Unions of Conjunctive Queries in $\text{SHOQ}$

Birte Glimm and Ian Horrocks
University of Oxford, UK

Ulrike Sattler
The University of Manchester, UK

Abstract

Conjunctive queries play an important role as an expressive query language in Description Logics (DLs). Decision procedures for expressive Description Logics are, however, only recently emerging and it is still an open question whether conjunctive queries are decidable for the DL $\text{SHOIQ}$ that underlies the OWL DL standard. In fact, no decision procedure was known for expressive DLs that contain nominals. In this paper, we close this gap by providing a decision procedure for entailment of unions of conjunctive queries in $\text{SHOQ}$. Our algorithm runs in deterministic time single exponential in the size of the knowledge base and double exponential in the size of the query, which is the same as for $\text{SHIQ}$. Our procedure also shows that $\text{SHOQ}$ knowledge base consistency is indeed $\text{ExpTime}$-complete, which was, to the best of our knowledge, always conjectured but never proved.

Introduction

Description Logics (Baader et al. 2003) are a well-established family of logic-based knowledge representation formalisms that have gained increased attention due to their usage as the logical underpinning of ontology languages such as OWL (Horrocks, Patel-Schneider, and van Harmelen 2003). A DL knowledge base consists of a TBox, which contains intensional knowledge such as concept definitions and general background knowledge, and an ABox, which contains extensional knowledge and is used to describe individuals. Using a database metaphor, the TBox corresponds to the schema, and the ABox corresponds to the data. Contrary to a typical database setting, the open world assumption is usually made in (Description) Logics, which means we have only incomplete knowledge about the modeled domain and models can be infinite.

In data-intensive applications, querying knowledge bases plays a central role. Instance retrieval is a basic reasoning task that supports querying—it allows for the retrieval of all certain instances of a given (possibly complex) concept $C$, i.e., it returns all individuals from the ABox that are an instance of $C$ in every model of the knowledge base. Technically, instance retrieval is well-understood. For the prominent DL $\text{SHIQ}$, which underlies OWL Lite, it is $\text{ExpTime}$-complete (Tobies 2001) and for $\text{SHOIQ}$, which underlies OWL DL, it is $\text{NExpTime}$-complete (Tobies 2001). Despite this high worst-case complexity, efficient implementations are available. Instance retrieval supports only limited forms of querying: concepts (roles) are used as queries, and thus we can only query for (pairs of) individual names and for tree-like structures that are invariant under (guarded) bisimulations. For this reason, many applications require conjunctive query answering as a stronger form of querying, i.e., computing the certain answers to a conjunctive query over a knowledge base.

Recently, it has been shown that conjunctive query answering is decidable in $\text{SHIQ}$ and, thus, in OWL Lite (Glimm et al. 2008; 2007; Calvanese, Eiter, and Ortiz 2007). To the best of our knowledge, it is, however, still an open problem whether this is also the case for $\text{SHOIQ}$ and, thus, OWL DL. In this paper, we make an important step in this direction by presenting an algorithm for unions of conjunctive queries in $\text{SHOQ}$, i.e., $\text{SHOIQ}$ without inverse roles. More precisely, we devise a decision procedure for entailment of unions of conjunctive queries by a $\text{SHOQ}$ knowledge base, where conjunctive query entailment is the decision problem corresponding to conjunctive query answering. It is well-known that decidability and complexity results carry over from entailment to answering (Calvanese, Eiter, and Ortiz 2007; Glimm et al. 2007; Glimm 2007). Our decision procedure is inspired by the query rewriting algorithm for $\text{SHIQ}$ (Glimm et al. 2008), but is adapted to handle the more complicated relational structures that nominals can express. In the query rewriting process, we reduce a conjunctive query to (possibly several) $\text{SHOQ}^1$-concepts, i.e., $\text{SHOQ}$-concepts with role conjunctions. Conjunctive query entailment is then reduced to consistency checking of $\text{SHOQ}^1$ knowledge bases. For this task, we devise an automata-based algorithm that runs in deterministic double exponential time in the size of the query and single exponential time in the size of the knowledge base. This result concerns the combined complexity, i.e., it is measured in the size of the knowledge base and the query. For $\text{SHIQ}$, the same upper bound holds and...
it is known to be tight (Lutz 2007).

For full proofs and more detailed definitions of the presented results we refer to (Glimm 2007).

Preliminaries

A signature \((N_C, N_R, N_I)\) consists of a set of concept names \(N_C\), role names \(N_R\), and individual names \(N_I\). The set of role names contains a subset \(N_{IR} \subseteq N_R\) of transitive role names. A role inclusion is of the form \(r \subseteq s\) with \(r, s \in N_R\). A role hierarchy \(R\) is a finite set of role inclusions. A role \(r\) is simple w.r.t. a role hierarchy \(R\) if there is no transitive role \(s \in N_R\) such that \(s \subseteq r \in R\). We define \(\equiv_R\) as the reflexive transitive closure of \(\subseteq\) w.r.t. \(R\).

An interpretation \(I = (\Delta^I, I)\) consists of a non-empty set \(\Delta^I\), the domain \(\Delta\), and a function \(\cdot^I\), which maps every concept name \(A \in N_C\) to a subset \(A^I \subseteq \Delta^I\), every role name \(r \in N_R\) to a binary relation \(r^I \subseteq \Delta^I \times \Delta^I\), and every individual name \(o \in N_I\) to an element \(a^I \in \Delta^I\). An interpretation \(I\) satisfies a role inclusion \(r \subseteq s\) if \(r^I \subseteq s^I\), and a role hierarchy \(R\) if it satisfies all role inclusions in \(R\).

\(SHOQ\)-concepts (or concepts for short) are built inductively using the following grammar, where \(o \in N_I\), \(A \in N_C\), \(n \in \mathbb{N}\), \(r \in N_R\), and \(s \in N_R\) is a simple role:

\[
C ::= A \mid \{o\} \mid \neg C \mid C \land C \mid \forall r.C \mid n.C.
\]

We use the following standard abbreviations: \(C_1 \lor C_2 := \neg(\neg C_1 \land \neg C_2)\), \(\exists r.C := \neg(\forall r.\neg C)\), and \(\forall n.s.C := \neg(\exists (n+1).s.C)\). The restriction to simple roles in number restrictions is necessary to ensure decidability of the standard reasoning tasks.

The semantics of \(SHOQ\)-concepts is defined as usual:

\[
\{o\}^I = \{a^I\}, (C_1 \land C_2)^I = C_1^I \land C_2^I, (\neg C)^I = \Delta^I \setminus C^I,
\]

\[
(\forall r.C)^I = \{d \in \Delta^I \mid \exists (s^I, (d, C)) \geq n\},
\]

where \(\sharp(M)\) denotes the cardinality of the set \(M\) and \(s^I(d, C) = \{d' \in \Delta^I \mid (d, d') \in s^I\} \subseteq C^I\).

The presented algorithm reduces a conjunctive query to concepts that may also contain role conjunctions in the place of role names. A role conjunction is an expression \(r_1 \cap \ldots \cap r_n\), where \(r_1, \ldots, r_n\) are role names. The interpretation function is extended to role conjunctions as follows:

\[
(r_1 \cap \ldots \cap r_n)^I = r_1^I \cap \ldots \cap r_n^I.
\]

A role conjunction is simple, if each role occurring in it is simple. We can then build complex \(SHOQ\)-concepts with the same grammar as above, just with \(r\) (being) a (simple) role conjunction.

