Status $\mathcal{QIO}$: An Update

Birte Glimm[1], Yevgeny Kazakov[1], and Carsten Lutz[2]

[1] Oxford University Computing Laboratory, UK
[2] Universität Bremen, Germany

Abstract. We prove co-N2ExpTime-hardness for conjunctive query entailment in the description logic $\mathcal{ALCOIF}$, thus improving the previously known 2ExpTime lower bound. The result transfers to OWL DL and OWL2 DL, of which $\mathcal{ALCOIF}$ is an important fragment. A matching upper bound remains open.

1 Introduction

Due to its importance for ontology-based data access and data integration conjunctive query (CQ) answering has developed into one of the most widely studied reasoning tasks in description logic (DL). Nevertheless, the precise complexity (and sometimes even decidability) of CQ answering in several important expressive DLs is still an open problem. In particular, this concerns fragments of the W3C-standardized OWL DL ontology language that comprise nominals, inverse roles, and number restrictions, a combination of expressive means that is notorious for interacting in intricate ways. In this paper, we concentrate on the basic such fragment $\mathcal{ALCOIF}$ in which number restrictions take the form of global functionality constraints.

Decidability of CQ answering in $\mathcal{ALCOIF}$ and its extension $\mathcal{ALCOIQ}$ with qualified number restrictions has been shown only very recently [1]. Since the proof is based on a mutual enumeration of finite models and theorems of first-order logic, it does not yield any upper complexity bound. The best known lower bound for CQ answering in $\mathcal{ALCOIF}$ is 2ExpTime, inherited from the fragment $\mathcal{ALCI}$ of $\mathcal{ALCOIF}$ that does not include nominals and functionality constraints [2, 3]. The aim of this paper is to improve upon this lower bound by establishing co-N2ExpTime-hardness. Note that CQ answering in the fragment $\mathcal{ALCIF}$ of $\mathcal{ALCOIF}$ that does not include nominals is in 2ExpTime [4], and the same is true for the fragment $\mathcal{ALCQO}$ that does not include inverse roles [5] and $\mathcal{ALCOI}$ that does not include functionality restrictions [6]. Thus, our result shows that the combination of nominals, inverse roles, and number restrictions leads to an increase of complexity of CQ answering from 2ExpTime to (at least) co-N2ExpTime. This parallels the situation for the subsumption problem, which is co-NExpTime-complete for $\mathcal{ALCOIF}$, but ExpTime-complete in any of $\mathcal{ALCIF}$, $\mathcal{ALCQO}$, and $\mathcal{ALCOI}$. Since $\mathcal{ALCOIF}$ is a fragment of OWL DL (in both the OWL1 and the OWL2 version), our co-N2ExpTime lower bound obviously also applies to CQ answering in this language.

We prove our result by a reduction of the tiling problem that requires to tile a torus of size $2^{2^n} \times 2^{2^n}$. Our construction combines elements of two existing hardness proofs, but also requires the development of novel ideas. We follow the general strategy of the proofs that show N2ExpTime-hardness of satisfiability in $\mathcal{SROIQ}$ [7] and in the extension of $\mathcal{SHOIF}$ with role conjunctions [8]. One central part of those proofs is the realization of a counter that counts up to 2^{2^n}. We realize this counter using a (rather subtle!) adaptation of the conjunctive queries that have been developed in [2, 3] to establish 2ExpTime-hardness of CQ-answering in $\mathcal{ALCI}$.

An extended technical report including proofs and further details is available [9].

2 Preliminaries

We assume standard notation for the syntax and semantics of $\mathcal{ALCOIF}$ knowledge bases [10]. The presence of nominals allows for only working with TBoxes, which consist of concept inclusions (CIs) $C \sqsubseteq D$. A *knowledge base (KB)* is then simply a TBox. Let N_V be a countably infinite set of variables. An atom is an expression $C(v)$ or $r(v, v')$, where C is a (potentially compound) $\mathcal{ALCOIF}$-concept, r is an atomic role, and $v, v' \in N_V$.[3] A conjunctive query q is a finite set of atoms. We use $\mathsf{Var}(q)$ to denote the set of variables that occur in the query q. Let $\mathcal{K}$ be an $\mathcal{ALCOIF}$ KB, $\mathcal{I} = (\cdot^{\mathcal{I}}, \Delta^{\mathcal{I}})$ a model of $\mathcal{K}$, q a conjunctive query, and $\pi \colon \mathsf{Var}(q) \to \Delta^{\mathcal{I}}$ a total function. We write $\mathcal{I} \models^{\pi} C(v)$ if $\pi(v) \in C^{\mathcal{I}}$ and $\mathcal{I} \models^{\pi} r(v, v')$ if $\langle \pi(v), \pi(v') \rangle \in r^{\mathcal{I}}$. If $\mathcal{I} \models^{\pi} at$ for all $at \in q$, we write $\mathcal{I} \models^{\pi} q$ and call π a *match* for $\mathcal{I}$ and q. We say that $\mathcal{I}$ *satisfies* q and write $\mathcal{I} \models q$ if there is a match π for $\mathcal{I}$ and q. If $\mathcal{I} \models q$ for all models $\mathcal{I}$ of a KB $\mathcal{K}$, we write $\mathcal{K} \models q$ and say that $\mathcal{K}$ *entails* q. The *conjunctive query entailment problem* is, given a knowledge base $\mathcal{K}$ and a query q, to decide whether $\mathcal{K} \models q$. This is the decision problem corresponding to query answering, see e.g. [4].

A *domino system* is a triple $D = (T, H, V)$, where $T = \{1, \ldots, k\}$ is a finite set of *tiles* and $H, V \subseteq T \times T$ are *horizontal* and *vertical matching relations*. A *tiling* of $m \times m$ for a domino system D with *initial condition* $c^0 = \langle t_1^0, \ldots, t_n^0 \rangle$, $t_i^0 \in T$ for $1 \leq i \leq n$, is a mapping $t \colon \{0, \ldots, m-1\} \times \{0, \ldots, m-1\} \to T$ such that $\langle t(i, j), t(i+1 \bmod m, j) \rangle \in H$, $\langle t(i, j), t(i, j+1 \bmod m) \rangle \in V$, and $t(i, 0) = t_{i-1}^0$ $(0 \leq i, j < m)$. There exists a domino system D_0 for which it is N2ExpTime-complete to decide, given an initial condition c^0 of length n, whether D_0 admits a tiling of $2^{2^n} \times 2^{2^n}$ with initial condition c^0 [11].

3 Conjunctive Query Entailment in $\mathcal{ALCOIF}$

Our aim is to construct, for an initial condition c^0 of length n, an $\mathcal{ALCOIF}$-KB $\mathcal{K}_0$ and conjunctive query q_0 such that $\mathcal{K}_0 \not\models q_0$ iff D_0 admits a tiling of $2^{2^n} \times 2^{2^n}$ with initial condition c^0.

Intuitively, the models of $\mathcal{K}_0$ that we are interested in have the form depicted in Figure 1: a torus of dimension $2^{2^n} \times 2^{2^n}$, where the lower left corner is identified by the nominal o, the upper right corner by the nominal e, each horizontal dashed arrow denotes the role h, and each vertical dotted arrow the role v. We will install two counters that identify the vertical and horizontal position of torus nodes. To store the counter values, we use binary trees of (roughly) depth n below the torus nodes, where each

[3] Complex concepts C in atoms $C(x)$ are used w.l.o.g.; to eliminate them, we can replace $C(x)$ with $A_C(x)$ for a fresh atomic concept A_C and add $C \sqsubseteq A_C$ to the TBox.

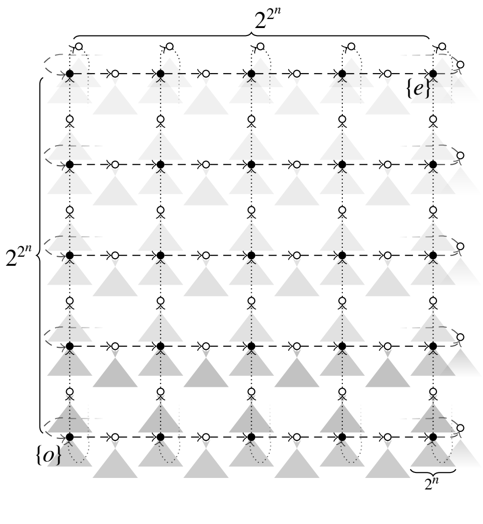


Fig. 1. Schematic depiction of the torus

of the 2^n leaves store one bit of each counter (represented via concept names X and Y). The filled circles in Figure 1 denote true torus nodes, which are labeled by a tile later on, while the unfilled circles denote auxiliary nodes that will help us in properly incrementing the counters. This incrementation is the main difficulty of the reduction, and it is achieved with the help of the query q_0. As the details are intricate, we defer a discussion of the details until later, and first concentrate on the construction of $\mathcal{K}_0$.

