Finite Model Reasoning in $\mathcal{ALCQI}$ is ExpTime-complete

Carsten Lutz[1], Ulrike Sattler[2], and Lidia Tendera[3]

[1] Institute for Theoretical Computer Science, TU Dresden, Germany
lutz@tcs.inf.tu-dresden.de

[2] Department of Computer Science, University of Manchester, UK
sattler@cs.man.ac.uk

[3] Institute of Mathematics and Informatics, Opole University, Poland
tendera@math.uni.opole.pl

1 Motivation

Most description logics (DLs) enjoy the finite model property (FMP). This is, for example, the case for $\mathcal{ALC}$ [14] and many of its extensions such as $\mathcal{ALCI}$ ($\mathcal{ALC}$ with inverse roles) and $\mathcal{ALCQ}$ ($\mathcal{ALC}$ with qualifying number restrictions): for any of these logics $\mathcal{L}$, each satisfiable $\mathcal{L}$-concept has a finite model. This even holds if we consider concept satisfiability w.r.t. general TBoxes. However, there also exist natural description logics that do not enjoy FMP. A rather prominent example is $\mathcal{ALCQI}$, which is obtained from $\mathcal{ALC}$ by adding both inverse roles and qualifying number restrictions. While $\mathcal{ALCQI}$ *without* TBoxes still has the FMP, this is no longer the case in the presence of general TBoxes: for example, the concept $\neg A \sqcap \exists R.A$ is satisfiable w.r.t. the TBox

$$\{A \doteq \exists R.A \sqcap (\leqslant 1\ R^{-}\ \top)\},$$

but each of its models contains an infinite R-chain.

The fact that $\mathcal{ALCQI}$ lacks FMP becomes particularly important if we consider this logic's most prominent application, which is reasoning about conceptual database models as proposed by Calvanese et al. [4]: if such a model is described by an ER diagram or a UML diagram, then it can be translated into a DL TBox, and DL reasoners such as FaCT and RACER can be used to detect inconsistencies and to infer implicit IS-A relationships between entities/classes. This useful and original application has led to the implementation of I.com, a tool that provides a GUI for specifying conceptual models, automatizing the translation into DLs, and displaying the information returned by the reasoner [7]. However, it is well-known that there exist quite simple ER and UML diagrams that are satisfiable only in infinite models, but not in finite ones [15]. Since all available DL reasoning systems are performing reasoning in arbitrary (as opposed to finite) models, this means that some inconsistencies and IS-A relationships will go unnoticed if standard reasoners are used, e.g. in conjunction with I.com.

The main reason for existing DL reasoners to perform only reasoning w.r.t. arbitrary models is that finite model reasoning in description logics such as $\mathcal{ALCQI}$ is not yet well-understood. The only known algorithm is presented by Calvanese in [2], where he proves that reasoning in $\mathcal{ALCQI}$ is decidable in 2-ExpTime. *The purpose of this paper is to establish tight* ExpTime *complexity bounds for finite model reasoning in $\mathcal{ALCQI}$.* More precisely, we develop an algorithm that is capable of deciding finite satisfiability of $\mathcal{ALCQI}$-concepts w.r.t. general TBoxes. Similar to Calvanese's approach, the core idea behind our algorithm is to translate a given satisfiability problem into a set of linear inequalities that can then be solved by linear programming methods. The main difference to Calvanese's approach is that our equation systems talk about different components of models, *mosaics*, which allows us to keep the size of equation systems exponential in the size of the input. In this way, we improve the best-known 2-ExpTime upper bound to a tight ExpTime one.