A general concept inclusion (GCI) is an expression \(C \subseteq D\), where both \(C\) and \(D\) are concepts. A finite set of GCIs is called a TBox. An assertion is an expression of the form \(A(a)\), \(\neg A(a)\), \(r(a, b)\), \(\neg r(a, b)\), or \(a \neq b\) with \(A \in N_C\), \(r \in N_R\), and \(a, b \in N_I\). An ABox is a finite set of assertions. Since, in the presence of nominals, the ABox can be internalized (e.g., \(A(a)\) is equivalent to the GCI \((a) \subseteq A\), \(r(a, b)\) to \((a) \subseteq \exists r.(b)\), etc.), we assume w.l.o.g. that a \(SHOQ\)-knowledge base \(K\) is a pair \((T, R)\) where \(T\) is a TBox and \(R\) is a role hierarchy. We use \(\text{rol}(K)\) for the set of role names used in \(K\) and \(\text{nom}(K)\) for the set of individual names (nominals) that occur in \(K\). We assume that \(\text{nom}(K)\) is non-empty. This is w.l.o.g. since we can always add an axiom \(\{o\} \subseteq T\) to \(T\) for a fresh nominal \(o \in N_I\).

An interpretation \(I\) satisfies a GCI \(C \subseteq D\) if \(C^I \subseteq D^I\) and it satisfies a TBox if it satisfies each GCI in it. An interpretation \(I\) is a model of a knowledge base \(K = (T, R)\), denoted as \(I \models K\), if it satisfies \(T\) and \(R\). A knowledge base is consistent if it has a model.

For a concept \(C\), we use \(\text{nnf}(C)\) to denote the negation normal form of \(C\). We define the closure \(\text{cl}(K)\) of \(K\) as the smallest set containing \(\text{nnf}(\neg(C \cup D))\) if \(C \subseteq D \subseteq T; D\) if \(D\) is a sub-concept of \(C\) and \(C \subseteq \text{cl}(K)\); and \(\text{nnf}(\neg C)\) if \(C \subseteq \text{cl}(K)\).

Let \(N_V\) be a countably infinite set of variables with \(v, v' \in N_V\) and let \((N_C, N_R, N_I)\) be a signature with \(A \in N_C\), \(r \in N_R\). A Boolean conjunctive query \(q\) is a non-empty set of atoms, where an atom is an expression \(A(v), r(v, v'), \text{or } v \equiv v'\). We refer to these three types of atoms as concept, role, and equality atoms respectively. We use \(\text{Vars}(q)\) to denote the set of variables occurring in \(q\). A sub-query of \(q\) is simply a subset of \(q\) (including \(q\) itself). A union of Boolean conjunctive queries is an expression \(q_1 \lor \ldots \lor q_n\), where each conjunct \(q_i\) is a Boolean conjunctive query.

Please note that we do not allow for constants (individual names) to occur in the position of variables. This is w.l.o.g. since our DL contains nominals: for each constant \(a\) in \(q\), we introduce a new variable \(x_a\), replace each occurrence of \(a\) with \(x_a\), and add a concept atom \(\{a\}(x_a)\).

Since equality is reflexive, symmetric and transitive, we define the equivalence relation \(\approx\) as the transitive, reflexive, and symmetric closure of \(\equiv\) over the variables in \(q\). We define the relation \(\in\) over atoms in \(q\) as follows:

\[
A(v) \in q \text{ if there is a variable } v' \in \text{Vars}(q) \text{ such that } v \approx v' \text{ and } A(v') \in q \text{ and similarly for role atoms.}
\]

Let \(I = (\Delta^I, I)\) be an interpretation. For a total function \(\pi: \text{Vars}(q) \rightarrow \Delta^I\), we write (i) \(I \models^\pi A(v)\) if \(\pi(v) \in A^I\); (ii) \(I \models^\pi r(v, v')\) if \(\{\pi(v), \pi(v')\} \subseteq r^I\); and (iii) \(I \models^\pi v \equiv v'\) if \(\pi(v) = \pi(v')\). If \(I \models^\pi\) for all atoms at \(q\), we write \(I \models\pi\). We say that \(I\) satisfies \(q\) and write \(I \models q\) if there exists a \(\pi\) such that \(I \models^\pi q\). We call such a \(\pi\) a match for \(q\) in \(I\). Let \(K\) be a \(SHOQ\) knowledge base and \(q\) a conjunctive query. If, for every interpretation \(I\), \(I \models K\) implies \(I \models q\), we say that \(K\) entails \(q\) and write \(K \models q\). \(K\) entails a union of conjunctive queries \(q_1 \lor \ldots \lor q_n\), written as \(K \models q_1 \lor \ldots \lor q_n\), if, for each model \(I\) of \(K\), there is some \(i\) with \(1 \leq i \leq \ell\) such that \(I \models q_i\).

The size of a knowledge base \(K\) (a query \(q\)), denoted \(|K|\langle q\rangle\), is simply the number of symbols needed to write it over the alphabet of constructors, concept, role, individual, and variable names that occur in \(K\) (\(q\)). We assume unary coding of numbers in number restrictions.
Canonical Models

We first show that we can restrict our attention to interpretations that have a kind of forest shape. Since $\text{SHOQ}$ allows for nominals, the forest structure is not directly obvious, but we can use a domain that consists of a set of trees. We also introduce forest bases, which are forest-shaped interpretations that interpret transitive roles in an unrestricted way, i.e., not necessarily in a transitive way. In a forest base, the elements within a tree can be related to their direct successor nodes and to some root/nominal nodes. Due to the allowed relations from any element to nominal nodes, even forest bases are strictly speaking not forests. For simplicity and w.l.o.g., we make the unique name assumption, i.e., for each $a, b \in N_I$ such that $a \neq b$ and each interpretation $I$, we assume that $a^I \neq b^I$.

Definition 1. Let $N^*$ be the set of all (finite) words over the alphabet $N$. A tree $T$ is a non-empty, prefix-closed subset of $N^*$. The empty word $\varepsilon$ is called the root of $T$. For $w, w' \in T$, we call $w'$ a successor of $w$ if $w' = w \cdot c$ for some $c \in N$, where "·" denotes concatenation. The branching degree $d(w)$ of a node $w \in T$ is the number of its successors. If there is a $k$ such that $d(w) \leq k$ for each $w \in T$, we say that $T$ has branching degree $k$. Given a set of elements $\rho = \{a_1, \ldots, a_n\}$, a forest $F$ w.r.t. $\rho$ is a subset of $\rho \times N^*$ such that, for each $a_i \in \rho$, $(a_i, \varepsilon) \in F$ and the set $\{w \mid (a_i, w) \in F\}$ is a tree.

Let $K$ be a $\text{SHOQ}$ knowledge base. A forest base for $K$ is an interpretation $J = (\Delta^J, \cdot^J)$ that interprets transitive roles in an unrestricted (i.e., not necessarily transitive) way and, additionally, satisfies the following conditions:

F1 $\Delta^J$ is a forest w.r.t. nom($K$);
F2 if $((o, w), (o', w')) \in r^J$, then either $w' = \varepsilon$ or $o = o'$ and $w'$ is a successor of $w$;
F3 for each $o \in \text{nom}(K)$, $o^J = (o, \varepsilon)$.

