Conjunctive Query Answering for Description Logics with Transitive Roles

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

Abstract

An important reasoning task, in addition to the standard DL reasoning services, is conjunctive query answering. In this paper, we present algorithms for conjunctive query answering in the expressive Description Logics $\mathcal{SHQ}$ and $\mathcal{SHOQ}$. In particular, we allow for transitive (or non-simple) roles in the query body, which is a feature that is not supported by other existing conjunctive query answering algorithms. For $\mathcal{SHQ}$, we achieve this by extending the logic with a restricted form of $\downarrow$ binders and state variables as known from Hybrid Logics. We also highlight, why the addition of inverse roles makes the task of finding a decision procedure for conjunctive query answering more complicated.

1 Introduction

Existing Description Logic (DL) reasoners[1] provide automated reasoning support for checking concepts for satisfiability and subsumption, and also for answering queries that retrieve instances of concepts and roles. The development of a decision procedure for conjunctive query answering in expressive DLs is, however, still a partially open question. *Grounded* conjunctive queries for $\mathcal{SHIQ}$ are supported by KAON2, Pellet, and Racer's query language nRQL. However, the semantics of grounded queries is different from the usually assumed open-world semantics in DLs since existentially quantified variables are always bound to named individuals.

*This work was supported by an EPSRC studentship.

[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/`

None of the existing conjunctive query answering techniques [14, 10, 4, 9, 13] is able to handle transitive roles or nominals in the query body.[2] In this paper, we present an extension of $\mathcal{SHQ}$ with a restricted form of the $\downarrow$ binder operator and state variables as known from Hybrid Logics [3]. This extended logic enables an extension of the rolling-up technique [14, 4] to transitive roles; a feature which was left open by Tessaris [14]. Unfortunately, the $\downarrow$ binder already makes the DL $\mathcal{ALC}$ undecidable. For query answering, however, the $\downarrow$ binder occurs in a very restricted form and, as we show in Section 4, this extension of $\mathcal{SHQ}$ is decidable. Further on, we adapt the tableaux algorithm for $\mathcal{SHOQ}$ [7] in order to decide conjunctive query entailment in $\mathcal{SHQ}$.

Query answering for a logic that includes nominals, e.g., a logic such as $\mathcal{SHOQ}$, has the additional difficulty that cycles are no longer limited to ABox individuals or shortcuts via transitive roles. Therefore, the non-deterministic assignment of individual names to variables in a cycle, as suggested for the DL $\mathcal{DLR}$ by Calvanese et al. [4], can no longer be applied. In Section 5, however, we show that, by also introducing new variables, the non-deterministic assignment of individual names to variables can be used in order to provide a query answering algorithm for $\mathcal{SHOQ}$.

Finally, we highlight why the extension of $\mathcal{SHQ}$ or $\mathcal{SHOQ}$ with inverse roles makes the design of a decision procedure for conjunctive query answering more complicated.

2 Preliminaries

We assume readers to be familiar with the syntax and semantics of the DL $\mathcal{SHOIQ}$ (for details see [1]) and of $\mathcal{SHOIQ}$ knowledge bases. Therefore, we introduce only the syntax and semantics of the $\downarrow$ binder and of conjunctive queries here.

Let $\mathcal{L}$ be a Description Logic, C an $\mathcal{L}$-concept, and N_V a finite set of variable names with $y \in N_V$. With $\mathcal{L}_\downarrow$, we denote the language obtained by allowing, additionally, y and $\downarrow y.C$ as $\mathcal{L}$-concepts. 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 $y^\mathcal{I} = \{d\}$. The $\mathcal{L}_\downarrow$-concept $\downarrow y.C$ is then interpreted as $(\downarrow y.C)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid d \in C^{\mathcal{I}_{[y/d]}}\}$.

Let $\vec{y}$ be a vector of *variables* and $\vec{c}$ a vector of individual names. 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 $1 \leq i \leq n$

[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 [8, 5].

[3] For readability, we sometimes abuse our notation and refer to $\vec{y}$ as a set. When referring

an atom. *Atoms* are either concept or role atoms: a *concept atom* has the form $t\colon C$, and a *role atom* the form $\langle t, t'\rangle\colon r$, for $\{t, t'\} \subseteq \mathbf{T}(q)$, C a possibly complex $\mathcal{L}$-concept, and r an $\mathcal{L}$-role.

