Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

Data Complexity of Query Answering in Expressive Description Logics via Tableaux

Magdalena Ortiz · Diego Calvanese · Thomas Eiter

Received: date / Accepted: date

Abstract The logical foundations of the standard web ontology languages are provided by expressive Description Logics (DLs), such as $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$. In the Semantic Web and other domains, ontologies are increasingly seen also as a mechanism to access and query data repositories. This novel context poses an original combination of challenges that has not been addressed before: (*i*) sufficient expressive power of the DL to capture common data modelling constructs; (*ii*) well established and flexible query mechanisms such as those inspired by database technology; (*iii*) optimisation 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, few tight characterisations of data complexity were available, and the problem was still open for most DLs of the $\mathcal{SH}$ family and for standard query languages like conjunctive queries and their extensions. We tackle this issue and prove a tight coNP upper bound for positive existential queries without transitive roles in $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$. We thus establish that, for a whole range of sublogics of $\mathcal{SHOIQ}$ that contain $\mathcal{AL}$, an-

This work has been partially supported by FET project TONES (Thinking ONtologiES), funded within the EU 6th Framework Programme under contract FP6-7603, by the PRIN 2006 project NGS (New Generation Search), funded by MIUR, and by the Mexican National Council for Science and Technology (CONACYT).

M. Ortiz
Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, Vienna, Austria
E-mail: ortiz@kr.tuwien.ac.at

D. Calvanese
Faculty of Computer Science
Free University of Bozen-Bolzano
Piazza Domenicani 3, Bolzano, Italy
E-mail: calvanese@inf.unibz.it

T. Eiter
Institute of Information Systems
Vienna University of Technology
Favoritenstraße 9-11, Vienna, Austria
E-mail: eiter@kr.tuwien.ac.at

swering such queries has coNP-complete data complexity. We obtain our result by a novel tableaux-based algorithm for checking query entailment, which uses a modified blocking condition in the style of Carin. The algorithm is sound for $\mathcal{SHOIQ}$, and shown to be complete for all considered proper sublogics in the $\mathcal{SH}$ family.

1 Introduction

Description Logics (DLs) [3] are specifically designed for representing structured knowledge in terms of concepts (i.e., classes of objects) and roles (i.e., binary relationships between classes). They have been initially developed to provide a formalisation of frame-based systems and semantic networks, and expressive variants of DLs were shown to be in tight correspondence with representation formalisms used in databases and software engineering [8, 4]. More recently, DLs gained increasing attention as the logical foundation for the standard Web ontology languages [22]. In fact, the most significant representatives of these languages, OWL-Lite and OWL-DL, are syntactic variants of DLs [27, 38]. In the Semantic Web and in other application domains such as Enterprise Application Integration and Data Integration [32], ontologies provide a high-level, conceptual view of the information relevant in a specific domain or managed by an organisation. They are increasingly seen also as a mechanism to access and query data repositories, while taking into account the constraints that are inherent in the common conceptualisation.

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

1) On the one hand, a DL should have sufficient expressive power to capture common constructs typically used in data modelling [5]. This calls for *expressive DLs* [6, 2], 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), may contain number restrictions (to state existence and functional dependencies and cardinality constraints on the participation to relationships in general), or may refer to specific objects that are of relevance at the intensional level. Such concepts are then used in the intensional component of a knowledge base (the TBox), which contains inclusion assertions between concepts and roles, while the extensional component (the ABox) contains assertions about the membership of individuals to concepts and roles.

2) On the other hand, the data underlying an ontology should be accessed using well established and flexible mechanisms such as those provided by database query languages. This goes well beyond the traditional inference tasks involving objects that have been considered and implemented in DL-based systems, like *instance checking* [16, 39]. Indeed, since explicit variables are missing, DL concepts have limited means for relating specific data items to each other. *Conjunctive queries* (CQs), i.e., plain select-project-join SQL queries, and unions of CQs (UCQs), i.e., a union of plain select-project-join SQL queries, provide a good trade-off between expressive power and nice computational properties, and are therefore adopted as core query languages in several contexts, such as data integration [32].

3) Finally, one has to take into account that data repositories can be very large and are usually much larger than the representation of the intensional level expressing

constraints on the data. Therefore, the contribution of the extensional level (i.e., the data) to the complexity of inference should be singled out, and one must pay attention to optimising 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 [43], where the relevant parameter is the size of the data, as opposed to *combined complexity*, which additionally considers the size of the query and of the schema.

Notable examples of expressive DLs are the ones in the so called $\mathcal{SH}$ family, which support all Boolean constructs over concepts and allow for asserting the transitivity of certain roles and containment between roles. The most expressive DL in this family is called $\mathcal{SHOIQ}$. In addition to the mentioned concept constructs and role assertions, it supports *nominals* ($\mathcal{O}$), which are concepts denoting a single individual [41], inverse roles ($\mathcal{I}$), and qualified number restrictions ($\mathcal{Q}$). By disallowing one of these three constructs, we obtain the sublogics known as $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$, respectively, which are three DLs with high and mutually incomparable expressive power. Note that $\mathcal{SHOIQ}$ essentially corresponds to OWL-DL, while $\mathcal{SHIQ}$ essentially corresponds to OWL-Lite [27, 38].[1] These languages have been promoted as standard Web ontology languages by the World Wide Web Consortium within the Semantic Web effort.[2]

For the $\mathcal{SH}$ family and other expressive DLs, TBox+ABox reasoning has been studied extensively in the last decade, using a variety of techniques ranging from reductions to reasoning in Propositional Dynamic Logic (PDL) [7, 6], over tableaux [2, 26] to automata on infinite trees [6, 42] and resolution [28, 30, 31]. For many of them, the combined complexity of instance checking (with both TBox and ABox) is ExpTime-complete, including $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$. Unfortunately, the interaction of nominals, inverse roles, and counting increases the computational complexity of inference in $\mathcal{SHOIQ}$ causing instance checking to be NExpTime-complete [41].

As for data complexity, it was shown in [16, 39] that instance checking is coNP-hard already in the rather weak DL $\mathcal{ALE}$, and in [11] that CQ answering is coNP-hard in the yet weaker DL $\mathcal{AL}$. Tight upper bounds were not known, since little attention had been paid to this problem. The data complexity was studied in the last years, but mostly for suitably tailored DLs [10, 11, 12]. In [11, 12], the *DL-Lite* family of DLs was considered, and two DLs were identified for which the problem is in LogSpace and can be effectively reduced to evaluating a UCQ over a database using standard relational database technology. Furthermore, [11] analysed which additions to the DL make the problem hard for NLogSpace, PTime, or coNP. The analysis essentially showed that the two identified DLs are the maximal ones with respect to allowed constructs enjoying so called *FOL-rewritability* of query answering, which implies LogSpace data complexity of this problem. Another interesting consequence of the results in [11] is that any further addition to the DL, such as universal quantification (a construct considered basic in DLs) makes the problem already coNP-complete, and therefore, as shown by our work, as hard as for the very expressive DLs that we consider here.

The data complexity of expressive DLs has not been studied in depth, and it only became a topic of interest in recent years. An ExpTime upper bound for data com-

[1] The OWL languages also support certain datatypes, which are important for applications and can be seen as a restricted form of concrete domains [1, 34]. On the other hand, the OWL-DL and OWL-Lite variants support only restricted forms of number restrictions, namely unqualified number restrictions ($\mathcal{N}$) and functionality ($\mathcal{F}$), respectively. Notice that the upcoming standard language OWL 2, instead, supports qualified number restrictions.

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

plexity of CQ answering in $\mathcal{DLR}$ follows from the results on CQ containment and view-based query answering in [7, 9].[3] They are based on a reduction to reasoning in PDL, which however prevents to single out the contribution to the complexity coming from the ABox. Similar considerations hold for the techniques in [25], which refined and extended the ideas introduced in [7], making the resulting algorithms better suited for implementation on top of tableaux-based algorithms. In [28, 30] a technique based on a reduction to Disjunctive Datalog was used for $\mathcal{SHIQ}$. It provides a (tight) coNP upper bound for data complexity of instance checking, since it allows to single out the ABox contribution. The result can be immediately extended to tree shaped conjunctive queries (without transitive roles), since they admit a representation as a DL concept, e.g., by making use of the notion of tuple-graph of [7], or via rolling up [25]. However, this is not the case for general CQs, resulting in a non-tight 2ExpTime upper bound. The first tight upper bounds for CQ answering in $\mathcal{SHIQ}$ were given in [37], but only for queries without transitive roles (though transitive roles may occur in the knowledge base), and generalised in [19] to all CQs (see Section 5 for further discussion).

Most of the results we have mentioned are quite recent, since the work on data complexity before this decade was rather scarce. The most notable exception is the seminal work on the Carin language for hybrid knowledge bases [33]. The authors showed a tight coNP upper bound for CQ answering in a DL called $\mathcal{ALCNR}$, which has no role hierarchies, does not support inverse roles, and has only a limited form of number restrictions. It is based on the tableaux algorithm for satisfiability of $\mathcal{ALCNR}$ knowledge bases, modifying the blocking condition in such a way that it can be used for deciding query entailment. The modified tableaux algorithm provides not only the first tight upper bounds for data complexity of query answering in DLs, but also the first algorithm for answering UCQs and for deciding the containment of UCQs over DL knowledge bases.

Tableaux algorithms play a very important role in DLs nowadays, and are one of the most popular reasoning techniques. Despite their high worst-case computational complexity, they are amenable to optimisations and the basis of many reasoning engines, which provide efficient implementations [23, 21][4]. For all DLs in the $\mathcal{SH}$ family, tableaux algorithms for checking satisfiability have been found. In particular, in [24] a tableaux algorithm for deciding satisfiability of $\mathcal{SHOIQ}$ knowledge bases was given, which generalises the previous algorithms for $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$. However, all these algorithms were devised for standard reasoning tasks like satisfiability and instance checking, and several interesting questions remained unaddressed. First, whether it is possible to apply the ideas and techniques for Carin to the DLs in the $\mathcal{SH}$ family in order to obtain (tableaux-based) algorithms for answering expressive queries over knowledge bases in these logics. Second, given that this is possible, what kind of queries may be handled. Third, whether any of the algorithms obtained would allow to derive, similarly as in the case of Carin, exact characterisations of the data complexity of query answering.

In this paper, we shed light on these questions, by simultaneously addressing the three challenges identified above. We show that the blocking conditions of [33] can be suitably generalised to very expressive DLs from the $\mathcal{SH}$ family. Technically speaking, the generalisation is not trivial. Indeed, we consider logics that have inverse roles, which as recently shown make answering CQs 2ExpTime-hard [35]. Some of the DLs have

[3] These results apply only to queries without transitive closure.

[4] See also `http://www.cs.man.ac.uk/~sattler/reasoners.html`.

no finite model property, and only weak forms of the ubiquitous forest model property. Furthermore, we consider *Positive Existential Queries* (PQs), a generalisation of UCQs that is not more expressive, but is exponentially more succinct.

Our main contributions are briefly summarised as follows:

- Building on the techniques of [26, 33], we present a novel tableaux-based algorithm for query answering in expressive DLs of the $\mathcal{SH}$ family. We prove that the algorithm is sound for answering PQs (and hence, also for UCQs and CQs) without transitive roles over $\mathcal{SHOIQ}$ knowledge bases, and thus in all DLs of the $\mathcal{SH}$ family. However, it does not work in general when the query contains transitive roles. This is because the blocking condition we use relies on the fact that the query can only distinguish patterns of bounded size in the model, where the bound depends on the query shape.
- We prove that the algorithm is complete for knowledge bases in the three DLs $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$. As a consequence, entailment of PQs without transitive roles over knowledge bases in these logics is decidable, which was open for $\mathcal{SHOI}$. This result extends also to deciding the containment and equivalence of PQs. In fact, the algorithm terminates if there is no simultaneous interaction of number restrictions, inverse roles, and nominals, and hence also works for larger classes of knowledge bases. However, for arbitrary $\mathcal{SHOIQ}$ knowledge bases termination is not established, as it seems that the interaction could indefinitely postpone the satisfaction of the blocking conditions.
- The novel algorithm provides us with a characterisation of the data complexity of query answering in expressive DLs. Specifically, we show that the data complexity of answering PQs without transitive roles over $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$ knowledge bases is in coNP, and thus is coNP-complete for all their sublogics that contain $\mathcal{AL}$.

This shows that the techniques introduced for Carin are indeed a suitable tool to provide tableaux-based algorithms and exact characterisations of the data complexity of answering large families of queries over a wide range of expressive DLs.

The rest of the paper is organised as follows. After technical preliminaries in Section 2, we present in Section 3 our algorithm for answering PQs over $\mathcal{SHOIQ}$ knowledge bases. In Section 4, we discuss the resulting complexity bounds for $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$, and in Section 5 we draw final conclusions. In order to increase readability, technical details of some proofs have been moved to an appendix.

2 Preliminaries

In this section, we introduce the technical preliminaries for the rest of the paper. We first introduce syntax and semantics of the Description Logic $\mathcal{SHOIQ}$ and its sublogics $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$, and then we define the query answering problem addressed in this work.

2.1 Description Logics

Description Logics (DLs) [3] are logics that are particularly well-suited for the representation of structured knowledge. The basic elements of DLs are *concepts*, denoting sets

of objects of the domain of interest, and *roles*, denoting binary relations between the instances of concepts. They are described by concept and role expressions built from concept names and role names, by applying concept and role *constructors*, respectively. The domain of interest is then modelled through a knowledge base, which comprises logical assertions both at the intensional level (specifying the properties of concepts and roles), and at the extensional level (specifying the properties of individuals and the relationships among individuals).

We assume that $\mathbf{R}$, $\mathbf{C}$, $\mathbf{I}$ are countable and pairwise disjoint sets of *role names*, *concept names*, and *individuals*, respectively, and that $\mathbf{R}_+ \subseteq \mathbf{R}$ is a set of *transitive role names*.

2.1.1 The Description Logic $\mathcal{SHOIQ}$

Definition 2.1 (Roles) A *role expression* R (or simply *role*) is either a role name $P \in \mathbf{R}$ or its *inverse*, denoted P^-. A *role inclusion axiom* is an expression of the form $R \sqsubseteq R'$ where R and R' are roles. A *role hierarchy* $\mathcal{R}$ is a set of role inclusion axioms.

As usual, we define $\mathsf{Inv}(R) = P^-$ if $R = P$ is a role-name, and $\mathsf{Inv}(R) = P$ if $R = P^-$ for some role name P.

The relation $\sqsubseteq^*_{\mathcal{R}}$ denotes the reflexive, transitive closure of $\sqsubseteq$ over a role hierarchy $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(R') \mid R \sqsubseteq R' \in \mathcal{R}\}$. If $R \sqsubseteq^*_{\mathcal{R}} R'$, then we call R a *sub-role* of R' and R' a *super-role* of R w.r.t. $\mathcal{R}$.

A role R is *transitive* w.r.t. a role hierarchy $\mathcal{R}$, denoted by $\mathsf{Trans}(R, \mathcal{R})$, if either R or $\mathsf{Inv}(R)$ belongs to $\mathbf{R}_+$, or the role hierarchy $\mathcal{R}$ implies that R is both a sub-role and a super-role of a transitive role; formally, $\mathsf{Trans}(R, \mathcal{R})$ holds iff $R \sqsubseteq^*_{\mathcal{R}} R'$ and $R' \sqsubseteq^*_{\mathcal{R}} R$ for some $R' \in \mathbf{R}_+ \cup \{R^- \mid R \in \mathbf{R}_+\}$.

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

In the following, we omit $\mathcal{R}$ when it is clear from the context, and use $\sqsubseteq^*$ and $\mathsf{Trans}(R)$ instead of $\sqsubseteq^*_{\mathcal{R}}$ and $\mathsf{Trans}(R, \mathcal{R})$, respectively.

Definition 2.2 (Concepts) $\mathcal{SHOIQ}$ concepts (or simply concepts) are defined inductively according to the following syntax:

C, C'	$\longrightarrow$	A	atomic concept	(S1)
		$\mid \{o\}$	nominal	(S2)
		$\mid C \sqcap C'$	conjunction	(S3)
		$\mid C \sqcup C'$	disjunction	(S4)
		$\mid \neg C$	negation	(S5)
		$\mid \forall R.C$	universal quantification	(S6)
		$\mid \exists R.C$	existential quantification	(S7)
		$\mid \geq n\, S.C \mid \leq n\, S.C$	(qualified) number restrictions	(S8)

where A is a concept name, C and C' are concepts, R is a role, S is a simple role, and $n \geq 0$ is an integer. An *atomic concept* is either a nominal $\{o\}$ with $o \in \mathbf{I}$ or a concept name $A \in \mathbf{C}$.

In DLs, the knowledge base about the domain of interest consists of an intensional component, called TBox, representing general knowledge about the domain, and an extensional component, called ABox, representing knowledge about specific objects. Additionally, in the DLs of the $\mathcal{SH}$ family, a role hierarchy might be present.

Definition 2.3 (Knowledge base) A *concept inclusion axiom* is an expression $C \sqsubseteq D$ where C and D are concepts. An *assertion* α is an expression $A(a)$, $P(a,b)$ or $a \not\approx b$, where A is a concept name, P is a role name, and a, b are individuals in $\mathbf{I}$. A *TBox*, or terminology, 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.

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. For a concept C, $NNF(C)$ denotes the NNF of C. For $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, we denote by $\mathbf{R}_K$ the set of roles occurring in $\mathcal{T}$ and $\mathcal{R}$, and of their inverses. Furthermore, $\mathbf{C}_K$ denotes the set of concept names occurring in K, and $\mathbf{I}_K$, $\mathbf{I}_\mathcal{A}$, and $\mathbf{I}_\mathcal{T}$ denote the sets of all individuals occurring in K, in $\mathcal{A}$, and in $\mathcal{T}$, respectively. Note that $\mathbf{I}_\mathcal{A} \cup \mathbf{I}_\mathcal{T} = \mathbf{I}_K$ for every K, and if K is a $\mathcal{SHIQ}$ knowledge base, then $\mathbf{I}_\mathcal{T} = \emptyset$ and $\mathbf{I}_\mathcal{A} = \mathbf{I}_K$.

2.1.2 The Description Logics $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$

The three sublogics $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$ of $\mathcal{SHOIQ}$ are obtained as follows.

Definition 2.4 (Sublogic Roles and Concepts) 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 is not available;
- in $\mathcal{SHIQ}$, nominals $\{o\}$ are not available, i.e., (S2) is not in the syntax of $\mathcal{SHIQ}$ concepts;
- in $\mathcal{SHOI}$, (qualified) number restrictions are not available, i.e., (S8) is not in the syntax of $\mathcal{SHOI}$ concepts;

Thus, in $\mathcal{SHIQ}$, only concepts names $A \in \mathbf{C}$ are atomic concepts.

