INFSYS
RESEARCH
REPORT

INSTITUT FÜR INFORMATIONSSYSTEME

ARBEITSBEREICH WISSENSBASIERTE SYSTEME

QUERY ANSWERING IN DESCRIPTION LOGICS WITH TRANSITIVE ROLES

Thomas Eiter Carsten Lutz Magdalena Ortiz
Mantas Šimkus

INFSYS RESEARCH REPORT 1843-09-02
APRIL 2009

Institut für Informationssysteme
AB Wissensbasierte Systeme
Technische Universität Wien
Favoritenstrassße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-09-02, APRIL 2009

QUERY ANSWERING IN DESCRIPTION LOGICS WITH TRANSITIVE ROLES

Thomas Eiter,[1] Carsten Lutz,[2] Magdalena Ortiz,[3] Mantas Šimkus[4]

Abstract. We study the computational complexity of conjunctive query answering w.r.t. ontologies formulated in fragments of the description logic $\mathcal{SHIQ}$. Our main result is the identification of two new sources of complexity: the combination of transitive roles and role hierarchies which results in 2-EXPTIME-hardness, and transitive roles alone which result in coNEXPTIME-hardness. These bounds complement the existing result that inverse roles make query answering in $\mathcal{SHIQ}$ 2-EXPTIME-hard. We also show that conjunctive query answering with transitive roles, but without inverse roles and role hierarchies, remains in EXPTIME if the ABox is tree-shaped.

[1]Institute of Information Systems, Vienna University of Technology, Austria. E-mail: eiter@kr.tuwien.ac.at.
[2]Fachbereich Informatik,Universität Bremen, Germany. E-mail: clu@informatik.uni-bremen.de
[3]Institute of Information Systems, Vienna University of Technology, Austria. E-mail: ortiz@kr.tuwien.ac.at.
[4]Institute of Information Systems, Vienna University of Technology, Austria. E-mail: simkus@kr.tuwien.ac.at.

Acknowledgements: This work has been partially supported by the Austrian Science Fund (FWF) grant P20840, the Mexican National Council for Science and Technology (CONACYT) grant 187697, and the EU project OntoRule (IST-2009-231875).

Some of these results were published in the preliminary report 1843-08-09.
This report is an extended version of a paper that has appeared in *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, C. Boutilier (Ed.); pp. 759-764. Pasadena, California. July 2009. AAAI Press 2009.

Contents

1 Introduction **1**

2 Preliminaries **1**

3 Query Answering in $\mathcal{SH}$ **3**

3.1 Building Computation Trees 6

3.1.1 Enforcing Configuration Nodes 6

3.1.2 Enforcing Computation Trees 7

3.2 Testing Properness of Computation Trees 9

4 Query Answering in $\mathcal{S}$ **11**

4.1 A Lower Bound 11

4.2 CQs over Tree-shaped ABoxes 16

4.2.1 Markings for canonical models 17

4.2.2 A knot-elimination algorithm 24

5 Related Work and Conclusions **27**

1 Introduction

One of the main applications of ontologies in computer science is in data access, where an ontology formalizes conceptual information about data stored in one or multiple data sources, and this information is used to derive answers when querying the sources. This general setup plays a central role e.g. in the Semantic Web, in ontology-based information integration, and in peer-to-peer data management. In all these areas, Description Logics (DLs) and in particular those of the OWL standard by the W3C are popular ontology languages, and conjunctive queries (CQs) are used as a fundamental querying mechanism, cf. [16, 4, 10] and references therein and below.

Inspite of the prominent applications, the study of algorithms for and the computational complexity of answering CQs over OWL ontologies has only recently gained momentum. In particular, it was shown that inverse roles have an impact on complexity: (a) CQ entailment (the decisional variant of CQ answering) over ontologies in the primary OWL fragment $\mathcal{SHIQ}$ is 2-EXPTIME-complete [4, 8]; and (b) the complexity drops to EXPTIME-complete if inverse roles are disallowed ($\mathcal{SHIQ}$ is replaced with $\mathcal{SHQ}$) and, additionally, the use of transitive roles in queries is disallowed or seriously restricted, cf. [8, 11].

From an application perspective, such restrictions are highly unsatisfactory, as transitive roles play a crucial role in most ontologies and are used to represent fundamental relations such as "part of" [14]. A main reason why they are often adopted is that that algorithms for CQ entailment with (unrestricted) transitive roles in the query become much more intricate, see e.g. [3, 1] where algorithms establishing 2-EXPTIME upper bounds for $\mathcal{SHIQ}$ were provided.

The aim of this paper is to study the computational complexity of CQ entailment in fragments of $\mathcal{SHIQ}$ with no restrictions on transitive roles in queries. Our main contribution is to identify two novel sources of complexity: (1) the combination of transitive roles and role hierarchies and, to a lesser degree, (2) transitive roles alone. More precisely, we first show that CQ entailment in $\mathcal{SH}$ ($\mathcal{SHIQ}$ without inverse roles and number restrictions, i.e., $\mathcal{ALC}$ extended with transitive roles and role hierarchies) is 2-EXPTIME-hard, and thus 2-EXPTIME-complete. Thus, inverse roles are *not* the only reason why CQ entailment in $\mathcal{SHIQ}$ is harder than standard reasoning tasks such as satisfiability and subsumption, which are EXPTIME-complete. Interestingly, 2-EXPTIME-hardness is hit already with a single role inclusion, or alternatively with a single left identity $r \circ t \sqsubseteq t$. Secondly, we prove that CQ entailment in $\mathcal{S}$ ($\mathcal{SH}$ without role hierarchies, i.e., $\mathcal{ALC}$ extended with transitive roles) is NEXPTIME-hard, and thus also harder than standard reasoning. The lower bound applies already to the case where TBoxes (which contain the conceptual information) are empty.

On the other hand, we show that CQ entailment in $\mathcal{S}$ ontologies where the ABoxes (data parts) have a tree-shaped relational structure is in EXPTIME, and thus EXPTIME-complete. This result is interesting for three reasons. Firstly, it is the first EXPTIME result for CQ entailment in an expressive DL with unrestricted transitive roles in queries. Secondly, to the best of our knowledge, this is the first case where CQ entailment for tree-shaped ABoxes is easier than the general case: in all existing lower bounds for CQ entailment in fragments of $\mathcal{SHIQ}$, the ABox contains *no* role assertions at all. Thirdly, EXPTIME membership may be viewed as an indication that the complexity in the general case is likely to be below 2-EXPTIME; a tight upper bound is currently open.

2 Preliminaries

Knowledge Bases. We assume standard notation for the syntax and semantics of $\mathcal{SH}$ knowledge bases [4]. In particular, $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ are countably infinite and disjoint sets of *concept names*, *role names*, and *individual names*. *Concepts* are inductively defined: (a) each $A \in \mathsf{N_C}$ is a concept, and (b) if C, D are concepts and $r \in \mathsf{N_R}$ is a role, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$ and $\exists r.C$ are concepts. A *TBox* is a set of concept inclusions $C \sqsubseteq D$, role inclusions $r \sqsubseteq s$, and transitivity statements $\mathsf{trans}(r)$. An *ABox* is a set of *assertions* $C(a)$ and $r(a,b)$. A *knowledge base (KB)* is a pair $(\mathcal{T}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We use $\mathcal{I}$ to denote an interpretation for a KB, $\Delta^{\mathcal{I}}$ for its domain, and $C^{\mathcal{I}}$ and $r^{\mathcal{I}}$ for the interpretation of a concept C and of a role r, respectively.

A role is *transitive* if there is some r' with $\mathsf{trans}(r) \in \mathcal{K}$ and $r' \sqsubseteq^* r$, where $\sqsubseteq^*$ denotes the reflexive transitive closure of $r' \sqsubseteq r \in \mathcal{K}$.

$\mathcal{S}$ is the fragment of $\mathcal{SH}$ that disallows role hierarchies.

We use the following notation. By $\mathsf{Ind}(\mathcal{A})$ we denote the set of all individual names in an ABox $\mathcal{A}$; by $\mathsf{sub}(\mathcal{K})$ we denote the set of all subconcepts of concepts occurring in a KB $\mathcal{K}$, and $\mathsf{Tr}(\mathcal{K}) := \{r \in \mathsf{N_R} \mid \mathsf{trans}(r) \in \mathcal{T}\}$.

Conjunctive Query Answering. Let $\mathsf{N_V}$ be a countably infinite set of *variables*. A *conjunctive query* (CQ) over a KB $\mathcal{K}$ is a finite set of atoms of the form $A(v)$ or $r(v, v')$, where $v, v' \in \mathsf{N_V}$, A is a concept name and r is a role, both occurring in $\mathcal{K}$.[1] For a CQ q over $\mathcal{K}$, let $\mathsf{Var}(q)$ denote the variables occuring in q. A *match for q in an interpretation $\mathcal{I}$* is a mapping $\pi : \mathsf{Var}(q) \to \Delta^{\mathcal{I}}$ such that (i) $\pi(v) \in A^{\mathcal{I}}$ for each $A(v) \in q$, and (ii) $(\pi(v), \pi(v')) \in r^{\mathcal{I}}$ for each $r(v, v') \in q$. We write $\mathcal{I} \models q$ if there is a match for q in $\mathcal{I}$. If $\mathcal{I} \models q$ for every model $\mathcal{I}$ of $\mathcal{K}$, then $\mathcal{K}$ *entails* q, written $\mathcal{K} \models q$. The *query entailment problem* is to decide, given $\mathcal{K}$ and q, whether $\mathcal{K} \models q$.

Forest Models. In many DLs, it suffices to concentrate on certain regular models of the input KB $\mathcal{K}$ when deciding CQ entailment. We describe these models for $\mathcal{SH}$. A *forest base for $\mathcal{K}$* is an interpretation $\mathcal{J}$ that satisfies:

(i) $\Delta^{\mathcal{J}}$ is a prefix-closed subset of ω^+; elements of $\Delta^{\mathcal{J}} \cap \omega$ are called the *roots* of $\mathcal{J}$.

(ii) If $(d, e) \in r^{\mathcal{J}}$ for some r, then either e and d are roots of $\mathcal{J}$, or $e = d \cdot c$ for some $c \in \omega$; in the latter case, there is no $r' \neq r$ with $(d, e) \in r'^{\mathcal{J}}$.

(iii) For every $a \in \mathsf{Ind}(\mathcal{A})$, $a^{\mathcal{I}}$ is a root of $\mathcal{J}$, and for every root d of $\mathcal{J}$, there is an $a \in \mathsf{Ind}(\mathcal{A})$ such that $a^{\mathcal{I}} = d$.

An interpretation $\mathcal{I}$ is a *forest model* of $\mathcal{K}$ if it is a model of $\mathcal{K}$ and there is a forest base $\mathcal{J}$ for $\mathcal{K}$ such that $\mathcal{I}$ is identical to $\mathcal{J}$ except that, for all transitive roles r, $r^{\mathcal{I}} = (r^{\mathcal{J}})^+$. Then, the *roots* of $\mathcal{I}$ are defined as the roots of $\mathcal{J}$.

Proposition 2.1 *Let $\mathcal{K}$ be an $\mathcal{SH}$-knowledge base and q a UCQ. If $\mathcal{K} \not\models q$, then there is a forest model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

Alternating Turing Machines. The 2-ExpTime-hardness result of this paper relies on a reduction from the word problem for *Alternating Turing machines* (ATMs) with exponential work space, whose definition we briefly recall; see e.g., [2] for background and details.

An ATM is given by a tuple $\mathcal{M} = (Q, \Sigma, q_0, \delta)$, where

- $Q = Q_\exists \uplus Q_\forall \uplus \{q_{\mathsf{acc}}\} \uplus \{q_{\mathsf{rej}}\}$, the set of *states*, consists of *existential states* in $Q_\exists$, *universal states* in $Q_\forall$, an *accepting state* q_{acc}, and a *rejecting state* q_{rej};
- Σ is the *alphabet* that additionally contains the *blank symbol* $\llcorner\!\lrcorner$;

[1]Individuals in q can be simulated and queries with answer variables can be reduced to the considered Boolean CQs as usual.

- $q_0 \in Q_\exists \cup Q_\forall$ is the *starting* state; and
- $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{+1, -1\}$ is the *transition relation*.

For later use, we define $\delta(q, \sigma) := \{(q', \sigma', M) \mid (q, \sigma, q', \sigma', M) \in \delta\}$.

A *configuration* of $\mathcal{M}$ is a word wqw' with $w, w' \in \Sigma^*$ and $q \in Q$, whose intended meaning is that the one-side infinite tape contains the string ww' with only blanks behind it, that the machine is in state q, and that the head is on the symbol just after w. The *successor configurations* of a configuration wqw' are defined in terms of δ as usual; without loss of generality, we assume that $\mathcal{M}$ is well-behaved and never attempts to move left if the head is on the left-most position. A *halting configuration* is of the form wqw' where $q \in \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}$.

A *computation* of an ATM $\mathcal{M}$ on a word w is a sequence of configurations $K_0, K_1, \ldots$ such that $K_0 = q_0 w$ (the *initial configuration*) and K_{i+1} is a successor configuration of K_i, for all $i \geq 0$. For our concerns, we may assume that all computations are finite (on any input), and define acceptance only for this case.

A configuration wqw' is *accepting*, if either (a) $q = q_{\mathsf{acc}}$, or (b) $q \in Q_\exists$ and at least one of its successor configurations is accepting, or (c) $q \in Q_\forall$ and all of its successor configurations are accepting. The ATM $\mathcal{M}$ *accepts* the input w, if the *initial configuration* is accepting. The *word problem of* $\mathcal{M}$ is, given $\mathcal{M}$ and w, to decide whether $\mathcal{M}$ accepts w. We use the following lemma.

Lemma 2.2 ([2]) *There is an ATM $\mathcal{M}$ for which the word problem is* 2-ExpTime*-hard and such that $\mathcal{M}$ works in exponential space, i.e., all configurations $w'qw''$ in computations on w fulfill $|w'w''| \leq 2^{|w|}$.*

3 Query Answering in $\mathcal{SH}$

It follows from a number of existing results that CQ entailment in $\mathcal{SH}$ is in 2-ExpTime [1, 3, 5, 11]. We provide a matching lower bound.

Theorem 3.1 *CQ entailment in $\mathcal{SH}$ is* 2-ExpTime*-complete.*

To prove the hardness part, we reduce the word problem for an exponentially space bounded ATM $\mathcal{M} = (Q, \Sigma, q_0, \delta)$ and an input word w (by Lemma 2.2, this shows 2-ExpTime-hardness).

Recall that the state set Q of an ATM is partitioned into *existential* ($Q_\exists$) and *universal* ($Q_\forall$) states. An ATM with only existential states can be viewed as a standard non-deterministic TM, which accepts a word iff there exists a sequence of successive configurations that starts in the *initial configuration*, with initial state q_0 and the input word w on the tape, and ends in an accepting state q_{acc}. For ATMs, these sequences become *trees* of configurations, where branching is caused by universal states (there is a successive configuration for each transition in $\delta(q, a)$ with $q \in Q_\forall$). Such a tree is a *computation tree*, and it is *accepting* if q_{acc} is reached on all paths. For details, please see [2].