An interpretation $I$ is canonical for $K$ if there exists a forest base $J$ for $K$ such that $I$ is identical to $J$ except that, for all non-simple roles $r$, we have

$$r^I = r^J \cup \bigcup_{s \subseteq r, s \in N_R} (s^J)^+,$$

where $^+$ denotes the transitive closure operator. In this case, we say that $J$ is a forest base for $I$ and, if $I \models K$, we say that $I$ is a canonical model for $K$. If, for each $o \in \text{nom}(K)$, the branching degree of the tree $\{w \mid (o, w) \in \Delta^J\}$ is bounded by some $k$, we say that $I$ has branching degree $k$.

We use the following running example throughout this paper.

Example 2. Let $K = (T, R)$ be a $\text{SHOQ}$ knowledge base with $t, t' \in N_T$

$$T = \{ \{o\} \subseteq \exists t. (C \sqcap \exists r. (\exists r. (D \sqcap \exists t. (\{o\}))) \}
\{o'\} \subseteq \exists s. T \sqcap \exists s. (\{o\}) \}
R = \{ \{r\} \subseteq t' \}$$

and $q = \{C(x), D(z), t'(x, z), t(z, x), r(x, y), r(y, z)\}$ with $\text{Vars}(q) = \{x, y, z\}$.

Figure 1 shows a graphical representation of a canonical model for $K$. Without the dashed lines, the figure would show the forest base for this canonical model. In what follows, we sometimes informally use the term shortcut, which, more precisely, means the following: let $K$ be a $\text{SHOQ}$ knowledge base and $K'$ the $\text{ALCHOQ}$ knowledge base obtained from $K$ by treating all roles as non-transitive. Now, let $I'$ be a model $K'$ that is minimal w.r.t. the interpretation of roles, i.e., there is no role $r$ and pair $(a, b) \in r^{I'}$ such that the interpretation obtained from $I'$ by excluding this pair from the interpretation of $r$ is a model of $K'$. If we have that $(a, b), (b, c) \in t^{I'}$ and $(a, c) \notin t^{I'}$ for a role $t$ that is transitive in $K$, then a model $I$ for $K$ obtained from $I'$ by transitively closing the interpretation of the roles that are transitive in $K$ contains the pair $(a, c)$ in the interpretation of $t$ and we would call this a "shortcut".
we transform each disjunct of $q$ and we can replace role atoms of the form $r(z, w)$. The second step is called shortcuts in a way that $SHOQ$-concepts. These queries are trivially false since their structure cannot be mapped to the canonical models of the knowledge base. The remaining queries are transformed into concepts by applying the rolling-up technique (Calvanese, De Giacomo, and Lenzerini 1998; Tessaris 2001). Finally, we use the obtained concepts to reduce conjunctive query entailment to knowledge base consistency checking.

Since the first query rewriting step, collapsing, is not new our running example focuses on the other rewriting steps. For the nominal rewriting step, we can choose to replace the conjunct $t(z, x)$, which bypasses the nominal node $(o, e)$ for the given mapping $\pi$ and canonical model $I$, with $t(z, x_r), t(x_r, x)$ for $x_r \in N_V$, a fresh variable. We call this nominal rewriting $q_{nr}$ (see Figure 3).

![Figure 2: A graphical representation of the query $q$ with a match in the canonical model (shown in grey, and without labels).](image)

![Figure 3: A graphical representation of the nominal rewriting $q_{nr}$ for $q$. The canonical model $I$ is shown in grey (without labels).](image)
(qs, \{x\}, \tau) with a grounding \(\tau\) that maps \(x\) to 0 results in the concept:
\[
\{0\} \cap \exists(C \cap \exists(r \cap t').(\exists(r \cap t').(D \cap \existst(\{0\})))
\]
Finally, we use the concepts from all possible rewritings and reduce the task of deciding query entailment to the task of deciding knowledge base consistency by augmenting \(K\) with an axiom of the form \(\top \subseteq \neg C_q\) for each obtained concept \(C_q\). If this extended knowledge base is satisfiable, then \(K \models q\) because any model \(I\) of the extended knowledge base cannot satisfy \(q\). If the extended knowledge base is unsatisfiable, then \(K \models \neg q\) because every model \(I\) of \(K\) satisfies \(q\) and, thus, violates one of the axioms \(\top \subseteq \neg C_q\) in the extended knowledge base.

We now give a precise definition of the rewriting steps. Please note that we assume that \(q\) is a conjunctive query and not a union of conjunctive queries since we apply the rewriting steps to each disjunct separately.

**Definition 4.** Let \(K\) be a \(SHOQ\) knowledge base and \(q\) a Boolean conjunctive query. A collapsing \(q_{co}\) of \(q\) is obtained by adding zero or more equality atoms of the form \(v \approx v'\) for \(v, v' \in Vars(q)\) to \(q\). We use \(co(q)\) to denote the set of all queries that are a collapsing of \(q\).

A nominal rewriting \(q_{nr}\) of \(q\) and \(\tau\) is obtained by choosing, for each role atom \(r(v, v') \in q\) such that there is a role \(s \in N_R\) and \(s \models_{\tau} r\) to either do nothing or to replace \(r(v, v')\) with \(s(v, v')\) and \(s(v', v)\) for \(v'' \in Vars_r\) a possibly fresh variable. We use \(nr_K(q)\) to denote the set of pairs \((q_{nr}, \rho)\) such that \(q_{nr}\) is a nominal rewriting of a query \(q_{co} \in co(q)\) and \(\rho\) is a subset of \(Vars(q_{nr})\).

We call \(\rho\) a root choice for \(q_{nr}\).

A shortcut rewriting of \(q\) and \(\tau\) is obtained from \(q\) by replacing each role atom \(t(v_1, v_n)\) from \(q\) for which there is a sequence \(r_1(v_1, v_2), \ldots, r_{n-1}(v_{n-1}, v_n) \in q\) and a role \(s \in N_R\) such that \(s \models_{\tau} t\) with \(n - 1\) role atoms \(s(v_1, v_2), \ldots, s(v_{n-1}, v_n)\). We use \(sr_K(q)\) to denote the set of all pairs \((q_{sr}, \rho)\) such that there is a pair \((q_{nr}, \rho) \in nr_K(q)\) and \(q_{sr}\) is a shortcut rewriting of \(q_{nr}\).

We assume that \(nr_K(q)\) contains no isomorphic queries, which are queries that differ only in newly introduced variable names.

A conjunctive query \(q\) is tree-shaped if there is a total function \(f\) from \(Vars(q)\) to a tree such that \(f\) is bijective modulo \(\cong\) and \(r(x, y) \in q\) implies that \(f(y)\) is a successor of \(f(x)\). A conjunctive query \(q\) is forest-shaped w.r.t. a root choice \(\rho\) if there is a total function \(f\) from \(Vars(q)\) to a forest \(F\) w.r.t. \(\rho\) s.t. \(f\) is bijective modulo \(\cong\) and, for each \(r(x, y) \in q\) with \(f(x) = (x_r, w)\), either \(f(y) = (y_r, c)\) with \(y_r \in \rho\) or \(f(y) = (x_r, w, c)\) for \(c \in \mathbb{N}\). We set \(fr_K(q) = \{(q_{fr}, \rho) \mid \rho \subseteq vars(q)\}\) and either \(\rho = \emptyset\) and \(q_{fr}\) is tree-shaped or \(q_{fr}\) is forest-shaped w.r.t. \(\rho\). We now build a query that consists only of concept atoms for queries in \(fr_K(q)\) by replacing the variables from the root choice \(\rho\) with variables from \(nom(\rho)\) and applying the rolling-up technique. Please note that although we say concept atoms, we actually allow queries in this intermediate step that contain complex concepts.