Let $\mathcal{K}$ be an $\mathcal{L}$-knowledge base ($\mathcal{L}$-KB), $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ a model for $\mathcal{K}$, and q a Boolean conjunctive query for $\mathcal{K}$. An *assignment in* $\mathcal{I}$ is a mapping $\cdot^{A} : \mathbf{T}(q) \to \Delta^{\mathcal{I}}$. We say that q *is true in* $\mathcal{I}$ and write $\mathcal{I} \models q$ if there exists an assignment $\cdot^{A}$ in $\mathcal{I}$ s.t. $t^A \in \{t^{\mathcal{I}}\}$ for every individual $t \in \vec{c}$, $t^A \in C^{\mathcal{I}}$ for every concept atom $t\colon C$ in q, and $\langle t^A, t'^A\rangle \in r^{\mathcal{I}}$ for every role atom $\langle t, t'\rangle\colon r$ in q. We call such an assignment *a q-mapping w.r.t. $\mathcal{I}$* . 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 $\mathcal{K} \not\models q$.

Since answering non-Boolean conjunctive queries can be reduced to answering (possibly several) Boolean queries, we consider only Boolean queries here.

3 Reducing Query Answering to Concept Unsatisfiability

The main ideas used in this paper were first introduced by Calvanese et al. [4] for deciding conjunctive query entailment for the expressive DL $\mathcal{DLR}_{reg}$. Since query answering can be reduced to query entailment, the algorithm in [4] is capable of deciding conjunctive query answering as well. The authors show how a query q can be expressed as a concept C_q, such that q is true w.r.t. a given knowledge base if adding $\top \sqsubseteq \neg C_q$ makes the KB unsatisfiable. In order to obtain the concept C_q, the query q is represented as a directed, labelled graph. This graph, called a tuple-graph or a query graph, is traversed in a depth-first manner and, during the traversal, nodes and edges are replaced with appropriate concept expressions, leading to the concept C_q after completing the traversal.

The nodes in a query graph correspond to the terms in the query and are labelled with the concepts that occur in the corresponding concept atoms. The edges correspond to the role atoms in q and are labelled accordingly. For example, let q_1 be the query $\langle\rangle \leftarrow x\colon C \wedge \langle x, y\rangle\colon s \wedge y\colon D$ and q_2 the query $\langle\rangle \leftarrow x\colon C \wedge \langle x, y\rangle\colon s \wedge \langle y, x\rangle\colon r \wedge y\colon D$. The query graphs for q_1 and q_2 are depicted in Fig. 1 and Fig. 2 respectively.

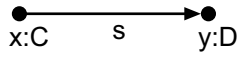

Figure 1: The (acyclic) query graph for q_1.

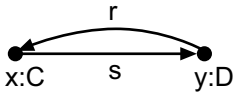

Figure 2: The (cyclic) query graph for q_2.

Since q_1 is acyclic, we can build the concept that represents q_1 as follows:

to a vector $\vec{y}$ as a set, we mean the set $\{y_i \mid y_i \text{ occurs in } \vec{y}\}$.

start at x and traverse the graph to y. Since y is a leaf node, remove y and its incoming edge and conjoin $\exists s.D$ to the label C of x, resulting in $C \sqcap \exists s.D$ for C_{q_1}. A given KB $\mathcal{K}$ entails q_1 iff $\mathcal{K} \cup \{\top \sqsubseteq \neg C_{q_1}\}$ is unsatisfiable. Since the graph is collapsed with each step, the technique is often called rolling-up. Full details can be found in the original paper [4], although the description there is slightly different since reification and name formulae are used in order to reduce $\mathcal{DLR}_{reg}$ (which allows also for roles of arity $n > 2$) to *converse*-PDL. Tessaris [14] proposes a similar approach for the DL $\mathcal{SHQ}$ (without transitive roles in the query).

This reduction is not directly extendable to cyclic queries since, due to the tree-model property of most DLs, a concept cannot capture cyclic relationships. However, for simpler DLs, this also means that cycles can only be enforced by ABox assertions and hence, Calvanese et al. suggest to replace variables in a cycle, i.e., x and y in q_2, non-deterministically with individual names from the KB (or with terms from the more specific query in the case of query entailment).