Definition 2.5 (Sublogic Knowledge Bases) For $\mathcal{L}$ being one of the logics $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, 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 $\mathcal{L}$.

Example 2.6 As a running example, we use the following two $\mathcal{SHOIQ}$ knowledge bases:

$$K_1 = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\neg A\},\ \{\},\ \{A(a)\} \rangle$$
$$K_2 = \langle \{A \sqsubseteq \exists P_1.A,\ A \sqsubseteq \exists P_2.\{o\}\},\ \{\},\ \{A(a)\} \rangle.$$

Note that K_1 is a $\mathcal{SHOQ}$, a $\mathcal{SHIQ}$, and a $\mathcal{SHOI}$ knowledge base, while K_2 is a $\mathcal{SHOQ}$ and a $\mathcal{SHOI}$ knowledge base, but not a $\mathcal{SHIQ}$ one.

We now define the semantics of knowledge bases, which is given in terms of first-order interpretations.

Definition 2.7 (Model of a knowledge base) An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$, the *domain*, and an *interpretation function* $\cdot^\mathcal{I}$ that

- maps each role $R \in \mathbf{R}$ to a set $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, such that $R^\mathcal{I} = (R^\mathcal{I})^+$ for each $R \in \mathbf{R}_+$ and $(R^-)^\mathcal{I} = \{\langle o', o \rangle \mid \langle o, o' \rangle \in R^\mathcal{I}\}$,
- assigns to each individual $o \in \mathbf{I}$ an element $o^\mathcal{I} \in \Delta^\mathcal{I}$,[5] and

[5] Notice that we do not enforce the unique name assumption, i.e., two individuals $o_1 \neq o_2$ may denote the same domain object $o_1^\mathcal{I} = o_2^\mathcal{I}$.

– assigns to each concept C' a set $C'^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ such that

$$\begin{aligned}
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{o \mid \text{ for all } o', \langle o, o' \rangle \in R^{\mathcal{I}} \text{ implies } o' \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{o \mid \text{ for some } o', \langle o, o' \rangle \in R^{\mathcal{I}} \text{ and } o' \in C^{\mathcal{I}}\} \\
(\geq n\, S.C)^{\mathcal{I}} &= \{o \mid |\{o' \mid \langle o, o' \rangle \in S^{\mathcal{I}} \text{ and } o' \in C^{\mathcal{I}}\}| \geq n\} \\
(\leq n\, S.C)^{\mathcal{I}} &= \{o \mid |\{o' \mid \langle o, o' \rangle \in S^{\mathcal{I}} \text{ and } o' \in C^{\mathcal{I}}\}| \leq n\} \\
\{o\}^{\mathcal{I}} &= \{o^{\mathcal{I}}\}.
\end{aligned}$$

Note that the interpretation of each nominal $\{o\}$ is a singleton.

An interpretation $\mathcal{I}$ *satisfies* a role inclusion axiom $R \sqsubseteq R'$, if $R^{\mathcal{I}} \subseteq R'^{\mathcal{I}}$; a concept inclusion axiom $C \sqsubseteq C'$, if $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$; and an assertion α, denoted $\mathcal{I} \models \alpha$, if:

$$\begin{aligned}
a^{\mathcal{I}} &\in A^{\mathcal{I}}, && \text{if } \alpha = A(a) \\
\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle &\in P^{\mathcal{I}}, && \text{if } \alpha = P(a, b) \\
a^{\mathcal{I}} &\neq b^{\mathcal{I}}, && \text{if } \alpha = a \not\approx b.
\end{aligned}$$

An interpretation $\mathcal{I}$ *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 $\mathcal{A}$.

Note that complex concepts and roles are not allowed in ABoxes. However, this is no limitation, since an assertion $C(a)$ with a complex concept C can always be replaced by an assertion $A_C(a)$ in the ABox, together with an inclusion assertion $A_C \sqsubseteq C$, where A_C is a new concept name. This transformation is model preserving.

Finally, we observe that, using nominals, an ABox $\mathcal{A}$ in $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ can be *internalised* in the TBox, yielding a knowledge base $K_{\mathcal{A}} = \langle \mathcal{T}_{\mathcal{A}}, \mathcal{R}, \emptyset \rangle$ with an empty ABox. Indeed, $\mathcal{T}_{\mathcal{A}}$ is obtained from $\mathcal{T}$ by adding, for each ABox assertion α in $\mathcal{A}$, the inclusion axiom

$$\begin{aligned}
\{a\} &\sqsubseteq A, && \text{if } \alpha = A(a) \\
\{a\} &\sqsubseteq \exists P.\{b\}, && \text{if } \alpha = P(a, b), \text{ and} \\
\{a\} &\sqsubseteq \neg\{b\}, && \text{if } \alpha = a \not\approx b.
\end{aligned}$$

If K is a $\mathcal{SHIQ}$ knowledge base, the resulting $K_{\mathcal{A}}$ is a $\mathcal{SHOIQ}$ knowledge base, where the only nominals are those resulting from ABox internalisation. It is easy to see that K and $K_{\mathcal{A}}$ have exactly the same models, so all reasoning services are preserved [40].

2.2 Positive Queries

We now introduce positive (existential) queries, which generalise both conjunctive queries and unions of conjunctive queries. We assume that **Var** is a countably infinite set of variable names.

Definition 2.8 (Positive Queries) Let $\mathbf{x}$ be a vector of variables from **Var**. A *positive (existential) query* (PQ) over a KB K is a formula $\exists \mathbf{x}.\varphi(\mathbf{x})$, where $\varphi(\mathbf{x})$ is built using $\wedge$ and $\vee$ from atoms $C(z)$ and $S(z, z')$, where C is a concept name in $\mathbf{C}_K$, S is a simple role name in $\mathbf{R}_K$, and z, z' are variables from $\mathbf{x}$ or individuals in $\mathbf{I}_K$.

Note that transitive roles and their super-roles are disallowed in queries. We denote by $\mathsf{VI}(Q)$ the set of variables and individuals in a query Q.

A PQ $Q = \exists \mathbf{x}.\varphi(\mathbf{x})$ is a *conjunctive query* (CQ), if $\varphi(\mathbf{x})$ is a conjunction of atoms, and a *union of conjunctive queries* (UCQ), if $\varphi(\mathbf{x})$ is in disjunctive normal form; every PQ can be easily rewritten to a UCQ, but the resulting query may be exponentially larger.

Queries are interpreted as usual. For an interpretation $\mathcal{I}$, let $\pi : \mathsf{VI}(Q) \to \Delta^{\mathcal{I}}$ be a total function such that $\pi(a) = a^{\mathcal{I}}$ for each individual a. We write $\mathcal{I}, \pi \models C(x)$ if $\pi(x) \in C^{\mathcal{I}}$, and $\mathcal{I}, \pi \models S(x, y)$ if $\langle \pi(x), \pi(y) \rangle \in S^{\mathcal{I}}$. Let γ be the Boolean expression obtained from φ by replacing each atom α in φ with $\top$ if $\mathcal{I}, \pi \models \alpha$, and with $\bot$ otherwise. We call π a *match for* $\mathcal{I}$ *and* Q, denoted $\mathcal{I}, \pi \models Q$, if γ evaluates to $\top$. Then $\mathcal{I}$ *is a model of* Q ($\mathcal{I} \models Q$), if there is a match π for $\mathcal{I}$ and Q.

Definition 2.9 (Query Entailment) Let Q be a query over a KB K. 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 $K \models Q$.

Example 2.10 Consider the following PQs:

$$Q_1 = \exists x, y, z.P_1(x, y) \wedge P_2(x, z) \wedge A(y);$$
$$Q_2 = \exists x, y, z.P_2(x, y) \wedge P_2(y, z);$$
$$Q_3 = \exists x, y.(P_1(x, y) \vee P_2(x, y)) \wedge P_2(y, o).$$

Note that Q_1 and Q_2 are CQs. First, we observe that $K_1 \models Q_1$. Indeed, due to the inclusion axiom $A \sqsubseteq \exists P_1.A$, in every model $\mathcal{I}$ of K_1 there is some instance o_1 of A that is connected to $a^{\mathcal{I}}$ via role P_1. By the axiom $A \sqsubseteq \exists P_2.\neg A$, there is also some element o_2 that is connected to $a^{\mathcal{I}}$ via role P_2. Setting $\pi(x) = a^{\mathcal{I}}$, $\pi(y) = o_1$, and $\pi(z) = o_2$, we have a match for $\mathcal{I}$ and Q_1. Similarly, $K_2 \models Q_1$: if $\mathcal{I}$ is a model of K_2, let o_1 be an instance of A that is connected to $a^{\mathcal{I}}$ via role P_1; such an o_1 exists by the axiom $A \sqsubseteq \exists P_1.A$. Then $\pi(x) = a^{\mathcal{I}}$, $\pi(y) = o_1$, and $\pi(z) = o^{\mathcal{I}}$ is a match for $\mathcal{I}$ and Q_1.

Next, we have $K_1 \not\models Q_2$. Indeed, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}} = \{o_1, o_2\}$ and $a^{\mathcal{I}} = o_1$, $A^{\mathcal{I}} = \{o_1\}$, $P_1^{\mathcal{I}} = \{\langle o_1, o_1 \rangle\}$, and $P_2^{\mathcal{I}} = \{\langle o_1, o_2 \rangle\}$, is a model of K_1 but not of Q_2. To see that $K_2 \not\models Q_2$, simply extend $\mathcal{I}$ to the nominal $\{o\}$ by setting $\{o\}^{\mathcal{I}} = \{o_2\}$; then we have a model of K_2 but not of Q_2. Finally, $K_2 \models Q_3$. (Note that Q_3 is not a query over K_1, since $o \notin \mathbf{I}_{K_1}$.) Indeed, in every model $\mathcal{I}$ of K_2, $a^{\mathcal{I}}$ must be connected to some instance o_1 of A via P_1 by the axiom $A \sqsubseteq \exists P_1.A$. The axiom $A \sqsubseteq \exists P_2.\{o\}$ ensures that o_1 is connected to $o^{\mathcal{I}}$ via role P_2. Therefore, $\pi(x) = a^{\mathcal{I}}$, $\pi(y) = o_1$, and $\pi(o) = o^{\mathcal{I}}$ is a match for $\mathcal{I}$ and Q_3.

The query entailment problem for a DL $\mathcal{L}$ is in a complexity class $\mathcal{C}$, if given a KB K in $\mathcal{L}$ and a query Q, deciding $K \models Q$ is in $\mathcal{C}$; this is also called *combined complexity*, while the *data complexity* is the complexity of deciding $K \models Q$ where Q and all of K except $\mathcal{A}$ are fixed.

Note that in Definition 2.8, queries have no distinguished (i.e., free) variables, so they are Boolean queries. For a query $Q = \exists \mathbf{x}.\varphi(\mathbf{y}, \mathbf{x})$ with distinguished variables $\mathbf{y}$, the *query answering problem* over K consists in finding all the possible tuples $\mathbf{t}$ of individuals (of the same length as $\mathbf{y}$) such that $K \models \exists \mathbf{x}.\varphi(\mathbf{t}, \mathbf{x})$ holds. Query answering can be reduced to answering all possible such Boolean queries with individuals appearing in K; that is, to polynomially many (in the size of the ABox) query entailment problems.

3 A Tableaux Algorithm for Query Entailment

In this section, we describe an algorithm to solve the query entailment problem for PQs in the DLs of the $\mathcal{SH}$ family we have introduced. As shown in this and the next section, it is sound and complete for $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$. For $\mathcal{SHOIQ}$ it is sound, while completeness is not guaranteed.

An important note is that the query entailment problem in all these DLs is not reducible to knowledge base satisfiability, since in general the negation of a query is not expressible as a part of a knowledge base. For this reason, the known algorithms for reasoning over knowledge bases are insufficient. In general, a knowledge base has infinitely many (possibly infinite) models, and in principle we have to verify whether the query is satisfied in all of them. Our technique builds on the tableaux algorithm for satisfiability of $\mathcal{SHOIQ}$ knowledge bases in [24]. Informally, the difference is that the latter algorithm only focuses on problems that are reducible to satisfiability checking; hence, it only needs to ensure that the algorithm obtains a model if the knowledge base is satisfiable. In our case this is not enough. We need to make sure that the algorithm obtains a set of models that suffices to check query entailment. This adaption to query answering is inspired by [33], yet we deal with DLs that lack the finite model property. Like the algorithm in [24] we use *completion graphs*, finite relational structures that represent sets of models of a knowledge base. Roughly, an initial completion graph $\mathcal{G}_K$ for K is built. Then, by applying *expansion rules* repeatedly, new completion graphs are generated. The application of the rules is non-deterministic, and sometimes new individuals are introduced. Since every model of K is represented in some completion graph that results from the expansion, $K \models Q$ can be decided by considering a set of sufficiently expanded graphs $\mathcal{G}$. From each such $\mathcal{G}$ a single *canonical model* is constructed. Semantically, the finite set of these canonical models is sufficient for answering all queries Q of bounded size. Furthermore, we prove that entailment in the canonical model obtained from $\mathcal{G}$ can be checked effectively via a syntactic mapping of the variables in Q to the nodes in $\mathcal{G}$.

As customary with tableau-style algorithms, we give blocking conditions on the rules that ensure that the expansion of the graphs terminates. They are more involved than those in [24], which serve for satisfiability checking but not for query entailment, and they involve a parameter n which depends on Q.

3.1 Completion Graphs

Let $\mathbf{VN}$ be a countably infinite set of *variable nodes*, disjoint from the vocabulary used in defining queries and knowledge bases. A *completion graph* $\mathcal{G}$ consists of a finite labelled directed graph $(\mathsf{nodes}(\mathcal{G}), \mathsf{arcs}(\mathcal{G}), \mathcal{L})$ such that $\mathsf{nodes}(\mathcal{G}) \subseteq \mathbf{VN} \cup \mathbf{I}$ and a binary relation $\not\approx$ on $\mathsf{nodes}(\mathcal{G})$.[6] Each node v of $\mathcal{G}$ is labelled with a finite set $\mathcal{L}(v)$ of concepts and each arc $v \to w$ of $\mathcal{G}$ with a finite set $\mathcal{L}(v \to w)$ of roles. The node w is a *successor* of v and v a *predecessor* of w. The union of the successor and predecessor relations is the *neighbour* relation, and their respective transitive closures are called *descendant* and *ancestor*. The *distance* between two nodes v, v' in $\mathcal{G}$ is defined as the

[6] The $\not\approx$ relation is used to state explicit inequalities between nodes, i.e., that two nodes of a graph must be interpreted as different individuals (there is no unique name assumption). It is tacitly assumed that $\not\approx$ is symmetric.

shortest path between them. We refer to $\mathsf{in}(\mathcal{G}) = \{v \in \mathsf{nodes}(\mathcal{G}) \mid \{o\} \in \mathcal{L}(v), o \in \mathbf{I}\}$ as the *individual nodes* in $\mathcal{G}$ and to $\mathsf{vn}(\mathcal{G}) = \mathsf{nodes}(\mathcal{G}) \setminus \mathsf{in}(\mathcal{G})$ as the *variable nodes* in $\mathcal{G}$.[7]

Now we introduce completion graphs for a $\mathcal{SHOIQ}$ knowledge base $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$. Our algorithm uses a set of *TBox concepts* $\mathsf{tcon}(K) = \{\neg C \sqcup D \mid C \sqsubseteq D \in \mathcal{T}\}$. By requiring that each individual belongs to all these concepts, satisfaction of the TBox is enforced. The *subconcept closure* of a concept C is the smallest set of concept expressions containing C that is closed under subconcepts and their negation (expressed in NNF). Given a concept C and a role hierarchy $\mathcal{R}$, $\mathsf{clos}(C, \mathcal{R})$ is the smallest set containing the subconcept closure of C and all concepts of the form $\forall R'.D$ for each R' occurring in $\mathcal{R}$ or in C and for each concept expression D such that $\forall R.D$ or $NNF(\forall R.D)$ is in the subconcept closure of C. The *closure of* K, denoted $\mathsf{clos}(K)$, is the union of all $\mathsf{clos}(C, \mathcal{R})$ for each concept C occurring in $\mathsf{tcon}(K)$. In the following, let $K_{\mathcal{A}} = \langle \mathcal{T}_{\mathcal{A}}, \mathcal{R}, \emptyset \rangle$ where $\mathcal{T}_{\mathcal{A}}$ is as in Section 2.1.

Definition 3.1 (Completion graph [24]) A *completion graph* $\mathcal{G}$ *for a knowledge base* K is a completion graph in which each node v is labelled with $\mathcal{L}(v) \subseteq \mathsf{clos}(K_{\mathcal{A}}) \cup \{\{o\} \mid o \in \mathbf{I}\} \cup \{\leq m\, R.C \mid \leq n\, R.C \in \mathsf{clos}(K_{\mathcal{A}}) \text{ and } m \leq n\}$, and in which each arc $v \to w$ has a label $\mathcal{L}(v \to w) \subseteq \mathbf{R}_{K_{\mathcal{A}}}$. If for two nodes v, w there is no arc $v \to w$ in $\mathcal{G}$, we consider $\mathcal{L}(v \to w) = \emptyset$. For each arc $v \to w$ and role R, 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.

In order to provide a method for verifying entailment of a conjunctive query Q in a knowledge base K, we first associate with K an initial completion graph and then we generate new completion graphs by applying *expansion rules*.

The *initial completion graph* $\mathcal{G}_K$ associated with K has a node a labelled with $\mathcal{L}(a) = \{\{a\}\} \cup \mathsf{tcon}(K_{\mathcal{A}})$, for each individual $a \in \mathbf{I}_K$, and the relation $\not\approx$ is empty.

Example 3.2 In our running example, $\mathcal{G}_{K_1}$ contains only the node a which has the label $\mathcal{L}(a) := \{\{a\}, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\neg A, \neg\{a\} \sqcup A\}$. $\mathcal{G}_{K_2}$ contains two nodes, a and o, with the labels $\mathcal{L}(a) := \{\{a\}, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\{o\}, \neg\{a\} \sqcup A\}$ and $\mathcal{L}(o) := \{\{o\}, \neg A \sqcup \exists P_1.A, \neg A \sqcup \exists P_2.\{o\}, \neg\{a\} \sqcup A\}$. In both graphs the $\not\approx$ relation is empty.

From this initial $\mathcal{G}_K$, we obtain new completion graphs by applying expansion rules, which may introduce new nodes. Variable nodes are always introduced as successors of exactly one existing node. Hence, the variable nodes in a completion graph form a set of trees that have individual nodes as roots. It may also happen that one of these variable nodes has an individual node as its successor, thus we have a tree of variable nodes that has a branch ending with an arc leading to an individual node. If a completion graph $\mathcal{F}$ for K has no such arcs, then $\mathcal{F}$ is a set of trees of variable nodes, whose roots are possibly interconnected individual nodes. This special kind of completion graphs are called *completion forests*.