Since our algorithm presupposes unary coding of the numbers occurring in qualifying number restrictions, it does not *immediately* yield an ExpTime-upper bound for the binary coding case. Therefore, we use a different approach to establish this bound, namely a reduction of $\mathcal{ALCQI}$-concept satisfiability w.r.t. TBoxes to $\mathcal{ALCFI}$-concept satisfiability w.r.t. TBoxes ($\mathcal{ALCFI}$ is obtained from $\mathcal{ALCQI}$ by allowing only the numbers 0 and 1 to be used in number restrictions). Since the latter problem is in ExpTime due to our initial result for $\mathcal{ALCQI}$ with unary coding, we obtain the desired bound. Finally, we show how to extend our ExpTime-upper bound to finite $\mathcal{ALCQI}$-ABox consistency w.r.t. general TBoxes (with numbers coded in binary).

All details and proofs can be found in the accompanying technical report [11].

2 The Algorithm for Unary Coding

We use the standard syntax and semantics for $\mathcal{ALCQI}$, assuming that $\exists R.C$ and $\forall R.C$ are just abbreviations for $(\geqslant 1\ R\ C)$ and $(\leqslant 0\ R\ \neg C)$, respectively. By "unary coding", we mean that $|(\geqslant n R.C)| = |(\leqslant n R.C)| = n + |C| + 1$.

As observed by Calvanese in [2], combinatorics is an important issue when deciding finite satisfiability of $\mathcal{ALCQI}$-concepts. To illustrate this, consider the TBox

$$\mathcal{T} := \{A \doteq (\geqslant 2\ R\ B), \quad B \doteq (\leqslant 1\ R^-\ A)\}.$$

It should be clear that, in any model of $\mathcal{T}$, there are at least twice as many objects satisfying $B \sqcap (\leqslant\ 1\ R^-\ A)$ as there are objects satisfying $A \sqcap (\geqslant\ 2\ R\ B)$. This simple example suggests that (i) *types* (i.e., sets of concepts satisfied by a particular object in a particular model) such as $\{A, (\geqslant 2\ R\ B)\}$ are a natural notion for dealing with finite satisfiability, and (ii) the combinatorics introduced by finite domains can be addressed with inequalities like $2 \cdot x_T \leq x_{T'}$, where the variable x_T describes the number of instance of a type T (e.g. $\{A, (\geqslant 2\ R\ B)\}$), while $x_{T'}$ describes the number of instances of another type T' (e.g. $\{B, (\leqslant 1\ R^-\ A)\}$).

Considering the above two points, a first idea to devise a decision procedure for finite satisfiability of $\mathcal{ALCQI}$-concepts w.r.t. TBoxes is to translate an input concept and TBox into a system of inequalities with one variable for each type, and then to use linear programming algorithms to check whether the equation system has a non-negative integer solution. For example, the satisfiability problem of the concept A

w.r.t. the TBox $\mathcal{T}$ above can be translated into the two inequalities

$$\sum_{\{T \mid (\geqslant 2\ R\ B) \in T\}} 2 \cdot x_T \leq \sum_{\{T \mid (\leqslant 1\ \mathsf{Inv}(R)\ A) \in T\}} x_T \quad \text{and} \quad \sum_{\{T \mid A \in T\}} x_T > 0$$

where the sums range over all types induced by the input concept A and TBox $\mathcal{T}$. It is not hard to see that any non-negative integer solution to this equation system can be used to construct a finite model for A and $\mathcal{T}$ and vice versa.

Unfortunately, there is a problem with this approach: assume that the input concept and TBox induce types T_1 to T_5 as follows: $(\geqslant 1\ R\ C) \in T_1$, $(\geqslant 1\ R\ D) \in T_2$, $(\leqslant 1\ \mathsf{Inv}(R)\ \top) \in T_3 \cap T_4 \cap T_5$, $C \in T_3 \cap T_4$, and $D \in T_4 \cap T_5$. The translation described above yields the inequalities

$$x_{T_1} \leq x_{T_3} + x_{T_4} \text{ and } x_{T_2} \leq x_{T_4} + x_{T_5},$$

which have $x_{T_1} = x_{T_2} = x_{T_4} = 1$ and $x_{T_3} = x_{T_5} = 0$ as an integer solution. Trying to construct a model with a_1, a_2, and a_4 instances of T_1, T_2, and T_4, respectively, we have to use a_4 as a witness of a_1 being an instance of $(\geqslant 1\ R\ C)$ and a_2 being an instance of $(\geqslant 1\ R\ D)$. However, this violates the $(\leqslant 1\ \mathsf{Inv}(R)\ \top)$ concept in T_4.