Although the technique presented by Calvanese et al. provides the basic foundation of several query answering algorithms, including the extensions presented here, it was found later that the algorithm in its original form is not a decision procedure. Horrocks et al. [8] point out that, by identifying variables with each other, some cyclic queries become acyclic, and hence a replacement of variables with named individuals might not find all solutions. For example, identifying y and y' in the query graph depicted in Fig. 3, leads to the acyclic query graph in Fig. 4, which can be expressed as $C \sqcap \exists r.\exists s.D$. Clearly, the KB containing the ABox assertion $a : \exists r'.(C \sqcap \exists r.\exists s.D)$ would make the query true, although in the relevant assignments none of the variables can be bound to the individual name a.

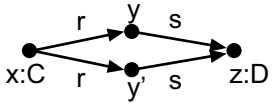

Figure 3: The original query graph.

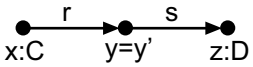

Figure 4: The query graph after identifying y with y'.

Another problem arises through the regular expressions in $\mathcal{DLR}_{reg}$, which are, for example, capable of expressing inverse roles in the query. Let q_3 be the query $\langle\rangle \leftarrow \langle x, y\rangle : r \wedge \langle y, x\rangle : s^-$ (see Fig. 5). Identifying x and y still leaves us with a cyclic query graph. Let $\mathcal{K}$ be a knowledge base containing the role axiom $r \sqsubseteq s$ and the ABox assertion $a : \exists r'.\exists r.\top$. Clearly, we have that $\mathcal{K} \models q_3$ (see Fig. 6), although again in the relevant assignments none of the variables can be bound to the individual name a. Similar examples can be given for regular expressions that use the Kleene star for expressing the transitive closure. These discoveries motivated the work presented in the following sections, which in particular targets the problems arising with cyclic queries.

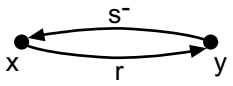

Figure 5: The query graph for q_3.

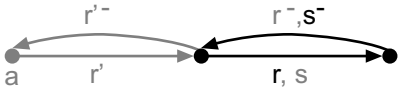

Figure 6: A representation of the canonical model of $\mathcal{K}$. The parts that make q_3 true are highlighted.

4 The Rolling-Up Technique with ↓ Binders

An extension of the rolling-up technique to cyclic queries is not directly possible, as we have tried to explain in the previous section. Due to the tree-model property of most DLs, a concept cannot capture cyclic relationships, but it is also not always correct to replace variable names in a cycle with individual names from the knowledge base. In this section, we show how cyclic queries for an $\mathcal{L}$-KB can be rolled-up into $\mathcal{L}_{\downarrow}$-concepts. The $\downarrow$ binder can label elements in a model with a variable, and this variable can be used to enforce a co-reference. Consider again the query $q_2\langle\rangle \leftarrow x\!:\!C \wedge \langle x,y\rangle\!:\!s \wedge \langle y,x\rangle\!:\!r \wedge y\!:\!D$ (see Fig. 2). Using the $\downarrow$ binder, we can construct the following concept C_{q_2} for q_2: $\downarrow x.(C \sqcap \exists s.(D \sqcap \exists r.x))$. In order to obtain this concept, we still traverse the query graph (see Fig. 7): we start at x, traverse to y, then we reach x again and replace the r-edge by conjoining $\exists r.x$ to the label of y, now y is a leaf node and we replace y and its incoming s-edge by conjoining $\exists r.\downarrow y.(D \sqcap \exists r.x)$ to the label of x, since x is the last node, we obtain C_{q_2} as $\downarrow x.(\exists r.\downarrow y.(D \sqcap \exists r.x))$. Since y never occurs as a variable, we can omit the $\downarrow$ binder for y. Now we have again that $\mathcal{K} \models q_2$ iff $\mathcal{K} \cup \{\top \sqsubseteq \neg C_{q_2}\}$ is unsatisfiable. Further details and a proof that C_{q_2} indeed captures q_2 can be found in a technical report [6].

