On the Complexity of Counting in Description Logics

Stephan Tobies
LuFg Theoretical Computer Science, RWTH Aachen, Germany
tobies@informatik.rwth-aachen.de

Abstract

Many Description Logics (DLs) allow for counting expressions of various forms that are important in many applications, e.g., for reasoning with semantic data models and for applications concerned with the configuration of technical systems. We present two novel complexity results for DLs that contain counting constructs:

(1) We prove that concept satisfiability for $\mathcal{ALCQI}$ is decidable in PSPACE even if binary coding of numbers in the input is assumed. (2) We prove that TBox consistency for $\mathcal{ALCQI}$ with cardinality restrictions is NEXPTIME-complete.

1 The Complexity of $\mathcal{ALCQI}$

Qualifying number restrictions [10] are a common generalisation of both role-quantification and standard number restrictions that are present in almost all DL systems. They provide an expressive means to describe objects by the number of other objects they are related to and are necessary for reasoning with semantic data models [5]. In [16] we have shown that—at least for $\mathcal{ALC}$—number restrictions can be replaced by qualifying number restrictions without increasing the (worst-case) complexity of the satisfiability problem. In this paper we extend this result to inverse roles.

Definition 1 (The DL $\mathcal{ALCQI}$) *Let N_C be a set of* atomic concepts *and N_R a set of* atomic roles. *The set of $\mathcal{ALCQI}$-*roles $\overline{N_R}$ *is $N_R \cup \{R^- \mid R \in N_R\}$.* Concepts *in $\mathcal{ALCQI}$ are built inductively using the following rules:*

1. *every $A \in N_C$ is an $\mathcal{ALCQI}$-concept, and*
2. *if C, D_1, D_2 are $\mathcal{ALCQI}$-concepts, $n \in \mathbb{N}$, and $R \in \overline{N_R}$ then $\neg C$, $D_1 \sqcap D_2$, $D_1 \sqcup D_2$, $(\geqslant n\ R\ C)$, and $(\leqslant n\ R\ C)$ are $\mathcal{ALCQI}$-concepts.*

We use $\bowtie$ as a placeholder for $\leqslant$ and $\geqslant$. For an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, we extend the usual semantics of $\mathcal{ALC}$-concepts to qualifying number restrictions by setting

$$(\bowtie n\ R\ C)^\mathcal{I} := \{x \in \Delta^\mathcal{I} \mid \sharp\{y \mid (x,y) \in R^\mathcal{I}, y \in C^\mathcal{I}\} \bowtie n\},$$

where $\sharp$ denotes the cardinality of a set. For roles names R, we define $(R^-)^\mathcal{I} := \{(y,x) \mid (x,y) \in R^\mathcal{I}\}$. With $\mathcal{ALCQ}$ we denote the fragment of $\mathcal{ALCQI}$ that does not contain inverse roles. With SAT*($\mathcal{ALCQ}$) and* SAT*($\mathcal{ALCQI}$) we denote the set of all satisfiable $\mathcal{ALCQ}$-, resp., $\mathcal{ALCQI}$-concepts.*

Obviously, existential and universal restrictions can be expressed in $\mathcal{ALCQI}$ with qualifying number restrictions using the equivalences $\exists R.C \equiv (\geqslant 1\ R\ C)$ and $\forall R.C \equiv (\leqslant 0\ R\ \neg C)$. In order to avoid considering roles such as R^{--}, we define a function Inv that returns the inverse of a role by setting $\mathsf{Inv}(R) := R^-$ if $R \in N_R$, and $\mathsf{Inv}(R) := S$ if $R = S^-$ for some $S \in N_R$.

Reasoning with Qualifying Number Restrictions In [10] a tableaux algorithm is presented that decides SAT($\mathcal{ALCQ}$) in polynomial space, provided that only unary coding of numbers in the input is allowed. In [6] it is conjectured that binary coding of numbers makes SAT($\mathcal{ALCQ}$) EXPTIME-hard. Why does the coding of numbers seem to be of such an importance for the problem? The answer lies in the nature of the tableaux algorithms for $\mathcal{ALCQ}$: Like other tableaux algorithms, they decide the satisfiability of a concept C by trying to construct a model for it. For an instance x of a concept $(\geqslant n\ R\ C)$, the algorithm in [10] generates n successors of x, and the correctness of the algorithms relies on the fact that they are kept in memory *simultaneously*. Assuming unary coding of numbers in the input, this is can be done in polynomial space because the number n will consume n bits in the input and hence the amount of memory needed for the n successors is polynomial in the size of the input. This changes if we assume binary coding of numbers: then n consumes only $\log_2 n$ bits in the input, making the amount of memory required to store n successors potentially exponential in the size of the input.