For any knowledge base K, the initial completion graph $\mathcal{G}_K$ is a completion forest. If K is a $\mathcal{SHIQ}$ knowledge base the expansion rules only introduce variable nodes and any completion graph obtained by applying the expansion rules is a completion forest. This is not the case if K is a knowledge base in some DL with nominals, since arcs from variable to individual nodes may be introduced.

[7] Our individual nodes correspond to *nominal nodes* in [24], and our variable nodes to *blockable nodes*.

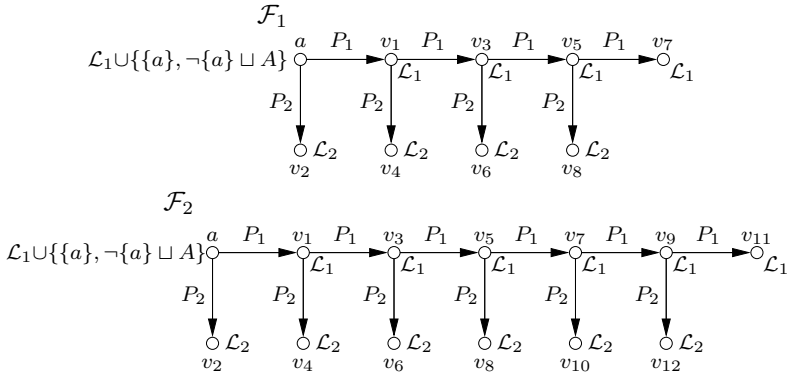

Fig. 3.1 Completion graphs for the example knowledge base K_1

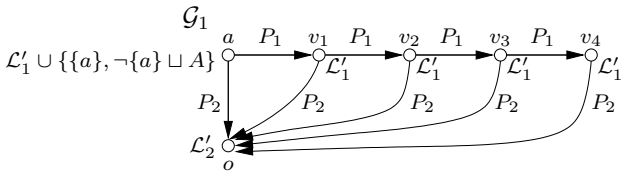

Fig. 3.2 A completion graph for the example knowledge base K_2

Example 3.3 In Figure 3.1, we show the completion graphs $\mathcal{F}_1$ and $\mathcal{F}_2$ for K_1, which have an empty $\not\approx$ relation (for simplicity, omitted in the figure), and where

$$\mathcal{L}_1 = \{A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A,\ \exists P_1.A,\ \exists P_2.\neg A\}$$
$$\mathcal{L}_2 = \{\neg A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\neg A\}.$$

Note that both $\mathcal{F}_1$ and $\mathcal{F}_2$ are completion forests. Figure 3.2 shows the completion graph $\mathcal{G}_1$, which has again an empty $\not\approx$ relation, and where

$$\mathcal{L}'_1 = \{A,\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\{o\},\ \exists P_1.A,\ \exists P_2.\{o\}\}$$
$$\mathcal{L}'_2 = \{\{o\},\ \neg A \sqcup \exists P_1.A,\ \neg A \sqcup \exists P_2.\{o\},\ \neg\{a\} \sqcup A,\ \neg A,\ \neg\{a\}\}.$$

Next, before giving the expansion rules, we define a notion of blocking which depends on a depth parameter $n \geq 0$. This notion generalises blocking in [24], where the parameter n is not present.

Definition 3.4 (Blockable n-graph, n-graph equivalence) Given an integer $n \geq 0$ and a completion graph $\mathcal{G}$, 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:

- $\psi(v) = v'$,
- for every $w \in \mathsf{nodes}(\mathcal{G}^{n,v})$, $\mathcal{L}(w) = \mathcal{L}(\psi(w))$,
- $\mathsf{arcs}(\mathcal{G}^{n,v'}) = \{\psi(w) \to \psi(w') \mid w \to w' \in \mathsf{arcs}(\mathcal{G}^{n,v})\}$,

– for every $w \to w' \in \mathsf{arcs}(\mathcal{G}^{n,v})$, $\mathcal{L}(w \to w') = \mathcal{L}(\psi(w) \to \psi(w'))$.

As discussed above, in the algorithm variable nodes occur only in tree-shaped structures. The n-graph of each variable 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. The leaves of the graph are the leaves of the tree in the usual sense. For the completion graph obtained from a $\mathcal{SHIQ}$ KB, since there are no arcs from a variable node to a nominal node, all n-graphs are actually trees of depth at most n.

Definition 3.5 (n-witness, graph-blocking) 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* ψ the node $\psi^{-1}(w)$ in $\mathcal{G}^{n,v}$.

Note that if some G' graph-blocks some G via a bijection ψ, then the particular ψ does not matter and any other bijection satisfying the three conditions of Definition 3.4 could be equivalently used. Therefore, we will always assume a fixed arbitrary bijection from a graph-blocked G to a graph-blocking G', and denote it ψ. Moreover, we often omit ψ and simply say G' graph-blocks G, v_1 graph-blocks v_2, etc.

Example 3.6 In $\mathcal{F}_1$, v_1 and v_5 are 1-graph equivalent, v_1 is a 1-witness of v_5 (but not vice versa); ${\mathcal{F}_1}^{1,v_1}$ graph-blocks ${\mathcal{F}_1}^{1,v_5}$; and v_1 (resp., v_3, v_4) graph-blocks v_5 (resp., v_7, v_8).

Definition 3.7 (n-blocking) For an integer $n \geq 0$ and a completion graph $\mathcal{G}$, 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 some blockable n-graph in $\mathcal{G}$ that is graph-blocked; in this case we say that v is (directly) n-blocked by $\psi(v)$ (i.e., by the node in $\mathcal{G}$ that graph-blocks v).[8] 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*, which is the blocking used in [26, 24]. When $n = 0$, then n-blocking corresponds to blocking by equal node labels (equality blocking [2]), which is a sufficient blocking condition in some DLs weaker than $\mathcal{SHIQ}$.

Example 3.8 Consider the completion forests $\mathcal{F}_1$ and $\mathcal{F}_2$ in Figure 3.1. The nodes v_7 and v_8 in $\mathcal{F}_1$ are (directly) 1-blocked. Similarly, v_{11} and v_{12} in $\mathcal{F}_2$ are (directly) 2-blocked. Consider the completion graph $\mathcal{G}_1$ in Figure 3.2. In it, ${\mathcal{G}_1}^{1,v_1}$ graph-blocks ${\mathcal{G}_1}^{1,v_3}$; v_4 is (directly) 1-blocked.

Now we can give our expansion rules, which are essentially the same as in [24]. The main differences are that "blocked" is uniformly replaced by "n-blocked" and that in the generating rules, the labels of the newly generated nodes must contain $\mathsf{tcon}(K)$ (because we don't assume an empty TBox). The rules use two operations on completion graphs called merge and prune (prune does not appear in the rules, but it is used by

[8] Note that the graph-blocking n-graph is unique, and thus by our assumption also the bijection ψ is unique.

merge). To illustrate the use of these operations, consider the $\leq$-rule. Suppose a node v is labelled by the concept $\leq 2\, S.C$ and has three successors v_1, v_2, v_3 labelled with C, and $v_2 \not\approx v_3$ does not hold. Then we can make v satisfy $\leq 2\, S.C$, by merging the nodes v_2 and v_3 into one. For this purpose, we use $\mathsf{merge}(v_2, v_3)$, which then applies $\mathsf{prune}(v_2)$. Intuitively, $\mathsf{merge}(v_2, v_3)$ merges the node v_2 into v_3: the label of v_2 is added to the label of v_3, all incoming arcs of v_2 are copied to v_3, and the outgoing arcs of v_2 to an individual node are also copied to v_3. After the merging, $\mathsf{prune}(v_2)$ removes v_2 from $\mathcal{G}$ and, recursively, all its variable successors.

Formally, for a completion graph $\mathcal{G}$ and $v, w \in \mathsf{nodes}(\mathcal{G})$, the operation $\mathsf{prune}(w)$ yields a graph that is obtained from $\mathcal{G}$ as follows:

1. For each successor w' of w, remove $w \to w'$ from $\mathsf{arcs}(\mathcal{G})$, and if $w' \in \mathsf{vn}(\mathcal{G})$, then $\mathsf{prune}(w')$.
2. Remove w from $\mathsf{nodes}(\mathcal{G})$.

The operation $\mathsf{merge}(w, v)$ yields a forest obtained from $\mathcal{G}$ as follows:

1. For each $w' \in \mathsf{nodes}(\mathcal{G})$ such that $w' \to w \in \mathsf{arcs}(\mathcal{G})$
   (a) if neither $v \to w'$ nor $w' \to v$ is in $\mathsf{arcs}(\mathcal{G})$, then add $w' \to v$ to $\mathsf{arcs}(\mathcal{G})$ and set $\mathcal{L}(w' \to v) := \mathcal{L}(w' \to w)$;
   (b) if $w' \to v$ is in $\mathsf{arcs}(\mathcal{G})$, then set $\mathcal{L}(w' \to v) := \mathcal{L}(w' \to v) \cup \mathcal{L}(w' \to w)$;
   (c) if $v \to w'$ is in $\mathsf{arcs}(\mathcal{G})$, then set $\mathcal{L}(v \to w') := \mathcal{L}(v \to w') \cup \{\mathsf{Inv}(R) \mid R \in \mathcal{L}(w' \to w)\}$;
   (d) remove $w' \to w$ from $\mathsf{arcs}(\mathcal{G})$.
2. For each $w' \in \mathsf{in}(\mathcal{G})$ such that $w \to w' \in \mathsf{arcs}(\mathcal{G})$
   (a) if neither $v \to w'$ nor $w' \to v$ is in $\mathsf{arcs}(\mathcal{G})$, then add $v \to w'$ to $\mathsf{arcs}(\mathcal{G})$ and set $\mathcal{L}(v \to w') := \mathcal{L}(w \to w')$;
   (b) if $v \to w'$ is in $\mathsf{arcs}(\mathcal{G})$, then set $\mathcal{L}(v \to w') := \mathcal{L}(v \to w') \cup \mathcal{L}(w \to w')$;
   (c) if $w' \to v$ is in $\mathsf{arcs}(\mathcal{G})$, then set $\mathcal{L}(w' \to v) := \mathcal{L}(w' \to v) \cup \{\mathsf{Inv}(R) \mid R \in \mathcal{L}(w \to w')\}$;
   (d) remove $w \to w'$ from $\mathsf{arcs}(\mathcal{G})$.
3. Set $\mathcal{L}(v) := \mathcal{L}(v) \cup \mathcal{L}(w)$.
4. Add $v \not\approx w'$ for each w' with $w \not\approx w'$.
5. $\mathsf{prune}(w)$.

To obtain new completion graphs from the initial $\mathcal{G}_K$, we apply the rules in Table 3.1. Note that their application is non-deterministic. Different choices for E in the $\sqcup$-rule and the *choose*-rule generate different graphs. The choice of the nodes to be merged in the $\leq$- and $\leq_o$-rules is also non-deterministic. The $\exists$-rule, the $\geq$-rule and the o?-rule are called *generating rules*, since they add new nodes to the graph. The $\leq$-rule, the o-rule and the $\leq_o$-rule are *shrinking rules*, since they merge two nodes of the graph into one.

Note that the o-rule merges two nodes whenever their labels share a nominal. Like in [24], we assume that whenever this rule is applicable, it is applied immediately. This consideration allows us to assume that, in every completion graph, each nominal occurs in the label of at most one node.

An important note is that the o?-rule is never applicable for $\mathcal{SHOQ}$, $\mathcal{SHIQ}$, and $\mathcal{SHOI}$ KBs, which allows us to prove termination (see below).[9] For $\mathcal{SHOIQ}$ KBs,

[9] This also holds for $\mathcal{SHOIQ}$ KBs without interaction between number restrictions, inverse roles, and nominals, in particular for $\mathcal{SHOIQ}$ KBs that result from internalising the ABox of a $\mathcal{SHIQ}$ KB, as described in Section 2.1.

$\sqcap$-rule:	if	$C_1 \sqcap C_2 \in \mathcal{L}(v)$, v is not indirectly n-blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(v)$,
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{C_1, C_2\}$.
$\sqcup$-rule:	if	$C_1 \sqcup C_2 \in \mathcal{L}(v)$, v is not indirectly n-blocked, and $\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset$,
	then	$\mathcal{L}(v) := \mathcal{L}(v) \cup \{E\}$ for some $E \in \{C_1, C_2\}$.
$\exists$-rule:	if	$\exists R.C \in \mathcal{L}(v)$, v is not n-blocked, and
		v has no n-safe R-neighbour w with $C \in \mathcal{L}(w)$,
	then	create a new node w with $\mathcal{L}(v \rightarrow w) := \{R\}$ and $\mathcal{L}(w) := \{C\} \cup \mathsf{tcon}(K)$.
$\forall$-rule:	if	$\forall R.C \in \mathcal{L}(v)$, v is not indirectly n-blocked, and
		v has an R-neighbour w with $C \notin \mathcal{L}(w)$,
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{C\}$.
$\forall_+$-rule:	if	$\forall R.C \in \mathcal{L}(v)$, v is not indirectly n-blocked,
		there is some R' with $\mathsf{Trans}(R')$ and $R' \sqsubseteq^* R$, and
		there is an R'-neighbour w of v with $\forall R'.C \notin \mathcal{L}(w)$,
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{\forall R'.C\}$.
choose-rule:	if	$\leq m\, S.C \in \mathcal{L}(v)$, v is not indirectly n-blocked, and
		there is an S-neighbour w of v with $\{C, NNF(\neg C)\} \cap \mathcal{L}(w) = \emptyset$,
	then	$\mathcal{L}(w) := \mathcal{L}(w) \cup \{E\}$ for some $E \in \{C, NNF(\neg C)\}$.
$\geq$-rule:	if	$\geq m\, S.C \in \mathcal{L}(v)$, v is not n-blocked, and
		there are not m n-safe S-neighbours $w_1, \ldots, w_m$ of v
		such that $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j$ for $1 \leq i < j \leq m$,
	then	create new nodes $w_1, \ldots, w_m$ with $\mathcal{L}(v \rightarrow w_i) := \{S\}$,
		$\mathcal{L}(w_i) := \{C\} \cup \mathsf{tcon}(K)$, and $w_i \not\approx w_j$ for $1 \leq i < j \leq m$.
$\leq$-rule:	if	$\leq m\, S.C \in \mathcal{L}(v)$, v is not indirectly n-blocked,
		$\lvert\{w \mid w \text{ is an } S\text{-neighbour of } v \text{ and } C \in \mathcal{L}(w)\}\rvert > m$, there are
		S-neighbours w, w' of v with not $w \not\approx w'$, and $C \in \mathcal{L}(w) \cap \mathcal{L}(w')$,
	then	(i) if $w \in \mathsf{in}(\mathcal{G})$, then $\mathsf{merge}(w', w)$; else
		(ii) if $w' \in \mathsf{in}(\mathcal{G})$ or w' is an ancestor of w, $\mathsf{merge}(w, w')$;
		else (iii) $\mathsf{merge}(w', w)$.
o-rule:	if	there are v, v' with not $v \not\approx v'$ and $\{o\} \in \mathcal{L}(v) \cap \mathcal{L}(v')$ for some $o \in \mathsf{in}(\mathcal{G})$,
	then	(i) if $v \in \mathbf{I}$, then $\mathsf{merge}(v', v)$; else (ii) $\mathsf{merge}(v, v')$.
o?-rule:	if	$\leq m\, S.C \in \mathcal{L}(v)$, $v \in \mathsf{in}(\mathcal{G})$, $v' \in \mathsf{vn}(\mathcal{G})$, $C \in \mathcal{L}(v')$,
		v' is an S-neighbour of v, v is a successor of v', and there is
		no m' with $1 \leq m' \leq m$ such that: (i) $\leq m'\, S.C \in \mathcal{L}(v)$ and
		(ii) v has m' S-neighbours $w_1, \ldots, w_{m'} \in \mathsf{in}(\mathcal{G})$
		with $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j$ for all $1 \leq j < i \leq m'$,
	then	guess $m' \leq m$, set $\mathcal{L}(v) := \mathcal{L}(v) \cup \{\leq m'\, S.C\}$, and
		create m' new nodes $w_1, \ldots, w_{m'}$ with $\mathcal{L}(v \rightarrow w_i) := \{S\}$,
		$\mathcal{L}(w_i) := \{C, \{o_i\}\} \cup \mathsf{tcon}(K)$ for some $o_i \in \mathbf{I} \setminus \mathsf{in}(\mathcal{G})$,
		and $w_i \not\approx w_j$ for all $1 \leq j < i \leq m'$.
$\leq_o$-rule:	if	$\leq m\, S.C \in \mathcal{L}(v)$, $v \in \mathsf{in}(\mathcal{G})$, $v' \in \mathsf{vn}(\mathcal{G})$ is an S-neighbour of v,
		$C \in \mathcal{L}(v')$, v has m S-neighbours $w_1, \ldots, w_m \in \mathsf{in}(\mathcal{G})$ with
		$C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j$ for all $1 \leq j < i \leq m$, and
		$w \in \mathsf{in}(\mathcal{G})$ is an S-neighbour of v, $C \in \mathcal{L}(w)$ and not $v' \not\approx w$,
	then	$\mathsf{merge}(v', w)$.

Table 3.1 Expansion Rules

however, the o?-rule is needed and the naive application of the expansion rules can lead to non-termination. Horrocks and Sattler in [24] give a prioritised strategy for rule application which guarantees termination of their satisfiability testing algorithm. Unfortunately, this strategy does not work for our query answering algorithm; we

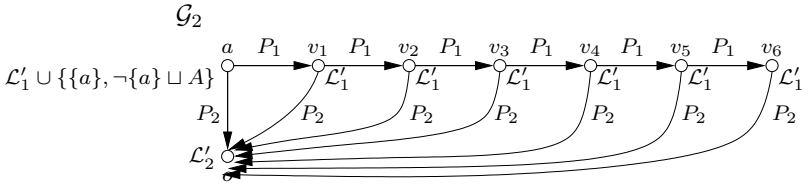

Fig. 3.3 2-complete completion graph for the example knowledge base K_2

cannot ensure that it terminates on $\mathcal{SHOIQ}$ KBs (although we believe that it will do so in many cases).