For each input w to $\mathcal{M}$, we define a KB $\mathcal{K}_w$ and a query q_w such that $\mathcal{M}$ accepts w iff $\mathcal{K}_w \not\models q_w$. In fact, each forest model $\mathcal{I}$ of $\mathcal{K}_w$ with $\mathcal{I} \not\models q_w$ will represent an accepting computation of $\mathcal{M}$ on w. More precisely, such a model is an accepting computation tree in which each node is the root of a *configuration tree*. The latter are binary trees of depth $m := |w|$ (length of w) that represent configurations using their 2^m leaves to store the tape contents. This is illustrated in Figure 1; the initial configuration tree is existential and thus has a single successor configuration tree. Its (magnified) successor is universal and has two successor configuration trees.

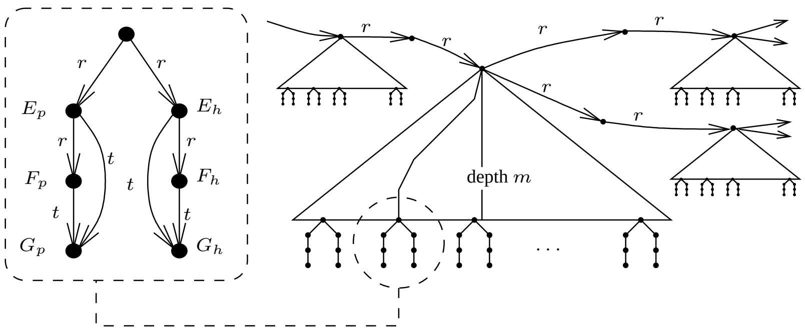


Figure 1: The structure of models.

To enforce this structure, we need some technical tricks. In particular, each configuration tree will represent *two* configurations: the *current configuration* K_h and the *previous* configuration K_p. We use $\mathcal{K}_w$ to ensure locally at each configuration tree that K_h is indeed a successor configuration of K_p. The query q_w is then used to globally guarantee that the K_p value of each configuration tree is identical to the K_h value of the predecessor in the computation tree. We will call a computation tree *proper*, if it satisfies the latter condition.

We now give a precise definition of how configuration trees and computation trees are represented as a model. A single, non-transitive role r is used for the edges of computation trees and of configuration trees. Observe that, as shown in Figure 1, we use *two* r-edges between two consecutive configuration trees. We also use a transitive role t, to be explained later. The alphabet symbols Σ of $\mathcal{M}$ and the states Q are used as concept names. We also use the concept names from $\mathbf{B} := \{B_1, \ldots, B_m\}$ to encode addresses of tape cells in binary. For a node n of a forest model $\mathcal{I}$ and $i < 2^m$, we write $\mathsf{adr}^{\mathcal{I}}(n) = i$ if the truth values of $B_1^{\mathcal{I}}, \ldots, B_m^{\mathcal{I}}$ at n encode the number i. A tape cell with address i and content $a \in \Sigma$ is represented by a node n with $\mathsf{val}^{\mathcal{I}}(n) = i$ that satisfies the concept name a. If the head is currently on the cell and $\mathcal{M}$'s state is q, then n also satisfies q; otherwise, n satisfies the concept name nil.

To later on ensure properness using the query, we use additional nodes and concept names. The latter are E_h, E_p, F_h, F_p, G_h, and G_p, used as markers; and the concept names from $\mathbf{Z} := \{Z_{a,q} \mid a \in \Sigma, q \in Q \cup \{nil\}\}$. The additional nodes are attached to the leaves of configuration trees, as indicated on the left-hand side of Figure 1 and detailed in the subsequent definition. Intuitively, nodes labeled E_h store the current configuration and nodes labeled E_p the previous.

Definition 3.2 [i-cell] Let $\mathcal{I}$ be an interpretation and $i < 2^m$. We call $n \in \Delta^{\mathcal{I}}$ an *i-cell* if the following hold:

(a) n has r-successors n_p and n_h with $\mathsf{adr}^{\mathcal{I}}(n_p) = \mathsf{adr}^{\mathcal{I}}(n_h) = i$ that satisfy E_p respectively E_h, and both satisfy exactly one $a \in \Sigma$ and exactly one $q \in Q \cup \{nil\}$.

(b) n_p (resp., n_h) has an r-successor n'_p (resp., n'_h) satisfying F_p (resp., F_h) and such that $\mathsf{adr}^{\mathcal{I}}(n'_p)$ (resp., $\mathsf{adr}^{\mathcal{I}}(n'_h)$) is the bit-wise complement of i. Furthermore, for all $a \in \Sigma$ and $q \in Q \cup \{nil\}$, we have:

  (i) n_h satisfies $Z_{a,q}$ iff n_h does not satisfy both a and q;

  (ii) n'_p satisfies $Z_{a,q}$ iff n_p does not satisfy both a and q;

  (iii) n'_h and n_p satisfy $Z_{a,q}$;

(c) n'_p (resp., n'_h) has a t-successor n''_p satisfying G_p (resp., n''_h satisfying G_h) such that n''_p (resp., n''_h) is also a t-successor of n_p (resp., n_h).

We simply speak of a cell if i is unimportant. Note that the ability of $\mathcal{SH}$ to express (c) in Definition 3.2 via the axioms $r \sqsubseteq t$ and $\mathsf{trans}(t)$ is crucial for the reduction. The same condition can be expressed via a so-called *left identity* $r \circ t \sqsubseteq t$.

We now define (q, a, i)-configuration nodes, which are the roots of configuration trees, and (models that encode) computation trees. A node n' is an r^m-successor of a node n, if n' is reachable from n by travelling m r-edges.

Definition 3.3 [(q, a, i)-configuration node, Computation tree] Let $\mathcal{I}$ be an interpretation. We call $n \in \Delta^{\mathcal{I}}$ a (q, a, i)-*configuration node* if (1) it has an r^m-successor that is a j-cell (called *j-cell of n*), for each $j < 2^m$ and (2) the E_h-node of the i-cell of n satisfies q and a, and all other j-cells have nil in their E_h-nodes.
We call $\mathcal{I}$ a *computation tree* for w if $\mathcal{I}$ is tree-shaped and

(I) the root ϵ of $\mathcal{I}$ has an r-successor n that is a $(q_0, a, 0)$-configuration node whose i-cells describe the initial configuration for input w;

(II) for each (q, a, p)-configuration node n, if $q \in Q_\exists$ (resp., $q \in Q_\forall$), then for some (resp., for each) tuple $(q', a', M) \in \delta(q, a)$ there exists an r^2-successor node n' that is an (q', a'', p')-configuration node with $p' = p + M$, where $M \in \{-1, +1\}$ is the executed move. Furthermore, the E_h node of a p-cell of n' satisfies a', and, for all remaining j-cells c of n' with $j \neq p$, if the E_p node of c satisfies $a \in \Sigma$, then the E_h node of c also satisfies a (i.e., a p-cell has the new symbol written, while for the remaining cells, the E_h nodes in the resulting configuration tree carry over the symbols from their respective E_p nodes).[2]

We call $\mathcal{I}$ *accepting*, if $q = q_{\mathsf{acc}}$ in each (q, a, i)-configuration for which there is no successor configuration. Furthermore, $\mathcal{I}$ is *proper*, if for each pair of successive configuration nodes n, n' as in Definition 3.3.II and each $i < 2^m$, the i-cell of n has the same (q, a)-label in its E_h-node as the i-cell of n' in its E_p-node.

It is not hard to see that there is a correspondence between accepting proper computation trees for w and accepting computations of $\mathcal{M}$ on w. The properness condition ensures that the *previous* configuration encoded in the E_p nodes of a configuration tree coincides with the *current* configuration encoded in the E_h nodes of the previous configuration tree. Then, due to the condition (II) in the above Definition 3.3, we get that each pair of successive configuration nodes encodes a correct transition of $\mathcal{M}$. On the other hand, given an accepting run of $\mathcal{M}$ on w, we can define an accepting computation tree.

Proposition 3.4 *$\mathcal{M}$ accepts w iff there exists an accepting proper computation tree for w.*

In the next section, we define an $\mathcal{SH}$ knowledge base capturing (proper and improper) computation trees, and in the subsequent section, we define a query for testing properness.

[2]The second part of condition (II) is not present in the submission, but is in fact needed for a complete argument. It realizes the intuition stated in the beginning of the section, viz. that the configuration encoded in the E_h nodes is a valid successor of the configuration encoded in the E_p nodes.

3.1 Building Computation Trees

Proposition 3.5 *Given w, we can build in polynomial time a KB $\mathcal{K}_w$ whose forest models are exactly the accepting computation trees for w.*

In the following, by constructing $\mathcal{K}_w$, we provide a proof of the above proposition. We define

$$\mathcal{K}_w = \langle \{a : I\}, \mathcal{T}_w \rangle$$

where a is an individual, I is a concept name (that identifies the initial node), and the TBox $\mathcal{T}_w$ contains the axioms described below.

3.1.1 Enforcing Configuration Nodes

Recall that configuration nodes are roots of binary trees of depth m whose leaves are i-cells corresponding to tape cells of $\mathcal{M}$. We next provide axioms enforcing conditions (1) and (2) in the definition of configurations nodes (see Definition 3.3). More precisely, nodes satisfying a special concept name R are forced to be configuration nodes. For technical reasons, the $m+1$ levels of a tree rooted at a configuration node are identified with concept names $L_0, \ldots, L_m$. For two concepts C and D, we use $C \rightarrow D$ as a shorthand for the concept $\neg C \sqcup D$. We introduce the following axioms, which generate a tree whose leaves cover the address range $0, \ldots, 2^m - 1$:

$$\begin{array}{rcll}
R & \sqsubseteq & L_0 & \\
L_i & \sqsubseteq & \exists r.(L_{i+1} \sqcap B_{i+1}) \sqcap \exists r.(L_{i+1} \sqcap \neg B_{i+1}) & \text{for all } 0 \leq i < m \\
L_i \sqcap B_j & \sqsubseteq & \forall r.(L_{i+1} \rightarrow B_j) & \text{for all } 0 < j \leq i < m \\
L_i \sqcap \neg B_j & \sqsubseteq & \forall r.(L_{i+1} \rightarrow \neg B_j) & \text{for all } 0 < j \leq i < m
\end{array}$$

Recall that the leaves of configuration trees must be i-cells as prescribed by Definition 3.2, and hence the properties (a)-(c) must be enforced. To enforce (a), we use the symbols from Σ, the states from Q and nil as concept names. We label such nodes with exactly one concept from Σ (the content of a cell c), and with exactly one concept from $Q^+ := Q \cup \{nil\}$; intuitively, the label $q \in Q$ means that the head of $\mathcal{M}$ is on the tape cell c and that $\mathcal{M}$ is in state q, while the label nil means that the head is not at position j. The above is realized as follows:

$$\begin{array}{rcl}
L_m & \sqsubseteq & \exists r.(E_p \sqcap E) \sqcap \exists r.(E_h \sqcap E) \\[1em]
E & \sqsubseteq & \bigsqcup_{a \in \Sigma} a \sqcap \bigsqcap_{a \neq a' \in \Sigma} \neg(a \sqcap a') \\[1em]
E & \sqsubseteq & \bigsqcup_{q \in Q^+} q \sqcap \bigsqcap_{q \neq q' \in Q^+} \neg(q \sqcap q').
\end{array}$$

To enforce the structures as prescribed in the remaining properties (b)-(c), we use the following axioms:

1. The existence of the required nodes is via the following axioms:

$$\begin{array}{rcl}
E_p & \sqsubseteq & \exists r.(F_p \sqcap \exists t.G_p) \\
E_h & \sqsubseteq & \exists r.(F_h \sqcap \exists t.G_h)
\end{array}$$

2. The address for E_p and E_h nodes, and its bitwise complement in F_p and F_h nodes is obtained by adding for each $1 \leq i \leq m$ the following axioms:

$$\begin{array}{rcl} L_m \sqcap B_i & \sqsubseteq & \forall r.B_i \\ L_m \sqcap \neg B_i & \sqsubseteq & \forall r.\neg B_i \\ E \sqcap B_i & \sqsubseteq & \forall r.\neg B_i \\ E \sqcap \neg B_i & \sqsubseteq & \forall r.B_i \end{array}$$

3. The conditions (b.i)-(b.iii) are enforced by the following axioms: for all $a \in \Sigma, q \in Q^+$,

$$\begin{array}{rcl} E_h & \sqsubseteq & (a \sqcap q) \leftrightarrow \neg Z_{a,q} \\ E_p & \sqsubseteq & (a \sqcap q) \rightarrow \forall r.(\neg Z_{a,q} \sqcap \bigsqcap_{(a,q)\neq(a',q')} Z_{a',q'}) \\ E_h & \sqsubseteq & \forall r.Z_{a,q} \\ E_p & \sqsubseteq & Z_{a,q} \end{array}$$

4. Finally, to enforce (c), we add $r \sqsubseteq t$ and $\mathsf{trans}(t)$.

It remains to ensure that each node n that satisfies R also satisfies that for exactly one address $i < 2^m$, the i-cell of n satisfies some $q \in Q$ and all j-cells, $j \neq i$, satisfy nil (cf. (2) in Definition 3.3). To achieve this, we use a concept name H (for the head position) and make sure that it occurs in the label of an L_m node iff its address is i, and that only an E_h successor of such an L_m node contains labels from Q.

$$\begin{array}{rcl} L_0 & \sqsubseteq & H \\ (L_i \sqcap H) & \sqsubseteq & (\forall r.((L_{i+1} \sqcap B_i) \rightarrow H) \sqcap \forall r.((L_{i+1} \sqcap \neg B_i) \rightarrow \neg H)) \\ & & \sqcup (\forall r.((L_{i+1} \sqcap \neg B_i) \rightarrow H) \sqcap \forall r.((L_{i+1} \sqcap B_i) \rightarrow \neg H)) \quad \text{for all } 0 \leq i < m \\ (L_i \sqcap \neg H) & \sqsubseteq & (\forall r.(L_{i+1} \rightarrow \neg H) \quad \text{for all } 1 \leq i < m \\ L_m \sqcap H & \sqsubseteq & \forall r.(E_h \rightarrow \bigsqcup_{q \in Q} q) \\ L_m \sqcap \neg H & \sqsubseteq & \forall r.(E_h \rightarrow nil) \end{array}$$

We remark here that for configurations represented by E_p nodes we omit here adding similar axioms. Indeed, the query q_w that we construct will, as a byproduct, also check whether a state $q \in Q$ is stored at exactly one address for E_p nodes.

3.1.2 Enforcing Computation Trees

To generate computation trees, we add axioms ensuring that tree-shaped models of $\mathcal{K}_w$ satisfy conditions (I) and (II) in Definition 3.3. In the following, we use $\forall r^i.C$ to denote the i-fold nesting $\forall r. \cdots \forall r.C$. In particular, $\forall r^0.C$ is C.

The initial configuration as described in (I) is ensured as follows. Let $w = a_0 \cdots a_n$ be the initial word. We will additionally keep track of the position of the R/W head of $\mathcal{M}$. To this end, we use concept names $Q'_1, \ldots, Q'_m$ and $Q_1, \ldots, Q_m$ for the previous position and the current position resulting due to a transition.