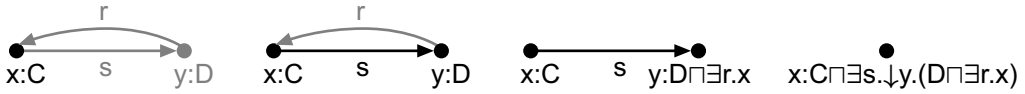

Figure 7: The graph traversal for the query graph of q_2 step by step. Parts already seen during the traversal are highlighted.

Even for cyclic queries, the rolling-up is now straightforward. If, however, the logic $\mathcal{L}$ under consideration does not provide for inverse roles, then this technique is only applicable in case each component in the query graph is also strongly connected. In the presence of weakly connected components, we would need inverse roles in order to roll-up a query. For example, let q_4 be the query $\langle\rangle \leftarrow x\!:\!C \wedge \langle x,y\rangle\!:\!s \wedge \langle x,y\rangle\!:\!r \wedge y\!:\!D$, i.e., both edges go now from x to y and x is not reachable from y. Therefore, the query graph is composed of two weakly connected components and a traversal of the query graph would get stuck at y. With inverse roles, q_4 could easily be expressed as $\downarrow x.(C \sqcap \exists s.(D \sqcap \exists \mathsf{Inv}(r).x))$. Tessaris [14] shows how arbitrary conjunctive queries for $\mathcal{SHQ}$ can be rolled-up

even without inverse roles, and an extension of this technique might work for $\downarrow$ binders and variables as well.

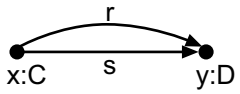

Figure 8: The query graph for q_4.

Unfortunately, there are no decision procedures for expressive DLs with the $\downarrow$ binder operator. It is even known that $\mathcal{ALC}$ extended with binders and state variables is undecidable [2]. However, we can observe some interesting properties for our query concepts. (1) After the rolling-up, all variables occur only positively. (2) Only existential restrictions are introduced in the rolling-up. Hence, negating the query concept for q_2 and transforming it into negation normal form yields $\downarrow x.(\neg C \sqcup \forall s.(\neg D \sqcup \forall r.\neg x))$ and we observe that all variables occur negated and are under the scope of universal quantifiers. For $\mathcal{ALC}$, these restrictions are enough to regain decidability [11].

4.1 A Tableaux Algorithm for $\mathcal{SHQ}$ Query Concepts

Our algorithm is based on the tableaux algorithm for $\mathcal{SHOQ}$ [7], which obviously decides KB satisfiability for $\mathcal{SHQ}$. Using this algorithm has the advantage that we can use nominals for the constants in the query. Alternatively, representative concepts and additional ABox assertions for each constant in a query could be used [14]. A tableaux algorithm builds a finite representation of a model, which is called a completion graph. The nodes in the completion graph represent elements in the domain and are labelled with a set of concepts. Labelled edges represent the relational structures among the elements. An initial completion graph is expanded according to a set of expansion rules. The expansion stops when an obvious contradiction, called a clash, occurs, or when no more rules are applicable. In the latter case, the completion graph is called complete. Termination is guaranteed by a technique called blocking. A complete and clash-free completion graph can be "unravelled" into a model for the given KB.

In order to handle query concepts that may contain binders and state variables, some adaptations of the existing algorithm are necessary. In order to store the bindings of variables, we modify the node labels s.t. they contain tuples of the form $\langle C, B\rangle$, where C is a concept and B is a (possibly empty) set of bindings of the form $\{y_1/v_1, \ldots, y_n/v_n\}$, where $y_1, \ldots, y_n$ are the free variables in C and $v_1, \ldots, v_n$ are nodes. The next obvious addition is a rule to handle concepts of the form $\downarrow y.C$: if $\langle \downarrow y.C, B\rangle$ is in the label of a node v, we add $\langle C, \{y/v\} \cup B\rangle$ and $\langle y, \{y/v\}\rangle$ to the label of v. The latter states that y is bound to v at v, and the former that the free variable y in C is bound to v. All other existing rules

have to propagate the bindings as well, e.g., the $\forall$-rule applied to $\langle \forall r.C, B\rangle$ in the label of a node v adds $\langle C, B\rangle$ to the labels of v's r-successors. The set B contains all and only the bindings for the free variables in C. Another obvious consequence is the addition of a new clash condition: if both $\langle y, \{y/v\}\rangle$ and $\langle \neg y, \{y/v\}\rangle$ are in the label of the node v, then this is a clash.

