Conjunctive Query Answering for the Description Logic $\mathcal{SHOIQ}$

Birte Glimm* Ian Horrocks
Ulrike Sattler
University of Manchester, UK
[glimm,horrocks,sattler]@cs.man.ac.uk

Contents

1 Introduction

The Semantic Web [2] aims at making web resources more accessible to automated processes by augmenting web pages with descriptions of their content. Ontologies are used to provide precisely specified meanings for these descriptions, and with the Web Ontology Language OWL [1] a standardised ontology building language is available. A notable feature of OWL is that two of the three OWL species (OWL Lite and OWL DL) correspond to Description Logics

*This work was supported by an EPSRC studentship.

(DLs) [6]. Existing DL reasoners[1] can, therefore, be used to provide automated reasoning support for OWL tools and applications [17, 13, 22]. Important reasoning tasks include not only checking concepts (classes in OWL) for satisfiability and subsumption, but also answering queries that retrieve instances of concepts and roles (properties in OWL).

Recently, a decision procedure for deciding satisfiability (and hence subsumption) of the DL $\mathcal{SHOIQ}$ has been introduced [8], which enables the implementation of reasoning procedures for the full expressivity of OWL DL. The development of a decision procedure for conjunctive query answering in $\mathcal{SHOIQ}$ is, however, still an outstanding issue; in fact it is not known if the problem is even decidable. Motik et al. [16] show that grounded conjunctive queries for $\mathcal{SHOIN}$ (and therefore for OWL DL) are decidable. However, the semantics of grounded queries is different from the usually assumed open-world semantics in DLs, since existentially quantified variables are always replaced with individual names. None of the existing conjunctive query answering techniques [20, 14, 4, 12] is able to handle transitive roles or nominals in the query body.[2]

In this paper, we present a decision procedure for conjunctive query answering for $\mathcal{SHOIQ}$ knowledge bases with a restriction on the use of transitive roles in the query body. We therefore provide the first decision procedure for conjunctive query answering in a logic that allows for nominals.

The algorithm extends the known tableaux algorithm for $\mathcal{SHOIQ}$ concept satisfiability [7] with a restricted form of the binder operator ($\downarrow$) and state variables known from Hybrid Logics [3]. Query entailment can then be reduced to deciding concept satisfiability for the extended DL. The binder operator is very powerful and leads, in an unrestricted form, to undecidability even for the simple DL $\mathcal{ALC}$. Decidability is regained by placing syntactic restrictions on the form of allowed concepts, while expressing conjunctive queries is still possible in the restricted fragment. Similar restrictions were already used to regain decidability for entailment in $\mathcal{ALC}$ (which is a syntactic variant of the modal logic **K**) extended with the binder and state variables [15]. For building a concept that represents a conjunctive query, we extend the so-called "rolling-up" technique [4, 20].

Compared to the technique used in CARIN for $\mathcal{ALCNR}$ [14] and its extension to $\mathcal{SHIQ}$ [18], the use of the hybrid logic binder ($\downarrow$) introduces far less non-determinism. It directly guides the tableaux algorithm towards producing a counter-model for the query. In addition, the presented technique integrates seamlessly into the existing tableaux algorithms, whereas, for the CARIN-style algorithm, checking the query entailment is an additional and completely separated step in the tableaux algorithm. We therefore believe that the presented algorithm is a more attractive choice from an implementation point of view.

[1] For example, FaCT++ `http://owl.man.ac.uk/factplusplus/`, KAON2 `http://kaon2.semanticweb.org/`, Pellet `http://www.mindswap.org/2003/pellet/`, or Racer Pro `http://www.racer-systems.com/`

[2] Although the algorithm presented by Calvanese et al. [4] allows the use of regular expressions in the query (in particular the transitive reflexive closure), it has been shown that the algorithm is incomplete [9, 5].

2 Preliminaries

Before we describe the extended tableaux algorithm, we first introduce the syntax and semantics of $\mathcal{SHOIQ}_{\downarrow}$, i.e., $\mathcal{SHOIQ}$ extended with the binder operator ($\downarrow$) and state variables, and conjunctive queries.

2.1 Syntax and Semantics of $\mathcal{SHOIQ}_{\downarrow}$

Definition 2.1 A $\mathcal{SHOIQ}$ Role Hierarchy
Let R be a set of *role names* with both transitive and normal role names $R_+ \cup R_p = R$, where $R_p \cap R_+ = \emptyset$. The set $\mathbf{R}$ of $\mathcal{SHOIQ}$-*roles* (or *roles* for short) is $R \cup \{r^- \mid r \in R\}$. A *role inclusion axiom* is of the form $r \sqsubseteq s$, for two roles r, s. A *role hierarchy or RBox* $\mathcal{R}$ is a finite set of role inclusion axioms.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$, which maps every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ s.t., for $p \in R$ and $r \in R_+$, $\langle x, y\rangle \in p^{\mathcal{I}}$ if $\langle y, x\rangle \in p^{-\mathcal{I}}$, and if $\langle x, y\rangle \in r^{\mathcal{I}}$ and $\langle y, z\rangle \in r^{\mathcal{I}}$, then $\langle x, z\rangle \in r^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ *satisfies a role hierarchy* $\mathcal{R}$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for each $r \sqsubseteq s \in \mathcal{R}$; such an interpretation is called a *model of* $\mathcal{R}$.

We introduce some notation to make the following considerations easier:

1. The inverse relation on roles is symmetric, and to avoid considering roles such as r^{--}, we define a function Inv which returns the inverse of a role. More precisely, we define $\mathsf{Inv}(r) := r^-$ if $r \in R$ and $\mathsf{Inv}(r) := s$ if $s = r^-$ for a role name s.

2. Since set inclusion is transitive, we define, for a role hierarchy $\mathcal{R}$, $\sqsubseteq^*_{\mathcal{R}}$ as the transitive reflexive closure of $\sqsubseteq$ over $\mathcal{R}$. We use $r \equiv_{\mathcal{R}} s$ as an abbreviation for $r \sqsubseteq^*_{\mathcal{R}} s$ and $s \sqsubseteq^*_{\mathcal{R}} r$.

3. For a role hierarchy $\mathcal{R}$ and a role s, we define $\mathsf{Trans}(s, \mathcal{R}) := \texttt{true}$ if $r \in R_+$ for some r with $r \equiv_{\mathcal{R}} s$ and $\mathsf{Trans}(s, \mathcal{R}) := \texttt{false}$ otherwise.

4. A role r is called *simple* w.r.t. a role hierarchy $\mathcal{R}$ if for each role s s.t. $s \sqsubseteq^*_{\mathcal{R}} r$ $\mathsf{Trans}(s) = \texttt{false}$.

5. In the following, if $\mathcal{R}$ is clear from the context, then we may abuse our notation and use $\sqsubseteq^*$ and $\mathsf{Trans}(s)$ instead of $\sqsubseteq^*_{\mathcal{R}}$ and $\mathsf{Trans}(s, \mathcal{R})$, and we say that "$s$ is a simple role" instead of "s is simple with respect to $\mathcal{R}$".

Definition 2.2 (Syntax and Semantics of $\mathcal{SHOIQ}_{\downarrow}$-Concepts)
Let N_C be a set of concept names with a subset $N_I \subseteq N_C$ of nominals and let N_V be a countable, finite set of variable names disjoint from the set of concept names N_C. The set of $\mathcal{SHOIQ}_{\downarrow}$-*concepts* (or concepts for short) is the smallest set built inductively from N_C and N_V using the following grammar, where $A \in N_C, y \in N_V, n \in \mathbb{N}, r \in \mathbf{R}$ is an arbitrary role, and $s \in \mathbf{R}$ is a simple role:

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall r.C \mid \exists r.C \mid \leqslant ns.C \mid \geqslant ns.C \mid y \mid \downarrow y.C.$$

We say that a variable y is *free* in a $\mathcal{SHOIQ}_{\downarrow}$-concept C if y is not bound by $\downarrow$. We assume, w.l.o.g., that every variable y is only bound once by an occurrence

of $\downarrow y$. Every concept D in which this is not the case can be transformed into an equivalent one by naming variables apart.

We assume all concepts to be in *negation normal form (NNF)*; any concept can be transformed into an equivalent one in NNF by pushing negation inwards, making use of de Morgan's laws and the duality between existential and universal restrictions, and between atmost and atleast number restrictions ($\leqslant nr.C$ and $\geqslant nr.C$ respectively) [11]. Since the binder ($\downarrow$) is self-dual, the concept $\neg\downarrow y.C$ is equivalent to $\downarrow y.\neg C$. For a concept C, we use $\dot{\neg}C$ to denote the NNF of $\neg C$.

The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ maps, additionally, every concept name to a subset of $\Delta^{\mathcal{I}}$. For an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$, an element $d \in \Delta^{\mathcal{I}}$, and a variable $y \in N_V$, we denote with $\mathcal{I}_{[y/d]}$ the interpretation that extends $\mathcal{I}$ such that $y^{\mathcal{I}} = \{d\}$. We can then define the semantics of $\mathcal{SHOIQ}_{\downarrow}$-concepts as:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \qquad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \qquad \sharp(o^{\mathcal{I}}) = 1 \text{ for all } o \in N_I \\
(\forall r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{ For all } d' \in \Delta^{\mathcal{I}}, \text{ if } \langle d, d'\rangle \in r^{\mathcal{I}}, \text{ then } d' \in C^{\mathcal{I}}\} \\
(\exists r.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \text{ There is a } d' \in \Delta^{\mathcal{I}} \text{ with } \langle d, d'\rangle \in r^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\} \\
(\leqslant nr.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \sharp(r^{\mathcal{I}}(d, C)) \leqslant n\} \\
(\geqslant nr.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \sharp(r^{\mathcal{I}}(d, C)) \geqslant n\} \\
(\downarrow y.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid d \in C^{\mathcal{I}_{[y/d]}}\}
\end{aligned}
$$

where, $n \in \mathbb{N}$, for a set M we denote the cardinality of M by $\sharp(M)$, and we define $r^{\mathcal{I}}(d, C)$ as $\{d' \in \Delta^{\mathcal{I}} \mid \langle d, d'\rangle \in r^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}$.

A concept C is called *satisfiable w.r.t. a role hierarchy* $\mathcal{R}$, if there is a model $\mathcal{I} = (\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ of $\mathcal{R}$ with $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a *model of* C *with respect to* $\mathcal{R}$. For C and D (possibly complex) concepts, $C \sqsubseteq D$ is called a *general concept inclusion* (GCI), and a finite set of GCIs is called a *TBox*. An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies each GCI in $\mathcal{T}$; such an interpretation is called *a model of* $\mathcal{T}$. A concept D subsumes a concept C with respect to $\mathcal{R}$ and $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{R},\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in every model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$.

An *ABox* is a partial instantiation of the schema defined by the TBox and RBox. In the presence of nominals, an ABox can be expressed in terms of TBox axioms [19]. Therefore, we do not consider ABoxes here and define a *knowledge base* (KB) as a tuple $\langle \mathcal{T}, \mathcal{R}\rangle$ with $\mathcal{T}$ a TBox and $\mathcal{R}$ a role hierarchy. For $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}\rangle$ a KB and $\mathcal{I} = (\Delta^{\mathcal{I}},\cdot^{\mathcal{I}})$ a model of $\mathcal{T}$ and $\mathcal{R}$, we say that $\mathcal{I}$ *is a model of* $\mathcal{K}$ and write $\mathcal{I} \models \mathcal{K}$; we say that $\mathcal{K}$ *is satisfiable* if there is a model $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{K}$ and we say that $\mathcal{K}$ *is unsatisfiable* otherwise.

In the presence of transitive roles, a TBox $\mathcal{T}$ can be *internalised* using an "approximation" of a universal role u [8, 11]. Therefore, testing the satisfiability of a concept with respect to a TBox and a role hierarchy can be reduced to testing satisfiability with respect to the role hierarchy only and we assume in the remainder that the TBox of a knowledge base is internalised.

2.2 Conjunctive Queries

Now that we have formally defined the syntax and semantics of $\mathcal{SHOIQ}_{\downarrow}$-concepts and knowledge bases, we are ready to formally introduce conjunctive

queries.

Definition 2.3 Conjunctive Queries
Let $\vec{y}$ be a vector of *non-distinguished variables*, and $\vec{c}$ a vector of nominals from N_I, both mutually disjoint. A *Boolean conjunctive query* q has the form $\langle\rangle \leftarrow conj_1(\vec{y};\vec{c}) \wedge \ldots \wedge conj_n(\vec{y};\vec{c})$. We call $\mathbf{T}(q) = \vec{y} \cup \vec{c}$ the set of *terms* in q,[3] and we call each $conj_i(\vec{y};\vec{c})$ for $1 \leq i \leq n$ an atom. Atoms are either concept or role atoms: a concept atom has the form $t_1 : C$, and a role atom the form $\langle t_1, t_2\rangle : r$, for $\{t_1, t_2\} \subseteq \mathbf{T}(q)$, C a $\mathcal{SHOIQ}$-concept and r a $\mathcal{SHOIQ}$-role.