The following concept inclusions (1) to (9) of $\mathcal{K}_0$ lay the foundation for enforcing the torus structure with attached trees. Successors in trees are connected via the composition of the roles r^- and r, from now on denoted by $r^-;r$. This is needed in the query construction later on, similar to the use of symmetric roles in [2, 3]. We call additional nodes between r^- and r the 'intermediate' tree nodes. Note that no branching occurs at intermediate nodes. Also for the query construction, the root of a tree below a true torus node is the torus node itself while the root of a tree below an auxiliary torus node is reachable by traveling one step along the role r (see Figure 1). To distinguish these two kinds of trees, we label trees of the former kind with the concept name B and call them black trees, and trees of the latter kind with the concept name W and call them white trees. Later on, we will use white trees that are on the vertical axis to increment the vertical counter and white trees that are on the horizontal axis to increment the horizontal counter. To support this, we further label white trees of the former kind with V and white trees of the latter kind with H. The basic idea for constructing the torus itself is similar to what is done in [12, 7, 8]: the maximum value of both counters (indicated by the concept names M_X and M_Y) identifies the upper right corner, which has to satisfy the nominal e and is thus unique. Inverse functionality for h and v then guarantees uniqueness of elements for all other values of the horizontal and vertical counters, and that the torus 'closes' in the expected way. We use concept names $L_0, \ldots, L_n$ to mark the levels of the trees, to deal with the symmetry of the composition $r^-;r$. Thus, the

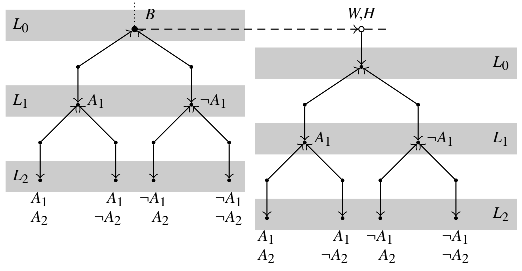


Fig. 2. A black and a white tree, for $n = 2$

concept $B \sqcap L_0$ identifies the true torus nodes.

$$\{o\} \sqsubseteq B \sqcap L_0 \tag{1}$$

$$B \sqcap L_0 \sqsubseteq \exists h.(W \sqcap \exists r.(H \sqcap W \sqcap L_0) \sqcap \exists h.(B \sqcap L_0)) \tag{2}$$

$$B \sqcap L_0 \sqsubseteq \exists v.(W \sqcap \exists r.(V \sqcap W \sqcap L_0) \sqcap \exists v.(B \sqcap L_0)) \tag{3}$$

$$B \sqcap L_0 \sqcap M_X \sqcap M_Y \sqsubseteq \{e\} \tag{4}$$

$$L_i \sqsubseteq \exists r^-.\exists r.(A_{i+1} \sqcap L_{i+1}) \sqcap \exists r^-.\exists r.(\neg A_{i+1} \sqcap L_{i+1}) \quad {\scriptstyle i<n} \tag{5}$$

$$A_i \sqcap L_j \sqsubseteq \forall r^-.\forall r.(L_{j+1} \rightarrow A_i) \qquad {\scriptstyle 1\leq i\leq j<n} \tag{6}$$

$$\neg A_i \sqcap L_j \sqsubseteq \forall r^-.\forall r.(L_{j+1} \rightarrow \neg A_i) \qquad {\scriptstyle 1\leq i\leq j<n} \tag{7}$$

$$\mathsf{C} \sqsubseteq \forall r.\mathsf{C} \sqcap \forall r^-.\mathsf{C} \qquad \text{for all } \mathsf{C} \in \{B, W, H, V\} \tag{8}$$

$$\top \sqsubseteq \leqslant 1h^-.\top \qquad \top \sqsubseteq \leqslant 1v^-.\top \tag{9}$$

Note that the concept names $A_1, \ldots, A_n$ implement a binary counter for the leafs of the trees, i.e., for counting the bit positions in the horizontal and vertical counters. In summary, the internal structure of the trees is as shown in Figure 2 where branching tree nodes have dark background and intermediate nodes have light background.

The next step is to make sure that the horizontal and vertical counter have value 0 at the origin and that M_X is true at the root of a tree when the horizontal counter has reached the maximum value, and similarly for M_Y. We use $\forall(r^-; r)^n.C$ to denote the $2n$-quantifier prefixed $\forall r^-.\forall r.\cdots\forall r^-.\forall r.C$. Recall that the concept name X represents the truth value of bits of the horizontal counter, and likewise for Y and the vertical counter.

$$\{o\} \sqsubseteq \forall(r^-; r)^n.(\neg X \sqcap \neg Y) \tag{10}$$

$$L_n \sqsubseteq (X \leftrightarrow M_X) \sqcap (Y \leftrightarrow M_Y) \tag{11}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap M_X \sqcap A_i) \sqcap \exists r^-.\exists r.(L_i \sqcap M_X \sqcap \neg A_i) \sqsubseteq M_X \quad {\scriptstyle 0<i\leq n} \tag{12}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap M_Y \sqcap A_i) \sqcap \exists r^-.\exists r.(L_i \sqcap M_Y \sqcap \neg A_i) \sqsubseteq M_Y \quad {\scriptstyle 0<i\leq n} \tag{13}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap \neg M_X) \sqsubseteq \neg M_X \quad {\scriptstyle 0<i\leq n} \tag{14}$$

$$L_{i-1} \sqcap \exists r^-.\exists r.(L_i \sqcap \neg M_Y) \sqsubseteq \neg M_Y \quad {\scriptstyle 0<i\leq n} \tag{15}$$

The general strategy for updating the horizontal and vertical counter is as follows. We introduce additional concept names X' and Y', which represent the truth value of the bits of two additional binary counters, the 'primed versions' of the horizontal and vertical counter. Using $\mathcal{K}_0$, we ensure that, in black trees, the X-counter has the same value as the X'-counter, and likewise for the Y- and Y'-counter. In white trees, we distinguish between horizontal incrementation indicated by the concept name H and vertical incrementation indicated by the concept name V: if the tree satisfies H, then the value of the X'-counter is the value of the X-counter incremented by one, while the values of the Y- and Y'-counter coincide; if the tree satisfies V, it is the other way around. The remaining job to be accomplished by the query q_0 is then to

($*$) ensure that the value of the X'-counter (resp. Y'-counter) in a (black or white) tree is identical to the value of the X-counter (resp. Y-counter) in its 'successor trees', i.e., in trees that can be reached by traveling a single step in the torus along the roles h or v.

This behavior of the counters, with the exception of ($*$), is implemented by the subsequent concept inclusions. To increment a counter, we use a concept name F to mark the bits that have to be flipped. Another concept name S, which is propagated down from the root to a single leaf, is used to mark the unique bit of the incremented counter such that (i) all bits to the right are flipped from 1 to 0, (ii) the bit itself is flipped from 0 to 1, and (iii) all bits to the left remain unchanged. As a special case, all bits flip when the maximum counter value has been reached. In the following, CIs (16) to (19) implement the proper marking by F and S, CIs (21) to (23) realize the actual incrementation of the X-counter to the X'-counter in (white) H-trees, and CI (24) ensures that the Y- and Y'-counters have the same value in H-trees and in black trees. We also need CIs (21) to (24) with H replaced by V, X by Y, X' by Y', Y by X, and Y' by X'.

$$L_0 \sqcap (\neg M_X \sqcup \neg M_Y) \sqsubseteq S \tag{16}$$

$$L_0 \sqcap (M_X \sqcap M_Y) \sqsubseteq F \sqcap \neg S \tag{17}$$

$$\begin{aligned} S \sqcap L_{i-1} \sqsubseteq \forall r^-.\forall r.(L_i \rightarrow [\, ((A_i \rightarrow \neg F \sqcap \neg S) \sqcap (\neg A_i \rightarrow S)) \\ \sqcup ((A_i \rightarrow S) \sqcap (\neg A_i \rightarrow F \sqcap \neg S)) \,]) \quad {\scriptstyle 0<i\leq n} \end{aligned} \tag{18}$$

$$F \sqcap \neg S \sqcap L_{i-1} \sqsubseteq \forall r^-.\forall r.(L_i \rightarrow F \sqcap \neg S) \quad {\scriptstyle 0<i\leq n} \tag{19}$$

$$\neg F \sqcap \neg S \sqcap L_{i-1} \sqsubseteq \forall r^-.\forall r.(L_i \rightarrow \neg F \sqcap \neg S) \quad {\scriptstyle 0<i\leq n} \tag{20}$$

$$H \sqcap L_n \sqcap F \sqcap \neg S \sqsubseteq X \sqcap \neg X' \tag{21}$$

$$H \sqcap L_n \sqcap S \sqsubseteq \neg X \sqcap X' \tag{22}$$

$$H \sqcap L_n \sqcap \neg F \sqcap \neg S \sqsubseteq (X \sqcap X') \sqcup (\neg X \sqcap \neg X') \tag{23}$$

$$(B \sqcup H) \sqcap L_n \sqsubseteq (Y \sqcap Y') \sqcup (\neg Y \sqcap \neg Y') \tag{24}$$

To enable the construction of a query q_0 that enforces ($*$), we add a further (single) $r^-; r$-successor to each leaf in each tree. At this extra node, which is marked with the concept name L_{n+1}, the truth value of all concept names A_i, X, X', Y, Y' is complemented compared to its predecessor L_n-node. We also introduce a marker concept Q that is true at the intermediate node between each L_n-node and L_{n+1}-node. This is similar to what is done in [2, 3]. We call such intermediate nodes Q-nodes.

$$L_n \sqsubseteq \exists r^-.(Q \sqcap \exists r.L_{n+1}) \tag{25}$$

$$L_n \sqcap \mathsf{C} \sqsubseteq \forall r^-.\forall r.(L_{n+1} \rightarrow \neg\mathsf{C}) \qquad L_n \sqcap \neg\mathsf{C} \sqsubseteq \forall r^-.\forall r.(L_{n+1} \rightarrow \mathsf{C}) \quad \text{for all } \mathsf{C} \in \{A_1, \ldots, A_n, X, X', Y, Y'\} \tag{26}$$

The construction of $\mathcal{K}_0$ is not yet finished. However, it will be more convenient to construct the remaining part along with the query q_0. The query is assembled from