A more challenging task is the extension of the blocking condition. For $\mathcal{SHQ}$, however, we argue that we can simply ignore the bindings, i.e., if $\langle C, B\rangle$ is in the label, we consider only C in the blocking condition. This clearly ensures termination. But why does this guarantee that we can unravel a complete and clash-free completion forest into a model? Obviously, in $\mathcal{SHQ}$, there is no way for a node to propagate information "back" to its ancestors, and clashes according to the new clash condition can only occur through a cyclic structure. This is so because a node v is only labelled with $\langle y, \{y/v\}\rangle$ by an application of the new $\downarrow$-rule to some concept $\langle \downarrow y.C, B\rangle$ in the label of v. Furthermore, the only way $\neg y$ can occur with v as a binding is when $\langle C, \{y/v\} \cup B\rangle$ is expanded to $\langle \neg y, \{y/v\}\rangle$ via a cyclic path back to v. This is obviously only possible among individual nodes in $\mathcal{SHQ}$ and, therefore, no clash in the tableau can by caused by unravelling. Hence, transitive roles alone are not causing major problems.

An interesting consequence is, however, that we loose the finite model property. For example, let $\mathcal{K}$ contain the axioms $\top \sqsubseteq \exists r.\top$ and $\top \sqsubseteq \neg C_q$ with r a transitive role and $\neg C_q := \downarrow x.(\forall r.\neg x)$. The first axiom enforces an infinite r-chain for every individual. Normally, a finite model could contain an r-cycle instead of an infinite chain, but this would clearly violate $\neg C_q$. Hence, every model of $\mathcal{K}$ must be acyclic and therefore contain an infinite r-chain.

5 A Rolling-up Technique for $\mathcal{SHOQ}$

In this section, we show how the rolling-up technique can be extended to the DL $\mathcal{SHOQ}$, i.e., $\mathcal{SHQ}$ plus nominals. We do not use $\downarrow$ binders here, but rather propose an extension of the guessing technique as sketched in Section 3. In the presence of nominals, a simple non-deterministic assignment of individual names to variables that occur in a cycle is not sufficient. For example, Fig. 9 represents a model for the KB containing the axioms $\{a\} \sqsubseteq \neg C \sqcap \neg D \sqcap \exists s.(C \sqcap \exists r.(D \sqcap \exists s.\{a\}))$ and $\mathsf{trans}(s)$. The query $\langle\rangle \leftarrow x : C \wedge y : D \wedge \langle x, y\rangle : r \wedge \langle y, x\rangle : s$ (see Fig. 10) would clearly be true, although in the relevant assignments a cannot be bound to either x or y. Since the query graph for this query contains just one strongly connected component, no inverse roles are required in the rolling-up and we should be able to deal with this query.

For $\mathcal{SHOQ}$, our main argument for the correctness of an extension of the guessing of variable assignments is that a cycle among new nodes can only occur due to a transitive role that provides a shortcut for "skipping" the nominal.

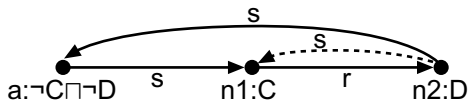

Figure 9: The dashed line indicates the relationship added due to s being transitive. Therefore, there is a cycle not directly containing the nominal a.

Hence, a nominal is always at least indirectly involved in a cycle. In this case, we have only one nominal a, and we may guess that it is either in the position of x, in the position of y, or it is "splitting" the (transitive) role s (see Fig. 11). In each case, we can roll-up the query graph into the nominal, obtaining the three query concepts $C_q^1 := \{a\} \sqcap C \sqcap \exists r.(D \sqcap \exists s.\{a\}), C_q^2 := \{a\} \sqcap D \sqcap \exists s.(C \sqcap \exists r.\{a\}), C_q^3 := \{a\} \sqcap \exists s.(C \sqcap \exists r.(D \sqcap \exists s.\{a\}))$. Identifying x with y gives the additional forth query concept $C_q^4 := \{a\} \sqcap C \sqcap D \sqcap \exists r.\{a\} \sqcap \exists s.\{a\}$. The query is true just in case one of these query concepts is entailed by the KB. Clearly, in our example, only the concept C_q^3 that corresponds to the "new" guess is entailed. If we had n nominals, then we would need to try $4n$ guesses.