We add the following:

$$\begin{array}{rcll}
I & \sqsubseteq & \exists r.R & \\
I & \sqsubseteq & \forall r^{m+1}.(\mathsf{pos} = i \rightarrow \forall r.(E_h \rightarrow a_i)) & \text{for all } i < n \\
I & \sqsubseteq & \forall r^{m+1}.(\mathsf{pos} = 0 \rightarrow \forall r.(E_h \rightarrow q_0)) & \\
I & \sqsubseteq & \forall r^{m+1}.(\mathsf{pos} \geq n \rightarrow \forall r.(E_h \rightarrow \llcorner\!\lrcorner)) & \\
I & \sqsubseteq & \forall r.\neg Q_i & \text{for all } 1 \leq i \leq m
\end{array}$$

where $(\mathsf{pos} = i)$ and $(\mathsf{pos} \geq n)$ are the obvious (Boolean) concepts expressing that the value of the address $B_1, \ldots, B_m$ equals i and is at least n, respectively (recall that $\llcorner\!\lrcorner$ is the blank symbol).

We turn to the condition (II) in Definition 3.3. In detail, to represent that a configuration node n' is a successor of a configuration node n upon taking the transition $(q', a', M) \in \delta(q, a)$, we label n' with the concept name $T_{q',a',M}$ and we connect n to n' via two consecutive r arcs. Furthermore, if q is existential, we enforce that some n' exists with suitable label $T_{q',a',M}$, and if q is universal, we enforce that for each $(q', a', M) \in \delta(q, a)$ some n' exists with label $T_{q',a',M}$; we exploit that the state q and the symbol a are stored in an E_h-node of n, for one unique address. We also ensure that the address of R/W head is copied to the follow-up configuration nodes.

$$\begin{array}{rcll}
R \sqcap \exists r^{m+1}.(E_h \sqcap q \sqcap a) & \sqsubseteq & \bigsqcup_{(q',a',M)\in\delta(q,a)} \exists r^2.(R \sqcap T_{q',a',M}) & \text{for all } q \in Q_\exists, a \in \Sigma, \\
R \sqcap \exists r^{m+1}.(E_h \sqcap q \sqcap a) & \sqsubseteq & \bigsqcap_{(q',a',M)\in\delta(q,a)} \exists r^2.(R \sqcap T_{q',a',M}) & \text{for all } q \in Q_\forall, a \in \Sigma. \\
Q_i & \sqsubseteq & \forall r^2 Q'_i & \text{for all } 0 < i < m \\
\neg Q_i & \sqsubseteq & \forall r^2 \neg Q'_i & \text{for all } 0 < i < m
\end{array}$$

Next we provide axioms that define the position of the R/W head resulting by transition. It is obtained by applying addition or subtraction to the address encoded by Q'_i concepts. We use $INV_1, \ldots, INV_m$ to decide on the bits that need to be inverted:

$$\begin{array}{rcll}
T_{q,a,+1} & \sqsubseteq & PLUS & \text{for all} q \in Q, a \in \Sigma, \\
T_{q,a,-1} & \sqsubseteq & MINUS & \text{for all} q \in Q, a \in \Sigma, \\
R \sqcap PLUS & \sqsubseteq & INV_m & \\
R \sqcap Q'_i \sqcap INV_i \sqcap PLUS & \sqsubseteq & INV_{i-1} & \text{for all } 1 < i \leq m \\
R \sqcap PLUS \sqcap (\neg Q'_i \sqcup \neg INV_i) & \sqsubseteq & \neg INV_{i-1} & \text{for all } 1 < i \leq m \\
R \sqcap MINUS & \sqsubseteq & INV_m & \\
R \sqcap \neg Q'_i \sqcap INV_i \sqcap MINUS & \sqsubseteq & INV_{i-1} & \text{for all } 1 < i \leq m \\
R \sqcap MINUS \sqcap (Q'_i \sqcup \neg INV_i) & \sqsubseteq & \neg INV_{i-1} & \text{for all } 1 < i \leq m \\
Q'_i \sqcap INV_i & \sqsubseteq & \neg Q_i & \text{for all } 1 \leq i \leq m \\
Q'_i \sqcap \neg INV_i & \sqsubseteq & Q_i & \text{for all } 1 \leq i \leq m \\
\neg Q'_i \sqcap INV_i & \sqsubseteq & Q_i & \text{for all } 1 \leq i \leq m \\
\neg Q'_i \sqcap \neg INV_i & \sqsubseteq & \neg Q_i & \text{for all } 1 \leq i \leq m
\end{array}$$

We also propagate the two addresses to the leaves by adding, for each $0 < j \leq m$, the following axioms:

$$\begin{array}{rcll} L_i \sqcap Q_j & \sqsubseteq & \forall r.(L_{i+1} \rightarrow Q_j) & \text{for all } 0 \leq i < m \\ L_i \sqcap \neg Q_j & \sqsubseteq & \forall r.(L_{i+1} \rightarrow \neg Q_j) & \text{for all } 0 \leq i < m \\ L_i \sqcap Q'_j & \sqsubseteq & \forall r.(L_{i+1} \rightarrow Q'_j) & \text{for all } 0 \leq i < m \\ L_i \sqcap \neg Q'_j & \sqsubseteq & \forall r.(L_{i+1} \rightarrow \neg Q'_j) & \text{for all } 0 \leq i < m. \end{array}$$

To enforce the second part of condition (II), we make sure that for a configuration node n satisfying $T_{q',a',M}$, the symbol in the previous position of the R/W head is changed to a', while the symbols in other positions are transferred from E_p nodes to E_h nodes. The first part is done by adding, for all $q' \in Q$, $a' \in \Sigma$, $M \in \{+1, -1\}$ the axioms:

$$T_{q',a',M} \sqsubseteq \forall r^m.(L_m \rightarrow T_{q',a',M}),$$

$$L_m \sqcap T_{q',a',M} \sqcap \vec{Q}' = \vec{B} \sqsubseteq \forall r.(E_h \rightarrow a'),$$

$$L_m \sqcap T_{q',a',M} \sqcap \vec{Q} = \vec{B} \sqsubseteq \forall r.(E_h \rightarrow q'),$$

where $\vec{Q} = \vec{B}$ stands for $\bigsqcap_{0<i\leq m}((Q_i \sqcap B_i) \sqcup (\neg Q_i \sqcap \neg B_i))$ and $\vec{Q}' = \vec{B}$ for $\bigsqcap_{0<i\leq m}((Q'_i \sqcap B_i) \sqcup (\neg Q'_i \sqcap \neg B_i))$.

All remaining tape cells do not change:

$$L_m \sqcap \exists r.(E_p \sqcap a \sqcap nil) \sqsubseteq \forall r.(E_h \rightarrow a) \quad \text{for all } a \in \Sigma.$$

This concludes the definition of the TBox $\mathcal{T}_w$, and hence of the KB $\mathcal{K}_w$. By construction, all forest-shaped models of $\mathcal{K}_w$ satisfy the conditions in Definition 3.3, and hence are computation trees.

3.2 Testing Properness of Computation Trees

As already mentioned, we use the query q_w to test whether the tree is proper. More precisely, q_w should have a match in a computation tree iff that tree is *not* proper. We start with a characterization of (im)properness in terms of the auxiliary concept names from above. In the following, we say that two cells n and n' are A-*conspicuous*, where A is a concept name, if

(†) A is true at the E_h-node of n and the E_p-node of n', or

(‡) A is true at the F_h-node of n and the F_p-node of n'.

Proposition 3.6 *A computation tree $\mathcal{I}$ is not proper iff ($\star$) there exist cells n and n' in successive configurations of $\mathcal{I}$ K such that n and n' are A-conspicuous for all $A \in \mathbf{B} \cup \mathbf{Z}$.*

Proof. The proposition holds due to the way auxiliary labels are defined. First note that if n, n' are cells of two successive configurations in $\mathcal{I}$, then the conditions imposed on $\mathsf{adr}^{\mathcal{I}}(\cdot)$ in Definition 3.2 imply that $\mathsf{adr}^{\mathcal{I}}(n) = \mathsf{adr}^{\mathcal{I}}(n')$ iff for all $A \in \mathbf{B}$, n and n' are A-conspicuous; this is because bit-wise complement is used for the addresses of F_p- and F_h-nodes.
($\Rightarrow$) Suppose that $\mathcal{I}$ is not proper. Then there exist two i-cells n and n' of two successive configurations of $\mathcal{I}$ such that the E_h-node of n and the E_p-node of n' satisfy different pairs (q, a) and (q', a'). As $\mathsf{adr}^{\mathcal{I}}(n) = \mathsf{adr}^{\mathcal{I}}(n')$, n and n' are A-conspicuous for all $A \in \mathbf{B}$. By (b.iii) of Definition 3.2, $Z_{q,a}$ is true at the F_h-node

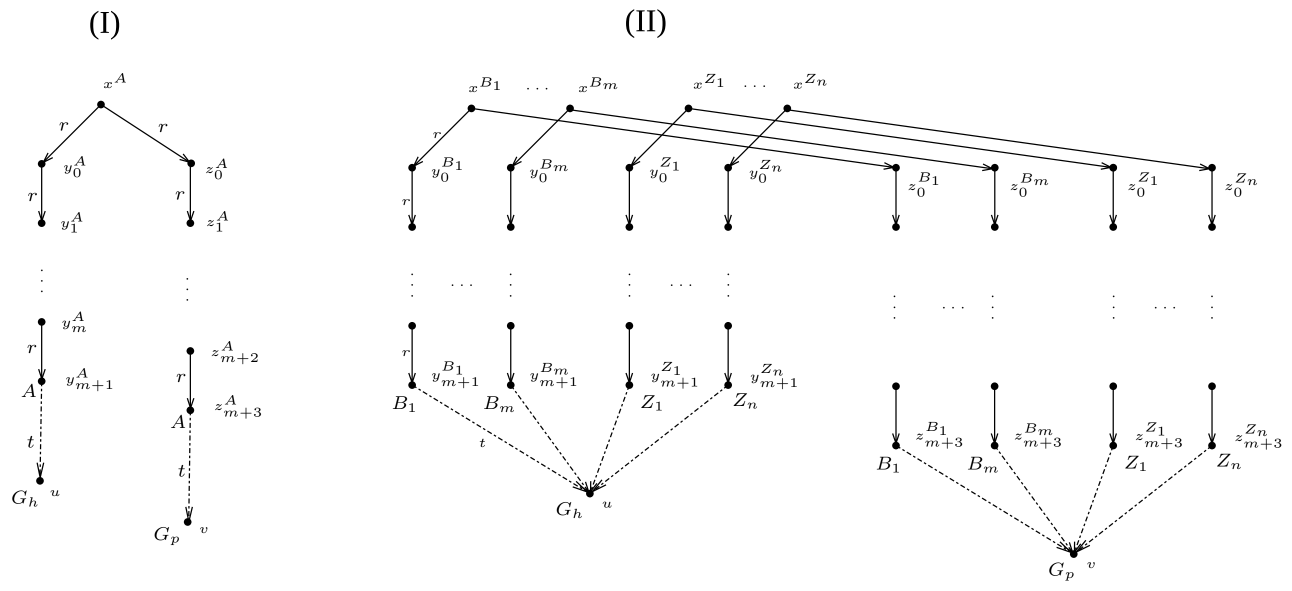


Figure 2: The basic query $q(A, u, v)$ and the final query q_w.

of n; by (b.ii) and since $(q, a) \neq (q', a')$, $Z_{q,a}$ is also true at the F_p-node of n' (recall that $Z_{q',a'}$ is false for at most one pair q', a'). We can argue similarly that $z_{a',q'}$ is true at the E_h-node of n and the E_p-node of n'. For $(q'', a'') \notin \{(q, a), (q', a')\}$, $Z_{a'',q''}$ holds at the E_h-, E_p-, F_h-, and F_p-nodes of both n and n'. In summary, n and n' are A-conspicuous for all $A \in \mathbf{Z}$. Hence, ($\star$) is true.
($\Leftarrow$) To show this, we prove the contrapositive. Suppose that $\mathcal{I}$ is proper and let n and n' be any cells of two successive configurations in $\mathcal{I}$. If n and n' are not A-conspicuous for some $A \in \mathbf{B}$ then ($\star$) is false; otherwise, $\mathsf{adr}^{\mathcal{I}}(n) = \mathsf{adr}^{\mathcal{I}}(n')$ holds, and as $\mathcal{I}$ is proper, the E_h-node of n and the E_p-node of n' satisfy the same $q \in Q$ and $a \in \Sigma$. By (b.i) of Definition 3.2, $Z_{a,q}$ is false at the E_h-node of n; by (b.ii), $Z_{a,q}$ is false at the F_p-node of n'. Hence, n and n' are not $Z_{a,q}$-conspicuous, which means that also in this case ($\star$) is false. $\square$

It thus remains to find a query q_w that has a match iff ($\star$) is satisfied. The structure of q_w is displayed in Figure 2(II).

We obtain q_w by taking, for each $A \in \mathbf{B} \cup \mathbf{Z}$, a copy of the basic query $q(A, u, v)$ in Figure 2(I) such that the different copies share only the variables u and v, and then taking the union. Intuitively, $q(A, u, v)$ deals with A-conspicuousness, and the shared variables u, v ensure that the different component queries speak about the same cells n, n'. In more detail, let n, n' be cells of two successive configurations that are A-conspicuous for all $A \in \mathbf{B} \cup \mathbf{Z}$. We can find a match for q_w as follows: start with matching u on the G_h-node of n and v on the G_p-node of n'. Now take an $A \in \mathbf{B} \cup \mathbf{Z}$. If (†) applies, then match y^A_{m+1} on the E_h-node of n and z^A_{m+1} on the E_p-node of n'; if (‡) applies, then match y^A_{m+1} on the F_h-node of n and z^A_{m+1} on the F_p-node of n'. The matches of all other variables are now uniquely determined by the (non-transitive) role edges in the query. In particular, the lengths of the role chains in the query ensure that x^A will be matched to the root of the configuration node in which n occurs in case (‡) and to the predecessor of this root node in case (†). Observe that the paths labeled with z-variables are exactly two steps longer than those labeled with y-variables, and thus the query only relates n and n' if they belong to successor configurations.

In summary, it is possible to show that

Proposition 3.7 *A computation tree $\mathcal{I}$ is proper iff $\mathcal{I} \not\models q_w$.*

Together with Propositions 3.3 and 3.5, this yields the desired reduction, establishing the lower bound from Theorem 3.1.

4 Query Answering in $\mathcal{S}$

In the next section we show that query non-entailment is NEXPTIME-hard for the DL $\mathcal{S}$, if arbitrary ABoxes are permited. We then show that for tree-shaped ABoxes, the complexity drops to EXPTIME-completeness.

4.1 A Lower Bound

We give a reduction from a NEXPTIME-complete variant of the tiling problem to query non-entailment in $\mathcal{S}$. Since the reduction does not require TBoxes, we will use ABoxes instead of knowlege bases.

