty set of models and thus the expansion can stop. If $\mathcal{G}$ does not contain a clash it is called *clash-free*. If $\mathcal{G}$ contains a clash or no more rules are applicable to it, then we say that it is *n-complete*. We denote by $\mathbb{G}_K$ the set of completion graphs that can be obtained from the initial $\mathcal{G}_K$ via the expansion rules, and by $\mathsf{ccf}_n(\mathbb{G}_K)$ the ones that are n-complete and clash free.

We view each graph in $\mathbb{G}_K$ as a representation of a (possibly infinite) set of models of K. Intuitively, the models of K are all the relational structures containing $\mathcal{A}$ that satisfy the constraints given by $\mathcal{T}$ and $\mathcal{R}$. Each completion graph $\mathcal{G}$ contains the initial $\mathcal{A}$ and additional constraints, implicit in $\mathcal{T}$ and $\mathcal{R}$, that were explicated by applying the

rules. When there is more than one way to apply a rule to a graph $\mathcal{G}$ (e.g. in the $\sqcup$-rule either C_1 or C_2 can be added), the models represented by $\mathcal{G}$ are 'partitioned' into the sets of models represented by each of the different graphs that can be obtained.[5]

Importantly, every model of K is represented by some $\mathcal{G}$ in $\mathbb{G}_K$. Thus, the union of all the models of the graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ coincides with all the models of K, independently of the value of n. Therefore, in order to decide query entailment, we can choose an arbitrary $n \geq 0$ and check all the models of all the completion graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$. This is still not enough to yield a decision procedure: although the set $\mathsf{ccf}_n(\mathbb{G}_K)$ is finite, we do not have an algorithm for deciding entailment of query Q in all (possibly infinitely many) models of a completion graph $\mathcal{G}$. However, if a suitable n is chosen, the latter can be effectively decided by finding a syntactic mapping of the query into $\mathcal{G}$.

Definition 14 (Query mapping). *Let $Q = \exists\overline{x}.\varphi(\overline{x})$ be a PQ and let $\mathcal{G}$ be a completion graph. Let $\mu : \mathsf{Vl}(Q) \rightarrow \mathsf{nodes}(\mathcal{G})$ be a total function such that $\{a\} \in \mathcal{L}(\mu(a))$ for each individual a in $\mathsf{Vl}(Q)$. We write $C(x) \hookrightarrow_\mu \mathcal{G}$ if $C \in \mathcal{L}(\mu(x))$, and $S(x, x') \hookrightarrow_\mu \mathcal{G}$ if $\mu(x')$ is an S-neighbour of $\mu(x)$. Let γ be the Boolean expression obtained from $\varphi(\overline{x})$ by replacing each atom α in φ with $\top$, if $\alpha \hookrightarrow_\mu \mathcal{G}$, and with $\bot$ otherwise. We say that μ is a* mapping for Q into $\mathcal{G}$*, denoted $Q \hookrightarrow_\mu \mathcal{G}$, if γ evaluates to $\top$.* Q can be mapped into $\mathcal{G}$*, denoted $Q \hookrightarrow \mathcal{G}$, if there is a mapping μ for Q into $\mathcal{G}$.*

It is not hard to see that if $Q \hookrightarrow \mathcal{G}$, then there is a mapping for Q in every model represented by $\mathcal{G}$. The converse is slightly more tricky and only holds if Q contains only simple roles and if a suitable value for the blocking parameter n is chosen. Very roughly, n-blocking ensures that all paths of length $\leq n$ that occur in the models of K are already found in some $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. Since the query contains only simple roles, matches for Q in a model do not require paths larger than the number $nr(Q)$ of role atoms in the largest disjunct when Q is transformed into a UCQ (which is in turn bounded by the total role atoms in Q). As a consequence, Q can not distinguish models that are equivalent up to $nr(Q)$-blocking. The following theorem is shown in [17]:

Theorem 1. *Let Q be a positive query where only simple roles occur, let K be a $\mathcal{SHOIQ}$ KB, and let $n \geq nr(Q)$. Then $K \models Q$ iff $Q \hookrightarrow \mathcal{G}$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.*

The theorem suggests to verify PQ entailment as follows: (*i*) obtain all the completion graphs in $\mathsf{ccf}_{nr(Q)}(\mathbb{G}_K)$, and (*ii*) check each of them for query mappability. This yields a decision procedure provided that both steps can be effectively executed. We show below that this is the case if the KB is in any of $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$.