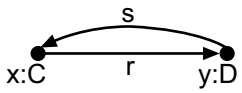

Figure 10: The original query graph.

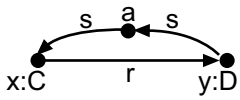

Figure 11: An alternative query graph in which the nominal a is assumed to be involved in the cycle.

Since it was, to the best of our knowledge, not even known if conjunctive query answering for logics with nominals and transitive roles is decidable, this technique is clearly valuable, although it is, due to its highly non-deterministic nature, not very practical.

6 The Challenges of Inverses and Nominals

Neither the arguments used for the extension of $\mathcal{SHQ}$ with binders (i.e., blocking can ignore different bindings), nor the extended guessing strategy for $\mathcal{SHOQ}$, can be used with inverse roles. Fig. 12 shows a representation of a model for the concept $\{a\} \sqcap \exists r.(\exists s.\top)$ for s a transitive and symmetric role. The query $\langle\rangle \leftarrow \langle x, x\rangle : s$ is obviously true in this model. The nominal a is, however, not even indirectly involved in the cycle.

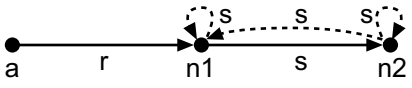

Figure 12: The dashed lines represent the additional relationships added for s being transitive and symmetric.

Since completion graphs are finite representations of models, the question is how far do we have to expand a completion graph before blocking is "safe", i.e., unravelling into a tableau does not lead to a clash.

In order to see the same problem from another perspective, we briefly sketch a different query answering algorithm in the style of CARIN [10]. CARIN provides a conjunctive query answering algorithm for the DL $\mathcal{ALCNR}$. Recall from Section 2 that a query q is true in a knowledge base $\mathcal{K}$, if there is a q-mapping for every model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{K}$. Similarly, we define a mapping ϕ from terms in a query to nodes in a completion graph $\mathbf{G}$ and we call ϕ *a q-mapping w.r.t.* $\mathbf{G}$ if, for each constant t in q, $\{t\}$ is in the label of $\phi(t)$ (i.e., constants are mapped to the corresponding nominal nodes), for each concept atom $t\!:\!C$ in q, C is in the label of $\phi(t)$ and, for each role atom $\langle t, t'\rangle\!:\!r$ in q, $\phi(t')$ is an r-descendant of $\phi(t)$, where r-descendant implicitly closes the transitive roles. Since ϕ is purely syntactic, we extend the KB with an axiom $\top \sqsubseteq C \sqcup \neg C$ for each concept C s.t. $t\!:\!C$ is a concept atom in q. Hence, we obtain a decision procedure for conjunctive query answering if we can show the following:

Claim 6.1
For each model $\mathcal{I}$ of $\mathcal{K}$, there is a q-mapping w.r.t. $\mathcal{I}$ iff for each completed and clash-free completion graph $\mathbf{G}$ of $\mathcal{K}$, there is a q-mapping w.r.t. $\mathbf{G}$.

In order to take the length of the paths in the query into account, blocking must be delayed appropriately. In CARIN (i.e., for a logic without transitive roles) this is achieved by using two isomorphic trees (instead of two isomorphic pairs as it is the case for normal pairwise blocking as in $\mathcal{SHIQ}$) s.t. the depth of the trees corresponds to the longest path in the query.

The "if" direction of Claim 6.1 is relatively straight forward but, for the "only if" direction, we have to show that (in contrapositive form), if there is a completion graph $\mathbf{G}$ for $\mathcal{K}$ s.t. there is no q-mapping ϕ w.r.t. $\mathbf{G}$, then there is a model $\mathcal{I}$ of $\mathcal{K}$ s.t. there is no q-mapping σ w.r.t. $\mathcal{I}$. We now give an example that shows that, even if we find such a completion graph $\mathbf{G}$, we cannot guarantee that there is no mapping σ w.r.t. the canonical model $\mathcal{I}$ of $\mathbf{G}$, if q contains a cycle with a transitive role occurring in it. Of course there may be another completion graph, for which this problem does not occur, but it is hard to prove that there is always a canonical counter-model for the query.