This example illustrates that "counting types" does not suffice: conflicts may arise if a type containing an at-most restriction (T_4) can be used as a witness for at-least restrictions in more than one type (T_1 and T_2). It is thus necessary to (additionally) fix the types that are actually used as witnesses for at-least restrictions. We achieve this by defining systems of inequalities that are based on chunks of models called *mosaics*, rather than being based directly on types. Intuitively, a mosaic describes the type of an object and fixes the type of "important" witnesses.

Before defining mosaics, we introduce some preliminaries. In the remainder of this paper, w.l.o.g. we assume concepts (also those appearing inside TBoxes) to be in negation normal form (NNF), i.e. negation is only allowed in front of concept names. We use $\dot{\neg}C$ to denote the NNF of $\neg C$. Moreover, we will only consider TBoxes of the form $\{\top \doteq C\}$. This can be done w.l.o.g. since every TBox $\mathcal{T} = \{C_i \doteq D_i \mid 1 \leq i \leq n\}$ can be rewritten as $\{\top \doteq \bigsqcap_{1 \leq i \leq n}(C_i \leftrightarrow D_i)\}$. For a concept C_0 and a TBox $\mathcal{T} = \{\top \doteq C_\mathcal{T}\}$, $\mathsf{cl}(C_0, \mathcal{T})$ is the smallest set containing all sub-concepts of C_0 and $C_\mathcal{T}$ that is closed under $\dot{\neg}$. We use $\mathsf{rol}(C_0, \mathcal{T})$ to denote the set of role names R and their inverses R^- occurring in C_0 or $\mathcal{T}$. Finally, we define a function Inv on roles such that $\mathsf{Inv}(R) = R^-$ if R is a role name, and $\mathsf{Inv}(R) = S$ if $R = S^-$.

Definition 1 (Types and Mosaics) *A* type *T for $C_0, \mathcal{T} = \{\top \doteq C_\mathcal{T}\}$ is a set $T \subseteq \mathsf{cl}(C_0, \mathcal{T})$ such that, for each $D, E \in \mathsf{cl}(C_0, \mathcal{T})$, we have*

(T1) *$D \in T$ iff $\dot{\neg}D \notin T$,*

(T2) *if $D \sqcap E \in \mathsf{cl}(C_0, \mathcal{T})$, then $D \sqcap E \in T$ iff $D \in T$ and $E \in T$,*

(T3) *if $D \sqcup E \in \mathsf{cl}(C_0, \mathcal{T})$, then $D \sqcup E \in T$ iff $D \in T$ or $E \in T$, and*

(T4) *$C_\mathcal{T} \in T$.*

We use $\mathsf{type}(C_0, \mathcal{T})$ *to denote the set of all types for* $C_0, \mathcal{T}$*. Let* T *be a type and* $\bowtie \in \{\leqslant, \geqslant\}$*. Then we use the following abbreviations:*

$$\mathsf{max}^{\bowtie}(T) := \max\{n \mid (\bowtie n\ R\ C) \in T\} \quad \textit{and} \quad \mathsf{sum}^{\bowtie}(T) := \sum_{(\bowtie n\ R\ C) \in T} n.$$

For types T_1, T_2 *and a role* R*, we write* $\lim_R(T_1, T_2)$ *(*T_2 *is a* limited ressource *for* T_1 *w.r.t.* R*) if* $C \in T_1$ *and* $(\leqslant n\ \mathsf{Inv}(R)\ C) \in T_2$ *for some* $C \in \mathsf{cl}(C_0, \mathcal{T})$ *and* $n \in \mathbb{N}$*.*