Definition 3.9 (Clash-free completion graph) A completion graph $\mathcal{G}$ contains a *clash* if one of the following holds:

1. For some $v \in \mathsf{nodes}(\mathcal{G})$ and some concept name A, $\{A, \neg A\} \subseteq \mathcal{L}(v)$.
2. For some $v \in \mathsf{nodes}(\mathcal{G})$ with $\leq n\, S.C \in \mathcal{L}(v)$, v has $n+1$ S-neighbours $w_0, \ldots, w_n$ such that, for all w_i, w_j with $0 \leq i < j \leq n$, $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j \in \mathcal{G}$.
3. For some $o \in \mathbf{I}$ and some $v, v' \in \mathsf{nodes}(\mathcal{G})$, $\{o\} \in \mathcal{L}(v) \cap \mathcal{L}(v')$ and $v \not\approx v' \in \mathcal{G}$.

If $\mathcal{G}$ does not contain a clash, then $\mathcal{G}$ is *clash-free*.

Definition 3.10 (n-complete completion graph) A completion graph $\mathcal{G}$ is n-complete, if no rule in Table 3.1 can be applied to it.

For a knowledge base K, we denote by $\mathbb{G}_K$ the set of all completion graphs that can be obtained from the initial $\mathcal{G}_K$ by applying the expansion rules, and by $\mathsf{ccf}_n(\mathbb{G}_K)$ the set of completion graphs in $\mathbb{G}_K$ that are n-complete and clash free.

Example 3.11 Both $\mathcal{F}_1$ and $\mathcal{F}_2$ can be obtained from $\mathcal{G}_{K_1}$ by applying the expansion rules, and they are both clash-free. $\mathcal{F}_1$ is 1-complete and $\mathcal{F}_2$ is 2-complete, so $\mathcal{F}_1 \in \mathsf{ccf}_1(\mathbb{G}_{K_1})$ and $\mathcal{F}_2 \in \mathsf{ccf}_2(\mathbb{G}_{K_1})$. Consider also the completion graphs $\mathcal{G}_1$ in Figure 3.2 and $\mathcal{G}_2$ in Figure 3.3 (where $\mathcal{L}'_1$ and $\mathcal{L}'_2$ are as in Example 3.3). Both can be obtained from $\mathcal{G}_{K_2}$ by means of the expansion rules. They are both clash-free completion graphs, and they are 1-complete and 2-complete respectively, so $\mathcal{G}_1 \in \mathsf{ccf}_1(\mathbb{G}_{K_2})$ and $\mathcal{G}_2 \in \mathsf{ccf}_2(\mathbb{G}_{K_2})$.

3.2 Models of a Completion Graph

Semantically, by viewing all the nodes of a completion graph as individuals, we can interpret a completion graph in a very similar way as we interpret a knowledge base. Intuitively, every individual in K is represented by a node of the completion graph, but the completion graph may have additional nodes. An interpretation of the individuals, concepts, and roles in $\mathcal{G}$ is an interpretation of K, possibly extended to interpret these additional nodes, and we can see it as a representation of a set of models of K.

Definition 3.12 (Model of a completion graph) An *extended interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation as in Definition 2.7 that in addition assigns to each node $v \in \mathbf{VN}$ an element $v^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Let $\mathcal{G} \in \mathbb{G}_K$. Then $\mathcal{I}$ is a *model* of $\mathcal{G}$ w.r.t. K, written $\mathcal{I} \models_K \mathcal{G}$, if:

1. $\mathcal{I} \models K$, and
2. for all $v, w \in \mathsf{nodes}(\mathcal{G})$, $\{C \in \mathcal{L}(v)\} \subseteq \{C \mid v^{\mathcal{I}} \in C^{\mathcal{I}}\}$, $\{R \in \mathcal{L}(v \to w)\} \subseteq \{R \mid \langle v^{\mathcal{I}}, w^{\mathcal{I}} \rangle \in R^{\mathcal{I}}\}$, and $v \not\approx w \in \mathcal{G}$ implies $v^{\mathcal{I}} \neq w^{\mathcal{I}}$.

We emphasize that, in order to be a model of a completion graph for K, an extended interpretation must include an ordinary interpretation that is a model of K (item 1).

We say that two extended interpretations $\mathcal{I}$ and $\mathcal{J}$ are *equal on a set* $N \subseteq \mathbf{VN} \cup \mathbf{I}$, if $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ and for every $v, w \in N$, $v^{\mathcal{I}} = v^{\mathcal{J}}$, $\{C \mid v^{\mathcal{I}} \in C^{\mathcal{I}}\} = \{C \mid v^{\mathcal{J}} \in C^{\mathcal{J}}\}$, and $\{R \mid \langle v^{\mathcal{I}}, w^{\mathcal{I}} \rangle \in R^{\mathcal{I}}\} = \{R \mid \langle v^{\mathcal{J}}, w^{\mathcal{J}} \rangle \in R^{\mathcal{J}}\}$. Furthermore, we call an extended interpretation $\mathcal{J}$ a *K-extension* of an ordinary interpretation $\mathcal{I}$, if $\mathcal{J}$ equals $\mathcal{I}$ on $\mathbf{I}_K$.

The initial completion graph $\mathcal{G}_K$ is just an alternative representation of the knowledge base, and it has exactly the same models. The following lemma is immediate from the definition of the semantics of knowledge bases and of $\mathcal{G}_K$.

Lemma 3.13 *For every (extended or ordinary) interpretation $\mathcal{I}$, $\mathcal{I} \models_K \mathcal{G}_K$ iff $\mathcal{I} \models K$.*

When we expand the graph, we make choices and obtain new graphs that represent a subset of the models of the knowledge base K. The union of the sets of models of all graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$, when restricted to the language of K, coincides with the set of models of K, independently of the value of n. Therefore, if we want to check all models of K, we must check all models of all graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$ for some n.

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

Proof Consider an interpretation $\mathcal{I}$ such that $\mathcal{I} \models K$. Intuitively, every K-extension $\mathcal{J}$ of $\mathcal{I}$ is a model of the initial $\mathcal{G}_K$, and $\mathcal{I}$ can be used to guide the non-deterministic choices when applying the expansion rules, in such a way that clashes are avoided until a complete graph is reached. This is the same intuition underlying the proof of completeness given in [24].[10] Formally, let $\mathbf{G}^k$ denote the set of completion graphs obtained from $\mathcal{G}_K$ by at most k applications of the expansion rules, and $\mathsf{cf}(\mathbf{G}^k)$ the set of these graphs that are clash-free. We prove the following claim by induction on $k \geq 0$:

Claim 1 If $\mathcal{I} \models K$, then for every $k \geq 0$ there is some K-extension $\mathcal{J}$ of $\mathcal{I}$ and some $\mathcal{G} \in \mathsf{cf}(\mathbf{G}^k)$ such that $\mathcal{J} \models_K \mathcal{G}$.

If $k = 0$, then $\mathsf{cf}(\mathbf{G}^k) = \{\mathcal{G}_K\}$ and the claim holds by Lemma 3.13. For the inductive step, we use the following fact:

Claim 2 Let $\mathcal{G} \in \mathbb{G}_K$, let $\mathcal{J} \models_K \mathcal{G}$, and let r be any rule in Table 3.1 that is applicable to $\mathcal{G}$. Then, there exist a completion graph $\mathcal{G}'$ obtainable from $\mathcal{G}$ by applying r and an extended interpretation $\mathcal{J}'$ equal to $\mathcal{J}$ on $\mathsf{nodes}(\mathcal{G})$ such that $\mathcal{J}' \models_K \mathcal{G}'$.

The (straightforward) proof of Claim 2 is given in the Appendix. Consider now $\mathcal{G} \in \mathsf{cf}(\mathbf{G}^k)$. If $\mathcal{J} \models_K \mathcal{G}$, then by Claim 2 there exist some $\mathcal{J}'$ equal to $\mathcal{J}$ on $\mathsf{nodes}(G)$ and some $\mathcal{G}' \in \mathbf{G}^{k+1}$ such that $\mathcal{J}' \models_K \mathcal{G}'$. As $\mathcal{J}$ is a K-extension of $\mathcal{I}$ and $\mathbf{I}_K \subseteq \mathsf{nodes}(\mathcal{G})$, also $\mathcal{J}'$ is a K-extension of $\mathcal{I}$ and a model of $\mathcal{G}'$ w.r.t. K. Hence, $\mathcal{G}' \in \mathsf{cf}(\mathbf{G}^{k+1})$ and Claim 1 holds.

[10] The details of the proof are quite different, however, since the authors of [24] use tableaux, while we use completion graphs as model representations.

Finally, it is easy to see that Claim 1 implies the statement of the proposition, as every sufficiently expanded completion graph that did not reach a clash will be n-complete, i.e., by taking a sufficiently large k, we can ensure that $\mathsf{cf}(\mathbf{G}^k) = \mathsf{ccf}_n(\mathbb{G}_K)$. □

3.3 Answering Positive Queries

Recall that for a knowledge base K and a query Q, $K \models Q$ holds iff $\mathcal{I} \models Q$ for every model $\mathcal{I}$ of K. We define an analogous notion of query entailment in a completion graph $\mathcal{G}$: $\mathcal{G} \models_K Q$ iff $\mathcal{I} \models Q$ for every model $\mathcal{I}$ of $\mathcal{G}$ w.r.t. K. We are interested in checking whether $K \models Q$, which means that entailment of Q has to be verified in every model of K. To this end, we may choose an arbitrary n and check entailment of Q in each graph $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. This is sound since all the models of K are represented by the graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$.

Proposition 3.15 *Let $n \geq 0$. Then $K \models Q$ iff $\mathcal{G} \models_K Q$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.*

Proof For the only if direction, assume $K \models Q$. Consider $\mathcal{G} \in \mathbb{G}_K$ and some $\mathcal{I}$ such that $\mathcal{I} \models_K \mathcal{G}$. Since $\mathcal{I} \models K$ by definition, $K \models Q$ implies that $\mathcal{I} \models Q$. Hence, $\mathcal{G} \models_K Q$. The if direction is shown by contraposition. If $K \not\models Q$, then there exists some model $\mathcal{I}$ of K such that $\mathcal{I} \not\models Q$. By Proposition 3.14, there is some K-extension $\mathcal{J}$ of $\mathcal{I}$ and some $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ such that $\mathcal{J} \models_K \mathcal{G}$. Note that $\mathcal{I} \not\models Q$ implies $\mathcal{J} \not\models Q$, since $\mathcal{J}$ and $\mathcal{I}$ can only differ in the interpretation of the nodes in $\mathbf{VN}$, which is irrelevant for Q. Thus, $\mathcal{G} \not\models_K Q$. □

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. In the rest of this section, we show that if a suitable n is chosen, entailment in all the models of K can be decided effectively by deciding the existence of a mapping of the query into each $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.

Definition 3.16 [Query mapping] Let $Q = \exists \mathbf{x}.\varphi(\mathbf{x})$ be a PQ and let $\mathcal{G}$ be a completion graph. Let $\mu : \mathsf{VI}(Q) \to \mathsf{nodes}(\mathcal{G})$ be a total function such that $\{a\} \in \mathcal{L}(\mu(a))$ for each individual a in $\mathsf{VI}(Q)$. We write $C(x) \xhookrightarrow{\mu} \mathcal{G}$ if $C \in \mathcal{L}(\mu(x))$, and $S(x, x') \xhookrightarrow{\mu} \mathcal{G}$ if $\mu(x')$ is an S-neighbour of $\mu(x)$. Let γ be the Boolean expression obtained from $\varphi(\mathbf{x})$ by replacing each atom α in φ with $\top$, if $\alpha \xhookrightarrow{\mu} \mathcal{G}$, and with $\bot$ otherwise. We say that μ is a *mapping for Q into $\mathcal{G}$*, denoted $Q \xhookrightarrow{\mu} \mathcal{G}$, 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 $\mathcal{G}$.

Note that $S(x, x') \xhookrightarrow{\mu} \mathcal{G}$ does not imply $S \in \mathcal{L}(\mu(x) \to \mu(x'))$, but only that a subrole of S occurs in the label. The correctness of the mapping for role atoms is thus related to the notion of S-neighbour (see Definition 3.1).

Example 3.17 We have that $Q_1 \xhookrightarrow{\mu_1} \mathcal{F}_1$ and $Q_1 \xhookrightarrow{\mu_1'} \mathcal{F}_2$, witnessed by $\mu_1(x) = \mu_1'(x) = a$, $\mu_1(y) = \mu_1'(y) = v_1$ and $\mu_1(z) = \mu_1'(z) = v_2$. Note that there is no mapping of Q_2 into $\mathcal{F}_2$ or $\mathcal{F}_1$ satisfying the above conditions. The mappings $\mu_2(x) = \mu_2'(x) = a$, $\mu_2(y) = \mu_2'(y) = v_1$ and $\mu_2(o) = \mu_2'(o) = o$ show that $Q_3 \xhookrightarrow{\mu_2} \mathcal{G}_1$ and $Q_3 \xhookrightarrow{\mu_2'} \mathcal{G}_2$.

Indeed, for completion graphs $\mathcal{G}$ for K, syntactic mappability $Q \hookrightarrow \mathcal{G}$ implies semantic consequence $\mathcal{G} \models_K Q$.

Lemma 3.18 *If $Q \hookrightarrow \mathcal{G}$, then $\mathcal{G} \models_K Q$.*

Proof Since $Q \hookrightarrow \mathcal{G}$, there is a mapping $\mu : \mathsf{VI}(Q) \to \mathsf{nodes}(\mathcal{G})$ satisfying Definition 3.16. Let $\mathcal{I}$ be a model of $\mathcal{G}$ w.r.t. K. Then $v^{\mathcal{I}} \in C^{\mathcal{I}}$ if $C \in \mathcal{L}(v)$; and if w is an R-neighbour of v, then $\langle w^{\mathcal{I}}, v^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$. We can define a match for $\mathcal{I}$ and Q by setting $\pi(x) = \mu(x)^{\mathcal{I}}$ for every $x \in \mathsf{VI}(Q)$. It satisfies $\pi(a) = a^{\mathcal{I}}$ for each individual a and $\mathcal{I}, \pi \models \alpha$ for each atom α such that $\alpha \overset{\mu}{\hookrightarrow} \mathcal{G}$. Hence, $\mathcal{I} \models_K Q$, which implies $\mathcal{G} \models_K Q$. □

Since every model of the KB K is represented by some completion graph, we already know that Q is entailed by K if there is a mapping for Q in each $\mathcal{G}$. We prove that the converse also holds. Now the blocking conditions come into play and the mapping will only be feasible if n is sufficiently large. We show that provided $\mathcal{G}$ has been expanded far enough, a suitable mapping μ into $\mathcal{G}$ can be constructed from a single model $\mathcal{I}_{\mathcal{G}}$ of K, which we call the *canonical model induced by* $\mathcal{G}$. In fact, entailment in this model implies entailment in the completion graph for *all* queries Q of bounded size. Indeed, we will see that the mapping μ can be constructed from any match for $\mathcal{I}_{\mathcal{G}}$ and Q.

3.3.1 Tableaux and Canonical Models

To build the canonical model induced by $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ (with $n \geq 1$), we *unravel* $\mathcal{G}$ into a tableau $T_{\mathcal{G}}$. This tableau induces a model for K.[11] Each path to a node in $\mathcal{G}$ is a node of $T_{\mathcal{G}}$. Every blocked node points back to the node that blocks it, creating a loop that generates infinite paths. Thus, if $\mathcal{G}$ has blocked nodes, its tableau is an infinite structure. Defining a model from T is straightforward. The definition of tableau is based on the one in [24].

Definition 3.19 (Tableau) A triple $T = \langle \mathbf{S}, \mathcal{L}, \mathcal{E} \rangle$ is a tableau for a KB $K = \langle \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle$, if $\mathbf{S}$ is a non-empty set; $\mathcal{L} : \mathbf{S} \to 2^{\mathsf{clos}(K_{\mathcal{A}})}$ maps each element in $\mathbf{S}$ to a set of concepts; and $\mathcal{E} : \mathbf{R}_{K_{\mathcal{A}}} \to 2^{\mathbf{S} \times \mathbf{S}}$ maps each role to a set of pairs of elements in $\mathbf{S}$. Furthermore, for all $s, t \in \mathbf{S}$; $C, C_1, C_2 \in \mathsf{clos}(K_{\mathcal{A}})$; and $R, R', S \in \mathbf{R}_{K_{\mathcal{A}}}$, T satisfies:

(P0) if $C \in \mathsf{tcon}(K)$ then $C \in \mathcal{L}(s)$.
(P1) if $C \in \mathcal{L}(s)$, then $\neg C \notin \mathcal{L}(s)$;
(P2) if $C_1 \sqcap C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ and $C_2 \in \mathcal{L}(s)$;
(P3) if $C_1 \sqcup C_2 \in \mathcal{L}(s)$, then $C_1 \in \mathcal{L}(s)$ or $C_2 \in \mathcal{L}(s)$;
(P4) if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(R)$, then $C \in \mathcal{L}(t)$;
(P5) if $\exists R.C \in \mathcal{L}(s)$, then $\langle s, t \rangle \in \mathcal{E}(R)$ and $C \in \mathcal{L}(t)$ for some $t \in \mathbf{S}$;
(P6) if $\forall R.C \in \mathcal{L}(s)$ and $\langle s, t \rangle \in \mathcal{E}(R')$ for some $R' \sqsubseteq^* R$ with $\mathsf{Trans}(R') = \mathit{true}$, then $\forall R'.C \in \mathcal{L}(t)$;
(P7) if $\leq n\, S.C \in \mathcal{L}(s)$, then $|\{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(t)\}| \leq n$;
(P8) if $\geq n\, S.C \in \mathcal{L}(s)$, then $|\{t \in \mathbf{S} \mid \langle s, t \rangle \in \mathcal{E}(S) \text{ and } C \in \mathcal{L}(t)\}| \geq n$;
(P9) if $\langle s, t \rangle \in \mathcal{E}(R)$ and $\leq n\, S.C \in \mathcal{L}(s)$, then $\{C, \mathit{NNF}(\neg C)\} \cap \mathcal{L}(t) \neq \emptyset$;
(P10) if $\langle s, t \rangle \in \mathcal{E}(R)$ and $R \sqsubseteq^* R'$ then $\langle s, t \rangle \in \mathcal{E}(R')$;
(P11) $\langle s, t \rangle \in \mathcal{E}(R)$ iff $\langle t, s \rangle \in \mathcal{E}(\mathsf{Inv}(R))$;
(P12) if $\{o\} \in \mathcal{L}(s) \cap \mathcal{L}(s')$ for some $o \in \mathbf{I}$, then $s = s'$;

[11] Note that we only use tableaux to define the canonical model .

(P13) if $o \in \mathbf{I}_K$, then $\{o\} \in \mathcal{L}(s)$ for some $s \in \mathbf{S}$.

We can easily obtain a canonical model of a KB K from every tableau for it.