Let $\mathcal{K}$ be a KB containing the axiom $\top \sqsubseteq \exists r.\top \sqcup \exists s.\top$ for r a transitive and symmetric role and let q be the query $\langle\rangle \leftarrow \langle x, x\rangle\!:\!r \wedge \langle x, y\rangle\!:\!s \wedge \langle y, z\rangle\!:\!r \wedge \langle z, z\rangle\!:\!r \wedge x\!:\!C \wedge z\!:\!C$, see Fig. 13 right. The upper part of Fig. 13 shows a possible (simplified) completion graph $\mathbf{G}$ for $\mathcal{K}$, where C or $\neg C$ is added to the node labels to allow for a purely syntactic mapping ϕ. The grey underlying triangle shapes illustrate the two isomorphic trees used for blocking, and clearly there is no q-mapping w.r.t. $\mathbf{G}$. The partial representation of a model depicted in the lower part of Fig. 13, however, shows that, by unravelling $\mathbf{G}$, we would get

a model $\mathcal{I}$ of $\mathcal{K}$ s.t. there is a q-mapping w.r.t. $\mathcal{I}$. This is the case because there is no longer only one element in the extension of C, and all role relationships for q are satisfied since r is transitive and symmetric (inferred relationships are only pictured where they are relevant for the mapping). Even choosing a larger depth for the two trees would allow this to happen, since the decision for C or $\neg C$ and for an r- or s-successor was purely non-deterministic.

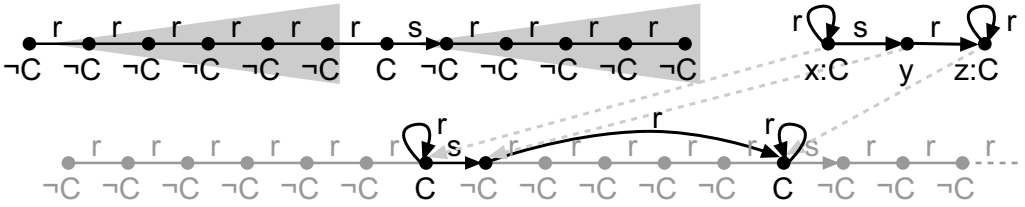

Figure 13: A completion graph **G** with trees for blocking (top), a partial unravelling of **G** (bottom), and a query graph (right).

Therefore, the absence of a q-mapping for the completion graph does not prove that there is a model for which there is no q-mapping. Requiring several isomorphic trees or pairs before blocking occurs obviously does not help either since only one part between the isomorphic trees/pairs could contain the twice required C in the node label. The problem is that, by repeating the unique part between the blocking and the blocked node, we can produce a structure that allows for a q-mapping.

A tempting solution for "safe" blocking conditions would be to require a path of n isomorphic sequences directly following each other, where n is the length of the longest path in the query. However, this will not guarantee termination, as the following argument shows: we could interpret each path from a root or nominal node as a word over the edge labels, e.g., the above example would produce a word starting with $rrrrrsrrrr$. For the suggested blocking condition, such a path must contain a substring of the form W^n, i.e., n repetitions of the word W, where W is any sequence of letters over our alphabet of roles. The additional blocking conditions for the node labels are irrelevant here. In 1944 Morse and Hedlund [12] showed that the so called Thue-Morse sequence, which forms a word over an alphabet of only two letters, is cube-free, which would correspond to a path of unbounded length in which no blocking occurs. Hence, the algorithm would not terminate for $n \geqslant 3$. For a three letter alphabet, this holds already for $n \geqslant 2$.

We encounter similar difficulties when using $\downarrow$ binders and variables for a logic like $\mathcal{SHIQ}$ or $\mathcal{SHOIQ}$. It is difficult to determine how "deep" the completion graph has to be before blocking is safe. If the completion graph is not "deep enough", unravelling does not yield a model.