We define the *semantics of Boolean conjunctive queries* as follows: Let $\mathcal{K}$ be a KB, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ a model for $\mathcal{K}$, q a Boolean conjunctive query for $\mathcal{K}$, and $\cdot^{A} : \mathbf{T}(q) \to \Delta^{\mathcal{I}}$ an assignment in $\mathcal{I}$. We say that q *is true in* $\mathcal{I}$ and write $\mathcal{I} \models q$, if there is an assignment $\cdot^{A}$ in $\mathcal{I}$ s.t. $t^A \in t^{\mathcal{I}}$ for every nominal $t \in \vec{c}$, $t^A \in C^{\mathcal{I}}$ for every concept atom $t : C$ in q, and $\langle t_1^A, t_2^A\rangle \in r^{\mathcal{I}}$ for every role atom $\langle t_1, t_2\rangle : r$ in q. For such an interpretation $\mathcal{I}$ and assignment $\cdot^{A}$, we write $\mathcal{I}$, A $\models q$. If $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models q$ for all models $\mathcal{I}$ of $\mathcal{K}$, then we say that q *is true in* $\mathcal{K}$, and write $\mathcal{K} \models q$; otherwise we say that q *is false in* $\mathcal{K}$, and write $\mathcal{K} \not\models q$.

Let $\vec{x}$ be a vector of *distinguished (free) variables*. The answer to a non-Boolean query $\langle\vec{x}\rangle \leftarrow conj_1(\vec{x};\vec{y};\vec{c}) \wedge \ldots \wedge conj_n(\vec{x};\vec{y};\vec{c})$ with respect to a KB $\mathcal{K}$ can be computed by using (possibly several) Boolean queries in which the distinguished variables are replaced with nominals from $\mathcal{K}$ in such a way that all possible answer tuples are tested. Given that there are a finite number of nominals in $\mathcal{K}$, there is a bound on the number of Boolean queries necessary to answer a non-Boolean query. This is clearly not very efficient, but optimisations can be used, e.g., to identify a (hopefully small) set of candidate tuples. In the following we will, therefore, consider only Boolean conjunctive queries, and if we write that q is a query, we implicitly assume that q is a Boolean conjunctive query.

Without loss of generality, we further assume that all queries are in a particular normal form. This simplifies several parts of the query answering algorithm. Therefore, we define some transformation for a given Boolean conjunctive query that can be used to obtain an equivalent query in the desired normal form.

We first write a Boolean conjunctive query q as the set

$$\{t : C \mid t : C \text{ is a concept atom in } q\} \cup \{\langle t, t'\rangle : r \mid \langle t, t'\rangle : r \text{ is a role atom in } q\}.$$

I.e., each conjunct in q is an element in the set. In the remainder, we often use this set notation and it should be clear from the context, when this is the case.

Lemma 2.4
Given a Boolean conjunctive query q, we can obtain an equivalent Boolean conjunctive query q' in which all terms are (non-distinguished) variables.

Proof. Let q be a query that contains constants, i.e., $\vec{c}$ is non-empty. We can eliminate all constants from the terms and obtain a Boolean conjunctive query

[3]For readability, we sometimes abuse our notation and refer to $\vec{y}$ as a set. When referring to a vector $\vec{y}$ as a set, we mean the set $\{y_i \mid y_i \text{ occurs in } \vec{y}\}$.

q' from q as follows:

$$\{y : C \mid y : C \text{ is a concept atom in } q \text{ and } y \in \vec{y}\} \cup \tag{1}$$

$$\{y : \{a\} \sqcap C \mid a : C \text{ is a concept atom in } q, \; a \in \vec{c}, \text{ and } y \in N_V \text{is new in } q\} \cup \tag{2}$$

$$\{\langle y, y' \rangle : r \mid \langle y, y' \rangle : r \text{ is a role atom in } q, \text{ and } y, y' \in \vec{y}\} \cup \tag{3}$$

$$\{y : \exists r.\{a\} \mid \langle y, a \rangle : r \text{ is a role atom in } q, \; a \in \vec{c}, \text{ and } y \in N_V\} \cup \tag{4}$$

$$\{y : \exists \mathsf{Inv}(r).\{a\} \mid \langle a, y \rangle : r \text{ is a role atom in } q, \; a \in \vec{c}, \text{ and } y \in N_V\} \cup \tag{5}$$

$$\{y : \{a\} \sqcap \exists r.\{b\} \mid \langle a, b \rangle : r \text{ is a role atom in } q, \; a, b \in \vec{c}, \text{ and } y \in N_V \text{is new in } q\}. \tag{6}$$

We now have to show that q' is equivalent to q:

The cases (1 and 3) for concept and role atoms in which all terms are variables from $\vec{y}$ are clear. These atoms are in q' iff they are in q.

For case 2: Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I} \models a : C$, we have that for each assignment $\cdot^{A}$, $a^{A} \in a^{\mathcal{I}}$ for $a \in N_I$ and $a^{A} \in C^{\mathcal{I}}$, i.e., $a^{A} \in a^{\mathcal{I}} \cap C^{\mathcal{I}}$. We can, therefore, clearly extend any assignment $\cdot^{A}$ to an assignment $\cdot^{A'}$ by mapping the new variable y to a. Now, let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \not\models$ a:C. This is the case if there is no assignment $\cdot^{A}$ such that $a^{A} \in a^{\mathcal{I}} \cap C^{\mathcal{I}}$ for $a \in N_I$. However, this means that $a^{\mathcal{I}} \cap C^{\mathcal{I}} = \emptyset$ and clearly there cannot be an assignment $\cdot^{A'}$ such that $y^{A} \in a^{\mathcal{I}} \cap C^{\mathcal{I}}$.

For case 4: Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I} \models \langle y, a \rangle : r$, we have that there exists an assignment $\cdot^{A}$ such that $\langle y^{A}, a^{A} \rangle \in r^{\mathcal{I}}$ and $a^{A} \in a^{\mathcal{I}}$ for $a \in N_I$. In q', $\langle y, a \rangle : r$ is replaced with $y : \exists r.\{a\}$. By definition of the semantics, we have that $(\exists r.\{a\})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists d'.\langle d, d' \rangle \in r^{\mathcal{I}} \text{ and } d' \in a^{\mathcal{I}}\}$. By setting $d = y^{A}$ and $d' = a^{A}$, it is clear that $d \in (\exists r.\{a\})^{\mathcal{I}}$ and clearly $\cdot^{A}$ is an assignment for q' and $\mathcal{I}$. Now, let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation such that $\mathcal{I} \not\models \langle y, a \rangle : r$. Hence, there is no assignment $\cdot^{A}$ such that $\langle y^{A}, a^{A} \rangle \in r^{\mathcal{I}}$ and $a^{A} \in a^{\mathcal{I}}$ for $a \in N_I$. For a contradiction, assume that $\mathcal{I} \models y : \exists r.\{a\}$ and that $\cdot^{A'}$ is the assignment in $\mathcal{I}$. Hence, $y^{A'} \in (\exists r.\{a\})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists d'.\langle d, d' \rangle \in r^{\mathcal{I}} \text{ and } d' \in a^{\mathcal{I}}\}$. Clearly, we can obtain a mapping $\cdot^{A}$ for q in $\mathcal{I}$ by setting $y^{A} = d$ and $a^{A} = d'$ such that $\langle y^{A}, a^{A} \rangle \in r^{\mathcal{I}}$ and $a^{A} \in a^{\mathcal{I}}$, contradicting the assumption.

The following two cases (5 and 6) are very similar to the previous one (4). □

Lemma 2.5
Given a Boolean conjunctive query q in which all terms are (non-distinguished) variables, we can obtain an equivalent Boolean conjunctive query q' in which each variable occurs exactly once in a concept atom.

Proof. We obtain q' from q by setting q' to

$$\{y : C_1 \sqcap \ldots \sqcap C_n \mid y : C_i \text{ is a concept atom in } q \text{ for } 1 \leqslant i \leqslant n\} \cup \tag{7}$$

$$\{y : \top \mid y \in \vec{y} \text{ and there is no concept atom } y : C \text{ in } q\} \cup \tag{8}$$

$$\{\langle y, y' \rangle : r \mid \langle y, y' \rangle : r \text{ is a role atom in } q'\}. \tag{9}$$

Clearly, each variable in q' occurs exactly once in a concept atom. It therefore remains to show that q' is equivalent to q:

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation such that $\mathcal{I} \models q$, let $\cdot^{A}$ be an assignment for q in $\mathcal{I}$, and let y be a variable in $\vec{y}$ such that $y : C_i$ for $1 \leqslant i \leqslant n$ are concept

atoms in q. By definition of the semantics, we have that $y^A \in C_i^{\mathcal{I},A}$ and hence $y^A \in C_1^{\mathcal{I},A} \cap \ldots \cap C_n^{\mathcal{I},A}$ for $1 \leqslant i \leqslant n$. However, then $y \in (C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I},A}$ and after replacing the concept atoms $y : C_i$ for $1 \leqslant i \leqslant n$ with a concept atom $y : C_1 \sqcap \ldots \sqcap C_n$ in q', we still have that $\mathcal{I}, A \models q'$. Let now $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation such that $\mathcal{I} \not\models q$, but $\mathcal{I} \models q'$, where q' is obtained by replacing the concept atoms $y : C_i$ for $1 \leqslant i \leqslant n$ in q with a concept atom $y : C_1 \sqcap \ldots \sqcap C_n$ in q'. Hence, there is an assignment $\cdot^A$ for q' in $\mathcal{I}$ such that $y^A \in (C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I},A} = C_1^{\mathcal{I},A} \cap \ldots \cap C_n^{\mathcal{I},A}$, i.e., $y \in C_i^{\mathcal{I},A}$ for $1 \leqslant i \leqslant n$. However, then $\cdot^A$ is also an assignment for q, contradicting the assumption.

Let y be a variable in q such that there is no concept atom $y : C$ for y in q, let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation such that $\mathcal{I} \models q$, and let $\cdot^A$ be an assignment for q in $\mathcal{I}$. However, since $\cdot^A$ is an assignment in $\mathcal{I}$, $y^A \in \Delta^{\mathcal{I}} = \top^{\mathcal{I}}$. Hence, $\cdot^A$ is also an assignment for the query q' obtained by adding $y : \top$ to q.

Since role atoms are in q' iff there are in q, we are done. □

Definition 2.6 Query Normal Form
Let q be a Boolean conjunctive query. We say that q is in *query normal form* (QNF) if all terms of q are (non-distinguished) variables and if each variable occurs exactly once in a concept atom.

Every Boolean conjunctive query q with $\vec{y}$ the vector of non-distinguished variables and $\vec{c}$ the vector of nominals from N_I can be transformed into an equivalent query q' in QNF as shown by Lemma 2.4 and Lemma 2.5. If not stated otherwise, we therefore assume in the remainder of this paper that all Boolean conjunctive queries are in QNF.

3 The Representation of Queries as $\mathcal{SHOIQ}_{\downarrow}$-Concepts

In this section, we introduce the relationship between $\mathcal{SHOIQ}$ conjunctive queries and $\mathcal{SHOIQ}_{\downarrow}$ concepts. Since we aim at finding a decision procedure for $\mathcal{SHOIQ}$ conjunctive queries and the $\downarrow$ binder makes already $\mathcal{ALC}$ undecidable, we also define a syntactic restriction on $\mathcal{SHOIQ}_{\downarrow}$-concepts. This restricted form, called the query form, corresponds exactly to the concepts that can express a conjunctive query and is far less expressive than the unrestricted $\mathcal{SHOIQ}_{\downarrow}$ DL.

3.1 Representing Queries as Graphs

We start with describing how a $\mathcal{SHOIQ}_{\downarrow}$-concept can be obtained from a given conjunctive query. This is easier to understand, if we represent a query as a graph.

Definition 3.1 Query Graph
A Boolean conjunctive query q in QNF can be represented as a labelled, directed multi-graph $\mathcal{G}(q) = \langle V, E, \mathcal{L} \rangle$ such that there is a bijective mapping $\sigma : V \to \vec{y}$ from the nodes in V to variables in q. Since q is in QNF, q contains only non-distinguished variables. Each node $v \in V$ is labelled with a concept $\mathcal{L}(v) = C$ such that $\sigma(v) : C$ is a concept atom in q. There is an edge $\langle v, v' \rangle \in E$ labelled

with $\mathcal{L}(\langle v, v' \rangle) = r$ for each role atom $\langle \sigma(y), \sigma(y') \rangle : r$ in q. We call the graph $\mathcal{G}(q)$ the *query graph for* q .

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. Similarly to the semantics of conjunctive queries, we say that $\mathcal{I}$ *is a model of* $\mathcal{G}(q)$, written as $\mathcal{I} \models \mathcal{G}(q)$, if there is an assignment $\cdot^A : V \to \Delta^{\mathcal{I}}$ such that for each node $v \in V$ with $\mathcal{L}(v) = C$, $\sigma(v)^A \in C^{\mathcal{I}}$ and for each edge $\langle v, v' \rangle \in E$ with $\mathcal{L}(\langle v, v' \rangle) = r$, $\langle \sigma(v)^A, \sigma(v')^A \rangle \in r^{\mathcal{I}}$. For such an interpretation $\mathcal{I}$ and assignment A, we write $\mathcal{I}, A \models \mathcal{G}(q)$.