In [16] we give an algorithm derived from the one presented in [10] that is capable of deciding SAT($\mathcal{ALCQ}$) in PSPACE, even if binary coding of numbers in the input is allowed. While still generating n successors for a concept $(\geqslant n\ R\ C)$, non-deterministic guessing of an assignment of relevant constraints to newly generated individuals is used to be able to generate them successively while re-using space.

This determines the complexity of SAT($\mathcal{ALCQ}$) as PSPACE-complete. As a result we may augment $\mathcal{ALC}$ with qualifying number restrictions without increasing the (worst-case) complexity of the satisfiability problem.

In this paper we present an extension of the algorithm in [16] that can, additionally, deal with inverse roles and runs in polynomial space. The so-called "reset-restart" technique used to deal with concepts moving "backwards" in the completion tree due to the *choose*-rule has already been used in [11] to handle inverse roles.

Definition 2 *An $\mathcal{ALCQI}$-concept C is in* negation normal form *(NNF) if negation occurs only in front of atomic concepts; we denote the NNF of $\neg C$ by $\sim C$. For a concept C in NNF we define clos(C) to be the smallest set of $\mathcal{ALCQI}$-concepts that contains C and is closed under sub-concepts and $\sim$.*

A completion tree *for an $\mathcal{ALCQI}$-concept D is a tree where each node x is labelled with a set $\mathcal{L}(x) \subseteq clos(D)$ and each edge $\langle x,y\rangle$ is labelled with a (possibly inverse) role name $\mathcal{L}(\langle x,y\rangle) = R$ for a role occurring in clos(D).*

*Given a completion tree, a node y is called an R-*successor *of a node x iff y is a successor of x and $\mathcal{L}(\langle x,y\rangle) = R$. A node y is called an R-*neighbour *of x iff y is an R-successor of x, or if x is an $\mathsf{Inv}(R)$-successor of y. Predecessors, ancestors, paths, etc., are defined as usual.*

A node x in $\mathbf{T}$ is said to contain a clash *if,*

- *for some atomic concept A, $\{A, \neg A\} \subseteq \mathcal{L}(x)$, or*
- *for some concept C, role R, and $n \in \mathbb{N}$, it holds that $(\leqslant\ n\ R\ C) \in \mathcal{L}(x)$ and $\sharp R^{\mathbf{T}}(x,C) > n$, where $R^{\mathbf{T}}(x,C) := \{y \mid y$ is R-neighbour of x in $\mathbf{T}$ and $C \in \mathcal{L}(y)\}$.*

A completion tree is called clash-free *iff none of its nodes contains a clash; it is called* complete *iff none of the expansion rules in Figure 1 is applicable to any of its nodes.*

To test the satisfiability of an $\mathcal{ALCQI}$-concept D, the algorithm starts with a completion tree consisting of a single node x_0 with $\mathcal{L}(x_0) = \{D\}$. It applies the expansion rules, stopping when a clash occurs, and answers "D is satisfiable" iff the completion rules can be applied in such a way that they yield a complete and clash-free completion tree.

$\sqcap$-rule:	if 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and 2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-rule:	if 1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$ and 2. $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$
choose-rule:	if 1. $(\bowtie\ n\ R\ C) \in \mathcal{L}(x)$ and 2. there is an R-predecessor y of x with $\{C, \sim C\} \cap \mathcal{L}(x) = \emptyset$ then $\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \sim C\}$ and delete *all* descendants of y.
$\geqslant$-rule:	if 1. $(\geqslant\ n\ R\ C) \in \mathcal{L}(x)$, none from the above rules is applicable to x or any of its ancestors, and 2. $\sharp R^{\mathbf{T}}(x, C) < n$ then create a new node y with $\mathcal{L}(\langle x,y\rangle) = R$ and $\mathcal{L}(y) = \{C, E_1, \ldots, E_n\}$ where $\{D_1, \ldots D_n\} = \{D \mid (\bowtie\ n\ R\ D) \in \mathcal{L}(x)\}$ and $E_i \in \{D_i, \sim D_i\}$.