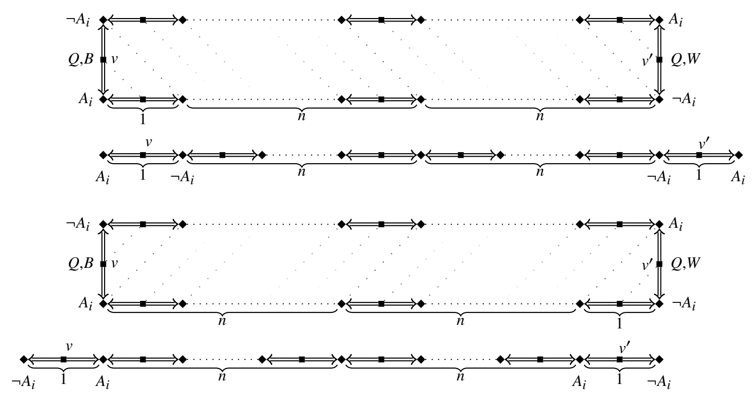


Fig. 3. A counting component of the query q_0 and the two ways to fold it

two types of components: *counting components* and *copying components*. We start with presenting and explaining a simplified version of counting components, which are then refined in a second step. The final query q_0 will contain one counting component for each bit of the counter $A_1, \ldots, A_n$ that counts the leaves of our trees. The simplified version of the counting component for A_i is shown as the topmost cycle in Figure 3, where, for the moment, every arrow should be interpreted as a role atom that uses the role name r. The goal is that matches of this query component should map (i) v to a Q-node of a black tree and v' to the Q-node of a white successor tree such that the two predecessor L_n-nodes agree on the value of A_i or, symmetrically, (ii) v' to a Q-node of a white tree and v to the Q-node of a black successor tree such that the two predecessor L_n-nodes agree on the value of A_i. By taking the union of all counting queries for $A_1, \ldots, A_n$ such that the variables v and v' are shared, we thus link leaves of successor trees that represent the same bit position for the horizontal and vertical counter, which is the first important step towards enforcing ($*$).

Due to the Q-concept at v and v', each variable labeled with A_i or $\neg A_i$ is matched to an L_n-node or an L_{n+1}-node. Ignoring the presence of the role names h and v in the torus and pretending that white trees are rooted directly on the torus, each match of the counting component gives rise to one of the two 'foldings' presented in Figure 3. These foldings are obtained by identifying variables that are matched to the same domain element, as indicated by the dotted lines. Intuitively, the two foldings correspond to the bit A_i being false (upper folding) and true (lower folding). For brevity, we omit the concept names Q, B, W in the foldings. Since the long sides of the counting component are of length $2n + 1$ (counted in terms of compositions $r^-; r$) and trees are of depth n, the two trees involved in a match cannot be further away than one step in the torus. Due to the use of B and W, they cannot be identical.

In the discussion of the simplified counting components above, we have neglected the presence of the roles h and v in the torus that we need to 'cross' when matching the query in the described way. Refining the counting queries to deal with these roles is the major challenge in the current reduction, compared to the 2ExpTime lower bound in [2, 3] where only a single role r is used. Note that we cannot just introduce a single

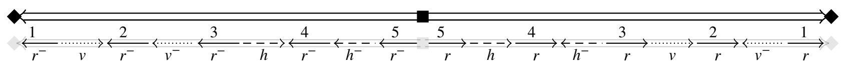


Fig. 4. Internal structure of a meta role consisting of 18 roles

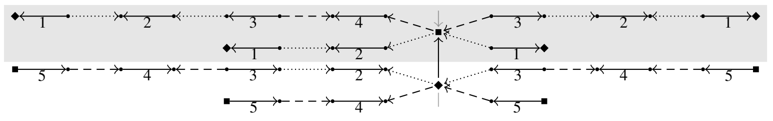


Fig. 5. Side chains for branching (upper) and intermediate (lower) nodes

h-arrow and v-arrow into the counting components since we want to match either h or v, but not both; moreover, the position of the h-arrow/v-arrow would shift back and forth with the different ways to fold the query. To solve the problem, we replace each role composition $r^-; r$ in the query (but not in the trees!) with a composition of 18 roles that we call a 'meta role', see Figure 4 where every solid edge denotes the role name r.

Note that the meta role is symmetric, like the composition $r^-; r$. The aim is that each meta-role in the refined counting query matches one $r^-; r$-role composition that connects two successor nodes in a tree. To resolve the mismatch between the role composition $r^-; r$ of length two and the meta role of length 18, the meta role is designed such that the remaining parts can be folded away into 'side chains' that we will add to the tree, i.e., chains of roles that start at each tree node. There are five ways to achieve such a folding, one for each corresponding pair of r^-- and r-arrows in the left and right half of the meta role. For example, we can use the 3rd r^--arrow and the 3rd r-arrow to match the $r^-; r$-composition in the tree, and then have to fold away the prefix composition $r^-; v; r^-; v^-$ before the 3rd r^--arrow, the infix composition $h; r^-; h^-; r^-; r; h; r; h^-$ between the 3rd r^--arrow and the 3rd r-arrow, and the postfix composition $v; r; v^-; r$ following the 3rd r-arrow. Observe that the infix composition is symmetric and thus can be folded into a chain. The postfix composition is the converse of the infix composition, which will allow us to leave a side chain that we have entered with the postfix composition using the prefix composition of the subsequent meta role. Similar foldings allow us to match the $r^-; r; h; r$-compositions required to move up one level in a black tree and then cross via an h-edge to the root of a white successor tree, the $r^-; h; r^-; r$-compositions that allows us to cross from the root of a white tree to a black tree and then move down one level, and to perform the two analogous crossing with h replaced by v.

The scheme for adding side chains is shown in Figure 5, where intermediate tree nodes (lower node on the center line) receive different chains than branching tree nodes (upper node on the center line). These chains are added to every node in the tree with the exception of the roots of black trees, as those are directly on the torus and adding side chains would violate inverse functionality of h and v. Note that the side chains attached to branching tree nodes are precisely the possible postfix compositions mentioned above, while the side chains attached to intermediate tree nodes are foldings of what we called infix compositions above. The chains are generated by the following CIs, to be added to $\mathcal{K}_0$. We use the concept $N_B = (L_0 \sqcap W) \sqcup \bigsqcup_{1 \leq i \leq n+1} L_i$ to identify

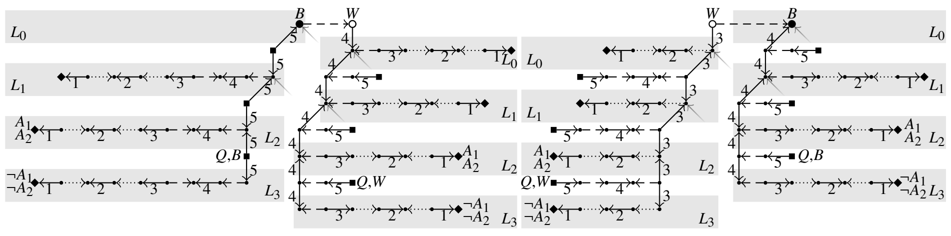


Fig. 6. From black to white trees via h (left) and from white to black trees via h (right)

branching tree nodes and $N_I = \exists r. \bigsqcup_{1 \le i \le n+1} L_i$ to identify intermediate tree nodes.

$$\begin{aligned} N_B \sqsubseteq\ & \exists h.\exists r.\exists h^-.\exists r.\exists v.\exists r.\exists v^-.\exists r.\top \sqcap \exists v.\exists r.\exists v^-.\exists r.\top \sqcap \\ & \exists h^-.\exists r.\exists v.\exists r.\exists v^-.\exists r.\top \sqcap \exists v^-.\exists r.\top \end{aligned} \tag{27}$$

$$\begin{aligned} N_I \sqsubseteq\ & \exists v.\exists r^-.\exists v^-.\exists r^-.\exists h.\exists r^-.\exists h^-.\exists r^-.\top \sqcap \exists h.\exists r^-.\exists h^-.\exists r^-.\top \sqcap \\ & \exists v^-.\exists r^-.\exists h.\exists r^-.\exists h^-.\exists r^-.\top \sqcap \exists h^-.\exists r^-.\top \end{aligned} \tag{28}$$

In matches of the refined query, the endpoints of the two foldings shown in Figure 3 will match at the end of (possibly empty) side chains at the level $n+1$, v and v' will match at the end of the side chains between the levels n and $n+1$, and the adjacent inner nodes labeled with $\neg A_i$ resp. A_i will match at the end of the side chains at the level n. For this reason, we propagate all relevant concept names to the end of those chains. For $\mathsf{C} \in \{A_1, \ldots, A_n, \neg A_1, \ldots, \neg A_n, X, \neg X, Y, \neg Y\}$, $\mathsf{C}' \in \{Q, B, W\}$, add the concept inclusions

$$\begin{aligned} (L_n \sqcup L_{n+1}) \sqcap \mathsf{C} \sqsubseteq\ & \forall h.\forall r.\forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.\mathsf{C} \sqcap \forall v.\forall r.\forall v^-.\forall r.\mathsf{C} \sqcap \\ & \forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.\mathsf{C} \sqcap \forall v^-.\forall r.\mathsf{C} \end{aligned} \tag{29}$$

$$\begin{aligned} Q \sqcap \mathsf{C}' \sqsubseteq\ & \forall v.\forall r^-.\forall v^-.\forall r^-.\forall h.\forall r^-.\forall h^-.\forall r^-.\mathsf{C}' \sqcap \forall h.\forall r^-.\forall h^-.\forall r^-.\mathsf{C}' \sqcap \\ & \forall v^-.\forall r^-.\forall h.\forall r^-.\forall h^-.\forall r^-.\mathsf{C}' \sqcap \forall h^-.\forall r^-.\mathsf{C}' \end{aligned} \tag{30}$$

Figure 6 shows, for $n = 2$, how to fold the refined counting query such that v is mapped to a Q-node of a black tree and v' to a Q-node of a white successor tree that can be reached via crossing an h-edge in the torus, and likewise for the case where v' is mapped to a white tree, and v to a black successor tree reachable via h. We display only those side chains that are needed for accommodating the query match. To get started, note that in the left part of Figure 6, the h-edge in the right half of a meta role as shown in Figure 3 is matched onto the crossing h-edge in the model. The square and diamond nodes indicate where the middle and end parts of each meta role in the query match. Crossings of v-edges are similar.