Due to Proposition 2, we can use the same decision procedure for the CARIN-$\mathcal{L}$ entailment problem. For any atom α, the number of atoms in each disjunct in $U_{\mathcal{P},\alpha}$ is bounded by $\mathsf{depth}(\mathcal{P})$. Trivially, if only simple roles occur in $\mathcal{P}$, the same holds for $U_{\mathcal{P},\alpha}$. Therefore, from Proposition 2 and Theorem 1, we easily obtain:

Corollary 1. *Let α be a ground atom, let $\mathcal{K} = \langle K, \mathcal{P}\rangle$ be a non-recursive* CARIN-*$\mathcal{SHOIQ}$ knowledge base where only simple roles occur in $\mathcal{P}$, and let $n \geq \mathsf{depth}(\mathcal{P})$. Then $\mathcal{K} \models \alpha$ iff $U_{\mathcal{P},\alpha} \hookrightarrow \mathcal{G}$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.*

[5] This view slightly differs from the more common one (e.g., [11]) in which a completion graph is a representation of one single model (the one obtained from $\mathcal{G}$ by standard unravelling).

Note that the outlined decision procedure requires that, for each given input the query α, $U_{\mathcal{P},\alpha}$ is built by unfolding $\mathcal{P}$. If several atoms are to be evaluated, a more efficient alternative can be to obtain the completion graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ and then evaluate all the rules of the program over each graph, in a bottom-up way. Roughly, for a completion graph $\mathcal{G}$ and a program $\mathcal{P}$, we can obtain the smallest set $S(\mathcal{G},\mathcal{P})$ of atoms that contains all the DL ground facts entailed by $\mathcal{G}$, and that contains the head of a rule r whenever there is a match of the body atoms to the atoms in the set $S(\mathcal{G},\mathcal{P})$ (under suitable substitutions). It is not hard to see that, for every atom α, $\alpha \in S(\mathcal{G},\mathcal{P})$ iff $K \models U_{\alpha,\mathcal{P}}$.[6]

4 Complexity of Reasoning in Hybrid KBs

We have shown that we can effectively solve the non-recursive-CARIN and the PQ entailment problem whenever we have an effective procedure or obtaining the graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ and deciding query mappability for each of them. The latter is trivially decidable if each $\mathcal{G}$ and the set $\mathsf{ccf}_n(\mathbb{G}_K)$ are finite (e.g., by traversing, for each $\mathcal{G}$, the finitely many possible mappings from the query variables to the nodes of $\mathcal{G}$).

As for the first part, it was shown in [17] that the expansion of an initial $\mathcal{G}_K$ into the set $\mathsf{ccf}_n(\mathbb{G}_K)$ terminates if there is no interaction between the number restrictions, inverses and nominals. Roughly, whenever variable nodes can cause a number restriction to be violated at an individual node a, the so-called $o?$-rule is applied to generate new individual neighbours for a. This rule is never applicable for $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$ KBs, allowing us to prove termination. For $\mathcal{SHOIQ}$, however, due to the mutual dependency between the depth of the forest and the number of individual nodes generated by the $o?$-rule that results from our modified blocking, we cannot ensure that it terminates (although we believe that, using the prioritised strategy for rule application of [11], it will do so in many cases).

The following bounds for the modified tableaux algorithm were shown in [17], while in the CARIN-entailment setting they are analysed in more detail in [16]. Given a KB $K{=}\langle\mathcal{T},\mathcal{R},\mathcal{A}\rangle$ and PQ Q, $||K,Q||$ denotes the combined size of the strings encoding the K and Q (assuming unary encoding of numbers in the number restrictions), and $|\mathcal{A}|$ the number of assertions in $\mathcal{A}$. Similarly, $||\mathcal{P}||$ denotes the size the string encoding a given DATALOG program $\mathcal{P}$.

Proposition 3. *The expansion of $\mathcal{G}_K$ into some $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq 0$, terminates in time triple exponential in $||K,Q||$ if n is polynomial in $||K,Q||$. If n is a constant and Q and all of K except $\mathcal{A}$ are fixed, then it terminates in time polynomial in $|\mathcal{A}|$.*

The same bounds apply to the number of nodes in each $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. Checking whether $Q \hookrightarrow \mathcal{G}$ can be easily done in time single exponential in the size of Q and polynomial in $|\mathsf{nodes}(\mathcal{G})|$; if Q is fixed, $Q \hookrightarrow \mathcal{G}$ can be tested in time polynomial in the size of $\mathcal{G}$ (as there are only polynomially many candidate assignments).

Theorem 2. *The PQ entailment problem is decidable if the input KB is in any of $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ and only simple roles occur in the query. Furthermore, it can be refuted non-deterministically in time polynomial in the size of the ABox.*

[6] This procedure has the same worst-case complexity as the one outlined above.

A matching lower bound holds already for instance checking in the very weak $\mathcal{AL}$ [1].