Figure 1: Tableaux expansion rules for $\mathcal{ALCQI}$

Correctness of the Algorithm In this paper we can only give the main ideas of the proofs, for details please refer to [17]. In order to prove the correctness of the algorithm it is necessary to show termination, soundness, and completeness. The following lemma states the termination of the algorithm an collects some facts that will be needed for the complexity analysis.

Lemma 3 *Let D be an $\mathcal{ALCQI}$-concept in NNF and $\mathbf{T}$ a completion tree that is generated for D by the tableaux algorithm.*

1. *$\sharp clos(D) = \mathcal{O}(|D|)$.*
2. *The length of a path in $\mathbf{T}$ is bounded by $|D|$.*
3. *The out-degree of $\mathbf{T}$ is limited by $\sharp clos(D) \times 2^{|D|}$.*
4. *For any concept D, the algorithm terminates.*

Proof. (1) is easily proved by observing that *clos*(D) = *sub*$(D) \cup \{\sim C \mid C \in$ *sub*$(D)\}$, where *sub*(D) is the set of all subconcepts of D which contains at most $|D|$ elements. (2) and (3) are simple consequence of the expansion rules. (4) is a consequence of the bounded size of the tree because each rule application either adds nodes to the tree or the adds concepts to the label of a node x which is always a subset of *clos*(D). The only problem is the deletion of nodes that is triggered by the *choose*-rule. However, if this rule deletes a node from the tree, also the label of an ancestor of this node grows, which prevents an infinite sequence of rule applications. □

Lemma 4 (Soundness) *If the rules can be applied to an $\mathcal{ALCQI}$-concept D such that they yield a complete and clash-free completion tree, then D is satisfiable.*

Proof. Let $\mathbf{T}$ be a complete and clash-free completion tree for D. A model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for D can be defined by choosing $\Delta^{\mathcal{I}}$ to be the nodes of $\mathbf{T}$ and by defining:

$$A^{\mathcal{I}} = \{x \mid A \in \mathcal{L}(x)\} \quad \text{for all atomic concepts } A$$
$$R^{\mathcal{I}} = \{\langle x,y\rangle \mid \mathcal{L}(\langle x,y\rangle) = R \text{ or } \mathcal{L}(\langle y,x\rangle) = \mathsf{Inv}(R)\}.$$

Inductively it can be shown that, for all $x \in \Delta^{\mathcal{I}}$ and all $C \in$ *clos*(D), $C \in \mathcal{L}(x)$ implies $x \in C^{\mathcal{I}}$. Since $D \in \mathcal{L}(x_0)$ it follows that $D^{\mathcal{I}} \neq \emptyset$ and, hence $\mathcal{I}$ is a model of D. □

Lemma 5 (Completeness) *Let D be an $\mathcal{ALCQI}$-concept: If D is satisfiable, then the expansion rules can be applied in such a way that they yield a complete and clash-free completion tree for D.*

Proof. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model for D. We will use $\mathcal{I}$ to guide the application of the non-deterministic completion rules. For this we incremently define a function π mapping the nodes in $\mathbf{T}$ to elements of $\Delta^{\mathcal{I}}$ such that at any given stage the following holds:

$$\left.\begin{array}{l} \text{if } C \in \mathcal{L}(x) \text{, then } \pi(x) \in C^{\mathcal{I}} \\ \text{if } \mathcal{L}(\langle x, y\rangle) = R, \text{ then } \langle \pi(x), \pi(y)\rangle \in R^{\mathcal{I}} \\ \text{if } y, z \text{ are } R\text{-neighbours of } x \text{ with } y \neq z, \\ \quad \text{then } \pi(y) \neq \pi(z) \end{array}\right\} (*)$$

CLAIM: Whenever $(*)$ holds for a tree $\mathbf{T}$ and a function π, and a rule is applicable to $\mathbf{T}$ then it can be applied in a way that maintains $(*)$.