We now define counting query parts in a more precise way. Note that each counting query consists of $4n+4$ meta roles. In the subsequent definition, $q_m^{i,j}$ is a meta role used in the counting query for A_i, where j ranges over $0, \ldots, 4n+3$.

Definition 1. *For all i, j with $1 \leq i \leq n$ and $0 \leq j < 4n+4$, put*

$$\begin{aligned} q_m^{i,j} := \{\, & r(v_1^{i,j}, v_0^{i,j}), v(v_1^{i,j}, v_2^{i,j}), r(v_3^{i,j}, v_2^{i,j}), v(v_4^{i,j}, v_3^{i,j}), r(v_5^{i,j}, v_4^{i,j}), h(v_5^{i,j}, v_6^{i,j}), \\ & r(v_7^{i,j}, v_6^{i,j}), h(v_8^{i,j}, v_7^{i,j}), r(v_9^{i,j}, v_8^{i,j}), r(v_9^{i,j}, v_{10}^{i,j}), h(v_{10}^{i,j}, v_{11}^{i,j}), r(v_{11}^{i,j}, v_{12}^{i,j}), \\ & h(v_{12}^{i,j}, v_{13}^{i,j}), r(v_{13}^{i,j}, v_{14}^{i,j}), v(v_{14}^{i,j}, v_{15}^{i,j}), r(v_{15}^{i,j}, v_{16}^{i,j}), v(v_{17}^{i,j}, v_{16}^{i,j}), r(v_{17}^{i,j}, v_0^{i,j+1})\} \end{aligned}$$

with $v_0^{i,4n+4} = v_0^{i,0}$. For each i with $1 \leq i \leq n$, the counting query for A_i *is*

$$q_c^i := \{\, A_i(v_0^{i,0}), \neg A_i(v_0^{i,1}), A_i(v_0^{i,2n+2}), \neg A_i(v_0^{i,2n+3})\} \cup \bigcup_{0 \leq j < 4n+4} q_m^{i,j}$$

with $v_9^{i,0} = v$, $v_9^{i,2n+2} = v'$. The counting query q_c for the whole counter is

$$q_c := \{B(v), Q(v), W(v'), Q(v')\} \cup \bigcup_{1 \leq i \leq n} q_c^i$$

Note that each counting query q_c^i is a cycle as intended since $v_0^{i,4n+4} = v_0^{i,0}$.

As explained above, the overall counting query q_c links a Q-node x_1 of a tree to the Q-nodes x_2 of its successor trees that represent the same bit position for the horizontal and vertical counters. To establish the central property ($*$) in models of $\mathcal{K}_0$ that do *not* match the query q_0 to be constructed, it thus remains to modify q_0 such that it matches only if the L_n-predecessor x_1' of x_1 has the same truth assignment of X' and Y' as the L_n-predecessor x_2' of x_2 for X and Y. This is achieved by the second type of component queries in q_0, the copying components.

Before we define the copying components, we prove that any match π for the query q_c is indeed such that the match connects nodes in neighboring trees that have the same counter value for the exponential counter.

To show this, we first define an *n-torus model* $\mathcal{T}_n = (\Delta^{\mathcal{T}_n}, \cdot^{\mathcal{T}_n})$ for our axioms. This n-torus model is the "canonical torus model" for a torus of size $2^{2^n} \times 2^{2^n}$ that reflects exactly the conditions we have so far defined and illustrated in the figures. The domain elements have the form (c, x, y, t), where $c = b$ for elements in black trees and $c = h$ ($c = v$) for elements in a horizontal (vertical) white trees, x and y indicate the horizontal and vertical counter value (starting from $0, 0$ for black trees and $1, 0$ $(0, 1)$ for horizontal (vertical) white trees), t is an empty string for elements that belong to the torus; for tree elements, t is non-empty. The real tree elements (as opposed to elements of a side chain) are strings from $\{0, 1, 2, 2^-\}^*$. Furthermore, if the size $|t|$ of t is m and t ends in 2, then the element is on level $m/2$, when t ends in 0 or 1, the element is between levels, and an element ending in 02 (12) is in the extension of $\neg A_{m/2}$ ($A_{m/2}$). The extra level is formed by appending 2^- and $2^- \cdot 2$ to elements of level n, where $\cdot$ denotes string concatenation. For brevity, we will simple write 2^-2 or $t_1 t_2$ to denote the concatenation of t_1 and t_2. Finally, for the side chains, we form a string that represents the roles in the chain, e.g., for the outgoing h-chain for a tree element that has only outgoing edges (c.f. Figure 5), t ends in a prefix of $vr^-v^-r^-$. Thus, for each torus element (c, x, y, ε), the set $\{t \mid (c, x, y, t) \in \Delta^{\mathcal{T}_n}\}$ is a tree, where we understand a tree to be a non-empty, prefix-closed subset of $\{0, 1, 2, 2^-, r, r^-, h, h^-, v, v^-\}^*$.

Definition 2. *Let $T_1 = \{0, 1, 2, 2^-\}$, $T_2 = \{r, r^-, h, h^-, v, v^-\}$, $T = T_1 \cup T_2$. The n-*torus interpretation *for $n \in \mathbb{N}$ and Axioms* (1) *to* (30) *is an interpretation $\mathcal{T}_n = (\Delta^{\mathcal{T}_n}, \cdot^{\mathcal{T}_n})$ with*

$\Delta^{\mathcal{T}_n} \subseteq \{b, h, v\} \times \{0, \ldots, 2^{2^n} - 1\} \times \{0, \ldots, 2^{2^n} - 1\} \times T^*$ *such that*

$$(b, x, y, \varepsilon) \in \Delta^{\mathcal{T}_n} \textit{ for each } 0 \leq x < 2^{2^n}, 0 \leq y < 2^{2^n} \tag{31}$$

$$(h, x, y, \varepsilon), (h, x, y, 2) \in \Delta^{\mathcal{T}_n} \textit{ for each } 0 < x < 2^{2^n}, 0 \leq y < 2^{2^n} \tag{32}$$

$$(v, x, y, \varepsilon), (v, x, y, 2) \in \Delta^{\mathcal{T}_n} \textit{ for each } 0 \leq x < 2^{2^n}, 0 < y < 2^{2^n} \tag{33}$$

$$(b, x, y, 0), (b, x, y, 1) \in \Delta^{\mathcal{T}_n} \textit{ for each } 0 \leq x < 2^{2^n}, 0 \leq y < 2^{2^n} \tag{34}$$

$$\begin{aligned}(c, x, y, t2) \in \Delta^{\mathcal{T}_n} &\textit{ if } (c, x, y, t) \in \Delta^{\mathcal{T}_n}, t = t'0 \textit{ or } t = t'1 \textit{ and either}\\ &(i)\ c = b \textit{ and } |t| < 2n \textit{ or } (ii)\ c \in \{h, v\} \textit{ and } |t| \leq 2n\end{aligned} \tag{35}$$

$$\begin{aligned}(c, x, y, t0), (c, x, y, t1) \in \Delta^{\mathcal{T}_n} &\textit{ if } (c, x, y, t) \in \Delta^{\mathcal{T}_n}, t = t'2 \textit{ and either}\\ &(i)\ c = b \textit{ and } |t| < 2n \textit{ or } (ii)\ c \in \{h, v\} \textit{ and } |t| \leq 2n\end{aligned} \tag{36}$$

$$\begin{aligned}(c, x, y, tt') \in \Delta^{\mathcal{T}_n} &\textit{ if } (c, x, y, t) \in \Delta^{\mathcal{T}_n}, t' \in \{2^-, 2^-2\} \textit{ and either } (i)\ c = b\\ &\textit{and } |t| = 2n \textit{ or } (ii)\ c \in \{h, v\} \textit{ and } |t| = 2n + 1\end{aligned} \tag{37}$$

$$\begin{aligned}(c, x, y, tt') \in \Delta^{\mathcal{T}_n} &\textit{ if } (c, x, y, t) \in \Delta^{\mathcal{T}_n}, t = t''2 \textit{ and}\\ &t' \in hrh^-rvrv^-r^* \cup h^-rvrv^-r^* \cup vrv^-r^* \cup v^-r^*\end{aligned} \tag{38}$$

$$\begin{aligned}(c, x, y, tt') \in \Delta^{\mathcal{T}_n} &\textit{ if } (c, x, y, t) \in \Delta^{\mathcal{T}_n}, \textit{ either } t = t''0 \textit{ or } t = t''1 \textit{ and}\\ &t' \in vrv^-r^-hr^-h^-r^{-*} \cup v^-r^-hr^-h^-r^{-*} \cup\\ &hr^-h^-r^{-*} \cup h^-r^{-*}\end{aligned} \tag{39}$$