Theorem 3. *For every DL extending $\mathcal{AL}$ and contained in $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOI}$, deciding the entailment of a PQ in which only simple roles occur has* CONP*-complete data complexity.*

For a non-recursive DATALOG program $\mathcal{P}$, $\mathsf{depth}(\mathcal{P})$ is finite and effectively computable, and $nr(U_{\alpha,\mathcal{P}}) \leq \mathsf{depth}(\mathcal{P})$. Although $\mathsf{depth}(\mathcal{P})$ is single exponential in $||\mathcal{P}||$, it is constant if $\mathcal{P}$ is fixed. Hence we obtain:

Theorem 4. *The non-recursive* CARIN-$\mathcal{L}$ *entailment problem is decidable if $\mathcal{L}$ is any of $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ and only simple roles occur in the rule component of the KB. Furthermore, it has* CONP*-complete data complexity if $\mathcal{L}$ is a DL extending $\mathcal{AL}$.*

This result provides an exact characterisation of the data complexity of the non-recursive CARIN-$\mathcal{L}$ entailment problem for a wide range of description logics. Unfortunately, our work does not provide optimal upper bounds with respect to the combined complexity. In fact, the tableaux algorithm from [11] on which our work is based terminates in non-deterministic double exponential time in the worst case, even if the input is a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ or $\mathcal{SHOI}$ knowledge base whose satisfiability problem is known to be EXPTIME-complete [21,7,9]. This suboptimality carries on to our results. Additionally, our reduction from the CARIN-entailment problem to UCQ entailment causes an exponential blow-up whose inevitability has not been explored.

The $\mathcal{DL}$+log Family In [19], Rosati introduced the $\mathcal{DL}$+log family of formalisms coupling arbitrary DLs with DATALOG rules. It allows for recursive programs and, in order to preserve decidability, imposes some *weak safety* conditions on the rules which are a relaxed version of CARIN's safety.

An $\mathcal{L}$+log knowledge base is composed of a knowledge base in the DL $\mathcal{L}$ and a set of weak-safe DATALOG rules (possibly with disjunction and negation as failure). Its decidability depends on the one of query containment in $\mathcal{L}$: as shown in [19] (Theorem 11), satisfiability in $\mathcal{L}$+log is decidable iff CQ/UCQ containment is decidable in $\mathcal{L}$.[7] From well known results that relate query containment and query answering, it follows that our method can be exploited for deciding this problem.

Theorem 5. *Satisfiability of an $\mathcal{L}$+log knowledge base is decidable if $\mathcal{L}$ is $\mathcal{SHOQ}$, $\mathcal{SHIQ}$ or $\mathcal{SHOI}$ and the* DATALOG *component contains only simple roles.*

Furthermore, it follows from [19] that whenever the data complexity of query entailment is strictly lower than that of reasoning in the rule component, the latter carries on to the overall data complexity of reasoning. As a consequence, it can also be concluded from our results that reasoning in the above setting has Σ_2^p-complete data complexity when the DATALOG component is a disjunctive program with negation.

Related Complexity Results Since this work started, many query answering algorithms have been proposed and new complexity bounds have been found. Due to Proposition 2 (which is independent of the particular DL in the DL component), the new decidability results for answering UCQs in DLs imply new decidability boundaries for

[7] In general, 'satisfiability' means under both FOL and NM semantics.

non-recursive CARIN, similarly as the decidability of CQ/UCQ containment carries on to the $\mathcal{DL}$+log setting. Furthermore, the data complexity of UCQ answering can be directly transferred to non-recursive CARIN.

From this and recent results, the decidability statements in Theorems 4 and 5 holds also in the presence of transitive roles in the query if $\mathcal{L}$ is $\mathcal{SHOQ}$ [6], $\mathcal{SHIQ}$ [5], or $\mathcal{ALCQIb}_{reg}$, another expressive DL [2]. Further interesting results can be obtained from the latter, which is to our knowledge the most general algorithm for query answering in DLs without nominals. In particular, the DL known as $\mathcal{SRIQ}$ [10] (closely related to the DL $\mathcal{SROIQ}$ underlying OWL 2) can be reduced to (a minor extension of) $\mathcal{ALCQIb}_{reg}$. Exploiting the regular expressions in the query atoms in [2], one can use that algorithm to decide PQ entailment and containment in $\mathcal{SRIQ}$ (note that containment of queries with regular expressions does not follow from [2] in general, but it does if the query on the left is a plain CQ); hence both CARIN and $\mathcal{DL}$+log are decidable for $\mathcal{SRIQ}$. We also note that the algorithm in [2] provides an optimal 2EXPTIME upper bound for satisfiability of $\mathcal{SRIQ}$ knowledge bases; this was reported open in [12].