**Definition 5.** Let \(fr_K(q) \subseteq fr_K(q)\). A grounding for \(fr\) w.r.t. a root choice \(\rho\) is a total function \(\tau: \rho \rightarrow nom(\rho)\) such that, for all \(v, v' \in \rho\), \((\tau(v), \tau(v'))\) implies \(v \cong v'\). We build \(con(q_{fr}, \rho, \tau)\) as follows:

1. For each \(v, v' \in \rho\), replace \(r(v, v')\) with \((\exists \tau(v_r))\)(\(\exists \tau(v_r)\))\).
2. For each \(v \in \rho\), add a concept atom \((\exists \tau(v_r))\)\(\exists \tau(v_r)\) to \(qc\).
3. Call the result of 1 and 2 \(q\).
4. We now inductively assign, to each \(v \in Vars(q)\) a concept \(con(v)\) as follows:
   - if there is no role atom \(r(v, v') \in q\), then \(con(v) = \bigcap_{c(v) \in q} C_c\),
   - if there are role atoms \(r(v, v_1), \ldots, r(v, v_k) \in q\), then
   \[
   con(v) = \bigcap_{c(v) \in q} C_c \bigcap_{1 \leq i \leq k} \exists\bigcap_{r(v, v_i) \in q} r(v, v_i).\con(v_i).
   \]
5. Finally, \(con(q_{fr}, \rho, \tau) = \{\langle con(v)\rangle(v) \mid v \in Vars(q)\}\) and there is no role atom \(r(v', v) \in q\).

We use \(con_K(q)\) for the set \(\{con(q_{fr}, \rho, \tau) \mid (q_{fr}, \rho) \in fr_K(q)\}\) and \(\tau\) is a grounding w.r.t. \(\rho\).

Please note that, after the first step, the resulting query consists of a set of unconnected components such that each component is a tree-shaped query with a distinguished root variable. This root variable need not necessarily belong to the root choice. In Step 4, we collect all query concepts for these root variables in the set \(con(q_{fr}, \rho, \tau)\). Hence \(con(q_{fr}, \rho, \tau)\) is a conjunctive query of the form \(\{C_1(v_1), \ldots, C_n(v_n)\}\) with \(v_i \neq v_j\) for \(1 \leq i < j \leq n\) and each \(C_i\) is a \(SHOQ\) concept. For a union of conjunctive queries \(q = q_1 \lor \ldots \lor q_k\), we define \(con_K(q)\) as \(\bigcup_{1 \leq i \leq k} con_K(q_i)\). The following theorem shows that the queries in \(con_K(q)\) can be used to decide entailment of \(q\) and that there is a bound on the cardinality of this set. We can then use
The standard methods for deciding entailment of tree-shaped conjunctive queries in order to decide entailment of arbitrary conjunctive queries in SHOQ.

**Theorem 6.** Let \( q \) be a Boolean union of conjunctive queries, \( K = (T, R) \) a SHOQ knowledge base, and \( \text{con}_K(q) = \{q_1, \ldots, q_t\} \). Then (1) \( K \models q \) iff \( K \models q_1 \lor \ldots \lor q_t \).

Due to space limitations, we just give a sketch of the proof for the above theorem. For the if direction: for the collapsing step, it is clear that if a model \( \mathcal{I} \) of \( K \) entails the collapsing, then it entails the query. Apart from the collapsing step, we replace only role atoms with non-simple roles with a sequence of role atoms that use one of the transitive sub-roles. It is, therefore, not hard to see that if a model \( \mathcal{I} \) of \( K \) satisfies a rewritten query, then it satisfies the original query. In the rolling-up step, the use of nominal concepts enforces the required co-references. The remaining tree-shaped parts can then straightforwardly be expressed as concepts. As for SHIQ, we can then show that a model of the knowledge base that satisfies such a rolled-up query concept, also satisfies the query. For the only if direction: we can, by Lemma 3, restrict our attention to the canonical models of the knowledge base. We can then use any of the canonical models as a “guide” for the rewriting process as we used the given canonical model in our running example. If the canonical model satisfies the query, the rewriting steps can be applied in such a way that we obtain a forest-shaped query as required.

By carefully analysing the definition of the rewriting steps, we get the following bounds on the size of the rewritten queries and the number of rewritings.

**Lemma 7.** Let \( q \) be a Boolean union of conjunctive queries, \( K = (T, R) \) a SHOQ knowledge base, and \( \text{con}_K(q) = \{q_1, \ldots, q_t\} \). Then the size of each \( q_i \in \text{con}_K(q) \) is polynomial in \( |q| \) and the cardinality of \( \text{con}_K(q) \) is at most polynomial in \( |K| \) and exponential in \( |q| \).

Please note that each query \( q_i \in \text{con}_K(q) \) is a set of concept atoms of the form \( \{C_1^1(x_1), \ldots, C_{n_i}^r(x_{n_i})\} \), i.e., each \( q_i \) contains \( n \) unconnected components. By transforming the disjunction \( q_1 \lor \ldots \lor q_t \) of queries in \( \text{con}_K(q) \) into conjunctive normal form (cf. (Tarsis 2001, 7.3.2)), we can reduce the problem of deciding whether \( K \models q_1 \lor \ldots \lor q_t \) to deciding whether \( K \) entails each \( q_i \) in the following sense: for each \( q_i \in \text{con}_K(q) \), we use \( \text{cnf}(\text{con}_K(q)) \) for the conjunctive normal form of \( q_i \lor \ldots \lor q_t \). We now show how we can decide entailment of unions of conjunctive queries, where each atom consists of exactly one concept atom only. This suffices to decide conjunctive query entailment for SHOQ.

**Definition 8.** Let \( K = (T, R) \) be a SHOQ knowledge base, \( q \) a union of Boolean conjunctive queries, and \( C_1(v_1) \lor \ldots \lor C_t(v_t) \) a query from \( \text{cnf}(\text{con}_K(q)) \). An extended knowledge base w.r.t. \( K \) and \( q \) is a pair \( (T \cup T_q, R) \) such that \( T_q = \{T \not\subseteq -C_i\} \) with \( 1 \leq i \leq t \).

We can now use the extended knowledge bases in order to decide conjunctive query entailment:

**Theorem 9.** Let \( K \) be a SHOQ knowledge base and \( q \) a union of Boolean conjunctive queries. Then \( K \models q \) iff each extended knowledge base \( K_q \) w.r.t. \( K \) and \( q \) is inconsistent.

By again carefully analysing the definition of the extended knowledge bases and by using the results from Lemma 7, we get the following bounds on the size of and the number of extended knowledge bases.

**Lemma 10.** Let \( K \) be a SHOQ knowledge base and \( q \) a union of Boolean conjunctive queries. The size of each extended knowledge base is at most polynomial in \( |K| \) and exponential in \( |q| \) and the number of extended knowledge bases w.r.t. \( K \) and \( q \) is at most exponential in \( |K| \) and double exponential in \( |q| \).

**SHOQ\(^\cap\)** Knowledge Base Consistency

We now present our automata based decision procedure for SHOQ\(^\cap\) knowledge base consistency. By Theorem 9 this gives us a decision procedure for entailment of unions of conjunctive queries in SHOQ.