We set $o^{\mathcal{T}_n} = (b, 0, 0, \varepsilon)$ *and* $e^{\mathcal{T}_n} = (b, 2^{2^n} - 1, 2^{2^n} - 1, \varepsilon)$. *For a decimal number d, we write* $\mathsf{bit}_i(d)$ *to return the* i^{th} *bit of the binary representation of d. We write* $t|_j$ *to denote the* j^{th} *character of a string t,* $c \in t$ $(c \notin t)$ *if the character c occurs (does not occur) in t, and define the interpretation of atomic concepts as*

$$B^{\mathcal{T}_n} = \{(c, x, y, t) \mid c = b\} \tag{40}$$

$$W^{\mathcal{T}_n} = \{(c, x, y, t) \mid c \in \{h, v\}\} \tag{41}$$

$$H^{\mathcal{T}_n} = \{(c, x, y, t) \mid c = h\} \tag{42}$$

$$V^{\mathcal{T}_n} = \{(c, x, y, t) \mid c = v\} \tag{43}$$

$$L_i^{\mathcal{T}_n} = \{(c, x, y, t) \mid c = b \textit{ and } i = |t|/2\} \cup \tag{44}$$

$$\{(c, x, y, t) \mid c \in \{h, v\} \textit{ and } i = (|t| - 1)/2\} \qquad {\scriptstyle 0 \leq i \leq n+1} \tag{45}$$

$$Q_0^{\mathcal{T}_n} = \{(c, x, y, t) \mid c = b \textit{ and } x + y \textit{ is even}\} \tag{46}$$

$$Q_1^{\mathcal{T}_n} = \{(c, x, y, t) \mid c \in \{h, v\} \textit{ and } x + y \textit{ is odd}\} \tag{47}$$

$$Q_2^{\mathcal{T}_n} = \{(c, x, y, t) \mid c = b \textit{ and } x + y \textit{ is odd}\} \tag{48}$$

$$Q_3^{\mathcal{T}_n} = \{(c, x, y, t) \mid c \in \{h, v\} \textit{ and } x + y \textit{ is even}\} \tag{49}$$

$$\begin{aligned}
A_i^{\mathcal{T}_n} =&\{(c,x,y,=t_1t_2) \mid t_1 \in T_1^*, t_2 \in T_2^*,\ \mathit{and\ either}\\
&c = b, 2^- \notin t_1, j = 2(i-1), |t_1| \geq j, t_1|_j = 1\ \mathit{or}\\
&c = b, 2^- \in t, j = 2(i-1), |t| \geq j, t|_j = 0\ \mathit{or}\\
&c \in \{h,v\}, 2^- \notin t, j = 2i, |t| \geq j, t|_j = 1\ \mathit{or}\\
&c \in \{h,v\}, 2^- \in t, j = 2i, |t| \geq j, t|_j = 0\} \qquad {\scriptstyle 1\leq i\leq n}
\end{aligned} \tag{50}$$

$$\begin{aligned}
X^{\mathcal{T}_n} =&\{(c,x,y,t_1t_2) \mid t_1 \in T_1^*, t_2 \in T_2^*\ \mathit{and}\\
&c = b, |t_1| = 2n, i = c^{\mathcal{T}_n}((c,x,y,t)), \mathsf{bit_i}(x) = 1\ \mathit{or}\\
&c = b, |t_1| = 2(n+1), t_1 = t2^-2, t \notin X^{\mathcal{T}_n}\ \mathit{or}\\
&c = h, |t_1| = 2n+1, i = c^{\mathcal{T}_n}((c,x-1,y,t)), \mathsf{bit_i}(x) = 1\ \mathit{or}\\
&c = h, |t_1| = 2(n+1)+1, t_1 = t2^-2, t \notin X^{\mathcal{T}_n}\ \mathit{or}\\
&c = v, |t_1| = 2n+1, i = c^{\mathcal{T}_n}((c,x,y,t)), \mathsf{bit_i}(x) = 1\ \mathit{or}\\
&c = v, |t_1| = 2(n+1)+1, t_1 = t2^-2, t \notin X^{\mathcal{T}_n}\}
\end{aligned} \tag{51}$$

$$\begin{aligned}
Y^{\mathcal{T}_n} =&\{(c,x,y,t_1t_2) \mid t_1 \in T_1^*, t_2 \in T_2^*\ \mathit{and}\\
&c = b, |t_1| = 2n, i = c^{\mathcal{T}_n}((c,x,y,t)), \mathsf{bit_i}(y) = 1\ \mathit{or}\\
&c = b, |t_1| = 2(n+1), t_1 = t2^-2, t \notin Y^{\mathcal{T}_n}\ \mathit{or}\\
&c = v, |t_1| = 2n+1, i = c^{\mathcal{T}_n}((c,x,y-1,t)), \mathsf{bit_i}(x) = 1\ \mathit{or}\\
&c = v, |t_1| = 2(n+1)+1, t_1 = t2^-2, t \notin Y^{\mathcal{T}_n}\ \mathit{or}\\
&c = h, |t_1| = 2n+1, i = c^{\mathcal{T}_n}((c,x,y,t)), \mathsf{bit_i}(y) = 1\ \mathit{or}\\
&c = h, |t_1| = 2(n+1)+1, t_1 = t2^-2, t \notin Y^{\mathcal{T}_n}\}
\end{aligned} \tag{52}$$

$$\begin{aligned}
X'^{\mathcal{T}_n} =&\{(c,x,y,t_1t_2) \mid t_1 \in T_1^*, t_2 \in T_2^*\ \mathit{and}\\
&c \in \{b,v\}, (c,x,y,t_1) \in X^{\mathcal{T}_n}\ \mathit{or}\\
&c = h, |t_1| = 2n+1, i = c^{\mathcal{T}_n}((c,x,y,t)), \mathsf{bit_i}(x) = 1\ \mathit{or}\\
&c = h, |t_1| = 2(n+1)+1, t_1 = t2^-2, t \notin X'^{\mathcal{T}_n}\}
\end{aligned} \tag{53}$$

$$\begin{aligned}
Y'^{\mathcal{T}_n} =&\{(c,x,y,t_1t_2) \mid t_1 \in T_1^*, t_2 \in T_2^*\ \mathit{and}\\
&c \in \{b,h\}, (c,x,y,t_1) \in Y^{\mathcal{T}_n}\ \mathit{or}\\
&c = h, |t_1| = 2n+1, i = c^{\mathcal{T}_n}((c,x,y,t)), \mathsf{bit_i}(y) = 1\ \mathit{or}\\
&c = h, |t_1| = 2(n+1)+1, t_1 = t2^-2, t \notin Y'^{\mathcal{T}_n}\}
\end{aligned} \tag{54}$$

$$\begin{aligned}
M_X^{\mathcal{T}_n} =&\{(c,x,y,t) \mid t \in T_1^*\ \mathit{and}\\
&c = b, |t| = 2n, (c,x,y,t) \in X^{\mathcal{T}_n}\ \mathit{or}\\
&c = b, |t| = 2(n-i), i > 0, (c,x,y,t02), (c,x,y,t12) \in M_X^{\mathcal{T}_n}\ \mathit{or}\\
&c \in \{h,v\}, |t| = 2n+1, (c,x,y,t) \in M_X^{\mathcal{T}_n}\ \mathit{or}\\
&c \in \{h,v\}, |t| = 2(n-i)+1, i > 0, (c,x,y,t02), (c,x,y,t12) \in M_X^{\mathcal{T}_n}\}
\end{aligned} \tag{55}$$

$$
\begin{aligned}
M_Y^{\mathcal{T}_n} = & \{(c,x,y,t) \mid t \in T_1^* \textit{ and} \\
& \quad c = b, |t| = 2n, (c,x,y,t) \in Y^{\mathcal{T}_n} \textit{ or} \\
& \quad c = b, |t| = 2(n-i), i > 0, (c,x,y,t02), (c,x,y,t12) \in M_Y^{\mathcal{T}_n} \textit{ or} \\
& \quad c \in \{h,v\}, |t| = 2n+1, (c,x,y,t) \in M_Y^{\mathcal{T}_n} \textit{ or} \\
& \quad c \in \{h,v\}, |t| = 2(n-i)+1, i > 0, (c,x,y,t02), (c,x,y,t12) \in M_Y^{\mathcal{T}_n}\} && (56) \\
S^{\mathcal{T}_n} = & \{(c,x,y,t) \mid t \in T_1^* \textit{ and} \\
& \quad c = b, |t| = 2n, t = (02)* \textit{ or} \\
& \quad c = h, |t| = 2n+1, (c,x,y,t) \notin X^{\mathcal{T}_n}), (c,x,y,t) \in X'^{\mathcal{T}_n}) \textit{ or} \\
& \quad c = v, |t| = 2n+1, (c,x,y,t) \notin Y^{\mathcal{T}_n}), (c,x,y,t) \in Y'^{\mathcal{T}_n}) \textit{ or} \\
& \quad \{(c,x,y,t02), (c,x,y,t12)\} \cap S^{\mathcal{T}_n} \neq \emptyset\} && (57) \\
F^{\mathcal{T}_n} = & \{(c,x,y,t) \mid t \in T_1^* \textit{ and} \\
& \quad c = b, |t| = 2n, t = (02)* \textit{ or} \\
& \quad c = h, |t| = 2n+1, (c,x,y,t) \in (X \sqcap \neg X')^{\mathcal{T}_n}) \textit{ or } (c,x,y,t) \in (\neg X \sqcap X')^{\mathcal{T}_n}) \textit{ or} \\
& \quad c = v, |t| = 2n+1, (c,x,y,t) \in (Y \sqcap \neg Y')^{\mathcal{T}_n}) \textit{ or } (c,x,y,t) \in (\neg Y \sqcap Y')^{\mathcal{T}_n}) \textit{ or} \\
& \quad \{(c,x,y,t02), (c,x,y,t12)\} \cap F^{\mathcal{T}_n} \neq \emptyset\} && (58) \\
Q^{\mathcal{T}_n} = & \{(c,x,y,t_1t_2) \mid t_1 \in T_1^*, t_2 \in T_2^*, t_1 = t2^-\} && (59) \\
N_B^{\mathcal{T}_n} = & \{(c,x,y,t) \mid t \in T_1^*, t = t'2\} && (60) \\
N_I^{\mathcal{T}_n} = & \{(c,x,y,t) \mid t \in T_1^*, t = t'0 \textit{ or } t = t'1\} && (61)
\end{aligned}
$$