The proof of this claim can be found in [17]. Lemma 5 is a consequence of this claim: obviously, $(*)$ holds for the initial tree with root node x_0 if we set $\pi(x_0) := s_0$ for an element $s_0 \in D^{\mathcal{I}}$ (such an element must exist because $\mathcal{I}$ is a model for D). Lemma 3 yields that each sequence of rule applications must terminate, and also each tree for which $(*)$ holds is necessarily clash-free because: (a) it cannot contain a clash of the form $\{A, \neg A\} \subseteq \mathcal{L}(x)$ since this would imply $\pi(x) \in A^{\mathcal{I}}$ and $\pi(x) \notin A^{\mathcal{I}}$; (b) it can neither contain a clash of the form $(\leqslant n\ R\ C) \in \mathcal{L}(x)$ and $\sharp R^{\mathbf{T}}(x, C) > n$ because π is injective on the set of all R-neighbours of a node x and hence $\sharp R^{\mathbf{T}}(x, C) > n$ implies $\sharp R^{\mathcal{I}}(x, C) > n$. This contradicts $\pi(x) \in (\leqslant n\ R\ C)^{\mathcal{I}}$. □

Theorem 6 *The tableaux algorithm is a non-deterministic decision procedure for $\mathcal{ALCQI}$-satisfiability.*

Complexity of the Algorithm What remains to show is that the algorithm can be implemented to run in polynomial space. Due to Savitch's theorem [13] (that states that PSPACE coincides with NPSPACE) it is sufficient to give a non-deterministic algorithm that runs in PSPACE. Still, as for $\mathcal{ALC}$, models for a $\mathcal{ALCQI}$-concept may be required to have exponential size so we have to develop a method that facilitates re-use of space while generating the completion tree.

Lemma 7 *The tableaux algorithm can be implemented to run in polynomial space.*

Proof. Let D be the $\mathcal{ALCQI}$-concept to be tested for satisfiability. We may assume D to be in NNF because the transformation of a formula to NNF can be performed in linear time and space.

The key idea for a PSPACE implementation is the *trace technique*, i.e., it is sufficient to keep only a single path (a trace) of $\mathbf{T}$ in memory at a given stage if the completion tree is generated in a depth-first manner. This technique has already been the key to a PSPACE upper bound for the propositional modal logic K_m and $\mathcal{ALC}$ in [14, 9] and it may be generalised to deal with inverse roles by a "reset-restart" technique as described in [11]. Moreover, to deal with the cardinality restrictions, we need to store the values for $\sharp R^{\mathbf{T}}(x, C)$ for each node x, each R which appears in *clos*(D), and each $C \in$ *clos*(D). By storing these values in binary form, we are able to keep information *about* exponentially many successors in memory while storing only a single path at any stage.

Once the existence of a complete and clash-free "subtree" for the constraints on a successor y of x has been established, this subtree will be discarded from memory. This is admissible since the tableaux rules can delete but will never modify this subtree once it is completed. This deletion is necessary because the *choose*-rule pushes concepts backwards which has an influence on other subtrees of the effected node. Since these may already have been discarded from memory they have to be regenerated.

Constraints in a subtree have no influence on the completeness or the existence of a clash in the rest of the tree, with the exception that a concept $C \in \mathcal{L}(y)$ for an R-neighbour y of x contributes to the value of $\sharp R^{\mathbf{T}}(x, C)$. These numbers play a rôle both in the definition of a clash and for the applicability of the $\geqslant$-rule. Hence, in order to re-use the space occupied by the subtree for y, it is necessary and sufficient to store only these numbers.

This algorithm consumes only polynomial space: (a) the depth of the tree is linearly bounded; (b) each node label is a subset of a set of linear size; (c) there are polynomially many relevant counters; and (d) the values of the counters do not exceed the out-degree of the tree and thus can be stored in polynomial size using binary coding of numbers. □

Obviously, satisfiability for $\mathcal{ALCQI}$ is at least as hard as for $\mathcal{ALC}$. Together with the previous lemma this yields:

Theorem 8 SAT*($\mathcal{ALCQI}$) is* PSPACE*-complete, even if numbers in the input are represented in binary coding.*

2 The Complexity of Cardinality Restrictions

Cardinality restrictions on concepts allows to restrict the model of a knowledge base with respect to the number of instances of complex concepts. They have first been introduced in [1] for $\mathcal{ALCQ}$ as a terminological formalism that is interesting for configuration applications. While the exact complexity of $\mathcal{ALCQ}$ with cardinality restrictions is unknown, we show that, for $\mathcal{ALCQI}$, reasoning is NEXPTIME-complete provided that unary coding of numbers in the input is assumed.