**Eliminating Transitivity**

Since automata cannot directly handle transitive roles, we first transform a SHOQ\(^\cap\) knowledge base \( K \) into an equisatisfiable \( \text{ALCHOQ}^\cap \) knowledge base \( \text{et}(K) \). In the presence of role conjunctions and nominals, it does not suffice to extend the standard encoding of transitivity (see, e.g., (Kazakov and Motik 2006)) in a trivial way. Such a naive extension would result in an \( \text{ALCHOQ}^\cap \) knowledge base \( \text{et}(K) \) that is obtained from \( K \) by treating all transitive roles as non-transitive and by adding an axiom

\[ \forall (r_1 \cap \ldots \cap r_n). C \subseteq (\forall t_1 \cap \ldots \cap t_n). (\forall (t_1 \cap \ldots \cap t_n). C) \]

for each concept \( \forall (r_1 \cap \ldots \cap r_n). C \in \text{cl}(K) \) and roles \( t_1, \ldots, t_n \in N_{IR} \) such that \( t_i \subseteq \text{r} r_i \) for \( 1 \leq i \leq n \).

The following example shows that such an encoding does not yield an equisatisfiable knowledge base. Let \( K = (T, R) \) be a SHOQ knowledge base with

\[ T = \{\{o\} \subseteq \exists r. (\exists t. (\{o\})), \{o\} \subseteq \exists r. (\{o\})\} \]

\[ R = \emptyset, \text{ and } t \text{ a transitive role}. \]

Figure 5 shows a representation of a model for \( K \), where the grey edge represents the role \( r \) and the black edges represent the role \( t \). The dashed black lines represent implicit (due to transitivity) instances of \( t \). It is not hard to check that adding the axiom \( \{o\} \subseteq \forall (r \cap t). (\{o\}) \) makes the knowledge base inconsistent.

The trivial encoding \( \text{et}(K) \) of \( K \) contains (among others) the additional axiom \( \forall t. (\{o\}) \subseteq \forall t. (\{o\}) \) since \( \forall t. (\{o\}) \in \text{cl}(K) \) and, thus, \( \forall t. (\{o\}) \in \text{cl}(K) \). Adding the axiom \( \{o\} \subseteq \forall (r \cap t). (\{o\}) \) to \( K \) does not yield any further axioms in \( \text{et}(K) \) since \( r \) is a simple role. Since none of the added axioms explicated the
implicit t relation between the nominals o and o', extending K with the axioms \{o\} ⊆ ∀(r ∩ t).¬{o'} yields a consistent knowledge base after applying the translation, contradicting our assumption that a knowledge base K is consistent iff \text{et}(K) is consistent.

Intuitively, this problem arises since we can have arbitrary relations between nominals. This can lead to situations where, as in the above example, we have an explicit relationship between two nominals, but only together with the implicit transitive shortcut can the universal quantifier over the role conjunction be applied. Hence, the above described encoding does not suffice.

We propose, therefore, a more involved encoding that explicates all transitive shortcuts between nominals.

**Definition 11.** Let K = (T, R) be a SHOQ\(^\upcorner\) knowledge base. The function \text{et}(K) yields the ALC\(\text{HOQ}^\upcorner\) knowledge base obtained from K as follows:

1. for each transitive role t and nominal o \in \text{nom}(K), add an axiom \exists t(\exists \{o\}) \subseteq \exists t(\{o\}),
2. for each concept \forall R.C \in c(K) with R = r_1 ∩ ... ∩ r_n and transitive roles t_1, ..., t_n such that t_i \subseteq R_i for each 1 \leq i \leq n, add an axiom \forall R.C \subseteq \forall T.(\forall T.C), where T = t_1 ∩ ... ∩ t_n, and
3. all roles in \text{et}(K) are non-transitive.

With the above definition, \text{et}(K) contains, additionally, the axiom \exists t(\exists \{o\}) \subseteq \exists t(\{o\}), which ensures that the implicit t-edges to nominals (the dashed lines in Figure 5) are made explicit. As a consequence, adding the axiom \{o\} \subseteq ∀(r ∩ t).¬{o'} indeed results in an inconsistent knowledge base.

Due to role conjunctions over non-simple roles, the encoding from Definition 11 yields not necessarily a knowledge base whose size is polynomial in the size of the input knowledge base. The number of transitive sub-roles for a role r that occurs in a role conjunction is bounded by m. Hence, we can use up to m transitive sub-roles for each of the at most n role conjuncts in the second step of the encoding, which results in at most \(m^n\) additional axioms in \text{et}(K). Since, in general, the length of the longest role conjunction can only also be bound by m, the encoding is exponential in m. To the best of our knowledge, it is unknown if this blow-up can be avoided.

**Lemma 12.** Let K be a SHOQ\(^\upcorner\) knowledge base with \(|K| = m\) and where the length of the longest role conjunction occurring in K is n. Then K is consistent iff \text{et}(K) is consistent and the size of \text{et}(K) is polynomial in m and exponential in n.

In the remainder we assume w.l.o.g. that K = (T, R) is an ALC\(\text{HOQ}^\upcorner\) knowledge base and that existential and universal restrictions in K are expressed using number restrictions.

**Alternating Automata**

In this section, we show how we can use (one-way) alternating automata (Muller and Schupp 1987) to decide the consistency of an ALC\(\text{HOQ}^\upcorner\) knowledge base. Alternating automata have the power of making both universal and existential choices. Informally, this means that in the transition function, we can create copies of the automaton, send them to successor nodes, and require that either some (existential) or all (universal) of them are accepting. We use, as usual, positive Boolean formulae as defined below in the specification of the transition function.

**Definition 13.** A labeled tree over an alphabet \(\Sigma\) is a pair \((T, \mathcal{L})\), where T is a tree and \(\mathcal{L}: T \rightarrow \Sigma\) maps each node in T to an element of \(\Sigma\).

Let X be a set of atoms. The set \(\mathcal{B}^+(X)\) of positive Boolean formulae is built over atoms from X, true, and false using only the connectives \(\land\) and \(\lor\). Let \(X^T\) be a subset of X. We say that \(X^T\) satisfies a formula \(\phi \in \mathcal{B}^+(X)\) if assigning true to all atoms in \(X^T\) and false to all atoms in \(X \setminus X^T\) makes \(\phi\) true.

Let \([k] = \{0, 1, ..., k\}\). An alternating looping tree automaton on k-ary \(\Sigma\)-labeled trees is a tuple \(A = (\Sigma, \delta, \delta_0)\), where Q is a finite set of states, \(\delta_0 \in Q\) is the initial state, and \(\delta: \Sigma \times Q \rightarrow \mathcal{B}^+(\{k\} \times Q)\) is the transition function.

A run of \(A\) on a \(\Sigma\)-labeled k-ary tree \((T, \mathcal{L})\) is a \((T \times Q)\)-labeled tree \((T_r, \mathcal{L}_r)\) that satisfies the following conditions:

- \(\mathcal{L}_r(\varepsilon) = (\varepsilon, \delta_0)\),
- if \(y \in T_r\) with \(\mathcal{L}_r(y) = (x, q)\) and \(\delta(q, \mathcal{L}(x)) = \phi\), then there is a (possibly empty) set \(S \subseteq [k] \times Q\) that satisfies \(\phi\) such that, for each \((c, q') \in S\), y has a successor \(y \cdot i\) in \(T_r\) with \(i \in N\) and \(\mathcal{L}_r(y \cdot i) = (x-c, q')\).

An automaton \(A\) accepts an input tree T if there exists a run of \(A\) on T. The language accepted by \(A\), \(\text{lang}(A)\), is the set of all trees accepted by \(A\).