Definition 4.1 [Domino System] A *domino system* $\mathfrak{D}$ is a triple (T, H, V), where $T = \{0, \dots, k-1\}$, $k \geq 0$, is a finite set of *tile types* and $H, V \subseteq T \times T$ represent the *horizontal and vertical matching conditions*. Let $\mathfrak{D}$ be a domino system and $c = c_0, \dots, c_{n-1}$ an *initial condition*, i.e. an n-tuple of tile types. A mapping $\tau : \{0, \dots, 2^{n+1}-1\} \times \{0, \dots, 2^{n+1}-1\} \to T$ is a *solution* for $\mathfrak{D}$ and c iff for all $x, y < 2^{n+1}$, the following holds (where $\oplus_i$ denotes addition modulo i):

- if $\tau(x, y) = t$ and $\tau(x \oplus_{2^{n+1}} 1, y) = t'$, then $(t, t') \in H$
- if $\tau(x, y) = t$ and $\tau(x, y \oplus_{2^{n+1}} 1) = t'$, then $(t, t') \in V$
- $\tau(i, 0) = c_i$ for $i < n$.

For a proof of NEXPTIME-hardness of this version of the domino problem, see e.g. Corollary 4.15 in [7].

We show how to translate a given domino system $\mathfrak{D}$ and initial condition $c = c_0 \cdots c_{n-1}$ into an ABox $\mathcal{A}_{\mathfrak{D},c}$ and query $q_{\mathfrak{D},c}$ such that each canonical model $\mathcal{I}$ of $\mathcal{A}_{\mathfrak{D},c}$ that satisfies $\mathcal{I} \not\models q_{\mathfrak{D},c}$ encodes a solution to $\mathfrak{D}$ and c, and conversely each solution to $\mathfrak{D}$ and c gives rise to a model of $\mathcal{A}_{\mathfrak{D},c}$ with $\mathcal{I} \not\models q_{\mathfrak{D},c}$. We start with discussing (a part of) the ABox $\mathcal{A}_{\mathfrak{D},c}$. Among others, it contains an assertion $C_{\mathfrak{D},c}(a)$, with $C_{\mathfrak{D},c}$ a conjunction $C^1_{\mathfrak{D},c} \sqcap \cdots \sqcap C^7_{\mathfrak{D},c}$ whose conjuncts we define in the following. For convenience, let $m = 2n+2$. The purpose of the first conjunct $C^1_{\mathfrak{D},1}$ is to enforce a binary tree of depth m whose edges are labeled with the transitive role r and whose leaves are labeled with the numbers $0, \dots, 2^m - 1$ of a binary counter C implemented by the concept names $T_0, \dots, T_{m-1}$ representing logical truth of a bit and concept names $F_0, \dots, F_{m-1}$ representing logical falsity. We use concept names $L_0, \dots, L_m$ to distinguish the different levels of the tree. This is necessary because we work with transitive roles.

$$
\begin{aligned}
C^1_{\mathfrak{D},c} \quad := \quad & L_0 \sqcap \exists r.(L_1 \sqcap T_1) \sqcap \exists r.(L_1 \sqcap F_1) \sqcap \\
& \bigsqcap_{i<m} \forall r.\big(L_i \to \big(\exists r.(L_{i+1} \sqcap T_i) \sqcap \exists r.(L_{i+1} \sqcap \neg F_i)\big)\big) \sqcap \\
& \bigsqcap_{i<m} \forall r. \bigsqcap_{j<i} \big((L_i \sqcap T_j) \to \forall r.(L_{i+1} \to T_j) \sqcap \\
& \qquad\qquad (L_i \sqcap F_j) \to \forall r.(L_{i+1} \to F_j)\big)
\end{aligned}
$$

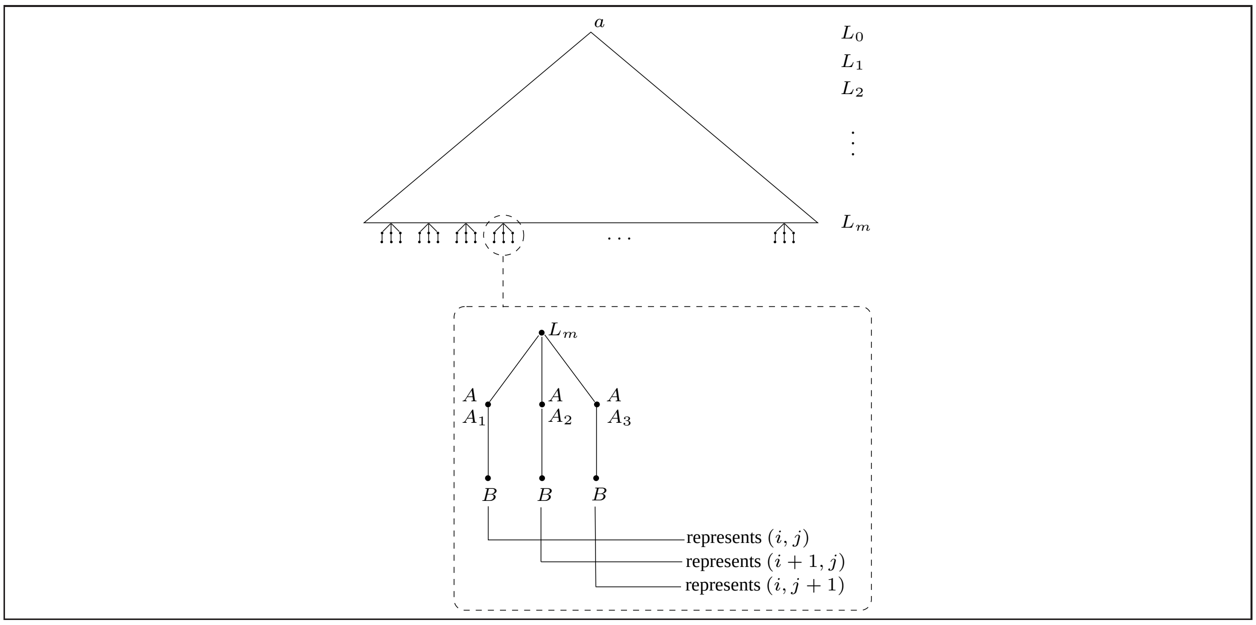


Figure 3: The structure encoding the $2^{n+1} \times 2^{n+1}$-grid.

From now on, leafs in this tree are called L_m-nodes. Intuitively, each L_m-node corresponds to a position in the $2^{n+1} \times 2^{n+1}$-grid that we have to tile: the counter C_x realized by the concept names $T_0, \ldots, T_n, F_0, \ldots, F_n$ binarily encodes the horizontal position, and the counter C_y realized by $T_{n+1}, \ldots, T_m, F_{n+1}, \ldots, F_m$ encodes the vertical position. We now extend the tree with some additional nodes. Every L_m-node gets three successor nodes labelled with A, and each of these A-nodes has a successor node labelled B. To distinguish the three different A-nodes below each L_m-node, we additionally label them with the concept names A_1, A_2, A_3.

$$C^2_{\mathfrak{D},c} \quad := \quad \forall r.\big(L_m \rightarrow \big(\bigsqcap_{1 \leq i \leq 3} \exists r.(A \sqcap A_i \sqcap \exists r.B)\big)\big)$$

We want that each A_1-node represents the grid position identified by its predecessor L_m-node, the sibling A_2 node represents the horizontal neighbor position in the grid, and the sibling A_3-node represents the vertical neighbor.

$$\begin{aligned} C^3_{\mathfrak{D},c} \quad := \quad \forall r.\big(L_m \rightarrow \big(& \bigsqcap_{i \leq n} \big((T_i \rightarrow \forall r.(A_1 \sqcup A_3 \rightarrow T_i)) \sqcap \\ & \qquad (F_i \rightarrow \forall r.(A_1 \sqcup A_3 \rightarrow F_i))\big) \sqcap \\ & \bigsqcap_{n < i < m} \big((T_i \rightarrow \forall r.(A_1 \sqcup A_2 \rightarrow T_i)) \sqcap \\ & \qquad (F_i \rightarrow \forall r.(A_1 \sqcup A_2 \rightarrow F_i))\big) \sqcap \\ & E_2 \sqcap E_3 \,\big)\big) \end{aligned}$$

where E_2 is an $\mathcal{ALC}$-concept ensuring that the C_x value at each A_2-node is obtained from the C_x-value of its L_m-predecessor by incrementing modulo 2^{n+1}; similarly, E_3 expresses that the C_y value at each A_3-node is obtained from the C_y-value of its L_m-node predecessor by incrementing modulo 2^{n+1}. It is not hard to work out the details of these concepts, see e.g. [9] for more details. The *grid representation* that we have enforced is shown in Figure 3. To represent tiles, we introduce a concept name D_i for each $i \in T$ and put

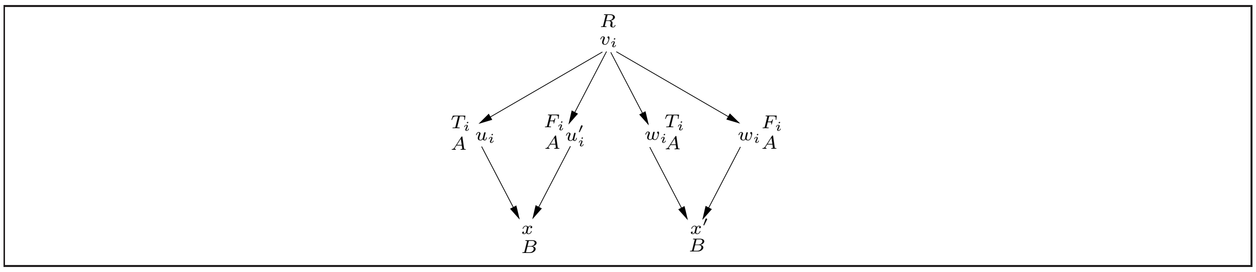


Figure 4: The query $q^i_{\mathfrak{D},c}$.

$$C^4_{\mathfrak{D},c} := \forall r.\big(A \to \big(\bigsqcup_{i \in T} D_i \sqcap \bigsqcap_{i,j \in T, i \neq j} \neg(D_i \sqcap D_j)\big)\big)$$

The initial condition $c = c_0 \cdots c_{n-1}$ is easily guaranteed by

$$C^5_{\mathfrak{D},c} := \bigsqcap_{i<n} \forall r.\big(\,\big(A \sqcap \bigsqcap_{j \leq n, \mathsf{bit}_j(i)=0} F_j \sqcap \bigsqcap_{j \leq n, \mathsf{bit}_j(i)=1} T_j \sqcap \bigsqcap_{n<j<m} F_j\big) \to T_{c_i}\big),$$

where $\mathsf{bit}_j(i)$ denotes the value of the j-th bit in the binary representation of i. To enforce the matching conditions, we proceed in two steps. First we ensure that they are satisfied locally, i.e., among the three A-nodes below each L_m-node:

$$\begin{aligned} C^6_{\mathfrak{D},c} \quad := \quad & \forall r.\big(L_m \to \big(\bigsqcap_{i \in T} \big(\exists r.(A_1 \sqcap D_i) \to \forall r.(A_2 \to \bigsqcup_{(i,j) \in H} D_j)\big) \sqcap \\ & \bigsqcap_{i \in T} \big(\exists r.(A_1 \sqcap D_i) \to \forall r.(A_3 \to \bigsqcup_{(i,j) \in V} D_j)\big)\big)\big) \end{aligned}$$

Second, we enforce the following condition, which together with local satisfaction of the matching conditions ensures their global satisfaction:

($*$) if the C_x and C_y-values of two A-nodes coincide, then their tile types coincide.

In ($*$), an A-node can by any of an A_1-, A_2-, or A_3-node. Note that ($*$) also ensures uniqueness of the tiling in the sense that if there are two A nodes with the C_x and C_y value, then they are labeled with the same tile type. To enforce ($*$), we use the query. Before we give details, let us finish the definition of the concept $C_{\mathfrak{D},c}$. The last conjunct $C^7_{\mathfrak{D},c}$ enforces a double labeling of tiles that will be exploited by the query. We introduce another concept name D'_i for each $i \in T$ and put

$$C^7_{\mathfrak{D},c} := \forall r.\big(A \to \bigsqcap_{i \in T} (D_i \leftrightarrow D'_i)\big)$$

We now construct the query $q_{\mathfrak{D},c}$ that does *not* match the grid representation iff ($*$) is satisfied. In other words, $q_{\mathfrak{D},c}$ matches the grid representation if there are two A-nodes that agree on the value of the counters C_x and C_y, but are labeled with different tile types. Because of Lemma 2.1, we can concentrate on the grid representation as shown in Figure 3 while constructing $q_{\mathfrak{D},c}$, and need not worry about models in which domain elements that are different in Figure 3 are identified.

The construction of $q_{\mathfrak{D},c}$ is in several steps, starting with the query $q^i_{\mathfrak{D},c}$ in Figure 4, where $i \in \{0, \ldots, m-$

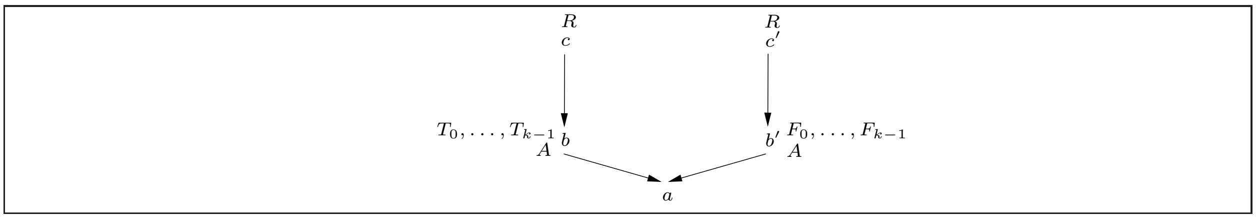


Figure 5: The part of $\mathcal{A}_{\mathfrak{D},c}$ used with $q^i_{\mathfrak{D},c}$.

$1\}$. In the queries $q^i_{\mathfrak{D},c}$, all the edges represent the role r and R is a concept name. Formally,

$$\begin{aligned} q^i_{\mathfrak{D},c} \ := \ \{ \ & R(v_i), r(v_i, u_i), r(v_i, u'_i), r(v_i, w_i), r(v_i, w'_i), \\ & T_i(u), F_i(u'), T_i(w), F_i(w'), \\ & A(u), A(u'), A(w), A(w'), \\ & r(u_i, x), r(u'_i, x), r(w_i, x'), r(w'_i, x'), \\ & B(x), B(x') \ \} \end{aligned}$$

The purpose of the query $q^i_{\mathfrak{D},c}$ is to relate any two A-nodes that agree on the i-th bit of the counter C. More precisely, x and x' are mapped to the B-node successors of two such A-nodes. To make the query work, we add some assertions to the ABox $\mathcal{A}_{\mathfrak{D},c}$, as shown in Figure 5. Note that a is the same individual as in the assertion $C_{\mathfrak{D},c}(a)$ discussed before. Formally, the added assertions are

$$\begin{aligned} & R(c), R(c'), \\ & r(c, b), r(c', b'), \\ & T_0(b), \dots, T_{k-1}(b), \\ & F_0(b'), \dots, F_{k-1}(b'), \\ & A(b), A(b'), \\ & r(b, a), r(b', a) \end{aligned}$$

To understand the query $q^i_{\mathfrak{D},c}$, assume that π is a match of this query in the model obtained by combining Figure 3 and 5. Due to the concept name R, $\pi(v_i)$ is either c or c'. First assume that it is c. Due to the concept name B, $\pi(x)$ and $\pi(x')$ are B-nodes, i.e., leaves in the tree below a. We claim that, at the A-node predecessors of both $\pi(x)$ and $\pi(x')$, the i-bit of the counter C is false (and thus has the same value). To see this, first note that the use of the concept name A ensures that $\pi(u_i)$ and $\pi(u'_i)$ can only be b or the A-node predecessor of $\pi(x)$. Since b does not satisfy F_i, $\pi(u'_i)$ must be the mentioned predecessor, which thus satisfies F_i, but not T_i. It follows that $\pi(u_i)$ is b. Argueing analogously, it can be shown that $\pi(w_i)$ is b and $\pi(w'_i)$ is the A-node predecessor of $\pi(x')$. Since both u'_i and w'_i have to mapped to nodes satisfying F_i, we are done. Now assume that $\pi(v_i)$ is c'. We can argue dually to the previous case to show that $\pi(x)$ and $\pi(x')$ are B-nodes and, at the A-node predecessors of both $\pi(x)$ and $\pi(x')$, the i-bit of the counter C is true (and thus has the same value).