Lemma 3.2
Let $\mathcal{G}(q)$ be the query graph for a Boolean conjunctive query q in QNF and let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation, then $\mathcal{I} \models q$ iff $\mathcal{I} \models \mathcal{G}(q)$.

Proof. The query graph is an equivalent representation for q since q is in QNF and hence contains only non-distinguished variables as terms, σ is a one-to-one mapping and for each node v, there is exactly on concept atom $\sigma(v) : C$ in q and $\mathcal{L}(v) = C$. A similar argument holds for the edges. Hence, an assignment $\cdot^A$ for q in $\mathcal{I}$ is by definition an assignment for $\mathcal{G}(q)$ and vice versa. □

3.2 Building a $\mathcal{SHOIQ}_{\downarrow}$-Concept from a Query Graph

In general, a query graph $\mathcal{G}(q)$ may be composed of several connected components $\mathcal{G}(q)_1, \ldots, \mathcal{G}(q)_n$. For each such component $\mathcal{G}(q)_i$ with $1 \leqslant i \leqslant n$, we build a $\mathcal{SHOIQ}_{\downarrow}$-concept C_i by traversing the component in a depth-first manner.

We first define two auxiliary functions breakCycle and rollUp. The former function is used when a cycle is detected and it replaces the edge that closes the cycle with a concept expression. More formally, for a node v and an edge $e = \langle v, v' \rangle$ or $e = \langle v', v \rangle$, we define breakCycle$(e, v)$ as:

1. if $e = \langle v, v' \rangle$ then
   set $\mathcal{L}(v) := \mathcal{L}(v) \sqcap \exists \mathcal{L}(e).\{\sigma(v')\}$
   else
   set $\mathcal{L}(v) := \mathcal{L}(v) \sqcap \exists \mathsf{Inv}(\mathcal{L}(e)).\{\sigma(v')\}$
2. remove e from E

Although the function removes an edge, a query q is true in the modified query graph iff q is true in the original query graph, as shown by the following lemma:

Lemma 3.3
Let q be a Boolean conjunctive query in QNF, $\mathcal{G}(q) = (V, E, \mathcal{L})$ the query graph for q, and $\mathcal{G}(q)'$ the graph obtained by applying breakCycle(e, v) for an edge $e = \langle v, v' \rangle \in E$ or $e = \langle v', v \rangle \in E$ and a node $v \in V$. For an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and an assignment $\cdot^A$ in $\mathcal{I}$, $\mathcal{I}, A \models \mathcal{G}(q)$ iff $\mathcal{I}, A \models \mathcal{G}(q)'$.

Proof. We first show that if $\mathcal{I}, A \models \mathcal{G}(q)$, then $\mathcal{I}, A \models \mathcal{G}(q)'$. Suppose first, that e is an outgoing edge from v, i.e., $e = \langle v, v' \rangle$, and let $\mathcal{L}(e) = r$. Since $\mathcal{I}, A \models \mathcal{G}(q)$, we have that $\langle \sigma(v)^A, \sigma(v')^A \rangle \in r^{\mathcal{I}}$. After applying breakCycle(e, v), we have that $\mathcal{L}(v) = \mathcal{L}(v) \sqcap \exists \mathcal{L}(e).\{\sigma(v')\}$. Therefore, we have to show that $\sigma(v)^A \in (\exists r.\{\sigma(v')\})^{\mathcal{I},A}$. However, $(\exists r.\{\sigma(v')\})^{\mathcal{I},A} = \{d \in \Delta^{\mathcal{I}} \mid \langle d, d' \rangle \in r^{\mathcal{I}} \text{ and } d' \in \{\sigma(v')^A\}\}$ and since $\langle \sigma(v)^A, \sigma(v')^A \rangle \in r^{\mathcal{I}}$ and $\sigma(v')^A \in \{\sigma(v')^A\}$, we clearly have that $\sigma(v)^A \in (\exists r.\{\sigma(v')\})^{\mathcal{I},A}$. Therefore, $\cdot^A$ is an assignment for $\mathcal{G}(q)'$ in

$\mathcal{I}$ as well, and $\mathcal{I} \models \mathcal{G}(q)'$ as required. The case with with $e = \langle v, v' \rangle$ is similar, just with the use of inverse roles. The same technique can be used to show the opposite direction of the proof. □

The function rollUp replaces a leaf node of the query graph and its incoming edge with a concept expression added to the the label of the parent node. More formally, for a query graph $\mathcal{G}(q) = (V, E, \mathcal{L})$, a node $v \in V$, and an edge $e = \langle v, v' \rangle \in E$ or $e = \langle v', v \rangle \in E$ such that e is the only edge for v', we define $\mathsf{rollUp}(e, v)$ as:

1. if $e = \langle v, v' \rangle$ then
   set $\mathcal{L}(v) := \mathcal{L}(v) \sqcap \exists \mathcal{L}(e).(\downarrow \sigma(v').(\mathcal{L}(v')))$
   else
   set $\mathcal{L}(v) := \mathcal{L}(v) \sqcap \exists \mathsf{Inv}(\mathcal{L}(e)).(\downarrow \sigma(v').(\mathcal{L}(v')))$
2. remove e and v' from E and V respectively

This function is very similar to the known rolling-up technique and results in an equi-satisfiable query graph:

Lemma 3.4
Let q be a Boolean conjunctive query in QNF, $\mathcal{G}(q) = (V, E, \mathcal{L})$ the query graph for q, and $\mathcal{G}(q)'$ the graph obtained by applying $\mathsf{rollUp}(e, v)$ for an edge $e = \langle v, v' \rangle \in E$ or $e = \langle v', v \rangle \in E$, a node $v \in V$, and a leaf node $v' \in V$. For an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and an assignment $\cdot^{A}$ in $\mathcal{I}$, $\mathcal{I}, A \models \mathcal{G}(q)$ iff $\mathcal{I}, A \models \mathcal{G}(q)'$.

Proof. We first show that if $\mathcal{I}, A \models \mathcal{G}(q)$, then $\mathcal{I}, A \models \mathcal{G}(q)'$. Suppose first, that e is an outgoing edge from v, i.e., $e = \langle v, v' \rangle$, and let $\mathcal{L}(e) = r$. Since $\mathcal{I}, A \models \mathcal{G}(q)$, we have that $\langle \sigma(v)^A, \sigma(v')^A \rangle \in r^{\mathcal{I}}$ and that $\sigma(v')^A \in \mathcal{L}(v')^{\mathcal{I},A}$. After applying $\mathsf{rollUp}(e, v)$, we have that $\mathcal{L}(v) = \mathcal{L}(v) \sqcap \exists \mathcal{L}(e).(\downarrow \sigma(v').(\mathcal{L}(v')))$ and v' and e are removed from $\mathcal{G}(q)$. Therefore, we have to show that $\sigma(v)^A \in (\exists r.(\downarrow \sigma(v').(\mathcal{L}(v'))))^{\mathcal{I},A}$. However, $(\exists r.(\downarrow \sigma(v').(\mathcal{L}(v'))))^{\mathcal{I},A} = \{d \in \Delta^{\mathcal{I}} \mid \langle d, d' \rangle \in r^{\mathcal{I}}$ and $d' \in (\downarrow \sigma(v').(\mathcal{L}(v')))^{\mathcal{I},A}\}$. Since $\langle \sigma(v)^A, \sigma(v')^A \rangle \in r^{\mathcal{I}}$, we have to show that $\sigma(v')^A \in (\downarrow \sigma(v').(\mathcal{L}(v')))^{\mathcal{I},A} = \{d \in \Delta^{\mathcal{I}} \mid d \in (\mathcal{L}(v'))^{\mathcal{I}_{[\sigma(v')/d]}}\}$ and clearly this is the case for $d = \sigma(v')$, since $\sigma(v')^A \in (\mathcal{L}(v'))^{\mathcal{I},A}$ if $\mathcal{I}, A \models \mathcal{G}(q)$. Therefore, $\cdot^A$ is an assignment for $\mathcal{G}(q)'$ in $\mathcal{I}$ as well, and $\mathcal{I} \models \mathcal{G}(q)'$ as required. The case with with $e = \langle v, v' \rangle$ is similar, just with the use of inverse roles and again, the same technique can be used to show the opposite direction of the proof. □

We are now ready to define the graph traversal algorithm. For $v \in V$, we recursively define the function $\mathsf{visit}(v)$ as follows:

1. mark v
2. while there is an unmarked edge $e = \langle v, v' \rangle$ or $e = \langle v', v \rangle$
   (a) if v' is marked then
      i. $\mathsf{breakCycle}(e, v')$
   (b) else
      i. mark e
      ii. $\mathsf{visit}(v')$, and

iii. $\mathsf{rollUp}(e, v')$

Lemma 3.5
Let $\mathcal{G}(q)_i$ be a component of a query graph $\mathcal{G}(q)$. The function $\mathsf{visit}(v)$, applied to a starting node v, terminates.

Proof. The while loop in step 2 works only on unmarked edges. However, in each loop, we either remove an edge with breakCycle or mark an edge, and since the number of role atoms in q gives a bound on the number of edges, there is a bound on how often we can go through the while loop. Since the recursive call of visit is in the while loop, there is a bound on the number of recursion steps as well. Both functions breakCycle and rollUp trivially terminate. □

Lemma 3.6
Let $\mathcal{G}(q)_i$ be a component of a query graph $\mathcal{G}(q)$. After applying the function $\mathsf{visit}(v)$ to a starting node v, $\mathcal{G}(q)_i$ is collapsed into v.

Proof. We have to show that we do not apply breakCycle to cut off unvisited parts of the component. However, breakCycle is only applied on unmarked edges between marked nodes, hence we only delete edges between parts of the component that were already seen. Further on, we cannot cut off parts of the component that are not completely finished and are used in a backtracking step, since we mark edges that were used to visit an unmarked node and breakCycle is only applied to unmarked edges. Since we visit all nodes in the component and mark each node and the edge that was used to visit this component, we finally remove all edges from a node, except the edge marked when visiting the node. Hence, rollUp can finally remove all but the starting node (that has no marked edge left) and the edges connecting these nodes. □

Definition 3.7 Query Concepts
Let $\mathcal{G}(q)$ be the query graph for a Boolean conjunctive query q in QNF and let $\mathcal{G}(q)_1, \ldots, \mathcal{G}(q)_n$ be the components of $\mathcal{G}(q)$. After traversing a component $\mathcal{G}(q)_i$ from a starting node v in $\mathcal{G}(q)_i$ with the function visit, the component is collapsed into v. We can now obtain a $\mathcal{SHOIQ}_{\downarrow}$-concept C_i for $\mathcal{G}(q)_i$ by setting $C_i = \downarrow \sigma(v).\mathcal{L}(v)$. If a concept C_i contains a binder $\downarrow y.$, but the variable y does not occur in C_i, then we omit $\downarrow y.$ from C_i. This does obviously not affect the satisfiability of C_i. We call each concept C_i obtained by traversing $\mathcal{G}(q)_i$ a *query concept* and we call the set $\{C_1, \ldots, C_n\}$ the *query concepts* for q.

Lemma 3.8
Let q be a Boolean conjunctive query in QNF, $\mathcal{G}(q)_i$ a component of the query graph for q, and $\mathcal{G}(q)'_i$ the collapsed component after traversing $\mathcal{G}(q)_i$ from a starting node v. For an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and an assignment $\cdot^{A}$ in $\mathcal{I}$, $\mathcal{I}, A \models \mathcal{G}(q)'_i$ iff $\mathcal{I}, A \models \downarrow \sigma(v).\mathcal{L}(v)$.

Proof. If $\mathcal{I}$ is a model for $\downarrow \sigma(v).\mathcal{L}(v)$, then there is an element $d \in \Delta^{\mathcal{I}}$ such that $\sigma(y)^A = d$ and $d \in (\downarrow \sigma(v).\mathcal{L}(v))^{\mathcal{I},A}$, i.e., $d \in \{(\mathcal{L}(v))^{\mathcal{I}_{[\sigma(v)/d]},A}\}$. However, this is exactly the case, when $\mathcal{I}, A \models \mathcal{G}(q)'_i$. □

We can now use the query concepts from the query graph traversal to decide query entailment for a given knowledge base.

Lemma 3.9
Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{R} \rangle$ be a $\mathcal{SHOIQ}$ knowledge base, q a Boolean conjunctive query, and $\{C_1, \ldots, C_n\}$ the query concepts for q. $\mathcal{K} \models q$ iff each $\mathcal{K}_i = \langle \mathcal{T} \cup \{\top \sqsubseteq \neg C_i\}, \mathcal{R} \rangle$ for $1 \leqslant i \leqslant n$ is unsatisfiable.