For alternating automata, the non-emptiness problem is the following: given an alternating automaton \(A\), is there a tree \((T, \mathcal{L})\) such that \(A\) has an accepting run on \((T, \mathcal{L})\)?

Please note that, since we use looping automata, we do not impose any acceptance conditions and each run is accepting, i.e., we require only that the conditions imposed on a run are satisfied.

**Tree Relaxations**

In this section, we show how we can obtain labeled k-ary trees from a canonical model for an ALC\(\text{HOQ}^\upcorner\) knowledge base that can be used as input for our automaton. Since the labeled trees that an automaton takes as input cannot have labeled edges, we additionally store,
in the label of a node, with which roles it is related to its predecessor. Unfortunately, this does not work for the nominal nodes since a nominal node can be the successor of arbitrary elements and does not necessarily have a unique predecessor. In a first step, we build, therefore, a relaxation for a canonical model where, for each relationship between a node and a nominal node, we create a dummy node that is a representative of the nominal node. The label of the representative node is the extension of the label for the nominal node with \( \text{rep} \) and the role names with which the node is related to the nominal.

In this section, we use the equisatisfiable \( \text{ALCHOQ} \) version of \( \text{et}(K) \) of the running example, which also contains the axioms \( \exists t.(\exists t.\{}\{o\}\}) \subseteq \exists t.\{}\{o\}\) and \( \exists t'.(\exists t'.\{}\{o\}\}) \subseteq \exists t'.\{}\{o\}\). Without the dashed lines Figure 1 would represent a canonical model for \( \text{et}(K) \) and Figure 6 shows a relaxation for \( \text{et}(K) \).

\[
\begin{align*}
(o, \varepsilon) & \quad o \\
(o, 1) & \quad t, C \\
(o, 11) & \quad r, t' \\
(o, 111) & \quad r, t', D \\
(o, 111) & \quad (o, \varepsilon) \\
(\varepsilon, \varepsilon) & \quad o' \\
(o', 1) & \quad s, \text{rep}, o \\
(o', \varepsilon) & \quad s, \text{rep}, o'
\end{align*}
\]

Figure 6: A graphical representation of a relaxation for \( K \).

**Definition 14.** A set \( H \subseteq \text{cl}(K) \) is called a Hintikka set for \( K \) if the following conditions are satisfied:

1. For each \( C \subseteq D \subseteq T \), \( \text{nnf}(\neg C \cup D) \subseteq H \).
2. If \( C \cap D \subseteq T \), then \( \{C, D\} \subseteq H \).
3. If \( C \cup D \subseteq T \), then \( \{C, D\} \cap H \neq \emptyset \).
4. For all \( C \in \text{cl}(K) \), either \( C \in H \) or \( \text{nnf}(\neg C) \in H \).

We use \( H(K) \) to denote the set of all Hintikka sets for \( K \). A relaxation \( R = (\Delta^T, L) \) for \( K \) with \( L : \Delta^T \rightarrow 2^{\text{cl}(K) \cup \text{rep}(K) \cup \{\text{root} \}} \) satisfies the following properties:

- (R1) Let \( D = \text{nom}(K) \times \mathbb{N}^+ \) and \( B = \{d \mid d' \mid d \in D \text{ and } d' \in \text{nom}(K) \times (\{\varepsilon\})\} \), then \( \Delta^T \subseteq D \cup B \).
- (R2) For each \( o \in \text{nom}(K) \), \( (o, \varepsilon) \in \Delta^T \).
- (R3) Each set \( \{w \mid \{o, w\} \in \Delta^T \cap D \} \) is a tree.
- (R4) If \( d \mid d' \in \Delta^T \cap B \), then \( \{d, d'\} \subseteq \Delta^T, L(d) \cup L(d') \cap \text{cl}(K) = L(d') \cap \text{cl}(K) \), and \( \text{rep} \in L(d) \cup L(d') \).
- (R5) For each \( d \in \Delta^T, L(d) \cap \text{cl}(K) \in H(K) \).

**R6** For each \( d \in \Delta^T, \) if \( r \subseteq s \in R \) and \( r \in L(d) \), then \( s \in L(d) \).

**R7** For each \( (o, \varepsilon) \in \Delta^T, L(o, \varepsilon) \cap \text{rol}(K) = \emptyset \).

(R8) If \( d = (o, w) \in \Delta^T \) and \( \{n \mid n \in \text{rol}(K) \} \subseteq L(d) \), then there are \( n \) distinct elements \( d_1, \ldots, d_n \in \Delta^T \) such that, for each \( i \) with \( 1 \leq i \leq n \), \( \{r_1, \ldots, r_k, C\} \subseteq L(d_i) \) and either \( d_i = (o, w \cdot c) \) with \( c \in \mathbb{N} \) or \( d_i = d(d' \in \Delta^T \cap B \).

(R9) If \( d = (o, w) \in \Delta^T \) and \( \{n \mid n \in \text{rol}(K) \} \subseteq L(d) \), then \( \{d' \mid d' = (o, w \cdot c) \) for some \( c \in \mathbb{N} \) or \( d' = d(d' \in \Delta^T \cap B \).

Given the above properties of relaxations and the results from Lemma 3, we can show the following:

**Lemma 15.** \( K \) has a relaxation if \( K \) is consistent.

In a second step, we build a tree relaxation, which is a labeled tree, from the relaxation. For this, we additionally add a dummy root node labeled with root that has all nominal nodes as successors, and we require that the domain is a tree.

\[
\begin{align*}
1 & \quad o \\
2 & \quad o' \\
11 & \quad t, C \\
21 & \quad s, \text{rep}, o \\
22 & \quad s \\
1111 & \quad t, \text{rep}, o
\end{align*}
\]

Figure 7: A tree relaxation for the relaxation from Figure 6.

Figure 7 shows a representation of a tree relaxation built from the relaxation for our running example. The tree relaxation can, additionally, have dummy nodes labeled with \( \# \), but we do not show any dummy nodes in the figure. For ease of presentation, we assume in the remainder that all tree relaxations are full trees, i.e., all non-leaf nodes have the same number of successors, and we add dummy nodes labeled with \( \# \) where necessary.

**Definition 16.** A tree relaxation for \( K \) is a labeled tree \( (T, L) \) with \( L : T \rightarrow 2^{\text{cl}(K) \cup \text{rep}(K) \cup \{\text{root} \}} \) that satisfies the following conditions:
For each $o \in \text{nom}(K)$, there is a unique $c \in \mathbb{N} \cap T$ with $o \in \mathcal{L}(c)$ and \{(rep, #), rol(K)\} \cap \mathcal{L}(c) = \emptyset.

If $c \in \mathbb{N} \cap T$ and $\text{nom}(K) \cap \mathcal{L}(c) = \emptyset$, then $\mathcal{L}(c) = \{\#\}$.

For each $w \in \mathbb{N}^+ \cap T$, $\sharp(\mathcal{L}(w) \cap \text{nom}(K)) \leq 1$.

For each $w = w' \cdot c \in T$ with $w' \in \mathbb{N}^+$ and $c \in \mathbb{N}$, if $\mathcal{L}(w) \cap \text{nom}(K) \neq \emptyset$, then $\text{rep} \in \mathcal{L}(w)$.

For each $w, w' \in T$ and $o \in \text{nom}(K)$, if $o \in \mathcal{L}(w) \cap \mathcal{L}(w')$, then $\text{cl}(K) \cap \mathcal{L}(w) = \text{cl}(K) \cap \mathcal{L}(w')$.