Now set $q_{\mathsf{cnt}} := \bigcup_{i<m} q^i_{\mathfrak{D},c}$. Observe that all queries $q^i_{\mathfrak{D},c}$, $i < m$, share the variables x and x'. It is not hard to verify that if there is a match of q_{cnt} in the model obtained by combining Figure 3 and 5, then x and x' are mapped to B-nodes whose A-predecessors agree on the value of all bits of the counter C. To achieve $(*)$, it just remains to enforce that these predecessors are labeled with different tile types. To this end, we further extend the query and the ABox.

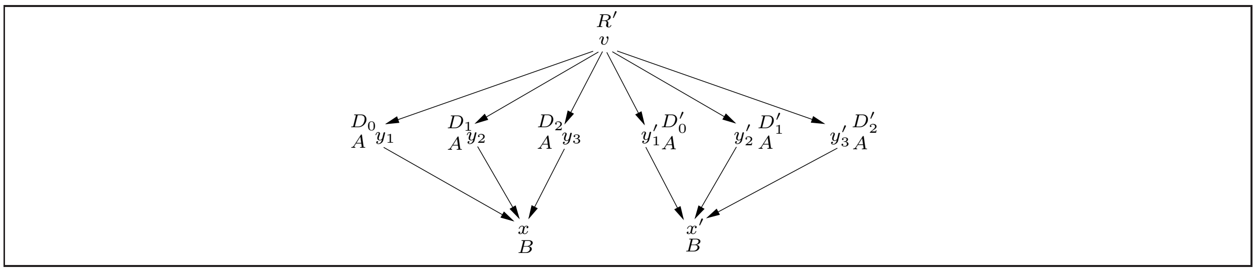
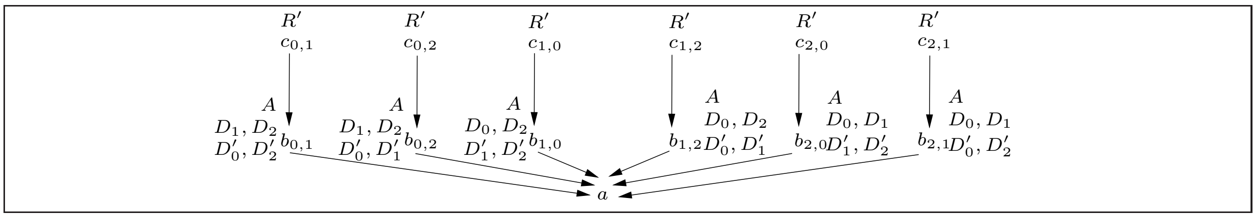


Figure 6: The query q_{tile}.



R' $c_{0,1}$
R' $c_{0,2}$
R' $c_{1,0}$
R' $c_{1,2}$
R' $c_{2,0}$
R' $c_{2,1}$
A D_1, D_2 D_0', D_2' $b_{0,1}$
A D_1, D_2 D_0', D_1' $b_{0,2}$
A D_0, D_2 D_1', D_2' $b_{1,0}$
$b_{1,2}$ A D_0, D_2 D_0', D_1'
$b_{2,0}$ A D_0, D_1 D_1', D_2'
$b_{2,1}$ A D_0, D_1 D_0', D_2'
a

Figure 7: The part of $\mathcal{A}_{\mathfrak{D},c}$ used with q_{tile}.

The query q_{tile} is given in Figure 6 for the case of three tiles, i.e., $T = \{0, 1, 2\}$. It shares the variables x and x' with q_{cnt}. In general, for $T = \{1, \ldots, k-1\}$, we define

$$\begin{aligned} q_{\mathsf{tile}} \quad := \{ & R(v), \\ & r(v, y_0), \ldots, r(v, y_{k-1}), \ldots, r(v, y_0'), \ldots, r(v, y_{k-1}') \\ & D_0(y_0), \ldots, D_{k-1}(y_{k-1}), D_0(y_0'), \ldots, D_{k-1}(y_{k-1}'), \\ & A(y_0), \ldots, A(y_{k-1}), A(y_0'), \ldots, A(y_{k-1}'), \\ & r(y_0, x), \ldots, r(y_{k-1}, x), \ldots, r(y_0', x'), \ldots, r(y_{k-1}', x') \\ & B(x), B'(x') \ \} \end{aligned}$$

To use q_{tile}, we further extend the ABox $\mathcal{A}_{\mathfrak{D},c}$ as shown in Figure 7 for the case of three tiles and where all individuals except a are fresh. Formally, we add the following assertions:

- $R(c_{i,j}), r(c_{i,j}, b_{i,j}), A(b_{i,j}), r(b_{i,j}, a)$ for all $i, j \in \{0, \ldots, k-1\}$ with $i \neq j$;
- $D_\ell(b_{i,j})$ for all $\ell, i, j \in \{0, \ldots, k-1\}$ with $i \neq j$ and $i \neq \ell$;
- $D_\ell'(b_{i,j})$ for all $\ell, i, j \in \{0, \ldots, k-1\}$ with $i \neq j$ and $j \neq \ell$.

Observe the similarity between q_{tile} and $q^i_{\mathfrak{D},a}$, and between the ABox extension for q_{tile} and that for $q^i_{\mathfrak{D},a}$. Let π be a match of q_{tile} in in the model obtained by combining Figure 3 and 7. Due to the concept name R', $\pi(v) = c_{i,j}$ for some i, j with $i \neq j$. Moreover, x and x' are mapped to a B-node in the tree below a, each y_ℓ is mapped either to $b_{i,j}$ or to the A-node predecessor of $\pi(x)$, and each y_i' either to $b_{i,j}$ or to the A-node predecessor of $\pi(x')$. Since $b_{i,j}$ does not satisfy D_i, $\pi(y_i)$ must be the A-node predecessor of $\pi(x)$, which thus satisfies D_i, but none of $D_0, \ldots, D_{i-1}, D_{i+1}, \ldots, D_{k-1}$. We can use the concept names $D_0', \ldots, D_{k-1}'$ to argue analogously that $\pi(y_j')$ is the A-node predecessor of $\pi(x')$, and that it satisfies D_j'. Since $i \neq j$, the A-node predecessors of $\pi(x)$ and $\pi(x')$ are labeled with different tile types.

Now, the desired query $q_{\mathfrak{D},c}$ is simply the union of q_{cnt} and q_{tile}. From what was already said about q_{cnt} and q_{tile}, it is easily derived that $q_{\mathfrak{D},c}$ does not match the grid representation iff Property $(*)$ is satisfied. It is possible to show that there is a solution for $\mathfrak{D}$ and c iff $(\emptyset, \mathcal{A}_{\mathfrak{D},c}) \not\models q_{\mathfrak{D},c}$. We have thus proved that query entailment in $\mathcal{S}$ is co-NEXPTIME-hard.

Theorem 4.2 *Query entailment in $\mathcal{S}$ is co-*NEXPTIME*-hard. This holds even for knowledge bases in which the TBox is empty.*

4.2 CQs over Tree-shaped ABoxes

In this section, we show that the hardness of the query answering drops to EXPTIME-hard if we restrict the shape of the ABox.

An ABox $\mathcal{A}$ is *tree-shaped*, if the directed graph with nodes $\mathsf{Ind}(\mathcal{A})$ and edges $\{(a,b) \mid r(a,b) \in \mathcal{A}\}$ is a tree, and $r(a,b)\ r'(a,b) \in \mathcal{A}$ implies $r = r'$. We aim to show the following.

Theorem 4.3 *In $\mathcal{S}$, CQ entailment is* EXPTIME*-complete if ABoxes are tree-shaped.*

It is well-known that CQ entailment in $\mathcal{S}$ is EXPTIME-hard even with empty ABoxes (which follows from the EXPTIME-completeness of knowledge base satisfiability in $\mathcal{ALC}$ [15]) and thus it remains to show the upper bound. We start with a simple observation.

Proposition 4.4 *For a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{A}$ is tree-shaped, we can build in polynomial time a KB $\mathcal{K}' = (\mathcal{T}, \{C_{\mathcal{A}}(a)\})$ such that $\mathcal{K} \models q$ iff $\mathcal{K}' \models q$ for every CQ q.*

It thus suffices to give an EXPTIME algorithm for CQ entailment in $\mathcal{S}$ with ABoxes of the form $\{C_0(a)\}$. From now on, let $\mathcal{K} = (\mathcal{T}, \{C_0(a)\})$ be a KB and q a CQ for which we decide $\mathcal{K} \models q$.

We assume w.l.o.g. that C_0 is in *negation normal form* (NNF), i.e. negation is only applied to concept names, and that $\mathcal{T}$ contains a single concept inclusion axiom of the form $\top \sqsubseteq C_{\mathcal{T}}$ with $C_{\mathcal{T}}$ in NNF. We may also assume w.l.o.g. that q is connected (a disconnected query can be answered by separately posing each connected subquery).

We can limit our attention to certain canonical models and a certain kind of query that we call a *pseudo-tree query*.

Definition 4.5 [Canonical Model] A *canonical model* for $\mathcal{K}$ is a model $\mathcal{I}$ of $\mathcal{A}$ such that

- $\mathcal{I}$ satisfies all concept inclusions in $\mathcal{T}$ (but not necessarily the transitivity axioms);
- $(\Delta^{\mathcal{I}}, \bigcup_{r \in \mathsf{N_R}} r^{\mathcal{I}})$ is a tree with root $a^{\mathcal{I}}$ and whose out-degree is bounded by the cardinality of $\mathsf{sub}(\mathcal{K})$;
- $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$ whenever $r \neq s$;
- for all $\forall t.C \in \mathsf{sub}(\mathcal{K})$ with $t \in \mathsf{Tr}(\mathcal{K})$ and all $(d,e) \in t^{\mathcal{I}}$, $d \in (\forall t.C)^{\mathcal{I}}$ implies $e \in (\forall t.C)^{\mathcal{I}}$.

Due to the non-transitivity of transitive roles in canonical models $\mathcal{I}$, we have to work with a relaxed version of a match that becomes a match when, for every $r \in \mathsf{Tr}(\mathcal{K})$, $r^{\mathcal{I}}$ is replaced with its transitive closure.

Definition 4.6 [Pre-match] Let $\mathcal{I}$ be a canonical model of $\mathcal{K}$. We call a mapping $\pi : \mathsf{Vars}(q) \to \Delta^{\mathcal{I}}$ a *pre-match* for q in $\mathcal{I}$, if (a) $\pi(u) \in A^{\mathcal{I}}$ for each $A(u) \in q$, (b) $(\pi(u), \pi(v)) \in r^{\mathcal{I}}$ for each $r(u,v) \in q$ with $r \notin \mathsf{Tr}(\mathcal{K})$, and (c) for each $t(u,v) \in q$ with $t \in \mathsf{Tr}(\mathcal{K})$ there is a sequence $d_0, \ldots, d_n \in \Delta^{\mathcal{I}}$, $n \geq 1$, such that $d_0 = \pi(u)$, $d_n = \pi(v)$ and $(d_i, d_{i+1}) \in t^{\mathcal{I}}$ for all $i < n$. We write $\mathcal{I} \models^{\mathsf{pre}} q$, if there is a pre-match for q in $\mathcal{I}$.

We now define pseudo-tree queries, based on role clusters.

Definition 4.7 [Role Cluster, Pseudo-tree Query] Let q be a CQ. For each $t \in \mathsf{Tr}(\mathcal{K})$, $\sim_t$ denotes the smallest equivalence relation over $\mathsf{Var}(q)$ such that $t(v, v') \in q$ implies $v \sim_t v'$. An equivalence class c_t of $\sim_t$ is called a *(transitive) cluster* of q. For each non-transitive role s, a *(non-transitive) cluster* of q is a set $c_s = \{u, v\}$ with $s(u, v) \in q$.

Now, a connected CQ q is a *pseudo-tree query* if it satisfies:

(a) if c_r is a cluster of q and $s(u, v), s'(u', v') \in q$ with $v, v' \in c_r$ and $s, s' \neq r$, then $s = s'$, $u = u'$, $v = v'$;

(b) q is acyclic, i.e., it does not contain atoms $r_0(v_0, v_1), \ldots, r_n(v_n, v_{n+1})$ with $v_{n+1} = v_0$.

A cluster c_r of q is *initial*, if no $v \in c_r$ has an incoming edge $r'(v', v) \in q$ with $r \neq r'$.

Intuitively, a pseudo-tree can be viewed as a tree of clusters with an additional root; the root is a predecessor of every initial cluster (there can be more than one) and there is an edge between two clusters if they share an element. Each transitive cluster in a pseudo-tree query describes a subquery that is an acyclic directed graph.

Over an interpretation $\mathcal{J}$ whose domain is a tree, the existence of a pre-match for any query can be reduced to the existence of a pre-match for a pseudo-tree query, which is obtained from q by identifying any two variables u, u' such that $\pi(u) = \pi(u')$ for every pre-match π for q in $\mathcal{J}$.

Definition 4.8 A CQ q' is *obtained from a CQ q by fork elimination*, if q' results from q by one of the following operations:

- select $r(u, v), r(u', v) \in q$ with $u \neq u'$ and $r \notin \mathsf{Tr}(\mathcal{K})$, and identify u and u';
- select $r(u, v), r(u', v') \in q$ with $v \neq v'$ and v, v' in a cluster c_s of q where $s \neq r$, and identify v and v';

We say that q' is a *maximal fork rewriting* of q, if q' is obtained from q by exhaustive fork elimination.

The entailment of a CQ q over the canonical models of $\mathcal{K}$ is invariant under fork-rewriting, and q can be entailed only if it can be turned into a pseudo-tree query by (maximal) fork rewriting.

It can be shown that the maximal fork rewriting is unique and computable in polynomial time. Moreover, it can be checked in polynomial time whether a query is a pseudo-tree query. Hence, the following proposition allows us to restrict our attention to canonical models and pseudo-tree queries.