We further define the interpretation of roles as

$$
\begin{aligned}
h^{\mathcal{T}_n} = \{\langle(c,x,y,t),(c',x',y',t')\rangle \mid\ & c = b, c' = h, t = t' = \varepsilon, x' = x+1 \textit{ or} \\
& c = h, c' = b, t = t' = \varepsilon, x = x' \textit{ or} \\
& c = c', x = x', y = y', t' = th \textit{ or} \\
& c = c', x = x', y = y', t = t'h^-\} \\
v^{\mathcal{T}_n} = \{\langle(c,x,y,t),(c',x',y',t')\rangle \mid\ & c = b, c' = v, t = t' = \varepsilon, y' = y+1 \textit{ or} \\
& c = v, c' = b, t = t' = \varepsilon, y = y' \textit{ or} \\
& c = c', x = x', y = y', t' = tv \textit{ or} \\
& c = c', x = x', y = y', t = t'v^-\} \\
r^{\mathcal{T}_n} = \{\langle(c,x,y,t),(c,x,y,t')\rangle \mid\ & t' = t \cdot 2 \textit{ or } t = t'2^- \textit{ or } t' = tr \textit{ or } t = t'r^-\}
\end{aligned}
$$

Lemma 1. *Let $n \in \mathbf{N}$ be some fixed number, $\mathcal{K}$ a knowledge base consisting of Axioms* (1) *to* (30) *and $\mathcal{T}_n$ an n-torus interpretation, then $\mathcal{T}_n \models \mathcal{K}$.*

The above lemma can be shown by exactly defining the canonical n-torus model $\mathcal{T}_n$ and by checking that all axioms are satisfied [9].

Proof. Shouldn't be too hard to show by going through the axioms...

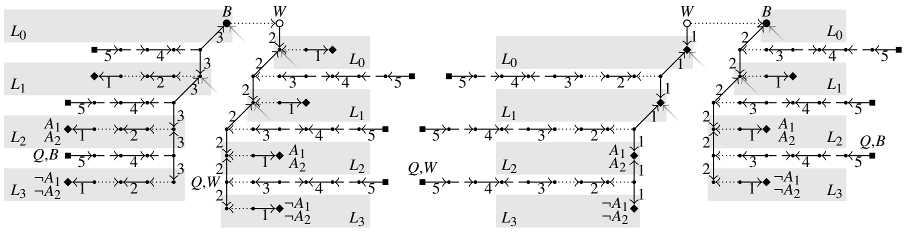


Fig. 7. The matching for the query from a black to a white tree via the role v (left-hand side) and the matching from a white to a black tree via the role v (right-hand side).

Lemma 2. *Let $n \in \mathbb{N}$ be some fixed number, $\mathcal{K}$ a knowledge base consisting of Axioms* (1) *to* (30)*, q_c as in Definition 1, and $\mathcal{T}_n$ an n-torus interpretation, then $\mathcal{T}_n \models q_c$.*

We can prove the lemma by analyzing how the query can be folded. As Figure 3 and 6 illustrate, there are only two ways of folding a query part for $(\neg)A_i$ depending on where the i^{th} bit if zero or one. In order to cross from a black to a horizontal white tree, the center point of the folded query has to fall into the longer white tree. Since the h-edge between the trees has an incoming r-edge from the black tree and an outgoing edge into the white tree, only the h-edge that follows the middle of the meta role to the right of the center point in Figure 3. This completely determines how the query has to be folded to reach the level $n+1$ nodes and to match Q between level n and $n+1$. As a consequence, each query atch is such that is connects two elements in neighboring trees with the same counter value.

Proof. We show that, depending on the trees in which v and v' match (black, white and horizontal, or white and vertical), we can fold the query only in two ways to have a long enough query. Shorter foldings will not be long enough to reach to level $n+1$. As a consequence, the query matches elements that have the same value for the exponential counter.

Let π be a match for q_c, i.e., $\mathcal{T}_n \models^{\pi} q_c$. Let $(c, x, y, t) = \pi(v)$, $(c', x', y', t') = \pi(v')$. Due to the atoms $B(v)$ and $W(v')$ in q_c, $\pi(v) \in B^{\mathcal{T}_n}$ and $\pi(v') \in W^{\mathcal{T}_n}$ and, more precisely, either $\pi(v') \in H^{\mathcal{T}_n}$ or $\pi(v') \in V^{\mathcal{T}_n}$. We consider the case $c = b$, $c' = h$ (by definition of $\mathcal{T}_n$, $(b, x, y, t) \in B^{\mathcal{T}_n}$ and $(h, x', y', t') \in H^{\mathcal{T}_n}$); we can proceed analogously for the case $c' = v$. Since q_c contains no chains of two h atoms (and the torus elements have no folding side chains that we could use to fold some query part away to come back to another h-edge), we can assume that $\pi(v)$ and $\pi(v')$ are in neighboring trees of the torus. Let $t = t_1t_2$ such that $t_1 \in T_1^*$ and $t_2 \in T_2^*$, i.e., if $\pi(v)$ matches on a side chain, (c, x, y, t_1) is the tree element between level n and $n+1$ since only for that tree element and the side chains starting from that element Q holds and q_c contains $Q(v)$. Furthermore, it is easily verified that (c, x, y, t_n) and (c, x, y, t_{n+1}) with $t_1 = t_n2^-$, $t_{n+1} = t_12$ are matched in each query part q_c^i with $1 \leq i \leq n$, i.e., the counter value $d = c^{\mathcal{T}_n}(c, x, y, t_n)$ can be uniquely determined from the query match. Similarly, we can define $d' = c^{\mathcal{T}_n}(c', x', y', t'_n)$ and we claim that $d = d'$.

We consider the case $x' = x + 1$, i.e., $\pi(v')$ matches in a successor tree of the tree in which $\pi(v)$ matches (see left-hand side of Figure 6). Let i be some fixed number such that $1 \leq i \leq n$. We show that there is a unique way for the matching q_c^i, determined by whether $(c, x, y, t_1) \in A_i^{\mathcal{T}_n}$ or $(c, x, y, t_1) \in (\neg A_i)^{\mathcal{T}_n}$. We distinguish the two cases:

1. Let $(c, x, y, t_1) \in A_i^{\mathcal{T}_n}$. We claim that meta role with $j = 0$ has to be folded onto meta role with $j = 1$, i.e., $v_0^{i,0} = v_0^{i,2}$ (the first variable of meta role with $j = 0$ matches the last variable of meta role with $j = 1$, which is the same as the first variable of meta role with $j = 2$) and $\pi(v) = \pi(v_9^{i,1})$ (not that $v = v_9^{i,0}$). Furthermore, the meta role with $j = 2$ has to be folded onto the last one with $j = 4n + 3$, i.e., $v_0^{i,3} = v_0^{i,4n+3}$. Accordingly, $v_0^{i,n+2} = v_0^{i,3n+4}$ becomes the centre point of the query, which has to match in the longer white tree. Lastly, $v_0^{i,2n+2} = v_0^{i,2n+4}$, i.e., $A_i(v_0^{i,2n+4})$ holds since $A_i(v_0^{i,2n+2}) \in q_c^i$ and $v' = v_9^{i,2n+2} = v_9^{i,2n+4}$. Since the center point of the query has to match in the longer white tree, we find that the two torus points (c, x, y, ε) and $(c', x', y', \varepsilon)$ fall into the meta role with $j = n + 1$ and the meta role with $j = 3n + 4$. Since (c, x, y, ε) must have an incoming r-edge, while $(c', x', y', \varepsilon)$ must have an outgoing r-edge and $\langle (c, x, y, \varepsilon), (c', x', y', \varepsilon) \rangle \in h^{\mathcal{T}_n}$ by assumption, the only suitable role atom in the meta role is $r(v_{10}^{i,n+1}, v_{11}^{i,n+1})$ (folded onto $r(v_7^{i,3n+4}, v_8^{i,3n+4})$). Note that the symmetry of the meta role makes the folding possible since the second half of the meta role with $j = n + 1$ can be folded onto the first half of the meta role with $j = 3n + 4$. If we make this decision, there is no choice for matching the adjacent variables, since the r-edges only go down into the trees on both sides and for the black side, the first bit of d, determines whether A_1 or $\neg A_1$ holds and, thus, which of the two r-edges we have to take. On the back side, we now have another r-edge to match, which means we reach a level 1 node. We now have to match an h-edge, which means we have to go into a side tree and again there is no choice about which side tree to take. Figure 5 illustrates that we have no choice but to match the last node of the meta role onto the last element of the side chain starting with an h-edge. Thus, $v_0^{i,n+1}$ matches $(c, x, y, 12hrh^{-}rvrv^{-}r)$ in case A_1 holds at (c, x, y, t_n) and $(c, x, y, 02hrh^{-}rvrv^{-}r)$ otherwise. There is again no choice but to match the side chain in the opposite direction. Once we are back at the tree node, we can go up or down an r-edge. If we go up, the query is not long enough to reach a node labeled with Q between level n and $n + 1$. Therefore, we go down into the tree with the direction again determined by the counter value d. After going an r^{-}-edge down, followed by an r-edge, we again have no choice but to match the query onto the same side chain, just one level down the tree, matching $v_0^{i,n}$ onto the end of the side chain on level 2. We proceed this way to match $v_0^{i,2} = v_0^{i,0}$ onto the side chain on level n. Finally, in order to reach a node with $\neg A_i$, we have to go down to level $n + 1$, via the Q node and match $v_0^{i,0}$ to the end of the h-side chain on level $n + 1$.Note that this immediately determines the matches for meta roles with $j = 3n + 5$ to $4n + 3$.
For the white tree, once we have matched the first r-edge, there is no choice but to match an h^{-}-edge, which means we have to go into the side chain and again there is no choice about which side chain to take. We match the center node $v_0^{i,n+2} = v_0^{i,3n+4}$ of the query, onto the last point of the h^{-} side chain. We can argue as for the black tree that there is no choice but to go down in the tree, matching the middle points of each meta role to a h^{-} side chain for a tree node with only outgoing r-edges and the end points to h^{-} side chains of tree nodes with only incoming r-edges.