For each $w \in \mathbb{N}^+ \cap T$, if \{rep, #\} \cap $\mathcal{L}(w) \neq \emptyset$, then, for each successor $w'$ of $w$, $\# \in \mathcal{L}(w')$.

For each $w \in T$, if $\mathcal{L}(w) \cap \{\#, \text{root}\} = \emptyset$, then $\mathcal{L}(w) \cap \text{cl}(K) = \text{H}(K)$.

For each $w \in T$ and $r \subseteq s \in \mathcal{R}$, if $r \in \mathcal{L}(w)$, then $s \in \mathcal{L}(w)$.

For each $w \in T$, if $(\geq n \ (r_1 \cap \ldots \cap r_m).C) \in \mathcal{L}(w)$, then there are at least $n$ distinct successors $w_1, \ldots, w_n$ of $w$ with $\{r_1, \ldots, r_m, C\} \subseteq \mathcal{L}(w_i)$, for each $i$ with $1 \leq i \leq n$.

For each $w \in T$, if $(\leq n \ (r_1 \cap \ldots \cap r_m).C) \in \mathcal{L}(w)$, then there are at most $n$ distinct successors $w_1, \ldots, w_n$ of $w$ with $\{r_1, \ldots, r_m, C\} \subseteq \mathcal{L}(w_i)$, for each $i$ with $1 \leq i \leq n$.

If $T$ has branching degree $k$, then we say that $(T, \mathcal{L})$ is a $k$-ary tree relaxation for $K$.

By using the above defined properties of tree relaxations and the results about the bounded branching degree of canonical models from Lemma 3, we get the following:

Lemma 17. Let $n_{\text{max}}$ be the maximal number occurring in a number restriction in $K$, and $k = n_{\text{max}} \cdot |K| + \sharp(\text{nom}(K))$. $K$ has a $k$-ary tree relaxation iff $K$ is consistent.

The Decision Procedure

It remains to devise a procedure that decides whether $K$ has a tree relaxation. For this, we define an alternating automaton that accepts exactly the tree relaxations of $K$. Our automaton combines and extends ideas from (Sattler and Vardi 2001) and (Calvanese, Eiter, and Ortiz 2007), where (two-way) alternating automata have been used for the hybrid $\mu$-calculus (Sattler and Vardi 2001) and for answering regular path queries in $ACQA2Q2_{\text{reg}}$. We first define two alternating automata $A_K$ and $B_K$, and then define an automaton $\mathcal{B}_K$ as their intersection. Informally, the automaton $A_K$ just checks that the input tree has a structure as required whereas the automaton $\mathcal{B}_K$ checks that the input is indeed a tree relaxation for $K$. For alternating automata, intersection is simple: we introduce a new initial state $q_0$ and set the transition function for $q_0$ and each letter $\sigma$ from the input alphabet $\Sigma$ to $\delta(q_0, \sigma) = (0, q_{(0,1)}) \wedge (0, q_{(0,2)})$, where $q_{(0,1)}$ and $q_{(0,2)}$ are the initial states of $A_K$ and $A_K$ respectively. The size of the resulting automaton is the sum of the sizes of $\mathcal{A}_K$ and $A_K$.

The automaton $A_K$ is relatively straightforward and helps to keep the definition of the automaton $\mathcal{A}_K$, where we do the real work, transparent. Informally, it guarantees the following:

- We distinguish root (state $q_r$), nominal (state $q_n$), nominal representative (state $q_{\text{rep}}$), dummy (state $q_{\#}$), and normal nodes (state $q_n$).

- The label root is only found in the root node.

- The level one nodes are either “real” nominal nodes (i.e., they are not marked as representatives with rep) with exactly one nominal and no roles in their label, or they are dummy nodes labelled with # only.

- The level one nominal nodes have either normal, nominal representative, or dummy nodes as successors.

- Nominal representative nodes are marked with rep, and have exactly one nominal in their label.

- Dummy nodes have only dummy nodes as successors. More precisely, let $n_{\text{max}}$ be the maximal number occurring in number restrictions in $K$, and $k = n_{\text{max}} \cdot |\text{cl}(K)| + \sharp(\text{nom}(K))$. The alphabet $\Sigma$ for both automata $\mathcal{A}_K$ and $A_K$ is

\[
\delta(q_r, \sigma) = \bigwedge_{i=1}^k (i, q_n) \lor (i, q_{\#})
\]

if $\sigma = \{\text{root}\}$ and it is false otherwise.

\[
\delta(q_n, \sigma) = \bigwedge_{i=1}^k ((i, q_n) \lor (i, q_{\text{rep}}) \lor (i, q_{\#}))
\]

if $\sharp(\text{nom}(K) \cap \sigma) = 1$ and $\{\text{root, rep, #}, \text{rol}(K)\} \cap \sigma = \emptyset$ and it is false otherwise.

\[
\delta(q_{\text{rep}}, \sigma) = \bigwedge_{i=1}^k (i, q_{\#})
\]

if $\sharp(\text{nom}(K) \cap \sigma) = 1$, $\text{rep} \in \sigma$, and $\{\text{root, #}\} \cap \sigma = \emptyset$ and it is false otherwise.

\[
\delta(q_{\#}, \sigma) = \bigwedge_{i=1}^k (i, q_{\#})
\]

if $\{\#\} = \sigma$ and it is false otherwise.

The automaton $A_K$ mainly checks the formulae occurring in the labels of the input. Hence, most of the
states correspond to formulae in cl(κ) and the transition function is more or less determined by the semantics and we only sketch its realization. For number restrictions, we use the same technique as (Calvanese, Eiter, and Ortiz 2007) that involves states that count how many successors have been checked and how many of the checked ones fulfill the requirements of the number restriction. In the root node, we additionally make a non-deterministic choice, for each nominal and each atomic concept, whether the concept or its negation holds at the nominal node. This choice is propagated downwards in the tree in order to ensure that the nominal representatives agree with their corresponding real nominal nodes on all atomic concepts. This also enables us to simply count over the successors of a node for the qualified number restrictions. We propagate the concepts via a kind of universal role and we assume that $\emptyset$ is a symbol that does not occur in cl(κ) or rol(κ). We define, therefore, the following set of auxiliary states

$$Q_{rep} = \{ \{\emptyset\} \cup \bigcup \{ o \in nom(\kappa) \text{ and } A \in N_T \cap cl(\kappa) \} \bigcup \{ \{\emptyset\} \cup \neg A \mid o \in nom(\kappa) \text{ and } A \in N_T \cap cl(\kappa) \} \}.$$  

We then define $A_{\kappa}$ as $(\Sigma, Q, \delta, q_0)$, where $q_0$ is the initial state and the set $Q$ of states is

$$\{ q_0 \} \cup cl(\kappa) \cup rol(\kappa) \cup \{ r \mid r \in rol(\kappa) \} \cup \{ q_T, q_R \} \cup \{ (\bowtie n R.C, i, j) \mid i \leq j \leq k \cup Q_{rep} \cup \{ v u. C \mid C \in Q_{rep} \} \},$$

where states of the form $(\bowtie n R.C, i, j)$ are used to check that the number restrictions are satisfied.

In the following, we give the definition of the transition function for each $\sigma \in \Sigma$ together with an explanation for each of the different types of states.