Proof. Lemma 3.9 is a consequence of the facts that $\mathcal{I} \models q$ iff $\mathcal{I} \models \mathcal{G}(q)$ (by Lemma 3.2), and $\mathcal{I} \models \mathcal{G}(q)_i$ iff $\mathcal{I} \models C_i$ for $1 \leqslant i \leqslant n$ (by Lemma 3.8). □

3.3 $\mathcal{SHOIQ}_{\downarrow}$-Concepts in Query Form

We are now ready to define exactly those $\mathcal{SHOIQ}_{\downarrow}$-concept that the tableaux algorithm presented in Section 5 is able to handle and we show that for each query concept C_i for a query q that contains only simple roles, $\dot{\neg} C_i$ has this form.

Definition 3.10 Query Form
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept. We say that D is in *query form*, if

1. D is in negation normal form,
2. no variable occurs free in D,
3. D is built according to the following grammar:
   $D ::= C \mid \neg y \mid C \sqcup \downarrow y.(\forall r_1.D \sqcup \ldots \sqcup \forall r_n.D)$,
   where C is a $\mathcal{SHOIQ}$-concept, $r_1, \ldots, r_n$ are roles, and $y \in N_V$, and
4. all roles that occur under the scope of a universal quantifier over a variable are simple.

For an example, let D be a $\mathcal{SHOIQ}$-concept, r, s simple roles, and $y \in N_V$. The concept $\downarrow y.(\neg D \sqcup \forall r.(\forall s.\neg y))$ is in query form, whereas the concept $\downarrow y.(\neg D \sqcup \exists r.(\forall s.\neg y))$ is not since $\neg y$ occurs in the scope of an existential quantifier.

Lemma 3.11
Let q be a Boolean conjunctive query in QNF with only simple roles occurring in the role atoms of q, let $\mathcal{G}(q) = (V, E, \mathcal{L})$ be the query graph for q with $\mathcal{G}(q)_1, \ldots, \mathcal{G}(q)_n$ the components of $\mathcal{G}(q)$, and let $C_1, \ldots, C_n$ be the query concepts for q. Each concept $\dot{\neg} C_i$ is a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form.

Proof. Each concept $\dot{\neg} C_i$ is in NNF by definition. We now show that no variable occurs free in $\dot{\neg} C_i$: During the graph traversal, we only use the functions breakCycle and rollUp. The function breakCycle introduces a variable for a marked node and the function rollUp introduces a $\downarrow$ binder for a variable. In Lemma 3.6, we have already shown that we do not apply breakCycle to cut off unvisited parts of a component and hence a $\downarrow$ binder is introduced for each variable represented by a node in the component. Since we use rollUp only on leaf nodes and remove the node and its incoming edge, no new references to the node can be introduces by breakCycle and the $\downarrow$ introduced in rollUp binds all the existing variables.

Each $\dot{\neg} C_i$ satisfies the given grammar, since the $\mathcal{SHOIQ}$ concept C that initially labels a node v is only conjoined with concepts obtained in the traversal of $\mathcal{G}(q)_i$. During the traversal, we only introduce positive variables, existential

quantifiers and conjunctions. Hence, after negating C_i and building the NNF, we have that all variables occur negatively, are universally quantified, and each conjuncts added during the traversal becomes a disjunct.

Finally, roles that occur under the scope of a universal quantifier over a variable are those introduced by removing an edge in $\mathcal{G}(q)_i$. These roles are simple, because all edges correspond to a role atom in q and all roles in the role atoms of q are simple roles by assumption. □

Since $\mathcal{SHOIQ}_{\downarrow}$-concepts in query form are not closed under negation, our definition of the closure of a concept with respect to a role hierarchy is slightly different from the one used for $\mathcal{SHOIQ}$.

Definition 3.12 Closure
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form and $\mathcal{R}$ a role hierarchy. We define the *closure* $\mathsf{cl}(D, \mathcal{R})$ *of* D *with respect to* $\mathcal{R}$, as follows:

$$\begin{aligned}\mathsf{cl}(D,\mathcal{R}) := \ & \mathsf{sub}(D) \cup \\ & \{\dot{\neg}C \mid C \in \mathsf{sub}(D) \text{ and } C \text{ is variable free}\} \cup \\ & \{\forall s.E \mid \forall r.E \in \mathsf{sub}(D) \text{ or } \dot{\neg}(\forall r.E) \in \mathsf{sub}(D) \text{ and} \\ & \ s \text{ occurs in } \mathcal{R} \text{ or } D\}.\end{aligned}$$

3.4 Role Rewriting for Query Concepts

The tableaux algorithm introduced in Section 5 builds a finite structure that represents a model. It uses a termination strategy called blocking, which is based on cycle detection. In the case of blocking, we have enough information for building an infinite model from the finite representation by repeating parts of the built structure. If C is a concept that contains a variable and C is a sub-concept of two different disjuncts D_1 and D_2, then it is important to know in the blocking condition, if C originates from D_1 or from D_2. For keeping the blocking conditions simple, we rewrite the roles occurring in one of the disjuncts and extend the role hierarchy of the given knowledge base, in order to preserve the original semantics.

Definition 3.13 Role Rewriting
Let $\mathcal{R}$ be a role hierarchy, D $\mathcal{SHOIQ}_{\downarrow}$-concept in query form such that $C \sqcup C'$ is a sub-concept of D, and $y \in N_V$ a variable. If there are two sub-concept C_1 and C_2 from the two different disjuncts C and C' such that y occurs in both of them and C_1 is a sub-concept of C_2, then we obtain a concept D' and role hierarchy $\mathcal{R}$' by renaming each role r quantifying over y in C_1 with a new role name r' not occurring in D or $\mathcal{R}$ and set $\mathcal{R}' = \mathcal{R} \cup \{r' \sqsubseteq r\}$.

Since the newly introduced role names do not occur in D or $\mathcal{R}$, we have that D is satisfiable with respect to $\mathcal{R}$ iff D' is satisfiable with respect to $\mathcal{R}$'.

4 A Tableau for $\mathcal{SHOIQ}_{\downarrow}$

Before we introduce the tableaux algorithm for $\mathcal{SHOIQ}_{\downarrow}$-concept satisfiability, we introduce tableaux as a close representation of models. Compared to a tableau for a $\mathcal{SHOIQ}$-concept, we now have to store the bindings for the variables. Further on, additional constraints are necessary for the $\downarrow$ binders and variables.

Definition 4.1 (Tableau)
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form, $\mathcal{R}$ a role hierarchy, $\mathbf{R}_D$ the set of role names occurring in D and $\mathcal{R}$, $\mathbf{S}$ a set of individuals, and $B_T := \{y/s \mid y \in N_V \text{ and } s \in \mathbf{S}\}$ the set of possible bindings. We define a *tableau* $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D and $\mathcal{R}$ such that $\mathcal{L} : \mathbf{S} \mapsto 2^{\mathsf{cl}(D) \times 2^{B_T}}$ maps each individual to a set of tuples $\langle C, B\rangle$, where $C \in \mathsf{cl}(D)$ and $B \subseteq B_T$, and $\mathcal{E} : \mathbf{R}_D \mapsto 2^{\mathbf{S}\times\mathbf{S}}$ maps each role in $\mathbf{R}_D$ to a set of pairs of individuals, and there is some individual $s_0 \in \mathbf{S}$ such that $\langle D, \emptyset\rangle \in \mathcal{L}(s_0)$.

For $\langle C, B\rangle \in \mathcal{L}(s)$, we call B the bindings for C, and for $b = y/s' \in B$, we call s' the binding for y in C. For $s \in \mathbf{S}$, the function $\mathcal{L}_C(s)$ returns the set of concepts that are in the label of s without their bindings, i.e., $\mathcal{L}_C(s) = \{C \mid \langle C, B\rangle \in \mathcal{L}(s)\}$. The function $\mathcal{L}_{subst}(s)$ returns the set of concepts with all variables replaced by their bindings, i.e., $\mathcal{L}_{subst}(s) = \{C_{[y_1/s_1,\ldots,y_n/s_n]} \mid \langle C, \{y_1/s_1, \ldots, y_n/s_n\}\rangle \in \mathcal{L}(s)\}$. If $\langle C, B\rangle \in \mathcal{L}(s)$ with $B = \{y_1/s_1, \ldots, y_n/s_n\}$, then we denote with $C_{[B]}$, the result of replacing all occurrences of y_i in C with s_i for $1 \leqslant i \leqslant n$. It is worth mentioning that for a concept C without variables and an individual $s \in \mathbf{S}$, $C \in \mathcal{L}_C(s)$ iff $C \in \mathcal{L}_{subst}(s)$. We assume that the set of bindings for a concept C contains only bindings for variables that occur in C. As a consequence, the set of bindings B for a concept name $C \in N_C$ is always the empty set. Due to the restriction of concepts in query form, it is always the case for a concept D in query form that sub-concepts of D occurring inside number or existential restrictions are variable-free. If we write $\langle C, \{y/s\} \cup B\rangle$, we assume that y occurs as a free variable in C and that B possibly contains bindings for other free variables in C.

For all $s, s' \in \mathbf{S}, C, C_1, C_2 \in \mathsf{cl}(D), r, r' \in \mathbf{R}_D, y \in N_V$, and

$$r^T(s, C) := \{s' \in \mathbf{S} \mid \langle s, s'\rangle \in \mathcal{E}(r) \text{ and } C \in \mathcal{L}_C(s')\},$$

it holds that:

(P1) if $\langle C, \emptyset\rangle \in \mathcal{L}(s)$ and $C \in N_C$, then $\langle \neg C, \emptyset\rangle \notin \mathcal{L}(s)$,
(P2) $\langle \neg y, \{y/s\}\rangle \notin \mathcal{L}(s)$,
(P3) if $\langle C_1 \sqcap C_2, B\rangle \in \mathcal{L}(s)$, then $\langle C_1, B'\rangle \in \mathcal{L}(s)$ and $\langle C_2, B''\rangle \in \mathcal{L}(s)$,
(P4) if $\langle C_1 \sqcup C_2, B\rangle \in \mathcal{L}(s)$, then $\langle C_1, B'\rangle \in \mathcal{L}(s)$ or $\langle C_2, B''\rangle \in \mathcal{L}(s)$,
(P5) if $\langle \forall r.C, B\rangle \in \mathcal{L}(s)$ and $\langle s, s'\rangle \in \mathcal{E}(r)$, then $\langle C, B\rangle \in \mathcal{L}(s')$,
(P6) if $\langle \exists r.C, \emptyset\rangle \in \mathcal{L}(s)$, then there is some $s' \in \mathbf{S}$ such that $\langle s, s'\rangle \in \mathcal{E}(r)$ and $\langle C, \emptyset\rangle \in \mathcal{L}(s')$,
(P7) if $\langle \forall r.C, B\rangle \in \mathcal{L}(s')$ and $\langle s, s'\rangle \in \mathcal{E}(r')$ for some $r' \sqsubseteq^* r$ with $\mathsf{Trans}(r')$, then $\langle \forall r'.C, B\rangle \in \mathcal{L}(s')$,
(P8) if $\langle \geqslant nr.C, \emptyset\rangle \in \mathcal{L}(s)$, then $\sharp(r^T(s, C)) \geqslant n$,
(P9) if $\langle \leqslant nr.C, \emptyset\rangle \in \mathcal{L}(s)$, then $\sharp(r^T(s, C)) \leqslant n$,
(P10) if $\langle \leqslant nr.C, \emptyset\rangle \in \mathcal{L}(s)$ and $\langle s, s'\rangle \in \mathcal{E}(r)$, then $\{\langle C, \emptyset\rangle, \langle \dot{\neg} C, \emptyset\rangle\} \cap \mathcal{L}(s') \neq \emptyset$,
(P11) if $\langle s, s'\rangle \in \mathcal{E}(r)$ and $r \sqsubseteq^* r'$, then $\langle s, s'\rangle \in \mathcal{E}(r')$,
(P12) $\langle s, s'\rangle \in \mathcal{E}(r)$ iff $\langle s', s\rangle \in \mathcal{E}(\mathsf{Inv}(r))$,
(P13) if $\langle o, \emptyset\rangle \in \mathcal{L}(s) \cap \mathcal{L}(s')$ for some $o \in N_I$, then $s = s'$,
(P14) if $o \in N_I$, then there is an $s \in \mathbf{S}$ s.t. $\langle o, \emptyset\rangle \in \mathcal{L}(s)$,
(P15) if $\langle \downarrow y.C, B\rangle \in \mathcal{L}(s)$, then $\langle C, B \cup \{y/s\}\rangle \in \mathcal{L}(s)$ and $\langle y, \{y/s\}\rangle \in \mathcal{L}(s)$,
(P16) if $\langle y, \{y/s'\}\rangle \in \mathcal{L}(s)$, then $s' = s$, and
(P17) for all $\langle C, B\rangle \in \mathcal{L}(s)$, $y/s' \in B$ iff y is free in C.

Property P15 asserts that for every concept $\downarrow y.C$, there is is an instantiated version of C in the label of the individual. Property P16 asserts that an unnegated atomic variable can only occur in the label of one individual, namely the one it is assigned to. All other variable occurrences can only be in negated form.