Definition 9 (Cardinality Restrictions) *A* cardinality restriction *is an expression of the form $(\geqslant\ n\ C)$ or $(\leqslant\ n\ C)$ where C is an $\mathcal{ALCQI}$-concept and $n \in \mathbb{N}$; a* TBox *T is a finite set of cardinality restrictions. An interpretation $\mathcal{I}$ satisfies $(\bowtie\ n\ C)$ iff $\sharp C^{\mathcal{I}} \bowtie n$; it is model of a TBox T ($\mathcal{I} \models T$) iff it satisfies all cardinality restrictions in T. A TBox T is called* consistent *iff there is a model $\mathcal{I}$ of T.*

Cardinality restrictions can express terminological axioms of the form $C = D$, which are the most expressive TBox formalisms usually studied in the DL context [7] as follows: two concepts C, D have the same extension in an interpretation $\mathcal{I}$ iff $\mathcal{I} \models (\leqslant\ 0\ (C \sqcap \neg D) \sqcup (\neg C \sqcap D))$. A standard inference service of a DL system is satisfiability of a concept C with respect to a TBox T (i.e., is there a model $\mathcal{I}$ of T with $C^{\mathcal{I}} \neq \emptyset$?). For TBoxes consisting of cardinality restrictions, this can be reduced to consistency of the TBox $T' := T \cup \{(\geqslant\ 1\ C)\}$.

Consistency of $\mathcal{ALCQI}$-TBoxes can naturally be reduced to satisfiability of C^2—the fragment of first order predicate logic restricted to two variables augmented with counting quantifiers—by an extension of the translation given in [4]. If we assume unary coding of numbers, this reduction yields a NExpTime upper bound [12] for the complexity TBox-consistency for $\mathcal{ALCQI}$. By reduction from a bounded domino problem [3], we show that NExpTime is also a lower bound for the problem. Again, we can only present the ideas of most proofs. Please refer to [15] for details.

Definition 10 (Domino System) *For $n \in \mathbb{N}$, let $\mathbb{Z}_n$ denote the set $\{0, \ldots, n-1\}$ and $\oplus_n$ denote the addition modulo n. A* domino system *is a triple $\mathcal{D} = (D, H, V)$, where D is a finite set (of tiles) and $H, V \subseteq D \times D$ are relations expressing horizontal and vertical compatibility constraints between tiles. For $s, t \in \mathbb{N}$, let $U(s, t)$ be the torus $\mathbb{Z}_s \times \mathbb{Z}_t$ and $w = w_0, \ldots, w_{n-1}$ be an n-tuple of tiles (with $n \leq s$). We say that $\mathcal{D}$* tiles *$U(s, t)$* with initial condition *w iff there exists a mapping $\tau : U(s, t) \to D$ such that, for all $(x, y) \in U(s, t)$,*

- *if $\tau(x, y) = d$ and $\tau(x \oplus_s 1, y) = d'$ then $(d, d') \in H$ (horizontal constraint);*
- *if $\tau(x, y) = d$ and $\tau(x, y \oplus_t 1) = d'$ then $(d, d') \in V$ (vertical constraint);*
- *$\tau(i, 0) = w_i$ for $0 \leq i < n$ (initial condition).*

Bounded domino systems are capable of simulating the computational behaviour of restricted, so called *simple*, Turing Machines (TM). This restriction is non-essential in the following sense: Every language accepted in time $T(n)$ and space $S(n)$ by some one-tape TM is accepted within the same time and space bounds by a simple TM, as long as $S(n), T(n) \geq 2n$ [3]. Exploiting the correspondence between computations of resource bounded TMs and tilings of bounded domino systems the following lemma can easily be shown.