Definition 3.20 (Canonical model) Let $T = \langle \mathbf{S}, \mathcal{L}, \mathcal{E} \rangle$ be a tableau for K. The *canonical model of* T, $\mathcal{I}_T = (\Delta^{\mathcal{I}_T}, \cdot^{\mathcal{I}_T})$, is defined as follows:

- $\Delta^{\mathcal{I}_T} = \mathbf{S}$,
- $A^{\mathcal{I}_T} = \{s \mid A \in \mathcal{L}(s)\}$ for all concept names A in $\mathsf{clos}(K_{\mathcal{A}})$,
- $a^{\mathcal{I}_T} = s \in \mathbf{S}, \{a\} \in \mathcal{L}(s)$, for all individual names a in $\mathbf{I}_K$, and
- $P^{\mathcal{I}_T} = \mathcal{E}(P)^{\oplus}$ for all role names P in $\mathbf{R}_{K_{\mathcal{A}}}$, where $\mathcal{E}(\cdot)^{\oplus}$ is the minimal extension of $\mathcal{E}(\cdot)$ such that $\mathcal{E}(R)^{\oplus}$ is transitively closed whenever $\mathsf{Trans}(R)$, and $\mathcal{E}(R')^{\oplus} \subseteq \mathcal{E}(R)^{\oplus}$ whenever $R' \sqsubseteq^* R$.

Please note that for each simple role S, $S^{\mathcal{I}_T} = \mathcal{E}(S)^{\oplus} = \bigcup_{S' \sqsubseteq^* S} \mathcal{E}(S')$. The next lemma follows from Lemma 4 in [24].

Lemma 3.21 *Let T be a tableau for K. Then $\mathcal{I}_T \models K$.*

Each completion graph $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ with $n \geq 1$ induces a tableau $T_{\mathcal{G}}$ that is the unravelling of $\mathcal{G}$, and which has as domain the set of paths in $\mathcal{G}$. The paths and the tableau are constructed as in [24]; each path comprises a sequence of pairs of nodes $\frac{v}{v'}$, in order to store which blocked nodes caused the loops in the path construction.

Definition 3.22 (Induced tableau) Let $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq 1$. In a sequence of pairs of nodes of the form $p = [\frac{v_0}{v'_0}, \ldots, \frac{v_m}{v'_m}]$, we define $\mathsf{tail}(p) = v_m$ and $\mathsf{tail}'(p) = v'_m$. By $[p \mid \frac{v_{m+1}}{v'_{m+1}}]$ we denote $[\frac{v_0}{v'_0}, \ldots, \frac{v_m}{v'_m}, \frac{v_{m+1}}{v'_{m+1}}]$. For a sequence of pairs of nodes p and a variable $v \in \mathsf{vn}(\mathcal{G})$, if v is not n-blocked and v is an R-successor of $\mathsf{tail}(p)$, then $[p \mid \frac{v}{v}]$ is an *R-step* of p; if v is directly n-blocked by w and v is an R-successor of $\mathsf{tail}(p)$, then $[p \mid \frac{w}{v}]$ is an *R-step* of p. The set of *paths in* $\mathcal{G}$, denoted $\mathsf{paths}(\mathcal{G})$, is inductively defined as follows:

- if $a \in \mathsf{in}(\mathcal{G})$, then $[\frac{a}{a}] \in \mathsf{paths}(\mathcal{G})$.
- if $p \in \mathsf{paths}(\mathcal{G})$, q is an R-step of p, $R \in \mathbf{R}_K$, then $q \in \mathsf{paths}(\mathcal{G})$.

The tableau $T_{\mathcal{G}} = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ induced by $\mathcal{G}$ is defined as follows:

$$
\begin{aligned}
\mathbf{S} &= \mathsf{paths}(\mathcal{G}),\\
\mathcal{L}(p) &= \mathcal{L}(\mathsf{tail}(p)),\\
\mathcal{E}(R) &= \{\langle p, q\rangle \in \mathbf{S}^2 \mid q \text{ is an } R\text{-step of } p, \text{ or } p \text{ is an } \mathsf{Inv}(R)\text{-step of } q,\\
&\qquad\qquad \text{or } q = [\tfrac{a}{a}] \text{ and } a \text{ is an } R\text{-successor of } \mathsf{tail}(p),\\
&\qquad\qquad \text{or } p = [\tfrac{a}{a}] \text{ and } a \text{ is an } \mathsf{Inv}(R)\text{-successor of } \mathsf{tail}(q)\}.
\end{aligned}
$$

Note that the definition of R-step requires w to be a variable node. Every path in $\mathsf{paths}(\mathcal{G})$ starts with a node $\frac{a}{a}$ for some individual a, and a node of this form only occurs at the first position in a path. The last two cases in the definition of $\mathcal{E}(R)$ are necessary in order to consider the arcs leading to individual nodes, which are not unravelled.

We use $\mathcal{I}_{\mathcal{G}}$ (instead of $\mathcal{I}_{T_{\mathcal{G}}}$) to denote the canonical model of the tableau $T_{\mathcal{G}}$ induced by $\mathcal{G}$.

Example 3.23 By unravelling $\mathcal{F}_1$, we obtain a model $\mathcal{I}_{\mathcal{F}_1}$ whose domain is the infinite set of paths from a to each v_i. When a node is not blocked, like v_1, the pair $\frac{v_1}{v_1}$ is added to the path. Every time a path reaches v_7, which is 1-blocked, we add $\frac{v_3}{v_7}$ to the path and 'loop' back to the successors of v_3. We thus obtain the following infinite set of paths:

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

The extension of each concept C is determined by the set of all p_i such that C occurs in the label of the last node in p_i. The extension of each role R is given by the pairs $\langle p_i, p_j \rangle$ such that p_j is an R-step of p_i. Therefore p_0, p_1, p_3, ... are in $A^{\mathcal{I}_{\mathcal{F}_1}}$; $\langle p_0, p_1 \rangle$, $\langle p_1, p_3 \rangle$, $\langle p_3, p_5 \rangle$, $\langle p_5, p_7 \rangle$, ... are in $P_1^{\mathcal{I}_{\mathcal{F}_1}}$ and $\langle p_0, p_2 \rangle$, $\langle p_1, p_4 \rangle$, $\langle p_3, p_6 \rangle$, $\langle p_5, p_8 \rangle$, ... are in $P_2^{\mathcal{I}_{\mathcal{F}_1}}$.
Analogously, by unravelling $\mathcal{G}_2$, we obtain the model $\mathcal{I}_{\mathcal{G}_2}$ whose domain is the infinite set of paths from a to each v_i, since there are no paths from o to any other node, i.e., the domain is:

$$
\begin{array}{ll}
p_0 = [\frac{o}{o}], & p_5 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}], \\
p_1 = [\frac{a}{a}], & p_6 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}, \frac{v_5}{v_5}] \\
p_2 = [\frac{a}{a}, \frac{v_1}{v_1}], & p_7 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}, \frac{v_5}{v_5}, \frac{v_3}{v_6}], \qquad \ldots \\
p_3 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}], & p_8 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}, \frac{v_5}{v_5}, \frac{v_3}{v_6}, \frac{v_4}{v_4}] \\
p_4 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}], & p_9 = [\frac{a}{a}, \frac{v_1}{v_1}, \frac{v_2}{v_2}, \frac{v_3}{v_3}, \frac{v_4}{v_4}, \frac{v_5}{v_5}, \frac{v_3}{v_6}, \frac{v_4}{v_4}, \frac{v_5}{v_5}]
\end{array}
$$

The extension of the concepts are $\{o\}^{\mathcal{I}_{\mathcal{G}_2}} = \{p_0\}$, $\{a\}^{\mathcal{I}_{\mathcal{G}_2}} = \{p_1\}$ and $A^{\mathcal{I}_{\mathcal{G}_2}} = \{p_i \mid i \geq 1\}$, and the extensions of the roles are $P_1^{\mathcal{I}_{\mathcal{G}_2}} = \{\langle p_i, p_{i+1} \rangle \mid i \geq 1\}$ and $P_2^{\mathcal{I}_{\mathcal{G}_2}} = \{\langle p_i, p_0 \rangle \mid i \geq 1\}$.

Lemma 3.24 *Let $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ with $n \geq 1$. Then $\mathcal{I}_{\mathcal{G}} \models K$.*

Proof First, it is proved as in [24] that every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ for $n \geq 1$ induces a tableau $T_{\mathcal{G}}$ for K. Note that since $n \geq 1$, pairwise blocking is subsumed. Since $T_{\mathcal{G}}$ is a tableau for K, it has a canonical model $\mathcal{I}_{\mathcal{G}}$, which by Lemma 3.21 is a model of K. □

Now we prove that, for a sufficiently large n, if Q is satisfied in the canonical model $\mathcal{I}_{\mathcal{G}}$ induced by an n-complete and clash-free graph $\mathcal{G}$, then we can map Q into $\mathcal{G}$. If $\mathcal{I}_{\mathcal{G}} \models Q$, then there is a match π for $\mathcal{I}_{\mathcal{G}}$ and Q. We show how to obtain a mapping μ witnessing $Q \hookrightarrow \mathcal{G}$ from π.

In this proof, the blocking parameter n is crucial. As we mentioned, it depends on Q. More specifically, it depends on the match π and what we call the *maximal π-distance*. Roughly, we consider the image of the query Q under π, restricted to the atoms that evaluate to true. If d is the length of the longest path between two (variable) nodes in this graph and the completion graph is (at least) d-complete, then it is large enough to construct a mapping $Q \overset{\mu}{\hookrightarrow} \mathcal{G}$ from π, which contains an isomorpic copy of the query image.

Definition 3.25 (Match graph, maximal π-distance) Let $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, where $n \geq 0$, such that $\mathcal{I}_\mathcal{G} \models Q$, and let π be a match for Q and $\mathcal{I}_\mathcal{G}$. Let Sat_π denote the set of atoms α in Q such that $\mathcal{I}_\mathcal{G}, \pi \models \alpha$. Then, the *match graph* G_π is the following (undirected) graph:
(i) its nodes are all $\pi(x)$ such that $x \in \mathsf{Vl}(Q)$ occurs in some $\alpha \in \mathsf{Sat}_\pi$; furthermore, if $\pi(x) = [\frac{a}{a}]$ for some $a \in \mathsf{in}(\mathcal{G})$, then $\pi(x)$ belongs to the set $\mathsf{ip}(G_\pi)$, otherwise to the set $\mathsf{vp}(G_\pi)$.
(ii) There is an edge between $\pi(x)$ and $\pi(y)$ iff $R(x, y)$ in Sat_π for some role R.

For every $x, y \in \mathsf{Vl}(Q)$, $d_\pi(x, y)$ is the length of the shortest path between $\pi(x)$ and $\pi(y)$ in G_π with nodes from $\mathsf{vp}(G_\pi)$ only, and -1 if no such path exists. Finally, the *maximal π-distance*, denoted $d^{\max}_\pi$, is the maximal $d_\pi(x, y)$ for all x, y in $\mathsf{Vl}(Q)$.

Note that the subgraph of G_π induced by $\mathsf{vp}(G_\pi)$ is acyclic (in fact, it is forest shaped), and thus shortest paths in it are unique.

Example 3.26 Consider a match π_1 for Q_1 and $\mathcal{I}_{\mathcal{F}_1}$ given as follows: $\pi_1(x) = p_7$, $\pi_1(y) = p_9$, and $\pi_1(z) = p_{10}$. Sat_{π_1} contains all atoms in Q_1 and the match graph G_{π_1} has the nodes p_7, p_9 and p_{10}, where $\mathsf{ip}(G_{\pi_1}) = \emptyset$ and $\mathsf{vp}(G_{\pi_1}) = \{p_7, p_9, p_{10}\}$, and the arcs $\langle p_7, p_9\rangle$ and $\langle p_7, p_{10}\rangle$. Moreover, $d_{\pi_1}(x, y) = 1$, $d_{\pi_1}(x, z) = 1$ and $d_{\pi_1}(y, z) = 2$, so $d^{\max}_{\pi_1} = 2$. Consider also the match π_2 for Q_3 and $\mathcal{I}_{\mathcal{G}_2}$, where $\pi_2(x) = p_7$, $\pi_2(y) = p_8$ and $\pi_2(o) = p_0$. $\mathsf{Sat}_{\pi_2} = \{P_1(x, y), P_2(y, o)\}$ and the match graph G_{π_2} has nodes p_0, p_7 and p_8, where $\mathsf{ip}(G_{\pi_2}) = \{p_0\}$ and $\mathsf{vp}(G_{\pi_2}) = \{p_7, p_8\}$, and arcs $\langle p_7, p_8\rangle$ and $\langle p_8, p_0\rangle$. Here, $d_{\pi_2}(x, y) = d^{\max}_{\pi_2} = 1$.

In the following, let $nr(Q)$ denote the number of role atoms in Q. Then, $d^{\max}_\pi$ is bounded by $nr(Q)$.[12] Since only simple roles occur in Q, arcs in G_π correspond to arcs in $\mathcal{G}$; thus in expanding the initial completion graph $\mathcal{G}_K$, it is sufficient to use n-blocking as a termination condition, for some arbitrarily chosen $n \geq nr(Q)$. Formally, we show:

Proposition 3.27 *Let $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$ with $n \geq nr(Q)$, and let $\mathcal{I}_\mathcal{G}$ be the canonical model of $\mathcal{G}$. If $\mathcal{I}_\mathcal{G} \models Q$ then $Q \hookrightarrow \mathcal{G}$.*

Proof As $\mathcal{I}_\mathcal{G} \models Q$, there is a match π for $\mathcal{I}_\mathcal{G}$ and Q. To define a mapping $\mu : \mathsf{Vl}(Q) \to \mathsf{nodes}(\mathcal{G})$, we consider the match graph G_π. Recall that, by construction, each node in G_π is from $\mathsf{paths}(\mathcal{G})$. Let G'_π be the subgraph of G_π induced by $\mathsf{vp}(G_\pi)$, and let $G_1, \ldots, G_n$ be the connected components of G'_π. Note that, since only simple role occur in Q, $d^{\max}_\pi \leq nr(Q) \leq n$ and each G_i has at most $nr(Q) \leq n$ edges.

Informally, the proof works as follows: $\mathcal{G}$ is n-complete, and by unravelling it we obtain the tableau $T_\mathcal{G}$ that induces $\mathcal{I}_\mathcal{G}$. Suppose there is a node v' in $\mathcal{G}$ directly n-blocked by some node v, and such that v' is not indirectly n-blocked; let S be the subgraph of $\mathcal{G}$ that includes every variable descendant of v that is not n-blocked. Then we can see $T_\mathcal{G}$ as having a branch composed of infinitely many adjacent non-overlapping copies of the path between v and v', where there are infinitely many copies $v_1, v_2, \ldots$ of v, and each v_i is the root of a copy S_i of the subtree S. The match π maps each $x \in \mathsf{Vl}(Q)$ to some element $\pi(x)$ of $T_\mathcal{G}$, which we now map to a node $\mu(x)$ in $\mathcal{G}$. There are two cases.

[12] For simplicity, we are using the number of role atoms in the query as a bound. A tighter bound would be the number of role atoms in the largest disjunct when the query is transformed into disjunctive normal form.

(1) If $\pi(x) \in \mathsf{ip}(G_\pi)$, we just set $\mu(x) = a$ where $\pi(x) = [\frac{a}{a}]$.

(2) If $\pi(x) \in \mathsf{vp}(G_\pi)$, consider the (unique) G_i containing $\pi(x)$. The bounded size of G_i ensures that it contains nodes from at most two copies of the subtree S in $T_\mathcal{G}$, and that if it contains nodes from two copies S_k and $S_{k'}$, $k \leq k'$, then $k' = k+1$. Consider two subcases. (2.1) G_i contains nodes from at most one copy of S, i.e., G_i is before the leaves of the first copy S_1 or fully within some S_k. Then we can map x in $\mathcal{G}$ to a node in S or above. (2.2) G_i includes nodes of two subtrees S_k and S_{k+1}, i.e., π maps some variables to nodes in S_k, which correspond to paths in $\mathcal{G}$ ending before or at v', and others to nodes in S_{k+1}, which correspond to paths ending at descendants of v (after passing through v'). We then ensure that μ maps the former to v or to nodes above v, and the latter to nodes in S.

Technically, let $\mathsf{blockedLeaves}(G_i)$ be the set of all nodes p of G_i such that $\mathsf{tail}(p) \neq \mathsf{tail}'(p)$, and let $\mathsf{afterblocked}(G_i)$ be the set of all nodes of G_i of the form $[\frac{v_0}{v'_0}, \ldots, \frac{v_m}{v'_m}, \ldots, \frac{v_{m+j}}{v'_{m+j}}]$ for some $[\frac{v_0}{v'_0}, \ldots, \frac{v_m}{v'_m}] \in \mathsf{blockedLeaves}(G_i)$ and $j > 0$. Intuitively, $\mathsf{blockedLeaves}(G_i)$ contains the paths $\pi(x)$ that end at some directly n-blocked node, i.e., at the end of a subtree S_k, and $\mathsf{afterblocked}(G_i)$ the paths $\pi(x)$ that go beyond these nodes, i.e., into the next subtree S_{k+1}.

If $\mathsf{afterblocked}(G_i) = \emptyset$, then the nodes of G_i are in at most one copy of S, and we are in case 2.1. For each variable x with $\pi(x)$ in G_i, we define $\mu(x) = \mathsf{tail}'(\pi(x))$, which is a node in or above S. Otherwise, we are in case 2.2 and consider two subcases: (2.2.1) if $\pi(x) \in \mathsf{afterblocked}(G_i)$, then we also define $\mu(x) = \mathsf{tail}'(\pi(x))$, which is a node in S; (2.2.2) if $\pi(x) \notin \mathsf{afterblocked}(G_i)$, then we define $\mu(x) = \psi(\mathsf{tail}'(\pi(x)))$, where ψ denotes the bijection via which $\mathsf{tail}'(\pi(x))$ is graph-blocked (thus $\mu(x)$ is a node above S). This is possible because the bounded size of G_i ensures that $\mathsf{tail}'(\pi(x))$ is a node in a blocked n-graph. Summing up, we define

$$\mu(x) = \begin{cases} \psi(\mathsf{tail}'(\pi(x))), & \text{if } \pi(x) \text{ is in some } G_i \text{ with } \mathsf{afterblocked}(G_i) \neq \emptyset \\ & \text{and } \pi(x) \notin \mathsf{afterblocked}(G_i), \\ \mathsf{tail}'(\pi(x)), & \text{otherwise.} \end{cases}$$

Now we prove the following:

a) For each individual a in $\mathsf{VI}(Q)$, $\pi(a) = a^{\mathcal{I}_\mathcal{G}}$ implies $\{a\} \in \mathcal{L}(\mu(a))$.
b) For each $C(x)$ in Sat_π, $C \in \mathcal{L}(\mu(x))$.
c) For each $R(x, y)$ in Sat_π, $\mu(y)$ is an R-neighbour of $\mu(x)$.