At the root node, we are in the initial state $q_0$ which has the following tasks: (a) we make the non-deterministic guesses for all atomic concepts, (b) we check that there is exactly one nominal node for each of the nominals in $nom(\kappa)$, and (c) we make sure that the axioms in $T$ and $R$ are satisfied in all non-dummy descendants. Let $\ell = \sharp(nom(\kappa))$.

$$\delta(q_0), \sigma) = \bigwedge_{A \in \kappa} \bigwedge_{i=1}^{\ell} ((0, \forall u.(\neg o_i) \cup A)) \lor (0, \forall u.(\neg o_i) \cup \neg A)) \wedge \bigwedge_{i=1}^{\ell} \bigwedge_{j=1}^{k} \bigwedge_{1 \leq i < j \leq k} (\bigwedge_{i=1}^{k} (j, o_i) \lor (j, \neg o_i)) \wedge (i, q_T) \lor (i, q_R)$$

Whenever we are in a state that is used to propagate information downwards through the whole tree via the “universal role” and we are not at a dummy node, we check that the required concept holds at the current node and also check all successors. More precisely, for each $C \in Q_{rep}$.

$$\delta(\forall u. C, \sigma) = (0, C) \land \bigwedge_{i=1}^{\ell} (i, \forall u. C)$$

if $\# \notin \sigma$,

$$\delta(\forall u. C, \sigma) = \bigwedge_{i=1}^{k} (i, \forall u. C)$$

if $\# \in \sigma$, and it is $true$ otherwise.

All non-dummy descendants of the root nodes must satisfy the TBox and RBox axioms.

$$\delta(q_T, \sigma) = \bigwedge_{C \in T} ((0, \text{nnf}(\neg C)) \lor (0, D)) \land \bigwedge_{i=1}^{k} (i, q_T)$$

if $\# \notin \sigma$ and it is $true$ otherwise.

$$\delta(q_R, \sigma) = \bigwedge_{r \in R} ((0, \neg) \lor (0, s)) \land \bigwedge_{i=1}^{k} (i, q_R)$$

if $\# \notin \sigma$ and it is $true$ otherwise.

The concepts that are used as states are inductively decomposed according to the semantics. We start by defining the base cases. For each $\alpha \in (N_T \cap cl(\kappa)) \cup rol(\kappa) \cup nom(\kappa)$: $\delta(\alpha, \sigma) = true$ if $\alpha \in \sigma$ and it is $false$ otherwise: $\delta(\neg \alpha, \sigma) = true$ if $\alpha \notin \sigma$ and it is $false$ otherwise. Since we use constructors for nominals, they are not handled as atomic concepts and we set, for each $o \in nom(\kappa)$, $\delta(\{o\}, \sigma) = (0, o); \delta(\neg \{o\}, \sigma) = (0, \neg o).

Conjunction and disjunction are handled in the straightforward way. For each $C_1 \land C_2 \in cl(\kappa)$, $\delta(C_1 \land C_2, \sigma) = (0, C_1) \land (0, C_2)$; for each $C_1 \lor C_2 \in cl(\kappa)$, $\delta(C_1 \lor C_2, \sigma) = (0, C_1) \lor (0, C_2)$.

For number restrictions, we have to use a more sophisticated technique that involves states that count how many successors have been checked and how many of the checked ones fulfill the requirements of the number restriction. This technique was introduced by Calvanese, De Giacomo, and Lenzerini (2002). More precisely, for each concept of the form $(\geq n R.C) \in cl(\kappa)$ with $R = r_1 \cap \ldots \cap r_m$.

$$\delta(\geq n R.C, \sigma) = (0, (\geq n R.C, 0, 0))$$

if $\# \notin \sigma$ and it is $true$ otherwise. For $1 \leq i \leq k$ and $1 \leq j \leq n$, $\delta(\geq n R.C, i, j, \sigma) = (i, \neg r_1) \lor \ldots \lor (i, \neg r_m) \lor$ (i, $\text{nnf}(\neg C))\lor$ (0, (\geq n R.C, i + 1, j + 1))}$

$$\bigwedge_{1 \leq i \leq k} \bigwedge_{1 \leq j \leq n} (i, \neg o_i) \lor (i, o_i) \land (i, C) \land (0, (\geq n R.C, i + 1, j + 1))$$

For $1 \leq i \leq k$, $\delta(\geq n R.C, i, n, \sigma) = true$. For $1 \leq j < n$, $\delta(\geq n R.C, i, j, \sigma) = false$.

Informally, we use the counter $i$ to count how many of the $k$ successors have already been checked and $j$ is increased for each successor that fulfills the requirements of the number restriction. The atmost number restrictions are handled similarly:

$$\delta(\leq n R.C, \sigma) = (0, (\leq n R.C, 0, 0))$$

if $\# \notin \sigma$ and it is $true$ otherwise. For $1 \leq i \leq k$ and $1 \leq j \leq n$, $\delta(\leq n R.C, i, j, \sigma) = (i, \neg r_1) \lor \ldots \lor (i, \neg r_m) \lor$ (i, $\text{nnf}(\neg C))\lor$ (0, (\leq n R.C, i + 1, j + 1))}$

$$\bigwedge_{1 \leq i \leq k} \bigwedge_{1 \leq j \leq n} (i, \neg o_i) \lor (i, o_i) \land (i, C) \land (0, (\leq n R.C, i + 1, j + 1))$$
For $1 \leq i \leq k$, $\delta((\leq n \ R.C, i, n+1), \sigma) = \text{false}$. For $1 \leq j < n$, $\delta((\leq n \ R.C, k, j), \sigma) = \text{true}$. We can now check whether the language accepted by the automaton $B_K$ is empty, which is enough to decide consistency of $K$:

**Theorem 18.** Let $B_K$ an alternating automaton as defined above. Then $K$ is consistent iff the language accepted by $B_K$ is non-empty.

Since looping alternating tree automata are a special case of alternating Büchi tree automata, we can use the result that, for an alternating Büchi automaton $A$ with $n$ states and input alphabet with $\ell$ elements, non-emptiness of the language accepted by $A$ is decidable in time exponential in $n$ and polynomial in $\ell$ (Vardi 1995). For the states, we mainly use formulae from $cl(K)$ and the size of the closure is linear in $|K|$ and, overall, the number of states for our automaton is polynomial in $|K|$. The input alphabet $2^{\text{rep. # root}} \cdot |\text{cl}(K)| \cdot |\text{rol}(K)| \cdot |\text{nom}(K)|$ is exponential in $|K|$. Hence, we get the following result:

**Theorem 19.** Checking the consistency of an ACHQ knowledge base $K$ can be done in deterministic time single exponential in the size of $K$ assuming unary coding of numbers in number restrictions.

By Lemma 12, we obtain the following upper bound on deciding consistency of SHOIQ knowledge bases.

**Theorem 20.** Let $K$ be a SHOIQ knowledge base where $|K| = m$ and the length of the longest role conjunction occurring in $K$ is $n$. Deciding the consistency of $K$ can be done in deterministic time single exponential in $m$ and double exponential in $n$ assuming unary coding of numbers in number restrictions.

The above result also shows that SHOQ, i.e., when no role conjunctions are used, is indeed EXPTIME-complete. This was always conjectured but, to the best of our knowledge, never proved.

**Conclusions and Future Work**

Although the presented upper bound for query entailment in SHOIQ agrees with the one for SHIQ (Lutz 2007) it is still an open whether this bound is tight. The hardness proof for SHIQ very much relies on the ability to propagate information upwards via inverse roles, which nominals can only simulate to some extend.

For SHOIQ, our algorithm runs in deterministic time double exponential in the size of the input knowledge base and, as for SHIQ, it is still an open question whether this blow-up due to role conjunctions over non-simple roles can be avoided.

**References**