Proposition 4.9 *Let q be a CQ, and let q' be the maximal fork rewriting of q. Then $\mathcal{K} \not\models q$ iff (i) q' is not a pseudo-tree query, or (ii) $\mathcal{I} \not\models^{\mathsf{pre}} q'$ for some canonical model $\mathcal{I}$ of $\mathcal{K}$.*

In what follows, assume that the input query q is a pseudo-tree query. We want to decide whether there is a canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models^{\mathsf{pre}} q$.

4.2.1 Markings for canonical models

In this subsection, we consider *markings* which witness the non-existence of pre-matches in canonical models. They are a stepping-stone to obtain the knot elimination algorithm in the short version of this paper. This ‘technical step’ was omitted from the latter due to the space restrictions, and is linked to the material in the submission in the next subsection.

First, we define some notions and notation for later use.

Definition 4.10 For $V \subseteq \mathsf{Vars}(q)$, we denote by $q|_V$ the restriction of q to the variables in V.

By $\mathsf{reach}_q(V)$ we denote the variables that are reachable in q from some variable of V, i.e., the smallest subset of $\mathsf{Vars}(q)$ with $V \subseteq \mathsf{reach}_q(V)$ and such that $r(v, v') \in q$ and $v \in \mathsf{reach}_q(V)$ imply $v' \in \mathsf{reach}_q(V)$.

We say that a variable $v \in V$ is *minimal* in $V \subseteq \mathsf{Vars}(q)$, if $v \notin \mathsf{reach}_q(V)$; we denote by $\mathsf{min}(V)$ the set of all variables that are minimal in V.

We consider some special subqueries of the query q. We also consider a restricted form of pre-matches for these subqueries, which require the minimal variables to be matched at some particular domain element.

Definition 4.11 [cluster part; pseudo-tree subquery $q(P)$] A *cluster part* (of q) is a pair $P = \langle c_r, V\rangle$ where c_r is a cluster of q and V is a nonempty subset of c_r. The subquery of q induced by P, denoted $q(P)$, is given by $q(P) = q|_{\mathsf{reach}_q(V)}$.

Note that for a pseudo-tree query q, also $q(P)$ is a pseudo-tree query.

Definition 4.12 [rooted pre-match] Let $\mathcal{J}$ be a canonical model for $\mathcal{K}$, and let $o, o' \in \Delta^{\mathcal{J}}$. For a transitive role r, we say that o' is r*-reachable* from o, if there is a sequence $o_1, \ldots, o_n$, $n \geq 1$, such that $o_1 = o$, $o_n = o'$ and $(o_i, o_{i+1}) \in r^{\mathcal{J}}$ for each $1 \leq i < n$.

A pre-match π for $q(P)$ is *rooted* at $o \in \Delta^{\mathcal{J}}$, if the following hold:

1. If $r \notin \mathsf{Tr}(\mathcal{K})$, then $\pi(v) = o$ for the unique v that is minimal in V.
2. If $r \in \mathsf{Tr}(\mathcal{K})$, then (a) for each $v \in \mathsf{min}(V)$, $\pi(v)$ is r-reachable from o; and (b) for every $r'(v', v) \in q$ such that $v \in V$ and $r' \neq r$, $\pi(v) = o$.

To conveniently describe rooted pre-matches for a pseudo-tree subquery q', we employ the *departs* relation between a variable and (cluster parts of) subqueries rooted at it, as well as local notions of matchability at a domain element.

Definition 4.13 [departs; label-matched] Given a a cluster c_r and a variable $v \in c_r$ such that q contains an atom $s(v, v')$ with $s \neq r$, we say that the cluster part $\langle c_s, c_s\rangle$ *departs from* v, where c_s is the s-cluster with $v, v' \in c_s$.

Let $\mathcal{J}$ be an interpretation and let $o \in \Delta^{\mathcal{J}}$. We say that the variable v is *label-matched* at o if $B(v) \in q$ implies $o \in B^{\mathcal{J}}$. For a set of variables V, we denote by $\mathsf{labelMatch}_o(V)$ the set of all $v \in V$ that are label-matched at o, and by let $M_o(V) := \mathsf{min}(V) \cap \mathsf{labelMatch}_o(V)$. Finally, we denote by $D_o(V)$ the set of all $v \in M_o(V)$ such that there is a pre-match rooted at o for each $q(P')$ such that P' departs from v.

Now we characterize rooted pre-matches for a subquery q' in terms of local conditions and rooted pre-matches for subqueries of q'.

Lemma 4.14 *Let $P = \langle c_r, V\rangle$ be a cluster part, let $\mathcal{J}$ be a canonical model for $\mathcal{K}$ and let $o \in \Delta^{\mathcal{J}}$. There exists a pre-match for $q(P)$ rooted at o iff the following hold:*

1. *if r is not transitive, $V = \{v, v'\}$ and $r(v, v') \in q$, then v is label-matched at o and there exists a rooted pre-match for $q(\langle c_r, \{v'\}\rangle)$ rooted at some $o' \in \Delta^{\mathcal{J}}$ with $(o, o') \in r^{\mathcal{J}}$;*

2. *if r is not transitive and $V = \{v\}$, then v is label-matched at o and there is a pre-match for $q(P')$ rooted at o for every P' that departs from v.*

3. *if r is transitive, then for each maximal connected component V' of $V \setminus D_o(V)$ (i.e., V' contains some $v_\downarrow$ that does not reach any other $v' \in c_r$, and it contains all variables that are connected to $v_\downarrow$ in $q|_{(V \setminus D_o(V))}$), there exists a pre-match for $q(\langle c_r, V' \rangle)$ rooted at some o' with $(o, o') \in r^{\mathcal{J}}$.*

Proof. If r is not transitive, then both directions are trivial. If r is transitive, then one direction is straightforward: item 3 implies the existence of a pre-match as desired. To show the other direction, it suffices to observe that if there is some pre-match for $q(P)$ rooted at o, then there is one such pre-match π such that $\pi(v) = o$ for every $v \in D_o(V)$. Such a π can be obtained by taking any rooted pre-match π', pulling up the match of each $v \in D_o(V)$ that was not matched at o, and setting the match of all variables that are in some $q' = q(P')$ such that P' departs from v, to coincide with the existing pre-match for q' rooted at o.

Clearly, all atoms of the form $A(u) \in q(P)$ are satisfied after this. As for the role atoms, recall that v is minimal in V and hence it has no incoming arcs $s(v', v) \in q(P)$. By this and the fact that $q(P)$ is a pseudo-tree query, each atom $s(u, u')$ in $q(P)$ is of one of the following three forms: (1) $u = v$, $u' \in c_r$ and $r = s$, (2) $u' \in c_r$ and u, u' are both in some $q' = q(P')$ such that P' departs from v, (3) $s(u, u')$ is not as in cases 1 or 2 above, which implies that $u, u' \neq v$ and neither u nor u' is in any q'. All outgoing edges $r(v, u')$ with $u' \in c_r$ (case 1) are satisfied, since $\pi'(u') = \pi(u')$ is r-reachable from $\pi'(v)$ and hence from o. Satisfaction of $s(u, u')$ as in case 2 is straightforward by the construction of π. For all other variables π and π' coincide, hence the satisfaction of all atoms in case 3 is also ensured.

The pre-match π, which maps each $v \in D_o(V)$ to o and induces rooted matches for each maximal connected component of $V \setminus D_o(V)$, witnesses item 3. $\square$

We define *markings* that witness the non-existence of rooted pre-matches for pseudo-tree subqueries in a canonical model. The negation of the statement of the lemma above provides the basis for defining conditions that correctly capture the non-existence of pre-matches.

In the following, we denote by $CP(q)$ (or simply CP) the set of all cluster parts of q.

Definition 4.15 [(spoiling) markings] Let $\mathcal{J}$ be a canonical model of $\mathcal{K}$. A *marking* (for $\mathcal{J}$ and q) is a relation $\mu \subseteq \Delta^{\mathcal{J}} \times CP(q)$; we use $\mu(o)$ to denote the set of cluster parts P with $(o, P) \in \mu$.

For $o \in \Delta^{\mathcal{J}}$ and P a cluster part of q, μ is $q(P)$-*spoiling at* o, if $P \in \mu(o)$ and μ is *spoiling*, i.e., for every $(o, P) \in \mu$ with $P = \langle c_r, V \rangle$, the following hold:

(S1) If $r \notin \mathsf{Tr}(\mathcal{K})$, $V = \{v, v'\}$ and $r(v, v') \in q$, then either (a) v is not label-matched at o, or (b) $\langle c_r, \{v'\} \rangle \in \mu(o')$ for all o' with $(o, o') \in r^{\mathcal{J}}$.

(S2) If $r \notin \mathsf{Tr}(\mathcal{K})$ and $V = \{v\}$, then either (a) v is not label-matched at o, or (b) $P' \in \mu(o)$ for some P' that departs from v.

(S3) If $r \in \mathsf{Tr}(\mathcal{K})$, then there is some *consumed set* $S \subseteq \mathsf{min}(V)$ such that:

- for each $v \in \mathsf{min}(V) \setminus S$, either (a) v is not label-matched at o, or (b) $P' \in \mu(o)$ for some P' that departs from v, and
- there is a $V' \subseteq V \setminus S$ that contains (a) some $v_\downarrow$ that does not reach any other $v' \in c_r$, and (b) all variables that are connected to $v_\downarrow$ in $q|_{(V \setminus S)}$, such that $\langle c_r, V' \rangle \in \mu(o')$ for all o' with $(o, o') \in r^{\mathcal{J}}$.

Next we show that there exists a marking μ for a canonical model $\mathcal{J}$ that is $q(P)$-spoiling at some o iff there exists no pre-match for $q(P)$ rooted at o.

We do this by induction, using a suitable notion of subquery size.

Definition 4.16 The *scope size* of a cluster part $P = \langle c_r, V\rangle$, denoted $\#s(P)$, is defined as $|V|+|\mathsf{reach}_q(V)|-1$. Observe that $\#s(P) = 1$ iff $P = \langle c_r, \{v\}\rangle$ and v has no outgoing edges $s(v, v')$ in q.

For the only if direction, we can actually prove something stronger and impose restrictions on μ, which ensure that the consumed set S in (S3) in Definition 4.15 is always as small as possible and that only relevant cluster parts appear in the markers of the nodes.

Definition 4.17 Let $\mathcal{J}$ be a canonical model for $\mathcal{K}$. Let $o \in \Delta^{\mathcal{J}}$, and let P be a cluster part of q. We say a marking μ is (o, P)-*austere*, if it is $q(P)$-spoiling at o and

1. for every $o' \in \Delta^{\mathcal{J}}$ and every $\langle c_r, V\rangle \in \mu(o')$ with $r \in \mathsf{Tr}(\mathcal{K})$, (S3) in Definition 4.15 is satisfied by taking $D_{o'}(V)$ as the consumed set S;
2. for every $o' \in \Delta^{\mathcal{J}}$, $\{\langle c, V\rangle, \langle c', V'\rangle\} \in \mu(o)$ implies $c \neq c'$;
3. no $\mu' \subsetneq \mu$ is $q(P)$-spoiling at o, hence each cluster part occurring in μ is *justified*, i.e., $\langle V, c\rangle \in \mu(o')$ implies that o' is inside the subtree of $\mathcal{J}$ rooted at o and one of the following holds:
   - c is the initial cluster of $q(P)$, or
   - c is an r-cluster and there is some $V \subseteq V'$ such that $\langle c, V'\rangle \in \mu(o'')$, or
   - $V = c$ and there is some r-cluster c', some $V' \subsetneq c'$, and some $v \in V'$ such that $\langle c, V\rangle$ departs from v and $\langle c', V'\rangle \in \mu(o')$,

   where o'' is the parent of o' in $\mathcal{J}$ and $(o'', o') \in r^{\mathcal{J}}$.

Now we show that the non-existence of a rooted pre-match for $q(P)$ is always witnessed by a (o, P)-austere marking.

Lemma 4.18 *Let $\mathcal{J}$ be a canonical model for $\mathcal{K}$, let $o \in \Delta^{\mathcal{J}}$ and let P be a cluster part of q. If there is no pre-match for $q(P)$ rooted at o, then there is an (o, P)-austere marking for $\mathcal{J}$.*

Proof. Let $P = \langle c_r, V\rangle$ and assume that there is no pre-match for $q(P)$ rooted at o. We prove the claim by induction on $\#s(P)$.

(Basis) Suppose that $\#s(P) = 1$. Then $V = \{v\}$ holds. If r is not-transitive, by Lemma 4.14 v is not label-matched at o, and thus $\mu = \{(o, P)\}$ is (o, P)-austere. If r is transitive, we simply set $\mu(o') = \{P\}$ for every o' that is r-reachable from o and where v is not matched, as well as for the first o' on each branch that is r-reachable from o and where v is matched (if any). Note we may encounter an infinite r-branch where v is never matched, and all its nodes are marked with P.

(Induction step) Suppose that $\#s(P) > 1$. We are in one of the following cases:

1. r is not transitive, $V = \{v, v'\}$ and $r(v, v') \in q$. By Lemma 4.14, either (*i*) v is not label-matched at o, or (*ii*) v is label-matched at o and there exists no pre-match for $q(\langle c_r, \{v'\}\rangle)$ rooted at some $o' \in \Delta^{\mathcal{J}}$ with $(o, o') \in r^{\mathcal{J}}$.

   If (*i*), we can define an (o, P)-austere marking μ as $\{(o, P)\}$.

   If (*ii*), by the induction hypothesis, for each $o' \in \Delta^{\mathcal{J}}$ with $(o, o') \in r^{\mathcal{J}}$ there is an $(o', \langle c_r, \{v\}\rangle)$-austere marking μ'. To obtain an (o, P)-austere μ, set $\mu(o) = \{P\}$ and $\mu(o'') = \mu'(o'')$ for each o'' that is below some o'.

2. r is not transitive and $V = \{v\}$. By Lemma 4.14, either (*i*) v is not label-matched at o, or (*ii*) v is label-matched at o and there is some P' that departs from v such that there is no pre-match for $q(P')$ rooted at o.

   If (*i*), then we proceed as above and define $\mu = \{(o, P)\}$.

   If (*ii*), then by the induction hypothesis, there is an (o, P')-austere marking μ'. We extend it to an (o, P)-austere μ by setting $\mu(o) = \{P\} \cup \mu'(o)$ and $\mu(o') = \mu'(o')$ for each o' that is below o.

3. r is transitive. According to Definition 4.17, we set $S = D_o(V)$ as the consumed set S. We must ensure that (S3) in Definition 4.15 is satisfied.

   First, observe that for each $v \in \mathsf{min}(V) \setminus S$, either (a) $v \notin M_o(V)$, i.e., v is not label-matched at o, or (b) $v \in M_o(V) \setminus S$. In case (b), there is some P_v that departs from v and such that no rooted pre-match for $q(P_v)$ exists, and hence by the induction hypothesis there is an (o, P_v)-austere marking μ_v. We can include these μ_v in the (o, P)-austere μ to satisfy the first item of (S3) in Definition 4.15.