Items a) – c) ensure that $\mathcal{I}_\mathcal{G}, \pi \models \alpha$ implies $\alpha \overset{\mu}{\hookrightarrow} \mathcal{G}$ for each atom α in Q. Since π is a match for Q and $\mathcal{I}_\mathcal{G}$, this is sufficient to prove $Q \hookrightarrow \mathcal{G}$.

The proof of items a) and b) is straightforward by the construction of $\mathcal{I}_\mathcal{G}$ and μ. Observe that for each individual a in $\mathsf{VI}(Q)$, $\pi(a) = a^{\mathcal{I}_\mathcal{G}}$, which implies $\{a\} \in \mathcal{L}(\pi(a))$. Since $\mathcal{L}(\pi(a)) = \mathcal{L}(\mu(a))$, we get $\{a\} \in \mathcal{L}(\mu(a))$. For every x in $\mathsf{VI}(Q)$, $\mathcal{I}_\mathcal{G} \models C(\pi(x))$ implies that $C \in \mathcal{L}(\pi(x))$. Again, as $\mathcal{L}(\pi(x)) = \mathcal{L}(\mu(x))$, we have $C \in \mathcal{L}((\mu(x)))$.

For c), by construction of $\mathcal{I}_\mathcal{G}$, we have that $\mathcal{I}_\mathcal{G} \models R(\pi(x), \pi(y))$ implies $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(R')^\oplus$. Since R is a simple role and $\mathcal{E}(R)^\oplus = \bigcup_{R' \sqsubseteq^* R} \mathcal{E}(R')$, $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(R')$ for some $R' \sqsubseteq^* R$ follows. We then prove:

Claim 3 If $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(R')$, then $\mu(y)$ is an R'-neighbour of $\mu(x)$.

As discussed above, μ is defined such that each variable preserves all its neighbours under the match π. A formal proof of Claim 3 is given in the Appendix. □

In the proof of Proposition 3.27, it was crucial that a match for the query on a canonical model only needs fragments of bounded size from the tableau. If this does not hold, as in the case of queries where non-simple roles occur, it is not clear whether this kind of technique can be used for deciding query entailment.

Example 3.28 For the match π_1 in Example 3.26, the single graph G_i for the match graph G_{π_1} as in the proof of Proposition 3.27 is G_{π_1}; recall that $\mathsf{ip}(G_{\pi_1}) = \emptyset$, and G_{π_1} is connected. We have $\mathsf{blockedLeaves}(G_{\pi_1}) = \{p_7\}$ and $\mathsf{afterblocked}(G_{\pi_1}) = \{p_9, p_{10}\}$. We obtain the mapping μ_1 from π_1 by: $\mu_1(x) = \psi(\mathsf{tail}'(p_7)) = v_3$; $\mu_1(y) = \mathsf{tail}'(p_9) = v_5$; $\mu_1(z) = \mathsf{tail}'(p_{10}) = v_6$. It satisfies the conditions of Definition 3.16, so $Q_1 \xhookrightarrow{\mu_1} \mathcal{F}_1$.

Now reconsider π_2 and G_{π_2} in Example 3.26. Removing the nodes $\mathsf{ip}(G_{\pi_2}) = \{p_0\}$ from G_{π_2}, the resulting graph G'_{π_2} is connected and hence the single graph G_i for G_{π_2} as in the proof of Proposition 3.27. We have $\mathsf{afterblocked}(G_1) = \{p_8\}$. We obtain from π_2 the mapping μ by: $\mu_2(x) = \psi(\mathsf{tail}'(p_7)) = v_3$, $\mu_2(y) = \mathsf{tail}'(p_8) = v_4$ and $\mu_2(o) = \mathsf{tail}(p_0) = o$. It also satisfies Definition 3.16, so $Q_3 \xhookrightarrow{\mu_2} \mathcal{G}_2$.

Summing up, to decide whether $K \models Q$, it is sufficient to choose an arbitrary $n \geq nr(Q)$ and then to check the existence of a mapping $Q \hookrightarrow \mathcal{G}$ for each $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.

Theorem 3.29 *Let Q be a positive query, let K be a $\mathcal{SHOIQ}$ KB, and let $n \geq nr(Q)$. Then $K \models Q$ iff $Q \hookrightarrow \mathcal{G}$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$.*

Proof Let $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. By Lemma 3.24, $\mathcal{I}_\mathcal{G} \models K$, and since $K \models Q$, it follows $\mathcal{I}_\mathcal{G} \models Q$. Since $n \geq nr(Q)$, by Proposition 3.27, $Q \hookrightarrow \mathcal{G}$. Conversely, from $Q \hookrightarrow \mathcal{G}$ and Lemma 3.18, we have that $\mathcal{G} \models Q$ for every $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$. By Proposition 3.15, this means $K \models Q$. □

Example 3.30 $K \models Q_1$, so $\mathcal{F}_1 \models Q_1$ must hold. This is witnessed by the mapping μ_1 in Example 3.28. Note that there are longer queries, like $Q' = \{P_1(a, x_0), P_1(x_0, x_1), P_1(x_1, x_2), P_1(x_2, x_3), P_1(x_3, x_4)\}$ such that $K \models Q'$ holds, but the entailment $\mathcal{F}_1 \models Q'$ cannot be verified by mapping Q' into $\mathcal{F}_1$ since $\mathcal{F}_1$ is 1-complete and $nr(Q') > 1$.

4 Termination and Complexity

The method from above yields a sound algorithm for answering PQs on $\mathcal{SHOIQ}$ KBs. As we show in this section, it always terminates for $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ KBs. Based on this, we prove our main results on the data complexity of query answering in these logics.

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

Proposition 4.1 *Let $\mathcal{G}$ be a (fixed) completion graph in $\mathbb{G}_K$ and let Q be a given CQ. Deciding whether $Q \hookrightarrow \mathcal{G}$ is* NP*-hard.*

Proof Deciding the existence of a mapping $Q \hookrightarrow \mathcal{G}$ is at least as hard as evaluating a CQ over a database (given by the ABox), which is NP-hard (w.r.t. query complexity) [14]. To verify this, consider the completion graph $\mathcal{G}_{col}$ associated to the ABox $\{E(c, c) \mid c, c' \in \{red, green, blue\}, c \neq c'\}$. Every directed graph G can be represented as a CQ

Q, where each node in G is associated with a distinct variable and for each arc $\langle x, y\rangle$ in G there is the literal $E(x, y)$ in Q. Then Q can be mapped into $\mathcal{G}_{col}$ iff G is 3-colourable. □

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

4.1 Bounding the size of completion forests and graphs

In what follows, we assume that K is a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ or $\mathcal{SHOI}$ knowledge base, such that $\mathbf{c} := |\mathsf{clos}(K)| \geq 1$ and $\mathbf{r} := |\mathbf{R}_K| \geq 1$. Let $\mathbf{m}$ denote the maximum between 1 and any number n occurring in concepts of the form $\leq n\, R.C$ or $\geq n\, R.C$ in K.

We first derive a bound on the possible size of a blockable n-graph, and then a bound on the size of the completion graphs in $\mathsf{ccf}_n(\mathbb{G}_K)$.

Claim 4 Let $\mathcal{G} \in \mathbb{G}_K$ and let $n \geq 0$. Then $\mathcal{G}$ has at most $H_n = 2^{p(\mathbf{c},\mathbf{r},\mathbf{m})^{n+1}}$ many non-isomorphic blockable n-graphs, for some polynomial $p(\mathbf{c}, \mathbf{r}, \mathbf{m})$ in $\mathbf{c}$, $\mathbf{r}$, and $\mathbf{m}$.

Proof First, we give a bound on the number of non-isomorphic node and arc labels that may occur in a blockable n-graph in $\mathcal{G}$. Recall that $K_\mathcal{A}$ is the KB obtained from K by internalising the ABox as in Section 2.1.2, which is used for constructing the initial $\mathcal{G}_K$.

The only expansion rule that can add some $C \notin \mathsf{clos}(K_\mathcal{A})$ to the label of a node is the o?-rule, which is never applied for a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOI}$ KB. Therefore, the label of every node v in a completion forest in $\mathbb{G}_K$ fulfills $\mathcal{L}(v) \subseteq \mathsf{clos}(K_\mathcal{A})$. By definition, every node v in a blockable n-graph is a successor of a variable node, and either (1) v is a variable node; or (2) v is an individual node that has a variable predecessor w. In case (1), v was created by a generating rule and its label was initialised with $\mathcal{L}(v) \subseteq \mathsf{clos}(K)$. Moreover, any concept added to its label will be from $\mathsf{clos}(K)$, unless it is merged into an existing individual whose label already contains some $C \in \mathsf{clos}(K_\mathcal{A}) \setminus \mathsf{clos}(K)$; the latter would imply that v is not a variable node. So we can conclude that every v of $\mathsf{vn}(\mathcal{G})$ fulfils $\mathcal{L}(v) \subseteq \mathsf{clos}(K)$. In case (2), if an individual node v is a successor of a variable node w, then $\{a\} \in \mathcal{L}(v)$ for some $\{a\} \in \mathsf{clos}(K)$. This is because arcs from variable to individual nodes can only be created by merging two nodes that share a nominal. The expansion rules can only cause this for nominals in $\mathsf{clos}(K)$, as they only add concepts from $\mathsf{clos}(K)$ to the node labels (except the o?-rule, which is never applied).

Consider two blockable n-graphs G_1 and G_2. Remove from them all arcs connecting two individual nodes, and restrict the labels of the individual nodes to $\mathsf{clos}(K)$. Suppose that the resulting graphs G_1' and G_2' are isomorphic. The label of each individual node in G_1' contains some nominal $\{a\}$ from $\mathsf{clos}(K)$, which must also be in the label of the isomorphic node in G_2'. As this $\{a\}$ can be in the label of only one node in $\mathcal{G}$ (by the assumption on the application of the o-rule), both nodes are the same node from $\mathcal{G}$. This ensures that G_1' and G_2' are isomorphic iff G_1 and G_2 are isomorphic. In general, G_1 and G_2 can only be isomorphic if they contain exactly the same set of individual nodes. Hence, when calculating the number of non-isomorphic blockable n-graphs, we can omit all arcs between individual nodes, and restrict their labels to the concepts in $\mathsf{clos}(K)$ (note that they will still be individual nodes after this restriction). Thus, we

consider only node labels that are subsets of $\mathsf{clos}(K)$, and there are $2^{\mathbf{c}}$ possible such labels. Similarly, each arc is labelled with a subset of $\mathbf{R}_{K_\mathcal{A}}$, but roles in $\mathbf{R}_{K_\mathcal{A}} \setminus \mathbf{R}_K$ occur only in arcs connecting two individual nodes, so we restrict our attention to $2^{\mathbf{r}}$ different arc labels that are subsets of $\mathbf{R}_K$.

Now we derive a bound on the out-degree of the variable nodes in $\mathcal{G}$. Every successor of such a node is generated by the application of a generating rule. Only two are feasible for K: the $\exists$-rule and the $\geq$-rule. Only concepts of the form $\exists R.S$ or $\geq n\, R.C$ trigger the application of these rules, and there are at most $\mathbf{c}$ such concepts. Each time one such rule is applied, it generates at most $\mathbf{m}$ R-successors for each role R. Note that if a node v is identified with another one by a shrinking rule, then the rule application which led to the generation of v will never be repeated [24], so a generating rule can be applied to each node at most $\mathbf{c}$ times. This gives a bound of $\mathbf{c}{\cdot}\mathbf{m}$ R-successors for each role R, and a total of $b = \mathbf{r}{\cdot}\mathbf{c}{\cdot}\mathbf{m} \geq 1$ for each variable node of $\mathcal{G}$.

Let h_n denote the number of non-isomorphic blockable n-graphs that may occur in $\mathcal{G}$. There are $2^{\mathbf{c}}$ different roots, each of which can have up to b successors. Each successor can be reached by any of the $2^{\mathbf{r}}$ possible arcs and can be the root of any of the h_{n-1} many different blockable $(n-1)$-graphs. Hence, there are at most $(2^{\mathbf{r}}{\cdot}h_{n-1})^b$ (ordered) combinations for each root. Thus we have

$$h_n = 2^{\mathbf{c}}{\cdot}(2^{\mathbf{r}}{\cdot}h_{n-1})^b = 2^{\mathbf{c}+\mathbf{r}{\cdot}b}{\cdot}(h_{n-1})^b$$

To simplify the notation, let $x = \mathbf{c} + \mathbf{r}{\cdot}b$. Then

$$h_n = 2^x{\cdot}(h_{n-1})^b = 2^{x+x{\cdot}b+\ldots+x{\cdot}b^{n-1}}{\cdot}(h_0)^{b^n} = 2^{x{\cdot}\sum_{i=0}^{n-1} b^i}{\cdot}(h_0)^{b^n}.$$

Since $t_0 = 2^{\mathbf{c}}$, we obtain for $b \geq 2$ that

$$h_n \leq (2^x{\cdot}h_0)^{b^n} = (2^{\mathbf{c}+\mathbf{r}{\cdot}b}{\cdot}2^{\mathbf{c}})^{b^n} \leq 2^{(2{\cdot}\mathbf{c}{\cdot}b+\mathbf{r}{\cdot}b^2)^{n+1}} = 2^{p(\mathbf{c},\mathbf{r},\mathbf{m})^{n+1}} \tag{4.1}$$

where $p(\mathbf{c},\mathbf{r},\mathbf{m}) = 2{\cdot}\mathbf{c}{\cdot}b + \mathbf{r}{\cdot}b^2 = 2{\cdot}\mathbf{c}^2{\cdot}\mathbf{r}{\cdot}\mathbf{m} + \mathbf{c}^2{\cdot}\mathbf{r}^3{\cdot}\mathbf{m}^2$. As (4.1) also holds for $b = 1$, we obtain the claimed bound $H_n = 2^{p(\mathbf{c},\mathbf{r},\mathbf{m})^{n+1}}$. □

In the rest of this section, we use $p(\mathbf{c},\mathbf{r},\mathbf{m})$ to denote the polynomial given above.

Claim 5 Let T be a tree of variable nodes rooted at some individual node in $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq 0$. Then the number of nodes in T is bounded by $(\mathbf{c}{\cdot}\mathbf{m}{\cdot}\mathbf{r})^{1+n{\cdot}2^{p(\mathbf{c},\mathbf{r},\mathbf{m})^{n+1}}}$.

Proof The claim is a consequence of the following properties:

i) The out-degree of T is bounded by $\mathbf{c}{\cdot}\mathbf{m}{\cdot}\mathbf{r}$. As shown above, each role R has at most $\mathbf{c}{\cdot}\mathbf{m}$ variable R-successors, and there are $\mathbf{r}$ roles.
ii) The depth of T is bounded by $d = (H_n + 1){\cdot}n$. This is because there are at most H_n non-isomorphic blockable n-graphs. If there was a path of length greater than $(H_n + 1){\cdot}n$ to a node v in T, then v would occur after a sequence of $H_n + 1$ non overlapping blockable n-graphs, and one of them would have been blocked so v would not have been generated.
iii) The number of variables in T is bounded by $(\mathbf{c}{\cdot}\mathbf{m}{\cdot}\mathbf{r})^{d+1}$. □

There can be one such tree rooted at each individual node, and since there is at most one individual node for each individual in $\mathbf{I}_K$, we easily get a bound on the size of a completion graph.

Lemma 4.2 *Let K be a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOI}$ KB and let $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq 0$. Then the number of nodes in $\mathcal{G}$ is bounded by*

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

Unfortunately, Lemma 4.2 does not apply to $\mathcal{SHOIQ}$ KBs. Indeed, our bound on the depth of completion graphs, relies on a fixed number of individual nodes. For $\mathcal{SHOIQ}$ KBs, the application of the o?-rule may introduce new individual nodes that lead to new n-blockable graphs non-isomorphic to previously present graphs. This potentially leads to non-termination. Note that in [24], the maximal depth of a variable node in the completion graphs does not depend on the number of individual nodes that can be generated. In turn, it is used to bound the number of nominals introduced by applying the o?-rule. The technique in [24] seems not to be applicable in our case, and it is not clear how termination could be achieved in general.

4.2 Complexity of the Query Entailment Algorithm

We now determine the complexity of deciding $K \models Q$ for a PQ Q. As for data complexity, the TBox, the RBox, and the query are considered fixed, while the ABox $\mathcal{A}$ is given as an input. The complexity bounds are given w.r.t. the size of this $\mathcal{A}$. In the following, we denote by $||K, Q||$ the total size of the string representing K and Q. Note that $\mathbf{m}$ is linear in $||K, Q||$ for unary number coding in number restrictions, and single exponential for binary number coding. In any case, if Q and all of K except $\mathcal{A}$ are fixed, $\mathbf{m}$ is a constant. Furthermore, $\mathbf{c}$ and $\mathbf{r}$ are linear in $||K, Q||$, but also constant in $|\mathcal{A}|$. Finally, $|\mathbf{I}_K|$ is linear in both. From this, and by Lemma 4.2, we know that the maximum number of nodes in a completion graph $\mathcal{G} \in \mathbb{G}_K$ is triple exponential in $||K, Q||$ if n is polynomial in $||K, Q||$. If n is a constant, then the size of $\mathcal{G}$ is linear in $|\mathcal{A}|$. We easily obtain:

Corollary 4.3 *Let $\mathcal{G} \in \mathsf{ccf}_n(\mathbb{G}_K)$, $n \geq 0$. Then the number of nodes in $\mathcal{G}$ is (i) at most triple exponential in $||K, Q||$, if n is polynomial in $||K, Q||$, and (ii) polynomial in $|\mathcal{A}|$, if n is a constant and Q and all of K except $\mathcal{A}$ is fixed.*

Moreover, we also obtain a bound on the number of rule applications to derive any clash-free n-complete completion graph.

Proposition 4.4 *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 $\mathcal{A}$ is fixed, then it terminates in time polynomial in $|\mathcal{A}|$.*

Proof The claim follows from the bound on the size of $\mathcal{G}$ given in Corollary 4.3, together with the following observations:

- Since the worst-case analysis of the size of $\mathcal{G}$ assumes that all possible successors are generated for every node, the shrinking of the completion graph by merging nodes can only lead to a smaller completion graph, and there is no additional effort in the regeneration of successors w.r.t. the worst-case estimate.