Lemma 4.2
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form and $\mathcal{R}$ a role hierarchy, then D is satisfiable with respect to $\mathcal{R}$ iff D has a tableau with respect to $\mathcal{R}$.

Proof. For the "if" direction:
If $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ is a tableau for D with respect to $\mathcal{R}$ with $D \in \mathcal{L}(s_0)$, we construct a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for D and $\mathcal{R}$ as follows:

$$\begin{array}{rrl} & \Delta^{\mathcal{I}} = & \mathbf{S} \\ \text{for } A \in N_C: & A^{\mathcal{I}} = & \{s \in \mathbf{S} \mid \langle A, \emptyset \rangle \in \mathcal{L}(s)\} \\ \text{for } s \in \mathbf{S}: & s^{\mathcal{I}} = & \{s\} \\ \text{for } r \in \mathbf{R}_D: & r^{\mathcal{I}} = & \begin{cases} \mathcal{E}'(r) & \text{if } \mathsf{Trans}(r) \\ \mathcal{E}(r) \cup \bigsqcup\limits_{r' \sqsubseteq^* r, r' \neq r} r'^{\mathcal{I}} & \text{otherwise} \end{cases} \end{array}$$

where $\mathcal{E}'(r)$ denotes the transitive closure of $\mathcal{E}(r)$. The interpretation of non-transitive roles is recursive in order to correctly interpret those non-transitive roles that have a transitive sub-role. From the definition of $r^{\mathcal{I}}$ and P7, it follows that, if $\langle s, s' \rangle \in r^{\mathcal{I}}$, then either $\langle s, s' \rangle \in \mathcal{E}(r)$ or there exists a path $\langle s, s_1 \rangle, \langle s_1, s_2 \rangle, \ldots, \langle s_n, s' \rangle \in \mathcal{E}(r')$ for some role r' with $\mathsf{Trans}(r')$ and $r' \sqsubseteq^* r$.

By induction on the structure of concepts, we show that if $\langle C, B \rangle \in \mathcal{L}(s)$ with $B = \{y_1/s_1, \ldots, y_n/s_n\}$, then $s \in C^{\mathcal{I}_{[y_1/s_1,\ldots,y_n/s_n]}}$, where $\mathcal{I}_{[y_1/s_1,\ldots,y_n/s_n]}$ denotes the interpretation obtained by extending $\mathcal{I}$ such that ${y_i}^{\mathcal{I}} = s_i$ for $1 \leqslant i \leqslant n$. This implies then that $D^{\mathcal{I}} \neq \emptyset$, because $\langle D, \emptyset \rangle \in \mathcal{L}(s_0)$ and $D^{\mathcal{I}} = D^{\mathcal{I}_{[B]}}$ for $B = \emptyset$, and hence $s_0 \in D^{\mathcal{I}}$.

Let $\langle C, B \rangle \in \mathcal{L}(s)$:

1. If $C = A \in N_C$ is a concept name, then $s \in C^{\mathcal{I}}$ by definition.
2. If $C = \neg A$ for $A \in N_C$, then $\langle A, \emptyset \rangle \notin \mathcal{L}(s)$ due to P1, so $s \in (\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}) = C^{\mathcal{I}}$.
3. If $C = y$, $y \in N_V$, then $B = \{y/s\}$ by P16 and P17. Thus $s \in C^{\mathcal{I}_B} = y^{\mathcal{I}_{y/s}} = \{s\}$.
4. If $C = \neg y$ for $y \in N_V$, then $B = y/s'$ by P17 and $s \neq s'$ by P2. Hence $s \in C^{\mathcal{I}_{[B]}} = (\neg y)^{\mathcal{I}_{[y/s']}} = \Delta^{\mathcal{I}} \setminus y^{\mathcal{I}_{[y/s']}} = \Delta^{\mathcal{I}} \setminus \{s'\}$.
5. If $C = (C_1 \sqcap C_2)$, then, due to P3, $\langle C_1, B' \rangle \in \mathcal{L}(s)$ and $\langle C_2, B'' \rangle \in \mathcal{L}(s)$, and hence, by induction and by the fact that B' and B'' contain exactly those bindings for the free variables in C_1 and C_2 respectively due to P17, $s \in C_1^{\mathcal{I}_{[B]}}$ and $s \in C_2^{\mathcal{I}_{[B]}}$. Thus, $s \in (C_1 \sqcap C_2)^{\mathcal{I}_{[B]}}$.
6. If $C = (C_1 \sqcup C_2)$, then, due to P4, $\langle C_1, B' \rangle \in \mathcal{L}(s)$ or $\langle C_2, B'' \rangle \in \mathcal{L}(s)$, and hence, by induction and by the fact that B' and B'' contain exactly those bindings for the free variables in C_1 and C_2 respectively due to P17, $s \in C_1^{\mathcal{I}_{[B]}}$ or $s \in C_2^{\mathcal{I}_{[B]}}$. Thus, $s \in (C_1 \sqcup C_2)^{\mathcal{I}_{[B]}}$.

7. If $C = \exists r.E$, $C =\geqslant nr.E$, $C =\leqslant nr.E$, or $C = o$ for $o \in N_I$, then $B = \emptyset$ and C is variable free, since variables can only occur universally quantified in concepts in query form. Hence, the proof is the same as the one for $\mathcal{SHOIQ}$ [8].

8. If $C = \forall r.E$ and $\langle s, s' \rangle \in r^{\mathcal{I}_{[B]}}$, then either

   (a) $\langle s, s' \rangle \in \mathcal{E}(r)$ and $\langle E, B \rangle \in \mathcal{L}(s')$ due to P5, or

   (b) $\langle s, s' \rangle \notin \mathcal{E}(r)$. This can only be the case if r is transitive and there exists a path of length $n \geqslant 1$ such that $\langle s, s_1 \rangle, \langle s_1, s_2 \rangle, \ldots, \langle s_n, s' \rangle \in \mathcal{E}(r)$. Due to P7, $\langle \forall r.E, B \rangle \in \mathcal{L}(s_i)$ for all $1 \leqslant i \leqslant n$, and we have $\langle E, B \rangle \in \mathcal{L}(s')$, again due to P5.

   In both cases, by induction $s' \in E^{\mathcal{I}_{[B]}}$ holds, and hence $s \in (\forall r.E)^{\mathcal{I}_{[B]}} = C^{\mathcal{I}_{[B]}}$.

9. If $C = \downarrow y.E$, then $\langle E, B \cup \{y/s\} \rangle \in \mathcal{L}(s)$ and $\langle y, \{y/s\} \rangle \in \mathcal{L}(s)$ by P15 and by induction $s \in E^{\mathcal{I}_{[B \cup \{y/s\}]}}$ and $s \in y^{\mathcal{I}_{[y/s]}}$ and hence $s \in (\downarrow y.E)^{\mathcal{I}_{[B]}} = C^{\mathcal{I}_{[B]}}$.

For the "only-if" direction:
Given a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for D with respect to $\mathcal{R}$, we can define a tableau $T = (\mathbf{S}, \mathcal{L}, \mathcal{E})$ for D with respect to $\mathcal{R}$ as follows:

$$
\begin{aligned}
\mathbf{S} &= \Delta^{\mathcal{I}} \\
\mathcal{E}(r) &= r^{\mathcal{I}} \\
\mathcal{L}(s) &= \{\langle C, B \rangle \mid\ C \in \mathsf{cl}(D) \\
&\qquad\qquad B = \{y/s' \in B_T \mid y \text{ is free in } C, y/s'' \notin B, \text{ and } s \in C^{\mathcal{I}_{[B]}}\}
\end{aligned}
$$

It remains to demonstrate that T is indeed a tableau for D:

1. P1 holds due to the semantics of $\mathcal{SHOIQ}_{\downarrow}$-concepts and the fact that a concept $C \in N_C$ is variable free.

2. To show a contradiction, assume that $\langle \neg y, \{y/s\} \rangle \in \mathcal{L}(s)$ for some $y \in N_V$ and $s \in \mathbf{S}$. Then $s \in (\neg y)^{\mathcal{I}_{[y/s]}} = \Delta^{\mathcal{I}} \setminus y^{\mathcal{I}_{[y/s]}} = \Delta^{\mathcal{I}} \setminus \{s\}$, which is a contradiction. Hence, $\langle y, \{\neg y/s\} \rangle \notin \mathcal{L}(s)$ and P2 holds.

3. If $\langle C_1 \sqcap C_2, B \rangle \in \mathcal{L}(s)$ for $(C_1 \sqcap C_2) \in \mathsf{cl}(D)$, then $\{C_1, C_2\} \subseteq \mathsf{cl}(D)$ and P3 holds, since $s \in (C_1 \sqcap C_2)^{\mathcal{I}_{[B]}}$ implies that $s \in C_1^{\mathcal{I}_{[B]}}$ and $s \in C_2^{\mathcal{I}_{[B]}}$ by the definition of the semantics.

4. P4, P5, and P6 hold as well by definition of the semantics.

5. For P7, if $s \in (\forall r.C)^{\mathcal{I}}$ and $\langle s, s' \rangle \in r'^{\mathcal{I}}$ with $\mathsf{Trans}(r')$ and $r' \sqsubseteq r$, then $s' \in (\forall r'.C)^{\mathcal{I}}$ if s' has no r'-successor or only r'-successor that are in $\mathrm{C}^{\mathcal{I}}$. The individual s cannot be in $(\mathrm{r'.C})^{\mathcal{I}}$ if there is some s'' such that $\langle s', s'' \rangle \in r'^{\mathcal{I}}$ and $s'' \notin C^{\mathcal{I}}$. In this case, since $\langle s, s' \rangle, \langle s', s'' \rangle \in r'^{\mathcal{I}}$, and $\mathsf{Trans}(r')$, it holds that $\langle s, s'' \rangle \in r'^{\mathcal{I}}$. Hence $\langle s, s'' \rangle \in r^{\mathcal{I}}$ and $s \notin \forall r.C^{\mathcal{I}}$ in contradiction to the assumption. Therefore, T satisfies P7.

6. P8, P9, and P10 hold as a consequence of the definition of the semantics of $\mathcal{SHOIQ}_{\downarrow}$-concepts and the fact that variables only occur universally quantified and are not allowed to appear inside an existential restriction or in number restrictions.

7. P11 is satisfied because $\mathcal{I} \models \mathcal{R}$ and set inclusion is a transitive property.

8. P12 and P13 are satisfied by the definition of the semantics.

9. For P15: If $\langle \downarrow y.C, B\rangle \in \mathcal{L}(s)$, then $\{C, y\} \subseteq \mathsf{cl}(D)$, $s \in C^{\mathcal{I}[B \cup \{y/s\}]}$, and hence $\langle C, \{y/s\} \cup B\rangle \in \mathcal{L}(s)$. Since $s \in y^{\mathcal{I}[y/s]} = \{s\}$, we have that $\langle y, \{y/s\}\rangle \in \mathcal{L}(s)$ as required.

10. We show P16 by contradiction. Assume that $\langle y, \{y/s'\}\rangle \in \mathcal{L}(s)$ and $s \neq s'$. Then, by definition of the labels, $s \in Y^{\mathcal{I}[y/s']} = \{s'\}$, which clearly is a contradiction for $s \neq s'$. Hence, P16 holds.

11. P17 holds by definition of the labels. In particular by the constraint that no variable $y \in NV$ can occur in the set of bindings for a concept C twice.

□

5 A Tableaux Algorithm for $\mathcal{SHOIQ}_{\downarrow}$-Concepts in Query Form

From Lemma 4.2, an algorithm which constructs a tableau for a $\mathcal{SHOIQ}_{\downarrow}$-concept D in query form can be used as a decision procedure for the satisfiability of D with respect to a role hierarchy $\mathcal{R}$. Such an algorithm will now be described in detail.

Definition 5.1 (Tableaux Algorithm)
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form, $\mathcal{R}$ a role hierarchy, and $\mathbf{R}_D$ the set of role names occurring in D and $\mathcal{R}$. A *completion graph* for D w.r.t. $\mathcal{R}$ is a directed graph $\mathbf{G} = (V, E, \mathcal{L}, \neq)$. Each node (vertex) $v \in V$ is labelled with a set $\mathcal{L}(v)$ of tuples $\langle C, B\rangle$, where $C \in \mathsf{cl}(D)$ and $B \subseteq \{y/v' \mid y \in N_V \text{ and } v' \in V\}$. We call B the *bindings* for C, and for $b = y/v' \in B$, we call v' the *binding* for y in C. Each edge $\langle v, v'\rangle \in E$ is labelled with a set of role names $\mathcal{L}(\langle v, v'\rangle) \subseteq \mathbf{R}_D$. The symmetric binary relation $\neq$ is used to keep track of inequalities between nodes of the graph $\mathbf{G}$.