   As for the second item of (S3), by Lemma 4.14, there is a $V' \subseteq V \setminus S$ such that:

   (a) V' contains some $v_\downarrow$ that does not reach any other $v' \in c_r$,

   (b) V' contains all the variables that are connected to $v_\downarrow$ in $q|_{(V \setminus S)}$, and

   (c) for each o' with $(o, o') \in r^{\mathcal{J}}$, there exists no pre-match for $q(\langle c_r, V'\rangle)$ rooted at o'.

   By the induction hypothesis, for each such o' an $(o', \langle c_r, V'\rangle)$-austere marking $\mu_{o'}$ exists. We can also include these $\mu_{o'}$ in the (o, P)-austere marking μ in order to satisfy the second item of (S3) in Definition 4.15.

   Hence the desired μ can be defined as

$$\{(o, P)\} \cup \bigcup_{(o,o') \in r^{\mathcal{J}}} \mu_{o'} \cup \bigcup_{v \in M_o(V) \setminus S} \mu_v.$$

   It can be easily verified that μ is (o, P)-austere.

□

The converse also holds, even if we drop the austerity restriction.

Lemma 4.19 *Let $\mathcal{J}$ be a canonical model for $\mathcal{K}$, let $o \in \Delta^{\mathcal{J}}$ and let $P = \langle c_r, V\rangle$ be a cluster part of q. If there is a pre-match for $q(P)$ rooted at o, then no marking for $\mathcal{J}$ is $q(P)$-spoiling at o.*

Proof. Let $P = \langle c_r, V\rangle$ and assume that there is a pre-match for $q(P)$ rooted at o. We prove the claim by induction on $\#s(P)$.

(Basis) Suppose $\#s(P) = 1$. Here $V = \{v\}$ and $v \in M_o(\{v\})$. If $r \notin \mathsf{Tr}(\mathcal{K})$, then it is clearly not possible to satisfy $P \in \mu(o)$ and (S2) in Definition 4.15. If $r \in \mathsf{Tr}(\mathcal{K})$, then $v \in S$ can not hold since there is no V' as required by (S3) in Definition 4.15. But if $v \notin S$, the first item of (S3) is not satisfied. This shows that there is no μ that is $q(P)$-spoiling at o.

(Induction step) We are in one of the following cases:

1. r is not transitive, $V = \{v, v'\}$ and $r(v, v') \in q$. Let $P' = \langle c_r, \{v'\}\rangle$. By Lemma 4.14, v is label-matched at o and there exists a rooted pre-match for $q(P')$ rooted at some $o' \in \Delta^{\mathcal{J}}$ with $(o, o') \in r^{\mathcal{J}}$. Any μ, to be $q(P)$-spoiling at o, must have $P \in \mu(o)$. Given that v is label-matched at o, to satisfy (S1) in Definition 4.15, $P' \in \mu(o')$ must hold at every o' with $(o, o') \in r^{\mathcal{J}}$. Thus, μ must also be $q(P')$-spoiling at o'; but as $\#s(P') < \#s(P)$, such a μ does not exist by the induction hypothesis.

2. r is not transitive and $V = \{v\}$. By Lemma 4.14, v is label-matched at o and for every P' that departs from v, there is a pre-match for $q(P')$ rooted at o.

   To be $q(P)$-spoiling at o, any μ must have $P \in \mu(o)$, and as v is label-matched at o, to satisfy (S2) in Definition 4.15, $P' \in \mu(o)$ must hold for some P' that departs from v. By definition, μ must be $q(P')$-spoiling at o, but as $\#s(P') < \#s(P)$, such a μ does not exist by the induction hypothesis.

3. r is transitive. By Lemma 4.14, for each maximal connected component V' of $V \setminus D_o(V)$, there exists a pre-match for $q(\langle c_r, V'\rangle)$ rooted at some o' with $(o, o') \in r^{\mathcal{J}}$.

   To be $q(P)$-spoiling at o, any μ must have $P \in \mu(o)$, and there must be some consumed set $S \subseteq \mathsf{min}(V)$ as required by condition (S3) in Definition 4.15.

   We show that $D_o(V) \subseteq S$. Towards a contradiction, suppose that some $v \in D_o(V) \setminus S$ exists. Then $P' \in \mu(o)$ must hold for some P' that departs from this v in order to satisfy the first item (note that v is label-matched at o as there is a pre-math for $q(P)$). However, the definition of $D_o(V)$ implies that there is a pre-match rooted at o for every such P'; as $\#s(P') < \#s(P)$, such a μ does not exist by the induction hypothesis, which is a contradiction. This shows $D_o(V) \subseteq S$.

   For μ to be $q(P)$-spoiling at o, the second item of condition (S3) implies that there must exists some maximal $V' \subseteq V \setminus S$ that contains (a) some $v_{\downarrow}$ that does not reach any other $v' \in c_r$, and (b) all variables that are connected to $v_{\downarrow}$ in $q|_{(V \setminus S)}$, such that $\langle c_r, V'\rangle \in \mu(o')$ for all o' with $(o, o') \in r^{\mathcal{J}}$. Clearly, this maximal connected component V' of $V \setminus S$ is contained in some maximal connected component V'' of $V \setminus D_o(V)$. The existence of a pre-match π for $q(\langle c_r, V''\rangle)$ rooted at some o' with $(o, o') \in r^{\mathcal{J}}$ implies the existence of a pre-match π' for $q(\langle c_r, V'\rangle)$. To see that π' is also rooted at o', observe that $v \in D_o(V)$ holds for every $v \in V$ with $s(v', v) \in q$ and $s \neq r$ (by the assumption that there is a pre-match for $q(\langle c_r, V\rangle)$ rooted at o) and hence there is no such v in V'. As $\#s(\langle c_r, V'\rangle) < \#s(\langle c_r, V\rangle)$, the induction hypothesis implies that μ can not be $q(\langle c_r, V'\rangle)$-spoiling at o'. This is a contradiction; it follows that a marking μ which is $q(P)$-spoiling at o does not exist.

$\square$

We have shown that the non-existence of a pre-match for $q(P)$ rooted at some specific o is correctly characterized by the existence of a marking that is $q(P)$-spoiling at this o. Now we introduce *global* markings to capture the non-existence of pre-matches at every o of $\mathcal{J}$.

Definition 4.20 A spoiling marking μ for $\mathcal{J}$ and q is *global*, if for each $o \in \mathcal{J}$, $\mu(o)$ contains some cluster part $\langle c_r, V\rangle$ where c_r is an initial cluster of q.

An easy consequence of the above is that $\mathcal{J}$ admits a global marking iff it is a countermodel of q.

Proposition 4.21 *$\mathcal{I} \not\models^{\mathsf{pre}} q$ iff there exists a global marking for $\mathcal{J}$ and q.*

Finally, we show that for each countermodel there is a global marking μ, which can be obtained by suitably composing austere markings, that does not assign to any o different parts of the same cluster. The latter is crucial for obtaining the desired complexity bounds.

Proposition 4.22 *There exists a global marking for a model $\mathcal{J}$ and a query q iff there exists a global marking for $\mathcal{J}$ and q that assigns to each node at most one part for each cluster of q.*

Proof. Suppose μ is a global marking for q. We gather a collection of austere markings that spoil a part of an initial cluster at each node. More precisely, for a node o, let $s_o \subseteq \Delta^{\mathcal{J}} \times CP$ be a marking that is $q(P)$-spoiling at o for some part P of an initial cluster of q. For each node o, such an s_o trivially exists and can be extracted from μ. Then by Lemmas 4.18 and 4.19, for each o there exists a $p_o \subseteq \Delta^{\mathcal{J}} \times CP$ which is (o, P)-austere for some part of an initial cluster of q. We define inductively a new marking $\nu = \bigcup_{i \geq 0} \nu_i$, where

1. $\nu_0 = p_{o_R}$ and o_R is the root of $\mathcal{J}$, and

2. for $i > 0$,
$$\nu_i = \nu_{i-1} \cup \bigcup_{o \in \gamma(i, \nu_{i-1})} p_o,$$
where $\gamma(i, \nu_{i-1})$ contains all $o \in \Delta^{\mathcal{J}}$ such that $|o| = i$ (i.e., o is at level i of the tree $\Delta^{\mathcal{J}}$) and ν_{i-1} contains no (o, P) where $P = \langle c_r, V \rangle$ and c_r is an initial cluster of q.

By construction, ν is a global marking for $\mathcal{J}$ and q. We verify that ν assigns each $o \in \Delta^{\mathcal{J}}$ at most one part per cluster of q.

Consider the tree T whose nodes are the clusters of q with an additional root; the root is a predecessor of every initial cluster (there can be more than one) and there is an edge between two clusters if they share an element. Intuitively, each iteration i in the construction of ν adds an austere marking for some initial cluster, but only inside a subtree of $\mathcal{J}$ rooted at some element where no parts of initial clusters occur. Hence, it is 'delayed' on the tree T w.r.t. all the markings that started higher in $\mathcal{J}$, and for each part $\langle c, V \rangle$ of q that is associated to a node o, c is at a strictly higher level in the tree T than all other clusters for which o had been assigned a part in any previous iteration $i' < i$.

Formally, we denote by $\leq_T$ the partial order over the clusters of q induced by T, and use $c \lneq_T c'$, $c + 1 =_T c'$ and $c \lessgtr_T c'$ to denote that c is, respectively, strictly smaller than c', a direct predecessor of c', or incomparable to c'. We show the following: ($\star$) for every i and for every cluster part $\langle c, V \rangle$, if $\langle c, V \rangle \in \nu_i(o) \setminus \nu_{i-1}(o)$ is added in the construction of ν_i, then for every cluster part $\langle c', V' \rangle \in \nu_{i-1}(o)$, either $c \lneq_T c'$ or $c \lessgtr_T c'$. This property is seen by induction. It is true for $i = 0$: as p_{o_R} is (o_R, P)-austere for some part P of an initial cluster, it assigns at most one part per cluster to each o.

For $i > 0$ assume, towards a contradiction, that j is the least i for which ($\star$) fails at some element of $\Delta^{\mathcal{J}}$. Let o^* be a shortest such element (i.e., minimal w.r.t. to its depth in the tree $\Delta^{\mathcal{J}}$), and let o' be its parent. By assumption, ($\star$) holds for o'. Assume $\langle c^*, V^* \rangle \in \nu_j(o^*) \setminus \nu_{j-1}(o^*)$, and that there is some $\langle c^-, V^- \rangle \in \nu_{j-1}(o^*)$ such that $c^- \leq_T c^*$. Let p_o be the restriction of $\nu_j \setminus \nu_{j-1}$ to the subtree of $\mathcal{J}$ rooted at the j-th level that contains o^*, i.e., o is the unique ancestor of o^* in $\gamma(j, \nu_{j-1})$, and p_o is an (o, P)-austere marking for some part P of an initial cluster of q. First we note that $o^* \neq o$, as $p_o(o)$ contains a part of an initial cluster but $\nu_{j-1}(o)$ does not, while $p_o(o^*)$ contains a part of some cluster c^* that has a predecessor c^- for which there is a part in $\nu_{j-1}(o^*)$. Since p_o is (o, P)-austere, $\langle c^*, V^* \rangle \in p_o(o^*)$ implies that there is some c such that $c = c^*$ or $c + 1 =_T c^*$, and $\langle c, V \rangle \in p_o(o')$, i.e., since c^* must be justified in $p_o(o^*)$, a part of itself or of its direct predecessor must occur in $p_o(o')$. Similarly, since ν_{j-1} is a union of austere markings, $\langle c^-, V^-\rangle \in \nu_{j-1}(o^*)$ implies that there is some c' with $c' = c^-$ or $c' + 1 =_T c^-$, and such that $\langle c', V'\rangle \in \nu_{j-1}(o')$ for some $V' \subseteq c'$.

The existence of either a part of c^* in $p_o(o')$, or a part of the direct predecessor c' of c^- in $\nu_{j-1}(o')$, is enough to contradict the fact that o' satisfies $(\star)$, i.e., if $c = c^*$ or $c' + 1 =_T c^-$ we get $c' \leq_T c$. Otherwise, there is no part of c^* in $p_o(o')$, $\langle c, V\rangle \in p_o(o')$ for the direct predecessor c of c^*, and $\langle c^-, V^-\rangle \in \nu_{j-1}(o')$. As c^* has the unique direct predecessor c and $c^- \leq c^*$, either $c^- \leq c$ or $c^- = c^*$. The former clearly contradicts the satisfaction of $(\star)$ at o'. We only have to analyze the case where $c^- = c^*$, i.e., there are markings $\langle c^*, V^*\rangle \in p_o(o^*)$, $\langle c^*, V^-\rangle \in \nu_{j-1}(o^*)$, and $\langle c^*, V'\rangle \in \nu_{j-1}(o')$, as well as $\langle c, V\rangle \in p_o(o')$ for the unique direct predecessor c of c^*. Let r be the role with $(o', o^*) \in r^{\mathcal{J}}$. As there is no part of c^* in $p_o(o')$ and $\langle c^*, V^*\rangle \in p_o(o^*)$, then c is an r-cluster, there is a part of c in $p_o(o^*)$, and $V^* = c^*$. Also $\langle c^*, V'\rangle \in \nu_{j-1}(o')$ together with $\langle c^*, V^-\rangle \in \nu_{j-1}(o^*)$ implies that c^* is an r-cluster. Hence r is not transitive (as by definition there are no adjacent c_r, $c_{r'}$ with $r = r'$ transitive), and $V' = c^*$ follows. We have $\langle c^*, c^*\rangle \in \nu_{j-1}(o')$, and the presence of this cluster part must be justified. As c is the predecessor of c^*, there must be some $\langle c, W\rangle \in \nu_{j-1}(o')$ with $W \subseteq c$. But this together with $\langle c, V'\rangle \in p_o(o')$ contradicts the assumption that o' satisfies $(\star)$. □

4.2.2 A knot-elimination algorithm

We present an algorithm which tests the existence of models that admit a global marking. It is based on knots [12].

Definition 4.23 [Knot] A $\mathcal{T}$-*type* is a set $\tau \subseteq \mathsf{sub}(\mathcal{T})$ that satisfies, for all $C, D \in \mathsf{sub}(\mathcal{T})$: (a) $C \in \tau$ implies $\neg C \notin \tau$, (b) if $C \sqcap D \in \tau$, then $\{C, D\} \subseteq \tau$, (c) if $C \sqcup D \in \tau$, then $C \in \tau$ or $D \in \tau$, and (d) $C_{\mathcal{T}} \in \tau$. A *knot for* $\mathcal{T}$ is a pair $\kappa = (\tau, S)$ with τ a $\mathcal{T}$-type and S a set of pairs (r, τ') such that r is a role name that occurs in $\mathcal{T}$, τ' is a $\mathcal{T}$-type, and in addition:

(1) if $\exists r.C \in \tau$, then $C \in \tau'$ for some $(r, \tau') \in S$;

(2) if $\forall r.C \in \tau$, then $C \in \tau'$ for all $(r, \tau') \in S$;

(3) if $\forall r.C \in \tau \wedge r \in \mathsf{Tr}(\mathcal{K})$, then $\forall r.C \in \tau'$ for all $(r, \tau') \in S$;

(4) $|S| \leq |\mathsf{sub}(\mathcal{K})|$.