– Shrinking rules do not cause repeated rule applications, by merging some node into another node that would later have to be regenerated. Indeed, a concept $C \in \mathcal{L}(v)$ can fire a generating rule r for node v at most once. Even if a shrinking rule is applied and a successor w of v is merged into a node w', then w' inherits the labels and inequalities of w, as well as all its neighbours that are not variable successors (which are removed by prune). This ensures that the conditions that triggered the application of r for v are not met again, and thus the rule application that led to the generation of w will not be repeated [24]. □

Checking whether $Q \hookrightarrow \mathcal{G}$ can be easily done in time single exponential in the size of Q. For $\mathcal{G} \in \mathsf{ccf}(\mathbb{G}_K)$ and a query Q with n variables, the naive search space has $|\mathsf{nodes}(\mathcal{G})|^n$ many candidate assignments, and each one can be polynomially checked. This is triple exponential in $||K, Q||$ if $|\mathsf{nodes}(\mathcal{G})|$ is. On the other hand, $Q \hookrightarrow \mathcal{G}$ can be tested in time polynomial in the size of $\mathcal{G}$ when Q is fixed. Therefore, we obtain the following result.

Theorem 4.5 *Given a $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOI}$ knowledge base K and a PQ Q in which all roles are simple, deciding whether $K \models Q$ is:*

1. *in* coN3ExpTime *w.r.t. combined complexity, for both unary and binary encoding of number restrictions in K.*
2. *in* coN2ExpTime *w.r.t. combined complexity for a fixed Q if number restrictions are encoded in unary.*
3. *in* coNP *w.r.t. data complexity.*

Proof If $K \not\models Q$, then there is a completion graph $\mathcal{G} \in \mathsf{ccf}_{nr(Q)}(\mathbb{G}_K)$ such that $Q \not\hookrightarrow \mathcal{G}$. By Proposition 4.4, this $\mathcal{G}$ can be obtained non-deterministically in time triple exponential in $\|K, Q\|$. Furthermore, $Q \hookrightarrow \mathcal{G}$ can be checked by naive methods in time triple exponential in $\|K, Q\|$ as well. Therefore, non-entailment of Q is in N3ExpTime, entailment in coN3ExpTime and item 1 holds.

Similarly, since $\mathbf{m}$ does not occur in the uppermost exponent of the bound in Lemma 4.2, each $\mathcal{G}$ in $\mathsf{ccf}_{nr(Q)}(\mathbb{G}_K)$ can be obtained in double exponential time when the conditions of item 2 hold.

As for item 3, under data complexity $nr(Q)$ is constant, since Q and all components of $K = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ except $\mathcal{A}$ are fixed. By Proposition 4.4, every $\mathcal{G} \in \mathsf{ccf}_{nr(Q)}(\mathbb{G}_K)$ can be nondeterministically generated in polynomial time. Since deciding whether $Q \hookrightarrow \mathcal{G}$ is polynomial in the size of $\mathcal{G}$, $K \models Q$ is in coNP. □

We note that $Q \hookrightarrow \mathcal{G}$ can also be tested in time polynomial in the size of $\mathcal{G}$ when Q is fixed, or when the expansion rules generate a completion graph whose size exponentially dominates the query size. Other particular cases can be solved in polynomial time as well. For example, when $\mathcal{G}$ is tree-shaped (i.e., the ABox is tree-shaped and there are no arcs from variable to individual nodes), then the complexity of the mapping corresponds to evaluating a conjunctive query over a tree-shaped database, which is polynomial in certain cases [20].

4.3 Data Complexity

The upper bound for data complexity given in Theorem 4.5 is worst-case optimal. In [16], coNP-hardness was proved for instance checking over $\mathcal{ALE}$ knowledge bases,

and in [11] this result has been extended to even less expressive DLs, like $\mathcal{AL}$. This allows us to state the following main result.

Theorem 4.6 *For KBs in any DL extending $\mathcal{AL}$ and contained in $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, or $\mathcal{SHOI}$, answering positive existential queries in which all roles are simple is* coNP*-complete w.r.t. data complexity.*

This result provides an exact characterisation of the data complexity of PQs for a wide range of description logics. An interesting observation is that once we allow for universal quantification, which is a basic constructor of DLs, then many other constructors can be added without affecting worst-case data complexity. Also, this result provides the first tight upper bound for data complexity of $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ and extends two previous coNP-completeness results w.r.t. data complexity: (*i*) for answering UCQs over $\mathcal{ALCNR}$ knowledge bases [33]. We extend this result to a query language allowing for arbitrary use of conjunction and disjunction, as well as to DLs including role hierarchies and some combinations of inverse roles and nominals. (*ii*) For answering atomic queries in $\mathcal{SHIQ}$ [30]. This can be immediately extended to tree-shaped CQs, as they admit a representation as a DL concept (e.g., by tuple-graphs of [7], or via rolling up [25]). However, an extension to all PQs without transitive roles remained open. We point out that [19] presented an algorithm for answering CQs with transitive roles in $\mathcal{SHIQ}$ KBs that also yields a coNP upper bound.

4.4 Combined Complexity

Theorem 4.5 does not provide optimal upper bounds with respect to the combined complexity of query answering. The main reason is that the tableaux algorithms in [26] and [24], which we extended, are also not worst-case optimal. They are both non-deterministically double exponential, while satisfiability of a knowledge base is ExpTime-complete for $\mathcal{SHIQ}$ [42] and NExpTime-complete for $\mathcal{SHOIQ}$ [41]. It is well known that tableaux algorithms for expressive DLs often do not yield optimal complexity bounds. However, they are easy to implement and amenable for optimisations [2]. Moreover, efficient reasoners implementing these algorithms are available [23, 21].

We want to point out that, in our algorithm, the witness of a blocked variable must be its ancestor. We use these rather strict conditions for blocking in order to make them similar to the conventional ones in DL tableaux, where it is usually required that the blocking and the blocked variable are on the same path, see e.g., [26] and [24]. This condition was relaxed in [36], resulting in an algorithm whose wort-case complexity is exponentially lower than in [26]. We conjecture that a similar blocking with any previous occurrence of an isomorphic n-tree could be used in our algorithm, without affecting its soundness and completeness. With this relaxed condition, we would obtain the same complexity upper bounds as those given in [33]. In fact, the absence of the 'blocking on the same path' requirement is the actual reason why the combined complexity bounds in [33] are exponentially lower than the ones we obtained. Our algorithms may be further optimised following the ideas in [15].

It was recently shown in [35] that answering CQs is 2ExpTime-hard for all DLs containing $\mathcal{ALCI}$, and thus also for $\mathcal{SHIQ}$ and $\mathcal{SHOI}$. As a consequence, the 2ExpTime upper bound given in [13] for answering PQs in $\mathcal{SHIQ}$ is tight, and similarly the ones given in [30, 19] for answering CQs in $\mathcal{SHIQ}$, and the ones given in [7] for containment

of CQs in $\mathcal{DLR}$. In the light of these results, and considering the coN2ExpTime upper bound discussed above and the intrinsic non-determinism of tableaux algorithms, it seems reasonable to conjecture that a 2ExpTime upper bound can be achieved for $\mathcal{SHOQ}$ and $\mathcal{SHOI}$. To our knowledge, the question remains open and this work provides the first upper bounds. We point out that decidability of CQs (with transitive roles) in $\mathcal{SHOQ}$ has been shown [18], but we are not aware of any emerging complexity results. As for $\mathcal{SHOI}$, no other decision procedures seem to be available, even for more restricted classes of queries. In any case, since CQ answering in $\mathcal{SHOI}$ is already 2ExpTime-hard, the gap to our coN2ExpTime upper bound is rather small. For $\mathcal{SHOQ}$ (in fact, for any logic containing $\mathcal{ALC}$) ExpSpace-hardness of PQ answering was shown in [13], thus the gap is still not large. A quite significant gap remains open for CQs, since only the ExpTime-hardness that follows from instance checking is known.

5 Conclusion

We have studied answering positive existential queries (PQs) over knowledge bases in the expressive DLs of the $\mathcal{SH}$ family, where we have focused on data complexity, i.e., measuring the complexity of query answering with respect to the size of the ABox while the query and the other parts of the knowledge base are fixed. This setting is gaining importance since DL knowledge bases are more and more used also for representing data repositories, especially in the context of the Semantic Web and in Enterprise Application Integration.

Generalising a technique presented in [33] for a DL which is far less expressive than $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$, and combining it with the techniques from [24], we have developed a novel tableaux-based algorithm for answering PQs without transitive roles. The algorithm manages the technical challenges caused by the simultaneous presence of inverse roles, number restrictions, and general knowledge bases, leading to DLs without the finite model property. We have presented blocking conditions that make it suitable for deciding query entailment. They are more involved than previous blocking conditions in [24] and use the query size as a parameter. Query answering itself is then accomplished by a technique that maps the query into completion graphs of bounded depth, which are constructed using tableaux-style rules. The technique provides a sound and complete algorithm for $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$, while for $\mathcal{SHOIQ}$ only soundness is established.

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 answering PQs for a wide range of DLs, including very expressive ones. Namely, for each DL of the $\mathcal{SH}$ family except $\mathcal{SHOIQ}$, answering PQs without transitive roles is coNP-complete with respect to data complexity. This narrows the gap between the known coNP lower bound and the ExpTime upper bound for even weaker DLs, towards a negative answer to the open issue whether the data complexity of expressive DLs will similarly increase as their combined complexity.

We point out that our method can also be exploited for deciding *containment* between PQs, i.e., given a knowledge base K and PQs Q_1 and Q_2, deciding whether $K \models Q_1$ implies that $K \models Q_2$. As a simple consequence, we also obtain decidability of the equivalence of positive queries Q_1 and Q_2 having only simple roles in $\mathcal{SHIQ}$,

$\mathcal{SHOQ}$, and $\mathcal{SHOI}$. This result can be exploited for query optimisation, and is to the best of our knowledge the first result in this direction for PQs in expressive DLs.

In this paper, roles in queries must be simple (this was also assumed e.g., in [29]), and a natural question is whether our results extend to queries with transitive roles. Unfortunately, as discussed in [17], these roles impose major difficulties in establishing a bound on the depth of completion graphs which need to be considered for answering a given query. The extension of these modified-tableaux techniques to general CQs is not apparent, and other techniques may be more adequate.

For example, the 'rolling-up' technique, which is related to the notion of tuple-graph of [7] and reduces the query answering problem to verifying the unsatifiability of a knowledge base, allowed the authors of [19] to obtain an algorithm for answering arbitrary CQs in $\mathcal{SHIQ}$. This technique was also exploited in [18] to provide an algorithm for arbitrary CQs in $\mathcal{SHOQ}$. Exploiting automata on infinite trees, an algorithm for answering positive 2-way regular path queries in the DL $\mathcal{ALCQIb}_{reg}$ was presented in [13]. This is, to our knowledge, the most general algorithm for query answering in DLs without nominals, and allows, e.g., to answer PQs in $\mathcal{SRIQ}$, a generalization of $\mathcal{SHIQ}$ closely related to the DL underlying OWL 2.

These techniques are both quite different from ours. The algorithms in [19] and [13] yield optimal 2ExpTime upper bounds w.r.t. combined complexity, while only the bound in [19] is known to be tight w.r.t. data complexity. Indeeed, we are not aware of any other tight data complexity bounds for query answering in $\mathcal{SHOQ}$ and $\mathcal{SHOI}$, neither of other algorithms for $\mathcal{SHOI}$. A terminating algorithm for query answering in $\mathcal{SHOIQ}$ remains to be found, either tableaux-based using suitable blocking conditions, or based on a different approach. It also remains to explore whether the proposed technique can be applied to yet more expressive DLs, e.g., allowing reflexive-transitive closure in the TBox (in the style of PDL), or to more expressive query languages. However, including inequality atoms in CQs is infeasible; as follows from results in [7], such queries are undecidable for every DL of the $\mathcal{SH}$ family.

Apart from the data complexity, also the combined complexity of query answering in expressive DLs remains for further investigation, since no tight bounds are known for $\mathcal{SHOQ}$ and $\mathcal{SHOI}$. Finally, an interesting issue is whether other techniques may be applied to derive results similar to ours. For instance, whether resolution-based techniques as in [28, 30] or techniques based on tree automata can be fruitfully applied. While the latter have already been successfully applied for answering PQs, allowing also for atoms that are regular expressions over roles, in very expressive DLs [13], it remains unclear how the contribution of the ABox may be singled out so as to establish data complexity.

Acknowledgements We thank Ian Horrocks and Birte Glimm for many fruitful and stimulating discussions, and are grateful to them for pointing out errors in preliminary work to this paper, and for helpful suggestions. We are also very grateful to the reviewers for their constructive comments, which greatly improved the presentation of this paper.

References

1. Baader F, Hanschke P (1991) A schema for integrating concrete domains into concept languages. In: Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91), pp 452–457

2. Baader F, Sattler U (2001) An overview of tableau algorithms for description logics. Studia Logica 69(1):5–40
3. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) (2003) The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press
4. Berardi D, Calvanese D, De Giacomo G (2005) Reasoning on UML class diagrams. Artificial Intelligence 168(1–2):70–118
5. Borgida A, Brachman RJ (2003) Conceptual modeling with description logics. In: [3], chap 10, pp 349–372
6. Calvanese D, De Giacomo G (2003) Expressive description logics. In: [3], chap 5, pp 178–218
7. Calvanese D, De Giacomo G, Lenzerini M (1998) On the decidability of query containment under constraints. In: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), pp 149–158
8. Calvanese D, Lenzerini M, Nardi D (1999) Unifying class-based representation formalisms. J of Artificial Intelligence Research 11:199–240
9. Calvanese D, De Giacomo G, Lenzerini M (2000) Answering queries using views over description logics knowledge bases. In: Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000), pp 386–391
10. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2005) *DL-Lite*: Tractable description logics for ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005), pp 602–607
11. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2006) Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp 260–270
12. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J of Automated Reasoning 39(3):385–429
13. Calvanese D, Eiter T, Ortiz M (2007) Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007), pp 391–396
14. Chandra AK, Merlin PM (1977) Optimal implementation of conjunctive queries in relational data bases. In: Proc. of the 9th ACM Symp. on Theory of Computing (STOC'77), pp 77–90
15. De Giacomo G, Massacci F (2000) Combining deduction and model checking into tableaux and algorithms for converse-PDL. Information and Computation 160(1–2):117–137
16. Donini FM, Lenzerini M, Nardi D, Schaerf A (1994) Deduction in concept languages: From subsumption to instance checking. J of Logic and Computation 4(4):423–452
17. Glimm B, Horrocks I, Sattler U (2006) Conjunctive query answering for description logics with transitive roles. In: Proc. of the 2006 Description Logic Workshop (DL 2006), CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-189/`, vol 189
18. Glimm B, Horrocks I, Sattler U (2007) Conjunctive query entailment for $\mathcal{SHOQ}$. In: Proc. of the 2007 Description Logic Workshop (DL 2007), CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-250/`, vol 250, pp 65–75
19. Glimm B, Horrocks I, Lutz C, Sattler U (2008) Conjunctive query answering for the description logic $\mathcal{SHIQ}$. J of Artificial Intelligence Research 31:151–198
20. Gottlob G, Koch C, Schulz KU (2004) Conjunctive queries over trees. In: Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004), pp 189–200
21. Haarslev V, Möller R (2001) RACER system description. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001), Springer, Lecture Notes in Artificial Intelligence, vol 2083, pp 701–705
22. Heflin J, Hendler J (2001) A portrait of the Semantic Web in action. IEEE Intelligent Systems 16(2):54–59
23. Horrocks I (1998) The FaCT system. In: de Swart H (ed) Proc. of the 7th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98), Springer, Lecture Notes in Artificial Intelligence, vol 1397, pp 307–312
24. Horrocks I, Sattler U (2007) A tableau decision procedure for $\mathcal{SHOIQ}$. J of Automated Reasoning 39(3):249–276

25. Horrocks I, Tessaris S (2000) A conjunctive query language for description logic ABoxes. In: Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000), pp 399–404
26. Horrocks I, Sattler U, Tobies S (2000) Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In: McAllester D (ed) Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000), Springer, Lecture Notes in Computer Science, vol 1831, pp 482–496
27. Horrocks I, Patel-Schneider PF, van Harmelen F (2003) From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. J of Web Semantics 1(1):7–26
28. Hustadt U, Motik B, Sattler U (2004) A decomposition rule for decision procedures by resolution-based calculi. In: Proc. of the 11th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004), pp 21–35
29. Hustadt U, Motik B, Sattler U (2004) Reducing $\mathcal{SHIQ}$-description logic to disjunctive datalog programs. In: Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004), pp 152–162
30. Hustadt U, Motik B, Sattler U (2005) Data complexity of reasoning in very expressive description logics. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp 466–471
31. Kazakov Y, Motik B (2008) A resolution-based decision procedure for $\mathcal{SHOIQ}$. J of Automated Reasoning 40(2–3):89–116
32. Lenzerini M (2002) Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002), pp 233–246
33. Levy AY, Rousset MC (1998) Combining Horn rules and description logics in CARIN. Artificial Intelligence 104(1–2):165–209
34. Lutz C (2003) Description logics with concrete domains: A survey. In: Balbiani P, Suzuki NY, Wolter F, Zakharyaschev M (eds) Advances in Modal Logics, vol 4, King's College Publications
35. Lutz C (2007) Inverse roles make conjunctive queries hard. In: Proc. of the 2007 Description Logic Workshop (DL 2007), CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-250/`, vol 250, pp 100–111
36. Motik B, Shearer R, Horrocks I (2007) Optimized reasoning in description logics using hypertableaux. In: Proc. of the 21st Int. Conf. on Automated Deduction (CADE 2007), Springer, Lecture Notes in Computer Science, vol 4603, pp 67–83
37. Ortiz MM, Calvanese D, Eiter T (2006) Characterizing data complexity for conjunctive query answering in expressive description logics. In: Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI 2006), pp 275–280
38. Patel-Schneider P, Hayes P, Horrocks I (2004) OWL Web Ontology Language semantics and abstract syntax. W3C Recommendation, available at `http://www.w3.org/TR/owl-semantics/`
39. Schaerf A (1993) On the complexity of the instance checking problem in concept languages with existential quantification. J of Intelligent Information Systems 2:265–278
40. Schaerf A (1994) Reasoning with individuals in concept languages. Data and Knowledge Engineering 13(2):141–176
41. Tobies S (2000) The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J of Artificial Intelligence Research 12:199–217
42. Tobies S (2001) Complexity results and practical algorithms for logics in knowledge representation. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany
43. Vardi MY (1982) The complexity of relational query languages. In: Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82), pp 137–146

Appendix

Claim 2 Let $\mathcal{G} \in \mathbb{G}_K$, let $\mathcal{J} \models_K \mathcal{G}$, and let r be any rule in Table 3.1 that is applicable to $\mathcal{G}$. Then, there exist a completion graph $\mathcal{G}'$ obtainable from $\mathcal{G}$ by applying r and an extended interpretation $\mathcal{J}'$ equal to $\mathcal{J}$ on $\mathsf{nodes}(\mathcal{G})$ such that $\mathcal{J}' \models_K \mathcal{G}'$.