A mosaic *for* $C_0, \mathcal{T}$ *is a triple* $M = (T_M, L_M, E_M)$ *where*

- $T_M \in \mathsf{type}(C_0, \mathcal{T})$,
- L_M *is a function from* $\mathsf{rol}(C_0, \mathcal{T}) \times \mathsf{type}(C_0, \mathcal{T})$ *to* $\mathbb{N}$*, and*
- E_M *is a function from* $\mathsf{rol}(C_0, \mathcal{T}) \times \mathsf{type}(C_0, \mathcal{T})$ *to* $\mathbb{N}$

such that the following conditions are satisfied:

(M1) *if* $L_M(R, T) > 0$*, then* $\lim_R(T_M, T)$ *and not* $\lim_{\mathsf{Inv}(R)}(T, T_M)$*,*

(M2) *if* $E_M(R, T) > 0$*, then* $\lim_{\mathsf{Inv}(R)}(T, T_M)$*,*

(M3) *if* $(\leqslant n\ R\ C) \in T_M$*, then* $n \geq \sum_{\{T \mid C \in T\}} E_M(R, T)$*,*

(M4) $\#\{(R, T) \mid L_M(R, T) > 0\} \leq \mathsf{sum}^{\geqslant}(T_M)$ *and* $\max(\text{range}(L_M)) \leq \mathsf{max}^{\geqslant}(T_M)$*.*

Consider a mosaic M and one of its "instances" d in some interpretation. While T_M is simply the type of d, L_M and E_M are used to describe certain "neighbors" of d, i.e. objects e reachable from d via a role. For a role R, there are three possibilities for the relationship between T_M and T, the type of e:

1. Not $\lim_R(T_M, T)$ and not $\lim_{\mathsf{Inv}(R)}(T, T_M)$. Then d may have an arbitrary number of R-neighbors of type T and every instance of T may have an arbitrary number of $\mathsf{Inv}(R)$-neighbors of type T_M. Intuitively, R-neighbors of type T are "uncritical" and not recorded in the mosaic.

2. $\lim_R(T_M, T)$ and not $\lim_{\mathsf{Inv}(R)}(T, T_M)$. Then d may have an arbitrary number of R-neighbors of type T, but every instance of T may only have a limited number of $\mathsf{Inv}(R)$-neighbors of type T_M. Thus, R-neighbors of type T are a limited ressource and we record in L_M the *minimal* number of R-neighbors of type T that d needs ("L" for "lower bound").

3. $\lim_{\mathsf{Inv}(R)}(T, T_M)$. Then d may only have a limited number of R-neighbors of type T. To prevent the violation of at-most restrictions in T_M, we record the *exact* number of d's R-neighbors of type T in E_M.

(M1) and (M2) ensure that L_M and E_M record information for the "correct" types as described above; (M3) ensures that at-most restrictions are not violated: by definition, this concerns only neighbors with E_M-types; finally, (M4) puts upper bounds on L_M to ensure that there exist only exponentially many mosaics (see below). At-least restrictions are not mentioned in the definition of mosaics and will be treated by the systems of inequalities to be defined later.

Now for the number of mosaics. First, the cardinality of $\mathsf{type}(C_0, \mathcal{T})$ is exponential in the size of C_0 and $\mathcal{T}$. Next, ($\mathsf{M2}$) and ($\mathsf{M3}$) imply $\#\{(R,T) \mid E_M(R,T) > 0\} \leq \mathsf{sum}^{\leqslant}(T_M)$ and $\max(\mathrm{range}(E_M)) \leq \mathsf{max}^{\leqslant}(T_M)$. Analogous bounds for L_M are enforced by ($\mathsf{M4}$). Now $\mathsf{max}^{\bowtie}(T)$ and $\mathsf{sum}^{\bowtie}(T)$ are linear in the size of C_0 and $\mathcal{T}$ for $\bowtie \in \{\leqslant, \geqslant\}$ since numbers are coded in unary, and thus the number of mosaics is bounded exponentially in the size of C_0 and $\mathcal{T}$.