For $v \in V$, the function $\mathcal{L}_C(v)$ returns the set of concepts that are in the label of v without their bindings, i.e., $\mathcal{L}_C(v) = \{C \mid \langle C, B\rangle \in \mathcal{L}(v)\}$. The function $\mathcal{L}_{subst}(v)$ returns the set of concepts with all variables replaced by their bindings, i.e., $\mathcal{L}_{subst}(v) = \{C_{[y_1/v_1,\ldots,y_n/v_n]} \mid \langle C, \{y_1/v_1, \ldots, y_n/v_n\}\rangle \in \mathcal{L}(v)\}$. If $\langle C, B\rangle \in \mathcal{L}(v)$ with $B = \{y_1/v_1, \ldots, y_n/v_n\}$, then we denote with $C_{[B]}$, the result of replacing all occurrences of y_i in C with v_i for $1 \leqslant i \leqslant n$. It is worth mentioning that for a concept C without variables and a vertex $v \in V$, $C \in \mathcal{L}_C(v)$ iff $C \in \mathcal{L}_{subst}(v)$.

In the following, we often use $r \in \mathcal{L}(\langle v_1, v_2\rangle)$ as an abbreviation for $\langle v_1, v_2\rangle \in E$ and $r \in \mathcal{L}(\langle v_1, v_2\rangle)$. If $\langle v_1, v_2\rangle \in E$, then v_2 is called a *successor* of v_1 and v_1 is called a *predecessor* of v_2. *Ancestor* is the transitive closure of predecessor, and *descendant* is the transitive closure of successor. A node v_2 is called an *r-successor* of a node v_1 if, for some role r' with $r' \sqsubseteq^{*} r$, $r' \in \mathcal{L}(\langle v_1, v_2\rangle)$.

For a role r and a node $v \in V$, we define the set of v's r-successors with C in their label, $r^{\mathbf{G}}$, as follows:

$$r^{\mathbf{G}}(v, C) := \{v' \mid v' \text{ is an } r\text{-successor of } v \text{ and } C \in \mathcal{L}(v')\}.$$

A completion graph $\mathbf{G} = (V, E, \mathcal{L}, \neq)$ is said to contain a *clash* if

1. for some concept name $A \in N_C$ and vertex $v \in V$, $\{A, \neg A\} \subseteq \mathcal{L}_C(v)$, or
2. for some role r and vertex $v \in V$, $\leqslant nr.C \in \mathcal{L}_C(v)$ and there are $n+1$ r-successor s $v_0, \ldots, v_n$ of v with $C \in \mathcal{L}_C(v_i)$ for each $0 \leqslant i \leqslant n$ and $v_i \neq v_j$ for each $0 \leqslant i < j \leqslant n$, or
3. for some $o \in N_I$, there are vertices $v \neq v'$ with $o \in (\mathcal{L}_C(v) \cap \mathcal{L}_C(v'))$, or
4. for $y \in N_V$, there is a vertex $v \in V$ with $\{v, \neg v\} \subseteq \mathcal{L}_{subst}(v)$.

If $o_1, \ldots, o_\ell$ are all the nominals occurring in D, then the tableaux algorithm starts with the completion graph $\mathbf{G} = (\{r_0, r_1, \ldots, r_\ell\}, \emptyset, \mathcal{L}, \emptyset)$ with $\mathcal{L}(r_0) = \{\langle D, \emptyset\rangle\}$ and $\mathcal{L}(r_i) = \{\langle o_i, \emptyset\rangle\}$ for $1 \leqslant i \leqslant \ell$. $\mathbf{G}$ is then expanded by repeatedly applying the expansion rules given in Table 5, stopping if a clash occurs.

The $\downarrow$-rule in Table 5 extends the expansion rules for $\mathcal{SHOIQ}$. Since the labels of the vertices are now tuples, the $\mathcal{L}_C$ and $\mathcal{L}_{subst}$ functions are used in some of the rules. We distinguish two types of nodes in $\mathbf{G}$: nominal nodes and blockable nodes. A node v is a nominal node if $\mathcal{L}(v)$ contains a nominal and v is a blockable node otherwise.

A vertex $v_d \in V$ is *directly blocked* in $\mathbf{G}$ if it has ancestors v'_d, v_b, and v'_b such that

1. v_d is a successor of v'_d and v_b is a successor of v'_b,
2. v_b, v_d and all nodes on the path from v_b to v_d are blockable,
3. $\mathcal{L}_C(v_d) = \mathcal{L}_C(v_b)$ and $\mathcal{L}_C(v'_d) = \mathcal{L}_C(v'_b)$, and
4. $\mathcal{L}(\langle v'_d, v_d\rangle) = \mathcal{L}(\langle v'_b, v_b\rangle)$.

In this case we say that v_b blocks v_d. A node is *blocked* if is directly blocked or if its predecessor is blocked.

The only difference to the blocking for $\mathcal{SHOIQ}$ is that the function $\mathcal{L}_C$, which does not consider the bindings, is used for comparing the labels of the vertices in $\mathcal{SHOIQ}_\downarrow$.

During the expansion it is sometimes necessary to merge two nodes or to delete (prune) a part of the graph. Since the node labels are now tuples, we adapt the according definitions from $\mathcal{SHOIQ}$. When a node v_r (the subscript r accounts for the fact that the node is to be removed) is merged into a node v_m (the subscript m indicates the node into which v_r is merged) by an application of the $\leqslant$-rule, we "prune" the completion graph by removing v_r and, recursively, all blockable successors of v_r to prevent a further rule application on this nodes. More precisely, pruning a node v_r (written $\mathsf{Prune}(v_r)$) in $\mathbf{G} = (V, E, \mathcal{L}, \not\approx)$ yields a graph that is obtained from $\mathbf{G}$ as follows:

1. for all successors v of v_r, remove $\langle v_r, v\rangle$ from E and, if v is blockable, $\mathsf{Prune}(v)$;
2. remove v_r from V.

Intuitively, when we merge a node v_r, we add $\mathcal{L}(v_r)$ to $\mathcal{L}(v_m)$, "move" all the edges leading to v_r so that they lead to v_m and "move" all the edges leading from v_r to nominal nodes so that they lead from v_m to the same nominal nodes; we then remove v_r (and blockable sub-trees below v_r) from the completion graph. More precisely, merging a node v_r into a node v_m (written $\mathsf{Merge}(v_r, v_m)$) in $\mathbf{G} = (V, E, \mathcal{L}, \not\approx)$ yields a graph that is obtained from $\mathbf{G}$ as follows:

1. for all nodes v such that $\langle v, v_r\rangle \in E$
   (a) if $\langle v, v_m\rangle \notin E$, then $E = E \cup \{\langle v, v_m\rangle\}$
   (b) $\mathcal{L}(\langle v, v_m\rangle) = \mathcal{L}(\langle v, v_m\rangle) \cup \mathcal{L}(\langle v, v_r\rangle)$,
   (c) remove $\langle v, v_r\rangle$ from E;
2. for all nominal nodes v such that $\langle v_r, v\rangle \in E$
   (a) if $\langle v_m, v\rangle \notin E$, then $E = E \cup \langle v_m, v\rangle$
   (b) $\mathcal{L}(\langle v_m, v\rangle) = \mathcal{L}(\langle v_m, v\rangle) \cup \mathcal{L}(\langle v_r, v\rangle)$,
   (c) remove $\langle v_r, v\rangle$ from E;
3. $\mathcal{L}(v_m) = \mathcal{L}(v_m) \cup \mathcal{L}(v_r)$
4. add $v_m \neq\!\!\!\!\cdot\;\; v$ for all v such that $v_r \neq\!\!\!\!\cdot\;\; v$; and
5. $\mathsf{Prune}(v_r)$.

$\sqcap$-rule	if 1. $\langle C_1 \sqcap C_2, B\rangle \in \mathcal{L}(v)$, v is not indirectly blocked, and 2. $\{\langle C_1, B_1\rangle, \langle C_2, B_2\rangle\} \not\subseteq \mathcal{L}(v)$, where $B_1 = \{y/v' \mid y \text{ is free in } C_1\}$ and $B_2 = \{y/v' \mid y \text{ is free in } C_2\}$ then $\mathcal{L}(v) = \mathcal{L}(v) \cup \{\langle C_1, B_1\rangle, \langle C_2, B_2\rangle\}$.
$\sqcup$-rule	if 1. $\langle C_1 \sqcup C_2, B\rangle \in \mathcal{L}(v)$, v is not indirectly blocked, and 2. $\{\langle C_1, B_1\rangle, \langle C_2, B_2\rangle\} \cap \mathcal{L}(v) = \emptyset$, where $B_1 = \{y/v' \mid y \text{ is free in } C_1\}$ and $B_2 = \{y/v' \mid y \text{ is free in } C_2\}$ then $\mathcal{L}(v) = \mathcal{L}(v) \cup \{\langle C_1, B_1\rangle\}$ or $\mathcal{L}(v) = \mathcal{L}(v) \cup \{\langle C_2, B_2\rangle\}$
$\exists$-rule	if 1. $\langle \exists r.C, \emptyset\rangle \in \mathcal{L}(v)$, v is not blocked, and 2. n has no safe r-neighbour v' with $\langle C, \emptyset\rangle \in \mathcal{L}(v')$, then create a new node v' and an edge $\langle v, v'\rangle$ with $\mathcal{L}(v') = \{\langle C, \emptyset\rangle\}$ and $\mathcal{L}(\langle v, v'\rangle) = \{r\}$.
$\forall$-rule	if 1. $\langle \forall r.C, B\rangle \in \mathcal{L}(v)$, v is not indirectly blocked, and 2. there is an r-neighbour v' of v with $\langle C, B\rangle \notin \mathcal{L}(v')$, then $\mathcal{L}(v') = \mathcal{L}(v') \cup \{\langle C, B\rangle\}$.
$\forall_+$-rule	if 1. $\langle \forall r.C, B\rangle \in \mathcal{L}(v)$, v is not indirectly blocked, and 2. there is some r' with $\mathsf{Trans}(r')$ and $r' \sqsubseteq^* r$ 3. there is an r'-neighbour v' of v with $\langle \forall r'.C, B\rangle \notin \mathcal{L}(v')$ then $\mathcal{L}(v') = \mathcal{L}(v') \cup \{\langle \forall r'.C, B\rangle\}$.
choose-rule	if 1. $\langle \leqslant nr.C, \emptyset\rangle \in \mathcal{L}(v)$, v is not indirectly blocked, and 2. there is an r-neighbour v' of v with $\{\langle C, \emptyset\rangle, \langle \dot{\neg}C, \emptyset\rangle\} \cap \mathcal{L}(v') = \emptyset$ then $\mathcal{L}(v') = \mathcal{L}(v') \cup \{\langle E, \emptyset\rangle\}$ for some $E \in \{C, \dot{\neg}C\}$
$\geqslant$-rule	if 1. $\langle \geqslant nr.C, \emptyset\rangle \in \mathcal{L}(v)$, v is not blocked, and 2. there are not n safe r-neighbours $v_1, \ldots, v_n$ of v with $\langle C, \emptyset\rangle \in \mathcal{L}(v_i)$ and $v_i \neq\!\!\!\!\cdot\;\; v_j$ for $1 \leqslant i < j \leqslant n$ then create n new nodes $v_1, \ldots, v_n$ with $\mathcal{L}(\langle v, v_i\rangle) = \{r\}$, $\mathcal{L}(v_i) = \{\langle C, \emptyset\rangle\}$ and $v_i \neq\!\!\!\!\cdot\;\; v_j$ for $1 \leqslant i < j \leqslant n$.
$\leqslant$-rule	if 1. $\langle \leqslant nr.C, \emptyset\rangle \in \mathcal{L}(v)$, v is not indirectly blocked, 2. $\sharp(r^{\mathbf{G}}(v, C)) > n$ and there are two r-neighbours v_1, v_2 of v with $\langle C, \emptyset\rangle \in (\mathcal{L}(v_1) \cap \mathcal{L}(v_2))$ and not $v_1 \neq\!\!\!\!\cdot\;\; v_2$, then a. if v_1 is a nominal node, then $\mathsf{Merge}(v_2, v_1)$ b. else if v_2 is a nominal node or an ancestor of v_1, then $\mathsf{Merge}(v_1, v_2)$ c. else $\mathsf{Merge}(v_2, v_1)$.

Table 1: The expansion rules for $\mathcal{SHOIQ}_{\downarrow}$ (continued next page).

Table 1– continued from previous page

o-rule	if	for some $o \in N_I$, there are two nodes v, v' with $\langle o, \emptyset\rangle \in (\mathcal{L}(v) \cap \mathcal{L}(v'))$ and not $v \neq\!\!\!\!\!= v'$
	then	$\mathsf{Merge}(v, v')$.
NN-rule	if	1. $\leqslant nr.C \in \mathcal{L}_C(v)$, v is a nominal node, and there is a blockable r-neighbour v' of v such that $C \in \mathcal{L}_C(v')$ and v is a successor of v',
		2. there is no m such that $1 \leqslant m \leqslant n$, $(\leqslant mr.C) \in \mathcal{L}_C(v)$, and there exist m nominal r-neighbours $v_1, \ldots, v_m$ of v with $C \in \mathcal{L}_C(v_i)$ and $v_i \neq\!\!\!\!\!= v_j$ for all $1 \leqslant i < j \leqslant m$.
	then	1. guess m with $1 \leqslant m \leqslant n$ and $\mathcal{L}(v) \rightarrow \mathcal{L}(v) \cup \{\langle \leqslant mr.C, \emptyset\rangle\}$
		2. create m new nodes $v'_1, \ldots, v'_m$ with $\mathcal{L}(\langle v, v'_i\rangle) = \{r\}$, $\mathcal{L}(v'_i) = \{\langle C, \emptyset\rangle, \langle o_i, \emptyset\rangle\}$ and each $o_i \in N_I$ new in $\mathbf{G}$, and $v'_i \neq\!\!\!\!\!= v'_j$ for $1 \leqslant i < j \leqslant m$.
$\downarrow$-rule	if	1. $\langle \downarrow y.C, B\rangle \in \mathcal{L}(v)$, v is not indirectly blocked, and
		2. $\{\langle y, \{y/v\}\rangle, \langle C, \{y/v\} \cup B\rangle\} \not\subseteq \mathcal{L}(v)$,
	then	$\mathcal{L}(v) = \mathcal{L}(v) \cup \{\langle y, \{y/v\}\rangle, \langle C, B \cup \{y/v\}\rangle\}$.