The proof of this claim is similar to the proof of completeness of the tableau algorithm for $\mathcal{SHOIQ}$, given in detail in [24]. Although the technical details are quite

different, the underlying intuition is essentially the same. The main difference is that the authors of [24] use a tableau T to represent an arbitrary model of the knowledge base, and they "steer" the application of the expansion rules through this T. In contrast, we follow an approach closer to [33] and look at completion graphs as a representation of a set of models of the knowledge base, thus we do the steering directly with a model. In [24], it was proved that there is a mapping π from the nodes of $\mathcal{G}$ to the elements of T, satisfying certain conditions, which can be extended after each rule application. The conditions imposed on π are closely related to those for a model of a completion graph. Here we prove that the interpretation $\mathcal{J}$ can be extended and modelhood is preserved after each rule application, similarly as this was proved for π.

Proof To prove Claim 2, we consider the different cases for an expansion rule r from Table 3.1. First we consider the cases where r is a deterministic, non-generating rule. There is only one completion graph $\mathcal{G}'$ which can be obtained from $\mathcal{G}$ by applying r, and the models of $\mathcal{G}$ are exactly the models of $\mathcal{G}'$. If r is the $\sqcap$-rule, there is some node v in $\mathcal{G}$ s.t. $C_1 \sqcap C_2 \in \mathcal{L}(v)$. Since $\mathcal{J} \models_K \mathcal{G}$, we have $v^{\mathcal{J}} \in (C_1 \sqcap C_2)^{\mathcal{J}}$. By the definition of interpretation, both $v^{\mathcal{J}} \in C_1^{\mathcal{J}}$ and $v^{\mathcal{J}} \in C_2^{\mathcal{J}}$ hold. The inequality relation and all labels in $\mathcal{G}'$ are exactly as in $\mathcal{G}$, the only change is that $\{C_1, C_2\} \subseteq \mathcal{L}(v)$ in $\mathcal{G}'$, so $\mathcal{J} \models_K \mathcal{G}'$.

The cases of the $\forall$-rule and the $\forall_+$-rule, are similar to the $\sqcap$-rule. The labels of all nodes in $\mathcal{G}$ are preserved in $\mathcal{G}'$, except for the node w to which the rule was applied, and we have in $\mathcal{G}'$ either $C \subseteq \mathcal{L}(w)$ or $\forall R'.C \subseteq \mathcal{L}(w)$ respectively. In the former case, since $\mathcal{J} \models K$, $v^{\mathcal{J}} \in (\forall R.C)^{\mathcal{J}}$, and w is an R-neighbour of v, it follows that $w^{\mathcal{J}} \in C^{\mathcal{J}}$. In the latter case, $v^{\mathcal{J}} \in (\forall R.C)^{\mathcal{J}}$ and w and R'-neighbour of v for some $R' \sqsubseteq^* R$, $\mathsf{Trans}(R')$, imply that $w^{\mathcal{J}} \in (\forall R'.C)^{\mathcal{J}}$. Thus $\mathcal{J} \models_K \mathcal{G}'$ in both cases.

For the non-deterministic, non-generating rules, there are two different completion graphs $\mathcal{G}'$ that can be obtained after the rule application, and $\mathcal{J} \models_K \mathcal{G}'$ for at least one of them, so we can choose to apply the rule in a way that such a $\mathcal{G}'$ is obtained. In particular, if r is the $\sqcup$-rule, there is some node v in $\mathcal{G}$ with $C_1 \sqcup C_2 \in \mathcal{L}(v)$. For every $\mathcal{J}$ such that $\mathcal{J} \models_K \mathcal{G}$ we have $v^{\mathcal{J}} \in (C_1 \sqcup C_2)^{\mathcal{J}}$. By definition, either $v^{\mathcal{J}} \in C_1^{\mathcal{J}}$ or $v^{\mathcal{J}} \in C_2^{\mathcal{J}}$ holds. If the former holds, we can apply r in such a way that we obtain a $\mathcal{G}'$ with $\{C_1\} \subseteq \mathcal{L}(v)$. If the latter holds, we can obtain a $\mathcal{G}'$ with $\{C_2\} \subseteq \mathcal{L}(v)$. In both cases, $\mathcal{J} \models_K \mathcal{G}'$ and the claim holds.

The proof for the choose rule is easy. Since either $v^{\mathcal{J}} \in C^{\mathcal{J}}$ or $v^{\mathcal{J}} \in \neg C^{\mathcal{J}}$ holds for every v, C, and $\mathcal{J}$, we can choose to apply r in such a way that we obtain a $\mathcal{G}'$ with $\{C\} \subseteq \mathcal{L}(v)$ if the former, or a $\mathcal{G}'$ with $\{NNF(\neg C)\} \subseteq \mathcal{L}(v)$ if the latter, so that $\mathcal{J} \models_K \mathcal{G}'$.

Now we show that if a shrinking rule r is applicable to $\mathcal{G}$ and $\mathcal{J} \models_K \mathcal{G}$, then there are two nodes v, v' in $\mathcal{G}$ such that $v^{\mathcal{J}} = v'^{\mathcal{J}}$. Thus the new $\mathcal{G}'$ can be obtained by merging these nodes and $\mathcal{J} \models_K \mathcal{G}'$. When the $\leq$-rule is applicable to a node v in $\mathcal{G}$, there is some concept $\leq n\, S.C \in \mathcal{L}(v)$ such that v has S-neighbours $w_1, \ldots, w_n, w_{n+1}$ labelled with C. As $\mathcal{J} \models_K \mathcal{G}$, it follows $v^{\mathcal{J}} \in (\leq n\, S.C)^{\mathcal{J}}$, which implies that there are at most $o_1, \ldots, o_n$ elements in $\Delta^{\mathcal{J}}$ such that $\langle v^{\mathcal{J}}, o_i \rangle \in S^{\mathcal{J}}$ and $o_i \in C^{\mathcal{J}}$. Thus v has S-neighbours w_i and w_j, $i \neq j$, which are instances of C such that $w_i^{\mathcal{J}} = w_j^{\mathcal{J}}$. Hence $\mathcal{J} \models_K \mathcal{G}'$, where $\mathcal{G}'$ is obtained from $\mathcal{G}$ by merging w_i into w_j.

The o-rule is applicable if $\{o\} \in \mathcal{L}(v) \cap \mathcal{L}(v')$ for some nominal $\{o\}$ and two nodes v and v'. Since $\mathcal{J} \models_K \mathcal{G}$, we have $v^{\mathcal{J}} \in \{o\}^{\mathcal{J}}$ and $v'^{\mathcal{J}} \in \{o\}^{\mathcal{J}}$, but since $\{o\}^{\mathcal{J}} = \{o^{\mathcal{J}}\}$, we have $v^{\mathcal{J}} = v'^{\mathcal{J}} = o^{\mathcal{J}}$. This ensures one can be merged into the other to obtain $\mathcal{G}'$, and $\mathcal{J} \models_K \mathcal{G}'$.

If the $\leq_o$-rule is applicable to a node v in $\mathcal{G}$, then $\leq m\, S.C \in \mathcal{L}(v)$ for some m. As $\mathcal{J} \models_K \mathcal{G}$ by assumption, there are only $m' \leq m$ elements $o_1, \ldots, o_{m'}$ in $\Delta^{\mathcal{J}}$ such that $\langle v^{\mathcal{J}}, o_i \rangle \in S^{\mathcal{J}}$ and $o_i \in C^{\mathcal{J}}$. Furthermore, since v has m S-neighbours $w_1, \ldots, w_m \in \mathsf{in}(\mathcal{G})$ with $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j$ for all $1 \leq j < i \leq m$, every o_j must be such that $w_i^{\mathcal{J}} = o_j$ for some w_i. Since $\langle v^{\mathcal{J}}, v'^{\mathcal{J}} \rangle \in S^{\mathcal{J}}$ and $v'^{\mathcal{J}} \in C^{\mathcal{J}}$ for any v' that satisfies the conditions of the rule, $v'^{\mathcal{J}} = o_j$ for some o_j, which implies that there is some w_i such that $w_i^{\mathcal{J}} = v'^{\mathcal{J}}$. Thus we can merge v' into w_i to obtain a $\mathcal{G}'$ such that $\mathcal{J} \models_K \mathcal{G}'$.

Next, we consider the generating rules. If r is the $\exists$-rule, since the propagation rule is applicable, there is some v in $\mathcal{G}$ such that $\exists R.C \in \mathcal{L}(v)$. Hence, some $o \in \Delta^{\mathcal{J}}$ exists such that $\langle v^{\mathcal{J}}, o \rangle \in R^{\mathcal{J}}$ and $o \in C^{\mathcal{J}}$. A completion graph $\mathcal{G}'$ can be obtained by adding a new node w to $\mathcal{G}$. $\mathcal{J}$ will be modified to $\mathcal{J}'$ by setting $w^{\mathcal{J}'} = o$, and thus $\mathcal{J}' \models_K \mathcal{G}'$.

The case of the $\geq$-rule is analogous to the $\exists$-rule: if $\mathcal{J} \models_K \mathcal{G}$ and $\geq n\, S.C \in \mathcal{L}(v)$, there are m elements, $o_1, \ldots, o_m \in \Delta^{\mathcal{J}}$, $m \geq n$, such that $\langle v^{\mathcal{J}}, o_i \rangle \in R^{\mathcal{J}}$ and $o_i \in C^{\mathcal{J}}$ for each o_i. To obtain $\mathcal{G}'$, we add new nodes $w_1, \ldots, w_n$ to $\mathcal{G}$, set the labels of each w_i and each $v \to w_i$ as required, and introduce new inequalities $w_i \not\approx w_j$ to $\mathcal{G}$ for each pair $i \neq j$. By setting $w_i^{\mathcal{J}'} = o_i$ for $1 \leq i \leq n$, $\mathcal{J}' \models_K \mathcal{G}'$ is ensured.

Finally, the $o?$-rule is only applicable to v if $\leq n\, S.C \in \mathcal{L}(v)$. As $\mathcal{J} \models_K \mathcal{G}$ by assumption, there is some $m \leq n$ such that there are exactly m elements $o_1, \ldots, o_m$ in $\Delta^{\mathcal{J}}$ with $\langle v^{\mathcal{J}}, o_i \rangle \in S^{\mathcal{J}}$ and $o_i \in C^{\mathcal{J}}$. We can guess this m and create m nominal S-successors $w_1, \ldots, w_m$ of v with $C \in \mathcal{L}(w_i)$ and $w_i \not\approx w_j$ for all $1 \leq j < i \leq m$ to obtain $\mathcal{G}'$. By setting $w_i^{\mathcal{J}'} = o_j$ for each i, we ensure that $\mathcal{J}' \models_K \mathcal{G}'$ as desired. □

Claim 3 If $\langle \pi(x), \pi(y) \rangle \in \mathcal{E}(R')$, then $\mu(y)$ is an R'-neighbour of $\mu(x)$.

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

We prove that (i) implies that $\mu(y)$ is an R'-successor of $\mu(x)$. Analogously, (ii) implies that $\mu(x)$ is an $\mathsf{Inv}(R')$-successor of $\mu(y)$. Together, these two facts complete the proof of the claim. We consider three cases:

1) $\pi(x) = [\frac{a}{a}] \in \mathsf{in}(G_\pi)$: then $\mu(x) = \mathsf{tail}'(\pi(x)) = \mathsf{tail}(\pi(x)) = a$. If $\mathsf{tail}'(\pi(y))$ is an R'-successor of $\mathsf{tail}(\pi(x)) = a$, then $\mathsf{tail}'(\pi(y))$ is an R'-successor of an individual node. This implies that either $\mathsf{tail}'(\pi(y))$ is also an individual node; or it is a variable node that is not n-blocked and $\pi(y)$ is in some G_i with $\mathsf{afterblocked}(G_i) = \emptyset$. In both cases $\mu(y) = \mathsf{tail}'(\pi(y)) = \mathsf{tail}(\pi(y))$ holds and thus $\mu(y)$ is an R'-successor of $\mu(x)$.
2) $\pi(y) = [\frac{a}{a}] \in \mathsf{in}(G_\pi)$: then $\mu(y) = \mathsf{tail}'(\pi(y)) = \mathsf{tail}(\pi(y)) = a$. By construction of $\pi(x)$, either $\mathsf{tail}(\pi(x)) = \mathsf{tail}'(\pi(x))$ or $\mathsf{tail}(\pi(x)) = \psi(\mathsf{tail}'(\pi(x)))$. The claim thus holds if $\mu(x) = \mathsf{tail}(\pi(x))$. Suppose this is not the case. Then there are two possibilities.
   2a) $\mu(x) = \mathsf{tail}'(\pi(x))$, $\mathsf{tail}'(\pi(x)) \neq \mathsf{tail}(\pi(y))$ and $\mathsf{tail}(\pi(x)) = \psi(\mathsf{tail}'(\pi(x)))$.
   In this case, $\mathsf{tail}'(\pi(x))$ is a leaf of a blocked n-graph, and it is blocked by $\mathsf{tail}(\pi(x)) = \psi(\mathsf{tail}'(\pi(x)))$. Since $\mu(y) = a$ is an R'-successor of $\mathsf{tail}(\pi(x)) = \psi(\mathsf{tail}'(\pi(x)))$, we have that $\psi^{-1}(a)$ is an R'-successor of $\mathsf{tail}'(\pi(x))$. Since $\psi^{-1}(a) = a$ (recall that nominals occur in at most one node label, thus an individual node can only be isomorphic to itself), we have that $a = \mu(y)$ is an R'-successor of $\mathsf{tail}'(\pi(x)) = \mu(x)$ as desired.

2b) $\mu(x) = \psi(\mathsf{tail}'(\pi(x)))$, $\psi(\mathsf{tail}'(\pi(x))) \neq \mathsf{tail}(\pi(y))$ and $\mathsf{tail}(\pi(x)) = \mathsf{tail}'(\pi(x))$. Then $\pi(x)$ is a node of some G_i with $\mathsf{afterblocked}(G_i) \neq \emptyset$, and $\pi(x) \notin \mathsf{afterblocked}(G_i)$. Also in this case, $\mathsf{tail}'(\pi(x))$ is blocked by $\psi(\mathsf{tail}'(\pi(x)))$. Thus, $\mu(y) = a$ an R'-successor of $\mathsf{tail}(\pi(x)) = \mathsf{tail}'(\pi(x))$ implies that $\psi(a)$ is an R'-successor of $\psi(\mathsf{tail}'(\pi(x)))$. As $\psi(a) = a$, we have that $a = \mu(y)$ is an R'-successor of $\psi(\mathsf{tail}'(\pi(x))) = \mu(x)$ and the claim holds.

3) If $\pi(x), \pi(y) \notin \mathsf{in}(G_\pi)$, then $\pi(x)$ and $\pi(y)$ are nodes of some G_i.
First, suppose that $\mathsf{afterblocked}(G_i) = \emptyset$. Then $\mu(x) = \mathsf{tail}'(\pi(x))$. Since $\pi(y)$ is an R'-step of $\pi(x)$, we have $\mathsf{tail}'(\pi(x)) = \mathsf{tail}(\pi(x))$ (otherwise $\pi(y) \in \mathsf{afterblocked}(G_i)$ would follow, contradicting $\mathsf{afterblocked}(G_i) = \emptyset$). Clearly, if $\mathsf{tail}'(\pi(y))$ is an R'-successor of $\mathsf{tail}(\pi(x))$, then $\mu(y) = \mathsf{tail}'(\pi(y))$ is an R'-successor of $\mu(x) = \mathsf{tail}'(\pi(x)) = \mathsf{tail}(\pi(x))$.
Now we assume $\mathsf{afterblocked}(G_i) \neq \emptyset$. We can further distinguish the following cases:

3a) $\{\pi(x), \pi(y)\} \subseteq \mathsf{afterblocked}(G_i)$.
In this case, by definition, $\mu(x) = \mathsf{tail}'(\pi(x))$ and $\mu(y) = \mathsf{tail}'(\pi(y))$. Note that, by the definition of n-blocking, if there is some p with $\mathsf{tail}(p) \neq \mathsf{tail}'(p)$ and some p' which is a descendant of p, then $\mathsf{tail}(p') \neq \mathsf{tail}'(p')$ can only hold if the distance between p and p' is greater than n. As a consequence, and since the path length of G_i is bounded by n, $\mathsf{tail}(p) = \mathsf{tail}'(p)$ holds for each $p \in \mathsf{afterblocked}(G_i)$. Clearly, if $\mathsf{tail}'(\pi(y))$ is an R'-successor of $\mathsf{tail}(\pi(x))$, we have that $\mu(y) = \mathsf{tail}'(\pi(y))$ is an R'-successor of $\mu(x) = \mathsf{tail}'(\pi(x)) = \mathsf{tail}(\pi(x))$ as desired.

3b) $\pi(x) \notin \mathsf{afterblocked}(G_i)$ and $\pi(y) \in \mathsf{afterblocked}(G_i)$.
In this case $\mu(x) = \psi(\mathsf{tail}'(\pi(x)))$ and $\mu(y) = \mathsf{tail}'(\pi(y))$. It is also easy to see that $\pi(x) \in \mathsf{blockedLeaves}(G_i)$, thus $\mathsf{tail}(\pi(x)) \neq \mathsf{tail}'(\pi(x))$ and $\mathsf{tail}(\pi(x)) = \psi(\mathsf{tail}'(\pi(x)))$. Hence if $\mathsf{tail}'(\pi(y))$ is an R'-successor of $\mathsf{tail}(\pi(x))$, then $\mu(y) = \mathsf{tail}'(\pi(y))$ is an R'-successor of $\mu(x) = \psi(\mathsf{tail}'(\pi(x)))$.

3c) $\{\pi(x), \pi(y)\} \cap \mathsf{afterblocked}(G_i) = \emptyset$.
By definition, $\mu(x) = \psi(\mathsf{tail}'(\pi(x)))$ and $\mu(y) = \psi(\mathsf{tail}'(\pi(y)))$ hold. We can also verify that $\mathsf{tail}(\pi(x)) = \mathsf{tail}'(\pi(x))$, as otherwise $\pi(y) \in \mathsf{afterblocked}(G_i)$ would hold. By the definition of n-graph equivalence, if $\mathsf{tail}'(\pi(y))$ is an R'-successor of $\mathsf{tail}(\pi(x)) = \mathsf{tail}'(\pi(x))$, then $\mu(y) = \psi(\mathsf{tail}'(\pi(y)))$ is an R'-successor of $\mu(x) = \psi(\mathsf{tail}'(\pi(x)))$ as desired.

Note that the case $\pi(y) \notin \mathsf{afterblocked}(G_i)$ and $\pi(x) \in \mathsf{afterblocked}(G_i)$ is not possible. This proves the claim. □