A knot $\kappa = (\tau, S)$ can be viewed as describing a fragment of a canonical model that consists of a node which satisfies the concepts in τ and its successors, as described by S. Our algorithm will represent canonical models as a set of knots. In fact, it is not hard to come up with conditions which guarantee that a given set of knots can be assembled into a canonical model. By transferring the marking conditions for canonical models (see Definition 4.15) to the setting of knots, we obtain conditions ensuring that a given knot set can be used to assemble a canonical model where there is a global marking for q.

Definition 4.24 [Marked Knot] A *marked knot* is a pair (κ, ν), where $\kappa = (\tau, S)$ is a knot and $\nu : S \cup \{\varepsilon\} \to 2^{PC}$ is a mapping such that, for any $e \in S \cup \{\varepsilon\}$, $\nu(e)$ contains no more than one part of each cluster of q.

As a convention, $\nu(\varepsilon)$ is the marking of the root of κ. The marking of a single knot must mimic a spoiling marking of a domain element and its immediate successors as given in Definition 4.15.

Definition 4.25 [q-avoiding] For a $\mathcal{T}$-type τ and a variable v, we say that v is *label-matched* at τ if $A(v) \in q$ implies $A \in \tau$ for any A. Then a marked knot (κ, ν) is *q-avoiding* (for $\mathcal{K}$), if (a) $\nu(\varepsilon)$ contains a part of an initial cluster of q, and (b) the following conditions hold for each $\langle c_r, V\rangle \in \nu(\varepsilon)$:

(K1) If $r \notin \mathsf{Tr}(\mathcal{K})$, $V = \{v, v'\}$ and $r(v, v') \in q$, then (a) v is not label-matched at τ, or (b) $(c_r, \{v'\}) \in \nu(\langle r, \tau'\rangle)$ for all $\langle r, \tau'\rangle \in S$;

(K2) If $r \notin \mathsf{Tr}(\mathcal{K})$ and $V = \{v\}$, then either (a) v is not label-matched at τ, or (b) $P' \in \nu(\varepsilon)$ for some P' that departs from v.

(K3) If $r \in \mathsf{Tr}(\mathcal{K})$, then there is some *consumed set* $S \subseteq \mathsf{min}(V)$ such that:

- for each $v \in \mathsf{min}(V) \setminus S$, either (a) v is not label-matched at τ, or (b) $P' \in \nu(\varepsilon)$ for some P' that departs from v, and
- there is some V' that contains some $v_\downarrow$ that does not reach any other $v' \in c_r$, and all variables that are connected to $v_\downarrow$ in $q|_{(V\setminus S)}$, and $\langle c_r, V'\rangle \in \nu(\langle r, \tau'\rangle)$ for all $\langle r, \tau'\rangle \in S$.

The following conditions ensure that sets of marked knots finitely represent marked canonical models.

Definition 4.26 A set $\mathfrak{K}$ of q-avoiding knots is *consistent* if the following are true:

1. for each $(\kappa, \nu) \in \mathfrak{K}$ with $\kappa = (\tau, S)$, and for each $(r, \tau') \in S$, there exists $(\kappa_s, \nu_s) \in \mathfrak{K}$ with $\kappa_s = (\tau_s, S_s)$ such that $\tau_s = \tau'$ and $\nu_s(\varepsilon) = \nu(r, \tau')$.
2. there is a $(\kappa, \nu) \in \mathfrak{K}$ with $\kappa = (\tau, S)$ such that $C_0 \in \tau$ (recall that the ABox is of the form $\{C_0(a)\}$).

Proposition 4.27 *There exists a global marking for a canonical model $\mathcal{J}$ of $\mathcal{K}$ and q iff there exists a consistent set of q-avoiding knots for $\mathcal{K}$.*

Proof. ($\Leftarrow$) Let $\mathfrak{K}$ be a consistent set of q-avoiding knots and let $b = \max\{|S| \mid (\tau, S) \in \mathfrak{K}\}$. To obtain a canonical model $\mathcal{J}$ for $\mathcal{K}$ and a global marking μ for $\mathcal{J}$ and q, we take the tree $\Delta = 1\cdot\{1, \ldots, b\}^*$. Let $\delta : \Delta \to \mathfrak{K}$ be a partial function inductively defined as follows:

- $\delta(1) = (\kappa, \nu)$ for some $(\kappa, \nu) \in \mathfrak{K}$ with $\kappa = (\tau, S)$ and $C_0 \in \tau$;
- if $\delta(o) = ((\tau, S), \nu)$ has been defined, then for each $(r, \tau') \in S$, we choose some $o \cdot i \in \Delta$ and set $\delta(o \cdot i) = (\kappa_s, \nu_s)$ for some $(\kappa_s, \nu_s) \in \mathfrak{K}$ with $\kappa_s = (\tau_s, S_s)$ such that $\tau_s = \tau'$ and $\nu_s(\varepsilon) = \nu(r, \tau')$.

Such a δ exists by the consistency of $\mathfrak{K}$ (Definition 4.26). Let $\Delta' = dom(\delta)$ ($\subseteq \Delta$) be the domain of δ.

We can now define a canonical model $\mathcal{J}_\delta$ for $\mathcal{K}$, by taking $\mathcal{J} = (\Delta', \cdot^{\mathcal{J}_\delta})$, $a^{\mathcal{J}_\delta} = 1$ and:

1. for each concept name $C \in \mathsf{sub}(\mathcal{K})$, $C^{\mathcal{J}_\delta} = \{o \in \Delta' \mid \delta(o) = ((\tau, S), \nu) \text{ and } C \in \tau\}$,
2. for each role r, $r^{\mathcal{J}_\delta} = \{(o, o \cdot i) \in \Delta' \times \Delta' \mid \delta(o) = ((\tau, S), \nu), (r, \tau') \in S, \delta(o \cdot i) = ((\tau_s, S_s), \nu_s), \tau_s = \tau', \nu_s(\varepsilon) = \nu(r, \tau')\}$.

To define a global marking μ_δ for $\mathcal{J}_\delta$ and q, we simply set $\mu_\delta(o) = \nu(\varepsilon)$ for each o with $\delta(o) = (\kappa, \nu)$.

($\Rightarrow$) Let $\mathcal{J}$ be a canonical model for $\mathcal{K}$ and let μ be a global marking for $\mathcal{J}$ and q. Due to Proposition 4.22 we can assume that μ assigns to each domain element no more than one part of each cluster of q. For $o \in \Delta^{\mathcal{J}}$, we define $\mathsf{type}(o) = \{C \in \mathsf{sub}(\mathcal{K}) \mid o \in C^{\mathcal{J}}\}$.

We define a marked knot $mk(o) = ((\tau_o, S_o), \nu_o)$ for each $o \in \Delta^{\mathcal{J}}$ as follows:

Algorithm Knot-Elim$(\mathcal{K}, q)$
Input: KB $\mathcal{K} = (\mathcal{T}, \{C_0(a)\})$, (pseudo-tree) query q
Output: "Yes" iff a consistent q-avoiding knot set for KB exists

```
Compute the set $\mathfrak{K}_0$ of marked $q$-avoiding knots for $\mathcal{T}$
$i := 0$
**repeat**
  $i := i + 1$
  $\mathfrak{K}_i := \mathfrak{K}_{i-1} \setminus \{(\kappa, \nu) \in \mathfrak{K}_{i-1} \mid (\kappa, \nu)$ bad in $\mathfrak{K}_{i-1}\}$
**until** $\mathfrak{K}_i = \mathfrak{K}_{i-1}$
**if** there exists some $(\kappa, \nu) \in \mathfrak{K}_i$ with $\kappa = (\tau, S)$ and $C_0 \in \tau$ then
  **return** "Yes"
**else return** "No"
```

Figure 8: The knot elimination algorithm.

- $\tau_o = \mathsf{type}(o)$
- $\nu_o(\varepsilon) = \mu(o)$,
- $S_o = \{(r, \tau') \mid \exists (o, o') \in r^{\mathcal{J}}, \mathsf{type}(o') = \tau', \nu_o(r, \tau') = \mu(o')\}$.

It can be easily verified that $\mathfrak{K} = \{mk(o) \mid o \in \Delta^{\mathcal{J}}\}$ is a consistent set of q-avoiding knots.

□

We now provide an algorithm for checking existence of consistent q-avoiding knot sets. It is a kind of type elimination as first used by Pratt [13] in the context of propositional dynamic logic, but works on marked knots instead of types. In a nutshell, we start with the set of all marked q-avoiding knots, and then repeatedly eliminate knots that cannot be part of any marked canonical model. In the end, we check whether there is a surviving knot that contains the concept C_0 from the ABox (cf. Definition 4.26). The following definition formalizes the condition for elimination.

Definition 4.28 [Bad] Let $\mathfrak{K}$ be a set of marked knots and let $(\kappa, \nu) \in \mathfrak{K}$ with $\kappa = (\tau, S)$. We say that (κ, ν) is *bad* in $\mathfrak{K}$, if there is some $(r, \tau') \in S$ for which there is no $(\kappa_s, \nu_s) \in \mathfrak{K}$ with $\kappa_s = (\tau_s, S_s)$ such that $\tau_s = \tau'$ and $\nu_s(\varepsilon) = \nu(r, \tau')$.

The algorithm is given in Figure 8. It is readily checked that it terminates, as there is only a finite set of marked knots for $\mathcal{T}$. If it answers "Yes," then the computed set of marked knots is consistent; furthermore, we can construct from it a canonical model $\mathcal{I}$ of $\mathcal{K}$ in which q has no pre-match (Proposition 4.27). Conversely, if $\mathcal{K}$ has a canonical model $\mathcal{I}$ such that $\mathcal{I} \not\models^{\mathsf{pre}} q$, then we can generate a set of marked q-avoiding knots from $\mathcal{I}$ and show that none of them is eliminated by the algorithm. As the algorithm terminates and one of the generated knots contains C_0 in the root type, it returns "Yes."

Proposition 4.29 *The algorithm* Knot-Elim *is sound, complete, and terminates.*

To establish Theorem 4.3, it remains to show that Knot-Elim runs in exponential time. Let n be the size of $\mathcal{K}$ and m the size of q. It suffices to show that the number of marked knots is single exponential in $n+m$; note that q-avoidance of a marked knot is easily checked nondeterministically in polynomial time in $n+m$. The number of $\mathcal{T}$-types is bounded by 2^n and the number of knots by $2^{\mathcal{O}(n^2)}$ (note condition (4.23.4)). There are at most m clusters for q, and each cluster has less than 2^m parts. A marking assigns to each node of a knot at most one part per cluster, thus we have at most $(2^m)^m = 2^{m^2}$ candidate markings per node. As the number of nodes in a knot is bounded by $\mathcal{O}(n)$, we have for each knot at most $2^{\mathcal{O}(m^2 n)}$ candidate markings. It follows that there are at most $2^{\mathcal{O}(n(n+m^2))}$ marked knots, which is bounded by $2^{\mathcal{O}(n^2 m^2)}$.

5 Related Work and Conclusions

We showed that deciding CQ non-entailment in $\mathcal{SH}$, which supports transitive roles and role hierarchies, is 2-EXPTIME-hard, and therefore provably harder (by one exponential) than the standard reasoning tasks, like satisfiability and instance checking, in a number of DLs for which the latter problems are EXPTIME-complete. We also showed that the problem is in EXPTIME for $\mathcal{S}$ for knowledge bases that have tree-shaped ABoxes, but is NEXPTIME-hard in general.

In the light of this, a natural question is under which other restrictions CQs non-entailment over $\mathcal{SH}$ knowledge bases has lower complexity. For $\mathcal{ALCI}$, where CQ non-entailment is as hard as in $\mathcal{SH}$, the complexity drops to NEXPTIME-complete if at least one variable must be mapped to the ABox [8]. As already remarked there, this does not reduce the worst case complexity in presence of role hierarchies and transitivity. In fact, the query q_w in our reduction above can be easily adapted, by adding a fresh variable x^r and atoms $t(x^r, x^A)$ that connect x^r to the roots x^A of all the components of q_w.

In [12], the *order-freeness degree (OFD)* was introduced as a measure of the structural complexity of CQs, which roughly is the maximum number of query variables that reach in the query graph a common sink via a transitive role, but mutually not each other. As shown there, entailment of CQs with OFD bounded by a constant is EXPTIME-complete for $\mathcal{SH}$ (note that the query q_w has unbounded OFD). As a simple consequence, all queries with at most constantly many variables in transitive role atoms are decidable in EXPTIME. This contrasts the result that CQ entailment in $\mathcal{SHIQ}$ is 2-EXPTIME-hard even for queries with only two variables of [6].

Finally, the 2-EXPTIME hardness of CQ entailment for $\mathcal{SH}$ and for $\mathcal{ALCI}$ [8] matches the known upper bounds for *unions of CQs* over $\mathcal{SHIQ}$ KBs [4] and the even more expressive *two-way positive regular path queries* over $\mathcal{ALCQI}b_{reg}$ KBs from [1]. This shows that, once either inverse roles or role hierarchies and transitivity are allowed, one can significantly extend both the query language and the Description Logic without further increase of the worst case complexity.

References

[1] D. Calvanese, T. Eiter, and M. Ortiz. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007)*, pages 391–396, 2007.

[2] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[3] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI 2007)*, pages 399–404, 2007.

[4] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic shiq. *Journal of Artificial Intelligence Research*, 31:157–204, 2008.

[5] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query entailment for $\mathcal{SHOQ}$. In *Proc. of the 2007 Description Logic Workshop (DL 2007)*, volume 250 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/Vol-250/`, pages 65–75, 2007.

[6] B. Glimm and Y. Kazakov. Role conjunctions in expressive description logics. Technical report, Oxford University Computing Laboratory, 2008.

[7] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, 2002.

[8] C. Lutz. The complexity of conjunctive query answering in expressive description logics. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR2008)*, number 5195 in LNAI, pages 179–193. Springer, 2008.

[9] C. Lutz, C. Areces, I. Horrocks, and U. Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research (JAIR)*, 23:667–726, 2005.

[10] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *J. of Automated Reasoning*, 41(1):61–98, 2008. `doi:10.1007/s10817-008-9102-9`. Preliminary version available as Tech.Rep. INFSYS RR-1843-07-07, Institute of Information Systems, TU Vienna, Nov. 2007.

[11] M. Ortiz, M. Šimkus, and T. Eiter. Conjunctive query answering in $\mathcal{SH}$ using knots. In F. Baader, C. Lutz, and B. Motik, editors, *Proceedings of the 21st International Workshop on Description Logics (DL2008), May 13-16, Dresden, Germany*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[12] M. Ortiz, M. Šimkus, and T. Eiter. Worst-case optimal conjunctive query answering for an expressive description logic without inverses. In D. Fox and C. P. Gomes, editors, *AAAI*, pages 504–510. AAAI Press, 2008.

[13] V. R. Pratt. Models of program logics. In *FOCS*, pages 115–122. IEEE, 1979.

[14] U. Sattler. Description logics for the representation of aggregated objects. In W. Horn, editor, *ECAI*, pages 239–243. IOS Press, 2000.

[15] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI 1991)*, pages 466–471, 1991.

[16] S. Tessaris. *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, University of Manchester, Department of Computer Science, Apr. 2001.