Table 1: The expansion rules for $\mathcal{SHOIQ}_{\downarrow}$.

6 Proof of the Algorithm's Correctness and Termination

In order to obtain a decision procedure, we have to prove that the algorithm is sound, complete, and terminating. We start with the termination proof and continue with the proofs for soundness and completeness.

Lemma 6.1
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form and $\mathcal{R}$ a role hierarchy, then the $\mathcal{SHOIQ}_{\downarrow}$ tableaux algorithm terminates, given D and $\mathcal{R}$ as input.

Proof. Since the blocking conditions consider the non-instantiated concepts only, the proof of termination works as for $\mathcal{SHOIQ}$. □

The next step towards showing that the algorithm as presented in Section 5 is a decision procedure for a $\mathcal{SHOIQ}_{\downarrow}$ concept D in query form with respect to a role hierarchy $\mathcal{R}$ is to prove the soundness of the algorithm.

Lemma 6.2
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form and $\mathcal{R}$ a role hierarchy. If the expansion rules can be applied to D and $\mathcal{R}$ such that they yield a complete and clash-free completion graph, then D has a tableau with respect to $\mathcal{R}$.

Proof. We can obtain a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ from a complete and clash-free completion graph $\mathbf{G} = (V, E, \mathcal{L}, \neq\!\!\!\!\!=)$ by unravelling blockable "tree" parts of the graph and by extending the labels with respect to binder concepts accordingly. As a result and in difference to tableau for $\mathcal{SHOIQ}$, there might be no bound on the size of the labels in a tableau, which is, however, not a problem. We define a trace as follows:
For a directly blocked node v, let $b(v)$ denote a node that blocks v. A *trace* in T is a sequence of pairs of blockable nodes of $\mathbf{G}$ of the form $t = \langle \frac{v_0}{v'_0}, \ldots, \frac{v_n}{v'_n}\rangle$. For such a trace, we define $\mathsf{Tail}(t) := v_n$ and $\mathsf{Tail}'(t) := v'_n$. With $\langle t \mid \frac{v_{n+1}}{v'_{n+1}}\rangle$ we

denote the trace $\langle \frac{v_0}{v'_0}, \ldots, \frac{v_n}{v'_n}, \frac{v_{n+1}}{v'_{n+1}} \rangle$. The set $\mathsf{Traces}(\mathbf{G})$ is defined inductively as follows:

- For each blockable node v of $\mathbf{G}$ that is a successor of a nominal node or a root node, $\langle \frac{v}{v} \rangle \in \mathsf{Traces}(\mathbf{G})$, and
- For a trace $t \in \mathsf{Traces}(\mathbf{G})$ and a blockable node v' in $\mathbf{G}$:
  - if v' is a successor of $\mathsf{Tail}(t)$ and v' is not blocked, then $\langle t \mid \frac{v'}{v'} \rangle \in \mathsf{Traces}(\mathbf{G})$, and
  - if v' is a successor of $\mathsf{Tail}(t)$ and v' is blocked, then $\langle t \mid \frac{b(v')}{v'} \rangle \in \mathsf{Traces}(\mathbf{G})$.

Due to the construction of $\mathsf{Traces}(\mathbf{G})$, all nodes occurring in a trace are blockable and, for $t \in \mathsf{Traces}(\mathbf{G})$ with $t = \langle t \mid \frac{v}{v'} \rangle$, v is not blocked, and v' is blocked iff $v \neq v'$. Furthermore, the blocking condition implies $\mathcal{L}_C(v) = \mathcal{L}_C(v')$.

We further on extend the function tail to be the identity on nominal nodes, i.e., $v \in \mathsf{Nom}(\mathbf{G})$, $\mathsf{Tail}(v) = v$. This makes that definition of the labels more convenient.

For concepts in a blocked node, we may have to make non-deterministic choices during the unravelling. For a trace t s.t. t contains free variables, we use the equivalent concept modulo bindings in $\mathsf{Tail}(t)$ to steer the non-deterministic decisions.

We now use $\mathsf{Nom}(\mathbf{G})$ for the set of nominal nodes in $\mathbf{G}$ and define a tableau $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ from $\mathbf{G}$ as follows:

$$
\begin{aligned}
\mathbf{S} &= \mathsf{Nom}(\mathbf{G}) \cup \mathsf{Traces}(\mathbf{G}) \\
\mathcal{L}'(s) &= \{\langle C, B \rangle \mid \langle C, B' \rangle \in \mathcal{L}(\mathsf{Tail}(s)) \text{ and } y/t \in B \text{ iff } y/\mathsf{Tail}(t) \in B'\} \\
\mathcal{E}(r) &= \{\langle t, t' \rangle \in \mathsf{Traces}(\mathbf{G}) \times \mathsf{Traces}(\mathbf{G}) \mid \\
&\qquad t' = \langle t \mid \tfrac{v}{v'} \rangle \text{ and } v' \text{ is an } r\text{-successor of } \mathsf{Tail}(t) \text{ or} \\
&\qquad t = \langle t' \mid \tfrac{v}{v'} \rangle \text{ and } v' \text{ is an } \mathsf{Inv}(r)\text{-successor of } \mathsf{Tail}(t')\} \cup \\
&\quad \{\langle t, v \rangle \in \mathsf{Traces}(\mathbf{G}) \times \mathsf{Nom}(\mathbf{G}) \mid v \text{ is an } r\text{-neighbour of } \mathsf{Tail}(t)\} \cup \\
&\quad \{\langle v, t \rangle \in \mathsf{Nom}(\mathbf{G}) \times \mathsf{Traces}(\mathbf{G}) \mid \mathsf{Tail}(t) \text{ is an } r\text{-neighbour of } v\} \cup \\
&\quad \{\langle v, v' \rangle \in \mathsf{Nom}(\mathbf{G}) \times \mathsf{Nom}(\mathbf{G}) \mid v' \text{ is an } r\text{-neighbour of } v\}
\end{aligned}
$$

We show that T is indeed a tableau for D with respect to $\mathcal{R}$ by showing that T satisfies all of the properties from Definition 4:

- P1 holds by the definition of $\mathcal{L}'$ and the fact that C is variable free since $C \in N_C$ and $\mathbf{G}$ is complete and clash-free.
- P2 holds by the definition of $\mathcal{L}'$ and the fact that $\mathbf{G}$ is clash-free. In particular, if $\{\langle y, \{y/t\}\rangle, \langle \neg y, \{y/t\}\rangle\} \subseteq \mathcal{L}'(t)$, then by definition of $\mathcal{L}'$, $\{\langle y, \{y/v\}\rangle, \langle \neg y, \{y/v\}\rangle\} \subseteq \mathcal{L}(v)$ for $v = \mathsf{Tail}(t)$, which clearly contradicts the fact that $\mathbf{G}$ is clash-free.
- As a consequence of the definition of concepts in query form, conjunctions occur only in concepts without free variables. Hence property P3 is satisfied because of the definition of $\mathcal{L}'$ and because $\mathbf{G}$ is complete.
- If $\langle C_1 \sqcup C_2, B \rangle \in \mathcal{L}'(t)$, we can, due to P17, partition B into B_1 and B_2 such that $\{y/s\} \in B_i$ iff y is a free variable in B_i for $i \in \{1, 2\}$. Then

by definition of $\mathcal{L}'$, $\langle C_1 \sqcup C_2, B'_1 \cup B'_2\rangle \in \mathcal{L}(\mathsf{Tail}(t))$ for $y/\mathsf{Tail}(s) \in B'_i$ iff $y/s \in B_i$ for $i \in \{1,2\}$. Completeness of $\mathbf{G}$ implies now that $\langle C_i, B'_i\rangle \in \mathcal{L}(\mathsf{Tail}(t))$ for $i \in \{1,2\}$. Since $y/\mathsf{Tail}(s) \in B'_i$ iff $y/s \in B_i$ and by definition of $\mathcal{L}'$, we have that $\langle C_i, B_i\rangle \in \mathcal{L}(\mathsf{Tail}(t))$ as required by P4.

- Property P5 holds since $\mathbf{G}$ is complete and by the definition of $\mathcal{L}'$.
- Since all concepts are in query form, all existentially quantified concepts are free of state variables and the set of bindings is empty. Hence the proof given for $\mathcal{SHOIQ}$ suffices.
- Due to the definition of concepts in query form, only simple roles can be used in quantifiers over state variables. Hence, property P7 applies only to tuples $\langle \forall r.C, B\rangle$ in which $B = \emptyset$ and again the proof given for $\mathcal{SHOIQ}$ suffices.
- Since the input concept is in query form, also all concepts that occur in the scope of a number restriction are free of state variables and the set of bindings is empty. Therefore, for P8, P9, and P10, we again have that if $\langle \bowtie r.C, \emptyset\rangle \in \mathcal{L}'(t)$, for $\bowtie$ a placeholder for $\leqslant$ or $\geqslant$, the standard proof for given $\mathcal{SHOIQ}$ is sufficient.
- Property P11 is satisfied due to the definition of an r-successor that, due to the use of $\overset{*}{\sqsubseteq}$, takes into account the role hierarchy.
- Property P13 holds since $\mathbf{G}$ is complete and clash-free.
- Property P14 holds due to the following arguments:
  - $\mathbf{G}$ is initialised such that for each nominal o_i for $1 \leqslant i \leqslant \ell$ there is a node $r_i \in V$ with $\langle o_i, \emptyset\rangle \in \mathcal{L}(r_i)$.
  - Pruning removes only blockable nodes and none of the nodes r_i is blockable.
  - If a nominal node r_j with $\langle o_j, \emptyset\rangle \in \mathcal{L}(r_j)$ was merged into another nominal node r_i with $\langle o_i, \emptyset\rangle \in \mathcal{L}(r_i)$, then the label of r_i after the merging is a union of the labels of the two nodes. Hence we have that $\langle o_i, \emptyset\rangle, \langle o_j, \emptyset\rangle \in \mathcal{L}'(r_i)$ in T.
- If $\langle \downarrow y.C, B\rangle \in \mathcal{L}'(s)$, then $\langle \downarrow y.C, B'\rangle \in \mathcal{L}(\mathsf{Tail}(s))$ and completeness implies that $\{\langle C, B' \cup \{y/\mathsf{Tail}(s)\}\rangle, \langle y, \{y/\mathsf{Tail}(s)\}\rangle\} \in \mathcal{L}(\mathsf{Tail}(s))$. Hence, by definition of $\mathcal{L}'$, also $\{\langle C, B \cup \{y/s\}\rangle, \langle y, \{y/s\}\rangle\} \in \mathcal{L}'(s)$ as required by property P15.
- The only way a non-negated variable can be added to the label of an individual $s \in \mathbf{S}$ is if $\langle \downarrow y.C, B\rangle \in \mathcal{L}'(s)$. Let now $v = \mathsf{Tail}(s)$. Even if a node v' is merged into a node v in $\mathbf{G}$ such that $\langle y, \{y/v'\}\rangle \in \mathcal{L}(v)$, then also $\langle \downarrow y.C, B'\rangle \in \mathcal{L}(v)$ and by the completeness of $\mathbf{G}$ we have that $\langle y, \{y/v\}\rangle \in \mathcal{L}(v)$. By definition of $\mathcal{L}'$, we then have that $\langle y, \{y/s\}\rangle \in \mathcal{L}'(s)$ iff $\langle y, \{y/\mathsf{Tail}(s)\}\rangle \in \mathcal{L}(\mathsf{Tail}(s)) = \mathcal{L}(v)$. Since the removed node v' can never be equal to $\mathsf{Tail}(s)$ for an individual $s \in \mathbf{S}$, property P16 holds.
- Property P17 holds due to the definition of concepts in query form and of $\mathcal{L}'$.

□

Finally, we now show the completeness of the algorithm, by showing that given a tableau for a $\mathcal{SHOIQ}_{\downarrow}$ concept D in query form with respect to a role hierarchy $\mathcal{R}$, we can use this tableau to steer the application of the non-deterministic rules. This technique is often used for showing the completeness of a tableaux algorithm for concept satisfiability [10, 8, 21].