As for data complexity, CONP completeness for CARIN entailment, $\mathcal{DL}$+log satisfiability and UCQ answering (with arbitrary query/rule component) for $\mathcal{SHIQ}$ follows from [5]. Most recently, the PTIME-complete data complexity of PQ answering in Horn-$\mathcal{SHIQ}$ (a disjunction-free fragment of $\mathcal{SHIQ}$) was established [4]; this carries on to both non-recursive CARIN entailment and $\mathcal{DL}$+log satisfiability. To our knowledge, no other tight bounds for the $\mathcal{SH}$ family have been established. CARIN entailment and $\mathcal{DL}$+log satisfiability (with a positive DATALOG component) are PTIME-complete for $\mathcal{EL}$ and some of its extensions contained in $\mathcal{EL}^{++}$ [14] and $\mathcal{ELI}^{f}$ [13], see also [20]. Finally, due to the results in [1], the non-recursive CARIN entailment problem is in LOGSPACE for the DLs of the DL-Lite family; their PTIME-completeness had already been established for $\mathcal{DL}$+log [19].

5 Conclusion

In this paper, we have presented an algorithm for the CARIN entailment problem in knowledge bases that combine a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ or $\mathcal{SHOI}$ KB with a non-recursive DATALOG program containing only simple roles. It relies on a tableaux-based algorithm for positive query entailment which builds on the techniques from [11] and generalises the existential entailment algorithm given in [15] for a DL which is far less expressive than $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$.

For the three mentioned sublogics of $\mathcal{SHOIQ}$, our algorithm is worst-case optimal in data complexity, and allows us to characterise the data complexity of reasoning with non-recursive DATALOG programs for a wide range of DLs, including very expressive ones. Namely, for all DLs of the $\mathcal{SH}$ family except $\mathcal{SHOIQ}$, the problem has CONP-complete data complexity, and is thus not harder than instance checking in $\mathcal{AL}$.

Combining the aforementioned DLs with recursive DATALOG results in an undecidable formalism. However, our results can be combined with those of Rosati [19] to show decidability if the rules are weakly safe and the query contains no transitive roles. Further decidability and data complexity results for reasoning in hybrid languages can be obtained from the reduction of non-recursive CARIN to UCQ entailment presented here and from the results in [19], some of them were discussed in this work.

Acknowledgements This paper presents some results obtained during the author's Masters thesis and complements those in [18,17]. The author wants to express her great gratitude to her thesis supervisors, Thomas Eiter and Diego Calvanese. She also thanks the JELIA organisers for the invitation to present these results, as well as the consortium of the European Masters in Computational Logic. This work was partially supported by the Austrian Science Fund (FWF) grant P20840, and the Mexican National Council for Science and Technology (CONACYT) grant 187697.

References

1. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. In *Proc. of KR'06*, pages 260–270, 2006.
2. Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proc. of AAAI'2007*, pages 391–396, 2007.
3. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In *Proc KR'04*, pages 141–151, 2004.
4. Thomas Eiter, Georg Gottlob, Magdalena Ortiz and Mantas Simkus. Query Answering in the Description Logic Horn-$\mathcal{SHIQ}$. In *Proc. of JELIA'08*, to appear.
5. Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In *Proc. of IJCAI'07*, pages 399–404, 2007.
6. Birte Glimm, Ian Horrocks, and Ulrike Sattler. Conjunctive query entailment for $\mathcal{SHOQ}$. In *Proc. of DL'07*, volume 250, pages 65–75, 2007.
7. Birte Glimm, Ian Horrocks, and Ulrike Sattler. Deciding $\mathcal{SHOQ}$ plus role conjunction knowledge base consistency using alternating automata. In *Proc. of DL'08*, 2008.
8. S. Heymans and D. Vermeir. Integrating ontology languages and answer set programming. In *Proceedings DEXA Workshops 2003*, pp. 584–588. IEEE Computer Society, 2003.
9. Jan Hladik. A tableau system for the description logic $\mathcal{SHIO}$. In *IJCAR Doctoral Programme*, volume 106 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
10. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The irresistible $\mathcal{SRIQ}$. 2005.
11. Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.
12. Yevgeny Kazakov. $\mathcal{SRIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In *Proc. DL'08*, 2008.
13. Adila Krisnadhi and Carsten Lutz. Data complexity in the $\mathcal{EL}$ family of description logics. In *Proc. of DL'07*, 2007.
14. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In *Proc. of ISWC/ASWC 2007*, pages 310–323, 2007.
15. Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.
16. Ortiz Magdalena, Diego Calvanese and Thomas Eiter. Data Complexity of Query answering in Expressive Description Logics via Tableaux. INFSYS Research Report 1843-07-07. Vienna University of Technology, November 2007.
17. Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Data complexity of query answering in expressive description logics via tableaux. *J. of Automated Reasoning*. June 2008.
18. Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of AAAI'06*, 2006.
19. Riccardo Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proc. of KR'2006*, pages 68–98, 2006.
20. Riccardo Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proc. of DL'07*, 2007.
21. Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, 2001.