Thus, there was indeed no choice as to how the query could be matched and folded. This also means that A_i holds at (c', x', y', t_n) and $\neg A_i$ holds at (c', x', y', t_{n+1}). By arbitrariness of the chosen i, we find that indeed $d = d'$, i.e., the query matches such that it connects two neighboring tree elements with the same value for the exponential counter.

2. The case for $(c, x, y, t_1) \in (\neg A_i)^{\mathcal{T}_n}$ is analogous, just folding the query as shown in the upper part of figure 3 since only then we have $\neg A_i$ on level n in both trees and A_i on level $n + 1$.

The proof for $(c', x', y', \varepsilon)$ matching to the predecesor of (c, x, y, ε) is analogues and illustrated on the right-hand side of Figure 6.

For the case of $c' = v$, we can also proceed analogously, using different side chains as for the previous two cases. Figure 7 illustrates which side chains have to be used depending on whether the match is from a black to a white or from a white to a black tree.

To prepare for these components, we introduce additional concept names $Q_0, \ldots, Q_3$ that realize a unary counter of trees, counting modulo 4. These concept names will be used to identify the successor relation between the trees in query matches. The counter value is incremented when moving from a tree to a successor tree and propagated to all L_n- and L_{n+1}-nodes inside each tree and the ends of their side chains. This is achieved by adding to $\mathcal{K}_0$ the following concept inclusions:

$$\{o\} \sqsubseteq Q_0 \tag{62}$$

$$\begin{aligned} Q_i \sqcap B \sqcap L_0 \sqsubseteq {} & \forall h.\forall r.Q_{i+1 \bmod 4} \sqcap \forall h.Q_{i+2 \bmod 4} \sqcap \\ & \forall v.\forall r.Q_{i+1 \bmod 4} \sqcap \forall v.Q_{i+2 \bmod 4} \qquad {\scriptstyle i=0,2} \end{aligned} \tag{63}$$

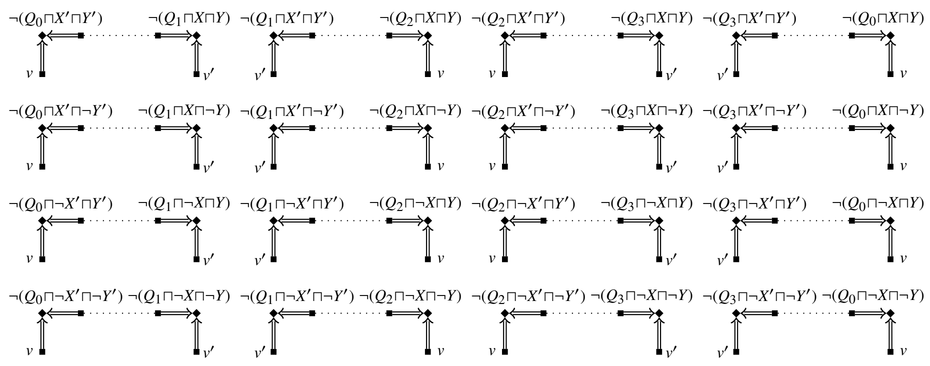
$$Q_i \sqcap L_0 \sqsubseteq \forall (r^-; r)^n.(L_n \rightarrow Q_i) \qquad {\scriptstyle 0 \le i \le 3} \tag{64}$$

$$Q_i \sqsubseteq \neg Q_j \qquad {\scriptstyle 0 \le i < j \le 3} \tag{65}$$

$$\begin{aligned} Q_i \sqcap (L_n \sqcup L_{n+1}) \sqsubseteq {} & \forall h.\forall r.\forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.Q_i \sqcap \forall v.\forall r.\forall v^-.\forall r.Q_i \sqcap \\ & \forall h^-.\forall r.\forall v.\forall r.\forall v^-.\forall r.Q_i \sqcap \forall v^-.\forall r.Q_i \qquad {\scriptstyle 0 \le i \le 3} \end{aligned} \tag{66}$$

The copying components take the form displayed in Figure 8, i.e., there are 16 such components in total. Each component is like the upper half of a counting component, except that the concept labels have changed to negated conjunctions. In Figure 8, the four copying components in each row take care of one possible truth assignment to X' and Y' and the corresponding assignment to X and Y. We need four queries per truth assignment to deal with the four possible ways in which a counting query can match, as induced by the concept names $Q_0, \ldots, Q_3$:

(a) v matches into a tree that satisfies B and Q_0, and v' into a successor tree that satisfies W and Q_1;
(b) v' matches into a tree that satisfies W and Q_1, and v into a successor tree that satisfies B and Q_2;
(c) v matches into a tree that satisfies B and Q_2, and v' into a successor tree that satisfies W and Q_3;
(d) v' matches into a tree that satisfies W and Q_3, and v into a successor tree that satisfies B and Q_0.



¬(Q0⊓X′⊓Y′) ¬(Q1⊓X⊓Y) ¬(Q1⊓X′⊓Y′) ¬(Q2⊓X⊓Y) ¬(Q2⊓X′⊓Y′) ¬(Q3⊓X⊓Y) ¬(Q3⊓X′⊓Y′) ¬(Q0⊓X⊓Y)
¬(Q0⊓X′⊓¬Y′) ¬(Q1⊓X⊓¬Y) ¬(Q1⊓X′⊓¬Y′) ¬(Q2⊓X⊓¬Y) ¬(Q2⊓X′⊓¬Y′) ¬(Q3⊓X⊓¬Y) ¬(Q3⊓X′⊓¬Y′) ¬(Q0⊓X⊓¬Y)
¬(Q0⊓¬X′⊓Y′) ¬(Q1⊓¬X⊓Y) ¬(Q1⊓¬X′⊓Y′) ¬(Q2⊓¬X⊓Y) ¬(Q2⊓¬X′⊓Y′) ¬(Q3⊓¬X⊓Y) ¬(Q3⊓¬X′⊓Y′) ¬(Q0⊓¬X⊓Y)
¬(Q0⊓¬X′⊓¬Y′) ¬(Q1⊓¬X⊓¬Y) ¬(Q1⊓¬X′⊓¬Y′) ¬(Q2⊓¬X⊓¬Y) ¬(Q2⊓¬X′⊓¬Y′) ¬(Q3⊓¬X⊓¬Y) ¬(Q3⊓¬X′⊓¬Y′) ¬(Q0⊓¬X⊓¬Y)
v
v′

Fig. 8. The 16 query copying components

To explain in detail how the copying queries work, consider case (a). For simplicity, in the explanation we largely ignore side chains and pretend that meta roles are compositions $r; r^-$, as in our initial, simplified view on counting components. Take the Q-node x_1 of a tree that satisfies B and Q_0 and the Q-node x_2 of a successor tree that satisfies W and Q_1. The relevant queries are those from the leftmost column in Figure 8. Let x'_i be the predecessor L_n-node of x_i, and x''_i the successor L_{n+1}-node of x_i. Due to the Q-label in the *counting* queries, the variable v can only be matched to x_1 and the variable v' can only be matched to x_2. Let u be the neighboring variable of v in the copying components, and u' the neighboring variable of v'. Then u can only be matched to either x'_1 or x''_1 while u' can only be matched to either x'_2 or x''_2. The match $u \mapsto x''_1$ and $u' \mapsto x''_2$ is excluded because the path from u to u' is not long enough. So a component has a match iff either x'_1 satisfies the label of u or x'_2 satisfies the label of u' (since x'_1 and x''_2 have complementary truth values for X' and Y', one of them always satisfy the label of u and similarly for x''_1, x''_2 and u'). The four copying components in the first column exclude exactly four situations when x'_1 has the same truth assignment of X' and Y' as x'_2 for X and Y. It remains to note that the copying components in the other columns always have a match in this situation due to our use of the concept names $Q_0, \ldots, Q_3$ in the labels, and thus do not interfere with the matches of the queries in the leftmost components.

We remark that, without the use of the concept names $Q_0, \ldots, Q_3$, it does not seem possible to ensure proper directionality of copying. For example, copying components that copy the X/Y-assignment from a black tree to the X'/Y'-assignment in white successor trees would also copy this assignment to the the X'/Y'-assignment in white *predecessor* trees. A formal definition of copying queries can be found in [9].

This finishes the construction of the query q_0 and of the part of $\mathcal{K}_0$ that enforces the torus structure.