Lemma 6.3
Let D be a $\mathcal{SHOIQ}_{\downarrow}$-concept in query form and $\mathcal{R}$ a role hierarchy. If D has a tableau with respect to $\mathcal{R}$, then the expansion rules can be applied to D and $\mathcal{R}$ such that they yield a complete and clash-free completion graph.

Proof. Let $T = (\mathbf{S}, \mathcal{L}', \mathcal{E})$ be a tableau for D with respect to $\mathcal{R}$. We use this tableau to guide the application of the non-deterministic rules in order to create a complete and clash-free completion graph $\mathbf{G}$.

If $o_1, \ldots, o_\ell$ are all the nominals occurring in D, then we start with an initial completion graph $\mathbf{G} = (\{r_0, r_1, \ldots, r_\ell\}, \emptyset, \mathcal{L}, \emptyset)$, where with $\mathcal{L}(r_0) = \{\langle D, \emptyset\rangle\}$ and $\mathcal{L}(r_i) = \{\langle o_i, \emptyset\rangle\}$ for $1 \leqslant i \leqslant \ell$.

We now define a mapping $\pi : V \to \mathbf{S}$ that is inductively extended with each application of a generating rule. Initially, we define $\pi(r_0) = s_0$, for some $s_0 \in \mathbf{S}$ with $\langle D, \emptyset\rangle \in \mathcal{L}'(s_0)$ and we define $\pi(r_i) = s_i$ for $1 \leqslant i \leqslant \ell$ such that $\langle o_i, \emptyset\rangle \in \mathcal{L}'(s_i)$. Clearly the individual s_0 exists, since T is a tableau for D with respect to $\mathcal{R}$. The nodes s_i for $1 \leqslant i \leqslant \ell$ exist due to P14.

We require that π satisfies the following conditions for each node $v \in V$ and each edge $\langle v, v'\rangle \in E$:

$$\left.\begin{array}{ll} 1. & \text{if } \langle D, \{y_0/v_0, \ldots, y_n/v_n\}\rangle \in \mathcal{L}(v), \text{ then} \\ & \langle D, \{y_0/\pi(v_0), \ldots, y_n/\pi(v_n)\}\rangle \in \mathcal{L}'(\pi(v)), \\ 2. & \text{if } v' \text{ is an } r\text{-neighbour of } v, \text{ then } \langle \pi(v), \pi(v')\rangle \in \mathcal{E}(r) \text{ and} \\ 3. & v \neq v' \text{ implies } \pi(v) \neq \pi(v') \end{array}\right\} (*)$$

It is easy to see that the initial mapping π satisfies $(*)$ since the set of bindings is initially empty and obviously, no completion graph that satisfies $(*)$ contains a clash as this would contradict one of P1, P2, P9, or P13.

Claim 6.4
Let $\mathbf{G}$ be a completion graph and π a function that satisfies $(*)$. If a rule is applicable to $\mathbf{G}$, then the rule is applicable to $\mathbf{G}$ in a way that yields a completion graph $\mathbf{G}'$ and an extension of π that also satisfies $(*)$.

We focus here only on the $\forall$-, $\sqcup$-, and $\downarrow$-rules, since this are the only rules that can be applied to concepts where the set of bindings is non-empty. For all other rules an extension of the known proof for $\mathcal{SHOIQ}$ is straightforward, since only the labels have to be changed to contain tuples with an empty set of bindings.

- If the $\forall$-rule is applicable to a node $v \in V$, i.e., $D = \langle \forall r.E, B\rangle \in \mathcal{L}(v)$ and v' is an r-successor of v such that $\langle E, B\rangle \notin \mathcal{L}(v')$, then $\langle \forall r.E, B'\rangle \in \mathcal{L}'(\pi(v))$ due to constraint 1 of $(*)$ and $\langle E, B'\rangle \in \mathcal{L}'(\pi(v'))$ due to P5. The $\forall$-rule adds $\langle E, B\rangle$ to $\mathcal{L}(v')$ and it is easy to see that $(*)$ is preserved.

- Assume that the $\sqcup$-rule is applicable to a node $v \in V$ for a concept $D = \langle C_1 \sqcup C_2, B\rangle \in \mathcal{L}(v)$, with $B = B_1 \cup B_2$ and $B_1 = \{y_0/v_0, \ldots, y_n/v_n\}$ is chosen such that $y_0, \ldots, y_n$ are exactly the free variables in C_1 and where $B_2 = \{y_{n+1}/v_{n+1}, \ldots, y_m/v_m\}$ is chosen such that $y_{n+1}, \ldots, y_m$ are exactly the free variables in C_2. Due to constraint 1 of $(*)$ we have that $\langle C_1 \sqcup C_2, B'\rangle \in \mathcal{L}'(\pi(v))$ such that B can be partitioned into B'_1 and B'_2 with $B'_1 = \{y_0/\pi(v_0), \ldots, y_n/\pi(v_n)\}$ and $B'_2 = \{y_{n+1}/\pi(v_{n+1}), \ldots, y_m/\pi(v_m)\}$. Due to property P4, we know that $\{\langle C_1, B'_1\rangle, \langle C_2, B'_2\rangle\} \cap \mathcal{L}'(\pi(v)) \neq \emptyset$. Hence, the $\sqcup$-rule can add $E \in \{\langle C_1, B_1\rangle, \langle C_2, B_2\rangle\}$ to $\mathcal{L}(v)$ such that $(*)$ is preserved.

- If the $\downarrow$-rule is applicable to a node $v \in V$, i.e., $D = \langle \downarrow y.C, B\rangle \in \mathcal{L}(v)$ and $\{\langle y, \{y/v\}\rangle, \langle C, \{y/v\} \cup B\rangle\} \not\subseteq \mathcal{L}(v)$, then $\langle \downarrow y.C, B'\rangle \in \mathcal{L}'(\pi(v))$ due to constraint 1 of $(*)$ and $\{\langle y, \{y/\pi(v)\}\rangle, \langle C, \{y/\pi(v)\} \cup B'\rangle\} \subseteq \mathcal{L}'(\pi(v))$ due to P15. The $\downarrow$-rule adds $\langle y, \{y/v\}\rangle$ and $\langle C, \{y/v\} \cup B\rangle$ to $\mathcal{L}(v)$ and it is easy to see that $(*)$ is preserved.

Since every rule application preserves $(*)$ and since the tableaux algorithm given in Definition 5.1 terminates, it is the case that any sequence of rule applications must terminate and result in a complete and clash-free completion graph. □

Since it is possible to internalise general TBoxes and ABoxes in $\mathcal{SHOIQ}_{\downarrow}$, the following theorem is now a simple consequence of Lemma 6.1, Lemma 6.2, and Lemma 6.3.

Theorem 6.5
The tableaux algorithm presented in Section 5 is a decision procedure for satisfiability of $\mathcal{SHOIQ}_{\downarrow}$-concepts in query form with respect to TBoxes, role hierarchies, and ABoxes.

Since, by Lemma 3.9, a query q is true in a knowledge base $\mathcal{K}$ iff $\mathcal{K}$ extended with the negated query concept for q is unsatisfiable and since the negated query concept is in query form, we get the following result with regard to query answering:

Corollary 6.6
Let q be a boolean conjunctive query and let $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ be a knowledge base. The tableaux algorithm presented in Section 5 is a decision procedure for deciding if $\mathcal{K} \models q$.

Now, since answering non-Boolean queries can be reduced to answering (possibly several) Boolean queries by answering one Boolean query for each possible substitution of the free variables with individual names, we obtain our desired result:

Corollary 6.7
Let q be a conjunctive query and let $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ be a knowledge base. The tableaux algorithm presented in Section 5 is a decision procedure for deciding if $\mathcal{K} \models q$.

7 Conclusions

In the previous sections we have presented what is, to the best of our knowledge, the first decision procedure for answering conjunctive queries for $\mathcal{SHOIQ}$ knowledge bases. In particular, the presented algorithm allows for nominals in the Description Logic, which was an open problem before.

We achieved this by extending the $\mathcal{SHOIQ}$ Description Logic with a restricted form of the Hybrid Logic binder ($\downarrow$) resulting in a logic, called $\mathcal{SHOIQ}_{\downarrow}$. $\mathcal{SHOIQ}_{\downarrow}$ is capable of expressing conjunctive queries and their negations, which is, for most queries, not possible in $\mathcal{SHOIQ}$, because there is no support for role negation. In $\mathcal{SHOIQ}_{\downarrow}$, it is then possible to reduce the problem of deciding if a knowledge base entails a conjunctive query to a concept satisfiability problem.

Adding the $\downarrow$ binder in an unrestricted form makes even $\mathcal{ALC}$ undecidable. In order to regain decidability, we define a syntactic restriction on $\mathcal{SHOIQ}_{\downarrow}$ concepts, which is still capable of expressing the negation of conjunctive queries. However, even in this very restricted form, the $\downarrow$ has a notable impact on the resulting logic. In the case of $\mathcal{SHOIQ}$, for example, the extension leads to the loss of the finite model property.

In Section 5, we present a novel tableaux algorithm for deciding concept satisfiability of $\mathcal{SHOIQ}_{\downarrow}$-concepts, which extends the tableaux algorithm for deciding $\mathcal{SHOIQ}$ concept satisfiability [7]. For query answering blocking has to be delayed depending on the size of the query (the longest path in the query). In CARIN [14] or its extension to $\mathcal{SHIQ}$ [18], for example, two isomorphic trees whose depths depends on the size of the query, are used in the blocking condition. This is a severe disadvantage from a practical point of view. In our approach, we can use the normal blocking conditions of $\mathcal{SHOIQ}$ (modulo the newly introduced bindings) and the length of the query is captured automatically in the query concept. Hence blocking is delayed only as much as necessary to achieve completeness. Further on, the algorithms in the style of CARIN require the addition of an axiom $\top \sqsubseteq C \sqcup \neg C$ for every concept C in the query, which can significantly increase the amount of non-determinism, whereas the query concept guides the algorithm towards producing a counter-model and enforces a decision about concept membership only if necessary. This makes $\downarrow$ binders and variables the more attractive choice from an implementation point of view.

Cycles in the query, in particular cycles containing transitive roles, have a significant impact on the blocking conditions as well. For cycles without transitive roles the blocking conditions for $\mathcal{SHOIQ}_{\downarrow}$ are comparable to the ones for $\mathcal{SHOIQ}$. With transitive roles, however, it is difficult to find a bound on the length of paths in a completion graph and the blocking conditions do no longer work. We therefore do not allow transitive roles to occur in cycles in the query and it is currently unknown, if conjunctive query answering without this restriction is still decidable. Or future work is aimed at finding an answer to this problem.

References

[1] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein.

OWL web ontology language reference. Technical report, W3C, February 10 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[2] Tim Berners-Lee, Mark Fischetti, and Michael L. Dertouzos. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco, 1999.

[3] Patrick Blackburn and Jerry Seligman. *Advances in Modal Logic*, volume 1, chapter What are hybrid languages?, pages 41–62. CSLI Publications, Stanford University, 1998. Kracht, Marcus and de Rijke, Maarten and Wansing, Heinrich and Zakharyaschev, Michael (editors.).

[4] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1998)*, pages 149–158. ACM Press, 1998.

[5] Birte Glimm and Ian Horrocks. Handling cyclic conjunctive queries. In *Proceedings of the 18th International Workshop on Description Logics (DL 2005)*, Edinburgh, Scotland, UK, July 26–28 2005. CEUR-Workshop Proceedings.

[6] Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *Journal of Web Semantics*, 1(4):345–357, 2004.

[7] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$ description logic. In Bernd Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001.

[8] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for SHOIQ. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, July 30 – August 5 2005.

[9] Ian Horrocks, Ulrike Sattler, Sergio Tessaris, and Stephan Tobies. How to decide query containment under constraints using a description logic. In *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR 2000)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000.

[10] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical Reasoning for Expressive Description Logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR 1999)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[11] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In David McAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE 2000)*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482–496. Springer-Verlag, 2000.

[12] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2004)*, volume 3452 of *Lecture Notes in Computer Science*. Springer, March 14–18 2004.

[13] Aditya Kalyanpur, Bijan Parsia, and James Hendler. A tool for working with web ontologies. *International Journal on Semantic Web and Information Systems*, 1(1):36–49, 2005.

[14] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *European Conference on Artificial Intelligence*, pages 323–327, 1996.

[15] Maarten Marx. Narcissists, stepmothers and spies. In *Proceedings of the 2002 Description Logic Workshop (DL 2002)*, volume 53. CEUR Workshop Proceedings, April 19–21 2002.

[16] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, November 2004.

[17] Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubzy, Ray W. Fergerson, and Mark A. Musen. Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[18] Maria M. Ortiz, Diego Calvanese, and Thomas Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI 2006)*, 2006.

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

[20] Sergio Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001.

[21] Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.

[22] Katy Wolstencroft, Andy Brass, Ian Horrocks, Phil Lord, Ulrike Sattler, Daniele Turi, and Robert Stevens. A Little Semantic Web Goes a Long Way in Biology. to appear in the International Semantic Web Conference, 2005, Galway, Ireland., 2005.