We now define a system of inequalities for a concept C_0 and a TBox $\mathcal{T}$.

Definition 2 (Equation System) *Let C_0 be an $\mathcal{ALCQI}$-concept and $\mathcal{T}$ a TBox. We introduce a variable x_M for each mosaic M for $C_0, \mathcal{T}$ and define the equation system $\mathcal{E}_{C_0,\mathcal{T}}$ by taking (i) the equation*

$$\sum_{\{M \mid C_0 \in T_M\}} x_M \geq 1, \tag{E1}$$

(ii) for each pair of types T, $T' \in \mathsf{type}(C_0, \mathcal{T})$ and role R such that $\lim_R(T,T')$ and not $\lim_{\mathsf{Inv}(R)}(T',T)$, the equation

$$\sum_{\{M \mid T_M = T\}} L_M(R,T') \cdot x_M \leq \sum_{\{M \mid T_M = T'\}} E_M(\mathsf{Inv}(R),T) \cdot x_M, \tag{E2}$$

and (iii) for each pair of types T, $T' \in \mathsf{type}(C_0, \mathcal{T})$ and role R such that $\lim_R(T,T')$ and $\lim_{\mathsf{Inv}(R)}(T',T)$, the equation

$$\sum_{\{M \mid T_M = T\}} E_M(R,T') \cdot x_M = \sum_{\{M \mid T_M = T'\}} E_M(\mathsf{Inv}(R),T) \cdot x_M. \tag{E3}$$

A solution of $\mathcal{E}_{C_0,\mathcal{T}}$ is admissible *if it is a non-negative integer solution and satisfies the following conditions: (i) for each pair of types T, $T' \in \mathsf{type}(C_0, \mathcal{T})$ and role R such that $\lim_R(T,T')$ and not $\lim_{\mathsf{Inv}(R)}(T',T)$,*

$$\textit{if} \sum_{\{M \mid T_M = T'\}} E_M(\mathsf{Inv}(R),T) \cdot x_M > 0, \ \textit{then} \sum_{\{M \mid T_M = T\}} x_M > 0; \tag{A1}$$

(ii) for each mosaic M and each role R, if $(\geqslant n \; R \; C) \in T_M$,

$$x_M > 0, \ \textit{and} \sum_{\{T \mid C \in T\}} L_M(R,T) + \sum_{\{T \mid C \in T\}} E_M(R,T) < n,$$

$$\textit{then} \sum_{\substack{\{M' \mid \ C \in T_{M'}, \ \textit{not} \ \lim_R(T_M,T_{M'}), \\ \textit{and not} \ \lim_{\mathsf{Inv}(R)}(T_{M'},T_M)\}}} x_{M'} > 0 \tag{A2}$$

While inequality (E1) guarantees the existence of an instance of C_0, inequalities (E2) and (E3) enforce the lower and exact bounds on the number of neighbors as described by L_M and E_M. A special case is treated by condition (A1): in inequality (E2), it may happen that the left-hand side is zero while the right-hand side is non-zero. In this case, there is an instance of a mosaic M' with $T_{M'} = T'$ and $E_M(\mathsf{Inv}(R), T) > 0$ (counted on the right-hand side), but there is no instance of a mosaic M with $T_M = T$

(counted on the left-hand side)—thus we cannot find any neighbors as required by $E_M(\mathsf{Inv}(R), T)$. To cure this defect, condition (A1) ensures that, if the right-hand side of (E2) is non-zero, then there is at least one instance of a mosaic M with $T_M = T$.[1] Finally, (A2) takes care of at-least restrictions in types T_M: if the number of R-neighbors enforced by L_M and E_M is not enough for some $(\geqslant n\ R\ C) \in T_M$, then we make sure that there is at least one instance of a mosaic M' such that $C \in T_{M'}$ and, for instances of M (M'), the number of R-neighbors ($\mathsf{Inv}(R)$-neighbors) that are instances of M' (M) is not limited.[1]

Lemma 1 *C_0 is finitely satisfiable w.r.t. $\mathcal{T}$ iff the equation system $\mathcal{E}_{C_0,\mathcal{T}}$ has an admissible solution.*

The proof of this lemma can be found in [11].