We define the copying components precisely as follows:

Definition 3. *Let $n \in \mathbb{N}$ and $A_1, \ldots, A_n$ the concepts to represent a counter with n bits and let $m := 2n + 2$. We reuse the definition of the cyclic query parts composed of meta* *roles, to define a query q_Q as follows:*

$$
\begin{aligned}
q_Q := \bigcup_{n<i\leq n+8} q_c^i \cup \\
&\{(\neg Q_1 \sqcup X')(v_0^{n+1,0}), (\neg Q_1 \sqcup \neg X')(v_0^{n+1,1}), (\neg Q_2 \sqcup \neg X)(v_0^{n+1,m}), (\neg Q_2 \sqcup X)(v_0^{n+1,m+1})\}\\
&\{(\neg Q_3 \sqcup X')(v_0^{n+2,0}), (\neg Q_3 \sqcup \neg X')(v_0^{n+2,1}), (\neg Q_4 \sqcup \neg X)(v_0^{n+2,m}), (\neg Q_4 \sqcup X)(v_0^{n+2,m+1})\}\\
&\{(\neg Q_1 \sqcup X)(v_0^{n+3,0}), (\neg Q_1 \sqcup \neg X)(v_0^{n+3,1}), (\neg Q_4 \sqcup \neg X')(v_0^{n+3,m}), (\neg Q_4 \sqcup X')(v_0^{n+3,m+1})\}\\
&\{(\neg Q_3 \sqcup X)(v_0^{n+4,0}), (\neg Q_3 \sqcup \neg X)(v_0^{n+4,1}), (\neg Q_2 \sqcup \neg X')(v_0^{n+4,m}), (\neg Q_2 \sqcup X')(v_0^{n+4,m+1})\}\\
&\{(\neg Q_1 \sqcup Y')(v_0^{n+5,0}), (\neg Q_1 \sqcup \neg Y')(v_0^{n+5,1}), (\neg Q_2 \sqcup \neg Y)(v_0^{n+5,m}), (\neg Q_2 \sqcup Y)(v_0^{n+5,m+1})\}\\
&\{(\neg Q_3 \sqcup Y')(v_0^{n+6,0}), (\neg Q_3 \sqcup \neg Y')(v_0^{n+6,1}), (\neg Q_4 \sqcup \neg Y)(v_0^{n+6,m}), (\neg Q_4 \sqcup Y)(v_0^{n+6,m+1})\}\\
&\{(\neg Q_1 \sqcup Y)(v_0^{n+7,0}), (\neg Q_1 \sqcup \neg Y)(v_0^{n+7,1}), (\neg Q_4 \sqcup \neg Y')(v_0^{n+7,m}), (\neg Q_4 \sqcup Y')(v_0^{n+7,m+1})\}\\
&\{(\neg Q_3 \sqcup Y)(v_0^{n+8,0}), (\neg Q_3 \sqcup \neg Y)(v_0^{n+8,1}), (\neg Q_2 \sqcup \neg Y')(v_0^{n+8,m}), (\neg Q_2 \sqcup Y')(v_0^{n+8,m+1})\}
\end{aligned}
$$

with $v_9^{i,0} = v, v_9^{i,2n+2} = v'$ for $n < i \leq n + 8$. We define the query q_0 is $q_c \cup q_Q$.

We now show that q_0 does not match in our canonical model. We know that the canonical model is a model of our axioms by Lemma 1 and that the query matches on the canonical model such that each match connects two elements in neighboring trees with the same counter value by Lemma 2. Thus, the next lemma shows that only if a double exponential counter is not incremented properly, the query can match in the canonical model.

Lemma 3. *Let $n \in \mathbf{N}$ be some fixed number, $\mathcal{K}$ a knowledge base consisting of Axioms* (1) *to* (30)*, q_0 as in Definition 3, and $\mathcal{T}_n$ an n-torus interpretation, then $\mathcal{T}_n \not\models q_0$.*

Proof. By Lemma 2, we know $\mathcal{T}_n \models q_c$. Since $q_0 = q_c \cup q_Q$, we only have to show that $\mathcal{T}_n \not\models q_Q$, i.e., some double exponential counter is wrong. Assume to the contrary of what is to be shown that there is some π such that $\mathcal{T}_n \models^\pi q_0$. By Lemma 2, we know that each match π is such that the nodes on level n in the match have the same counter value. By the definition of $\mathcal{T}_n$ ((51), (52), (53), and (54)), we know that X, Y, X', and Y' are set correctly. Assume that $\pi(v)$ matches in a black tree labeled with Q_0 and $\pi(v')$ in a horizontal white tree labeled with Q_1 (cf. Figure 8). Thus, we have that either

1. $\neg X'$ holds for the level n element above $\pi(v)$, X' holds at the level $n + 1$ element below $\pi(v)$, X holds at the level n element above $\pi(v')$, and $\neg X$ at the level $n + 1$ element below $\pi(v')$ or
2. X' holds for the level n element above $\pi(v)$, $\neg X'$ holds at the level $n + 1$ element below $\pi(v)$, $\neg X$ holds at the level n element above $\pi(v')$, and X at the level $n + 1$ element below $\pi(v')$.

Assume the former. Since a tree labeled with Q_1 can only be a successor tree for a tree labeled with Q_0 when π matches and since the level n elements in the range of π have the same value for the exponential counter by Lemma 2, we have that the X' counter in the black tree does not coincide with the X counter in the successor white tree, which contradicts the definition of $\mathcal{T}_n$.

We can proceed analogously for the second folding and for other matches of q_Q.

It remains to encode tilings of the domino system D_0.

$$\top \sqsubseteq T_1 \sqcup \cdots \sqcup T_k \qquad T_i \sqcap T_j \sqsubseteq \bot \qquad 1 \leq i < j \leq k \tag{67}$$

$$T_i \sqcap \exists h.T_j \sqsubseteq \bot \qquad T_k \sqcap \exists v.T_\ell \sqsubseteq \bot \qquad \langle i, j\rangle \notin H, \langle k, \ell\rangle \notin V \tag{68}$$

Finally, we enforce the initial condition $c^0 = \langle t_1^0, \ldots, t_n^0\rangle$ of the torus.

$$\{o\} \sqsubseteq T_{t_1^0} \sqcap \forall h.(T_{t_2^0} \sqcap \forall h.(T_{t_3^0} \sqcap \forall h.(T_{t_4^0} \sqcap \ldots \forall h.T_{t_n^0} \ldots))) \tag{69}$$

More details regarding the correctness of the reduction can be found in [9]. The most challenging issue is to show that when D_0 admits a tiling with initial condition c^0 and we build a model $\mathcal{I}$ of $\mathcal{K}$ that has the intended torus shape, then $\mathcal{I} \not\models q_0$: we need to prove that there are no unintended foldings and matchings of the query q_0.

Lemma 4. *Let c^0 be an initial condition of size n for the domino system D_0, $\mathcal{K}$ a knowledge base consisting of Axioms* (1) *to* (69)*, q_0 a query as in Definition 3. If D_0 admits a tiling of $2^{2^n} \times 2^{2^n}$ for c^0, then $\mathcal{K} \not\models q_0$.*

Proof. We can use the given titling to extend the n-torus model $\mathcal{T}_n$ to $\mathcal{T}_n'$ such that Axioms(67) to (69) are also satisfied. Since the query does not use any of the newly extended symbols, Lemma 3 immediately gives us the desired result.

Lemma 5. *Let c^0 be an initial condition of size n for the domino system D_0, $\mathcal{K}$ a knowledge base consisting of Axioms* (1) *to* (69)*, and q_0 a query as in Definition 3. If $\mathcal{K} \not\models q_0$, then D_0 admits the tiling of $2^{2^n} \times 2^{2^n}$ for c^0.*

Proof. Since $\mathcal{K} \not\models q_0$, there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I} \not\models q_0$. We establish a homomorphism from our n-torus model $\mathcal{T}_n$ to $\mathcal{I}$, showing that $\mathcal{T}_n \not\models q_0$. Since $\mathcal{T}_n \not\models q_0$, all counters are set correctly. Thus, $\mathcal{T}_n$ has a grid of double exponential size and since $\mathcal{T}_n \models \mathcal{K}$, we can read of the tile types to construct the solution for D_0.

Combining Lemma 4 and 5 then gives the desired result.

Theorem 1. *Conjunctive query entailment by $\mathcal{ALCOIF}$ knowledge bases is* co-N2ExpTime-*hard.*

4 Conclusions

We have shown that conjunctive query answerin in $\mathcal{ALCOIF}$ is hard for co-N2ExpTime. The challenging problem of finding a matching upper bound, or in fact *any* elementary upper bound, remains open.

Acknowledgments B. Glimm is supported by EPSRC grant EP/F065841/1, Y. Kazakov by EPSRC grant EP/G02085X, and C. Lutz by DFG SFB/TR 8 “Spatial Cognition”.

References

1. Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries. J. of Artificial Intelligence Research **39** (2010) 429–481
2. Lutz, C.: Inverse roles make conjunctive queries hard. In: Proc. of the 2007 Description Logic Workshop (DL 2007). (2007)

3. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-08), LNCS (2008) 179–193
4. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. J. of Artificial Intelligence Research **31** (2008) 151–198
5. Glimm, B., Horrocks, I., Sattler, U.: Unions of conjunctive queries in $\mathcal{SHOQ}$. In: Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08), AAAI Press/The MIT Press (2008)
6. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI-09). (2009) 714–720
7. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc. of the 11th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-08), AAAI Press/The MIT Press (2008)
8. Glimm, B., Kazakov, Y.: Role conjunctions in expressive description logics. In: Proc. of the 15th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 2008). Volume 5330 of LNCS., Springer (2008) 391–405
9. Glimm, B., Kazakov, Y., Lutz, C.: Status $\mathcal{QIO}$: An update. Technical report, The University of Oxford (2011) `http://www.comlab.ox.ac.uk/files/3969/GlKL11a.pdf`.
10. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook. Cambridge University Press (2003)
11. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Perspectives in Mathematical Logic. Springer (1997) Second printing, Springer Verlag, 2001.
12. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. of Artificial Intelligence Research **12** (2000)