Lemma 11 *There is a domino system $\mathcal{D}$ such that the following is a* NExpTime*-hard problem. "Given an initial condition $w = w_0, \ldots, w_{n-1}$ of length n. Does $\mathcal{D}$ tile $U(2^{n^d+1}, 2^{n^d+1})$ with initial condition w?"*

Defining a Torus of Exponential Size Just as defining infinite grids is the key problem in proving undecidability by reduction of unbounded domino problems, defining a torus of exponential size is the key to obtaining a NExpTime-hardness proof by reduction of bounded domino problems.

To apply Lemma 11 to TBox consistency for $\mathcal{ALCQI}$, we must characterise the torus $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$ with a TBox of polynomial size. We will use $2n$ atomic concepts $X_0, \ldots, X_{n-1}$ and $Y_0, \ldots, Y_{n-1}$, where X_i codes the ith bit of the binary representation of the X-coordinate of an element a:

For an interpretation $\mathcal{I}$ and $a \in \Delta^{\mathcal{I}}$, we define $pos(a)$ by

$$\mathit{pos}(a) := (\mathit{xpos}(a), \mathit{ypos}(a)) := \left(\sum_{i=0}^{n-1} x_i \cdot 2^i, \sum_{i=0}^{n-1} y_i \cdot 2^i\right),$$

where $x_i := 0$ if $a \notin X_i^{\mathcal{I}}$ and 1 otherwise, and y_i is defined analogously.

Let $C_{(0,0)}$ be a concept that is satisfied by all elements a of the domain with $pos(a) = (0, 0)$ and, similarly, $C_{(2^n-1,2^n-1)}$ is a similar concept, which is satisfied if $pos(a) = (2^n - 1, 2^n - 1)$. Let D_{east} (resp. D_{north}) be concepts enforcing that along the role *east* (*north*) the value of *xpos* (resp. *ypos*) increases by one while the value of *ypos* (resp. *xpos*) stays the same. This can expressed using a characterisation of binary addition from [9]:

$$\begin{aligned}
C_{(0,0)} &= \prod_{k=0}^{n-1} \neg X_k \sqcap \prod_{k=0}^{n-1} \neg Y_k, \\
D_{east} &= \prod_{k=0}^{n-1} (\prod_{j=0}^{k-1} X_j) \to ((X_k \to \forall east.\neg X_k) \sqcap \\
&\qquad\qquad (\neg X_k \to \forall east.X_k)) \sqcap \\
&\quad \prod_{k=0}^{n-1} (\bigsqcup_{j=0}^{k-1} \neg X_j) \to ((X_k \to \forall east.X_k) \sqcap \\
&\qquad\qquad (\neg X_k \to \forall east.\neg X_k)) \sqcap \\
&\quad \prod_{k=0}^{n-1} ((Y_k \to \forall east.Y_k) \sqcap (\neg Y_k \to \forall east.\neg Y_k)).
\end{aligned}$$

We define the TBox T_n to consist of the following cardinality restrictions:

$$\begin{array}{ll}
(\forall\ (\geqslant 1\ \mathit{east}\ .)\top), & (\forall\ (\geqslant 1\ \mathit{north}\ .)\top), \\
(\forall\ (= 1\ \mathit{east}^{-1}.\top)), & (\forall\ (=\ 1\ \mathit{north}^{-1}.\top)), \\
(\geq 1\ C_{(0,0)}), & (= 1\ C_{(2^n-1,2^n-1)}), \\
(\forall\ D_{east} \sqcap D_{north}), &
\end{array}$$

where we use the following abbreviations: $(\forall\ C)$ is an abbreviation for the cardinality restriction $(\leq 0\ \neg C)$; $\top$ stands for a concept that is satisfied everywhere in all interpretations (e.g., $A \sqcup \neg A$). The TBox T_n defines a torus of exponential size in the following sense:

Lemma 12 *Let T_n be the TBox as introduced above. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of T_n.*

- *The size $|T_n|$ (i.e., the number of symbols necessary to write down T_n) is quadratic in n.*

- $(\Delta^{\mathcal{I}}, east^{\mathcal{I}}, north^{\mathcal{I}}) \cong (U(2^n, 2^n), S_1, S_2)$, *where* $U(2^n, 2^n)$ *is the torus* $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$ *and* S_1, S_2 *are the horizontal and vertical successor relations on* $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$.