Although for the CARIN technique, some canonical models with a q-mapping have completion graphs without a q-mapping, it seems as if by testing all possi-

ble completed and clash-free completion graphs, one would always find at least one completion graph such that also its canonical model does not provide for a q-mapping. However, this is only an observation, which could support the view that the problem is, despite all these problems, decidable. A different proof technique might be necessary to show this, and our future work is aimed at investigating this.

7 Conclusions

In the previous sections, we have presented two ideas that allow an extension of the rolling-up technique also to cyclic conjunctive queries for $\mathcal{SHQ}$ and $\mathcal{SHOQ}$. For $\mathcal{SHQ}$, we achieved this by extending the logic with a restricted form of $\downarrow$ binders and state variables as known from Hybrid Logics. This allows the expression of cyclic queries as concepts since, with $\downarrow$ binders and variables we can express a co-reference. Query entailment can then be reduced to deciding concept satisfiability for the extended DL. However, adding the $\downarrow$ binder, even in the very restricted form needed for query answering, has a notable impact on the resulting logic; e.g., for $\mathcal{SHQ}$, this extension leads to the loss of the finite model property. In Section 4.1, we illustrate how a tableaux algorithm for $\mathcal{SHQ}$ can be extended in order to handle query concepts. Although each $\downarrow$ introduces a fresh nominal on the fly, we show that, for $\mathcal{SHQ}$, termination can be regained by ignoring them in the blocking condition. Thus we have shown how conjunctive queries with transitive roles in the query body can be answered.

In Section 5, we extend the work by Calvanese et al. [4] to $\mathcal{SHOQ}$, i.e., to a logic that allows for nominals. We do this by proposing a more sophisticated guessing technique, which then again enables the rolling-up of a query into a $\mathcal{SHOQ}$-concept.

In Section 6, we highlight why none of the proposed techniques extends easily to a logic with inverse roles. We also show this for a query entailment algorithm in the style of CARIN [10]. The main problem is to decide when a completion graph is expanded "far enough" in order to safely predict that the query is also not entailed in its possibly infinite canonical model.

The rolling-up technique with binders and variables integrates seamlessly into the existing tableaux algorithms, whereas, for the CARIN-style algorithms, the search for a q-mapping is an additional and completely separated step. Moreover, the added axioms $\top \sqsubseteq C \sqcup \neg C$ for every concept C in the query, can significantly increase the amount of non-determinism. This makes binders and variables the more attractive choice from an implementation point of view.

Our future work will include efforts to show that answering arbitrary conjunctive queries for $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$ is decidable. If that is the case — and we believe it is — we aim at extending the rolling-up technique with binders

and state variables to $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOIQ}$. This would provide a (hopefully practical) decision procedure for arbitrary conjunctive queries over OWL-DL knowledge bases.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P.F̃. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995. Special issue on decompositions of first-order logic.

[3] P. Blackburn and J. Seligman. *Advances in Modal Logic*, volume 1, chapter What are hybrid languages?, pages 41–62. CSLI, 1998.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158. ACM, 1998.

[5] B. Glimm and I. Horrocks. Handling cyclic conjunctive queries. In *Proc. of DL'05*, Edinburgh, UK, July 26–28 2005. CEUR.

[6] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHOIQ}$. Technical report, The University of Manchester, 2006. Available online at `http://www.cs.man.ac.uk/~glimmbx/download/GlHS06b.pdf`.

[7] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$ description logic. In *Proc. of IJCAI'01*, pages 199–204. Morgan Kaufmann, 2001.

[8] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proc. of LPAR'00*, LNAI. Springer, 2000.

[9] U. Hustadt, B. Motik, and U. Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of LPAR'04*, volume 3452 of *LNCS*. Springer, 2004.

[10] A. Y. Levy and M.-C. Rousset. CARIN: A representation language combining horn rules and description logics. In *European Conf. on Artificial Intelligence*, pages 323–327, 1996.

[11] M. Marx. Narcissists, stepmothers and spies. In *Proc. of DL'02*, volume 53. CEUR, 2002.

[12] M. Morse and G. A. Hedlund. Unending chess, symbolic dynamics, and a problem in semigroups. *Duke Mathematical Journal*, 11(1):1–7, 1944.

[13] M. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence (AAAI'06)*, 2006. to appear.

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