We will only sketch the proof of this lemma. The first part is immediate from the construction of T_n. The second part is established by showing that the function *pos* is an isomorphism from $\Delta^{\mathcal{I}}$ to $U(2^n, 2^n)$. That *pos* is a homomorphism follows immediately from the definition of D_{east} and D_{north}. Injectivity of *pos* is established by showing that each element $(x, y) \in U(2^n, 2^n)$ is the image of at most one element of $\Delta^{\mathcal{I}}$ by induction over the Manhattan distance of (x, y) to the upper right corner $(2^n - 1, 2^n - 1)$ of the torus. The base case is trivial because T_n contains the cardinality restrictions $(\leq\ 1\ C_{(2^n-1, 2^n-1)})$. The induction step follows from the fact that each element $a \in \Delta^{\mathcal{I}}$ has exactly one *east*- and *north*-predecessor (since $(\forall\ (=1\ east^{-1}.\top)), (\forall\ (=1\ north^{-1}.\top)) \in T_n$). Surjectivity is established similarly starting from $(0, 0)$.

Reducing Domino Problems to TBox Consistency Once Lemma 12 has been proved, it is easy to reduce the bounded domino problem to TBox consistency. We use a reduction similar to the on in [2], which uses the fact that we can express terminological axioms.

Lemma 13 *For a domino system* $\mathcal{D} = (D, V, H)$ *and* $w = w_0, \ldots, w_{n-1} \in D^*$, *there is a TBox* $T(n, \mathcal{D}, w)$ *with*

- $T(n, \mathcal{D}, w)$ *is consistent iff* $\mathcal{D}$ *tiles* $U(2^n, 2^n)$ *with initial condition* w, *and*
- $T(n, \mathcal{D}, w)$ *can be computed in time polynomial in* n.

Together with Theorem 11 this yields:

Theorem 14 *Consistency of* $\mathcal{ALCQI}$*-TBoxes is* NEXPTIME*-hard, even if unary coding of numbers is used in the input.*

Satisfiability of $\mathcal{ALCQI}$-concepts with respect to TBoxes consisting of terminological axioms is known to be EXPTIME-complete [8]. This indicates that cardinality restrictions, although interesting for knowledge representation, increase the complexity of the inference problem. While $\mathcal{ALCQI}$ with cardinality restriction has the same worst-case complexity as C^2, we have shown that it does not reach the expressiveness of the latter [15].

References

[1] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.

[2] F. Baader and U. Sattler. Number restrictions on complex roles in description logics. In *Proceedings of KR'96*, Morgan Kaufmann Publishers, 1996.

[3] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer-Verlag, 1997.

[4] A. Borgida. On the relative expressiveness of description logics and first order logics. *Artificial Intelligence*, 82:353–367, 1996.

[5] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proceedings of KR-94*, Morgan Kaufmann Publishers, 1994.

[6] W. Van der Hoek and M. De Rijke. Counting objects. *J. of Logic and Computation*, 5(3):325–345, June 1995.

[7] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proceedings of KR'96*, Morgan Kaufmann Publishers, 1996.

[8] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, to appear.

[9] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for model logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, April 1992.

[10] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proceedings of KR'91*, 1991.

[11] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for description logics with functional restrictions, inverse and transitive roles, and role hierarchies. In *Proceedings of the 1999 Workshop Methods for Modalities (M4M-1)*, Amsterdam, 1999.

[12] L. Pacholski, W. Szwast, and L. Tendera. Complexity of two-variable logic with counting. In *Proceedings of LICS'97*, IEEE Computer Society Press, 1997.

[13] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, April 1970.

[14] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[15] S. Tobies. A NExpTime complete description logic strictly contained in C^2. In *Proceeding of CSL-99*, LNCS. Springer Verlag, 1999. To appear.

[16] S. Tobies. A PSPACE algorithm for graded modal logic. In *Proceedings of CADE-99*, LNCS. Springer-Verlag, 1999. To appear.

[17] S. Tobies. A PSPACE-algorithm for ALCQI-satisfiability. LTCS-Report 99-09, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1999. See http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html