Since the number of mosaics is exponential in the size of C_0 and $\mathcal{T}$, the size of $\mathcal{E}_{C_0,\mathcal{T}}$ and of the admissibility condition is also exponential in the size of C_0 and $\mathcal{T}$. To prove an ExpTime upper bound for the finite satisfiability of $\mathcal{ALCQI}$-concepts, it thus remains to show that the existence of an admissible solution for the equation systems $\mathcal{E}_{C_0,\mathcal{T}}$ can be decided in deterministic polynomial time.

We assume linear inequalities to be of the form $\Sigma_i c_i x_i \geq b$. A system of linear inequalities is described by a tuple $(V, \mathcal{E})$, where V is a set of variables and $\mathcal{E}$ a set of inequalities using variables from V. Such a system is called *simple* if only non-negative integers occur on the right-hand side of inequalities and all coefficients are (possibly negative) integers. A *side condition* for an inequality system $(V, \mathcal{E})$ is a constraint of the form $x > 0 \Longrightarrow x_1 + \cdots + x_\ell > 0$, where $x, x_1, \ldots x_\ell \in V$. It is not hard to check that the inequalities (Ei) can be polynomially transformed into simple ones, and that the conditions (Ai) can be polynomially transformed into side conditions; more details are given in [11]. The proof of the following lemma is by reduction to linear programming and can be found in [11].

Lemma 2 *Let $(V, \mathcal{E})$ be a simple equation system and I a set of side conditions for $(V, \mathcal{E})$. Then the existence of a non-negative integer solution for $(V, \mathcal{E})$ satisfying all constraints from I can be decided in (deterministic) time polynomial in $\#V + \#\mathcal{E} + \#I$.*

Since satisfiability of $\mathcal{ALC}$ w.r.t. TBoxes in arbitrary models is ExpTime-hard [13] and this DL has the finite model property, combining Lemmas 1 and 2 yields the following theorem:

Theorem 1 *Finite satisfiability of $\mathcal{ALCQI}$-concepts w.r.t. TBoxes is* ExpTime*-complete if numbers are coded in unary.*

3 Binary Coding

If numbers in number restrictions are coded binarily (i.e. $|(\geqslant nR.C)| = |(\leqslant nR.C)| = \log(n) + |C| + 1$), Theorem 1 does no longer apply: in this case, the number of mosaics is double exponential in the size of the input, and thus the algorithm used in Section 2 yields only a 2-ExpTime upper bound. For reasoning w.r.t arbitrary models, a variety

[1]To see why a single instance suffices, consult the proof sketch of Lemma 1 in [11].

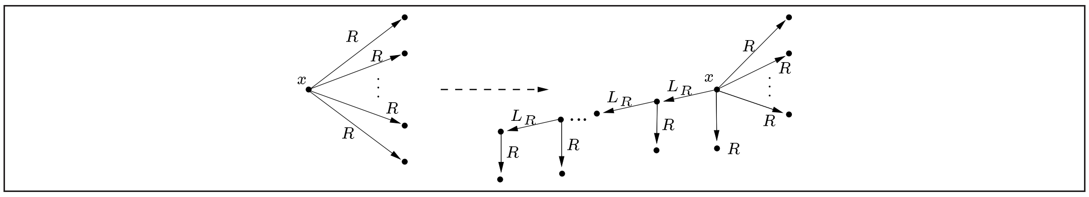

Figure 1: Representing role successor relationships.

of results are known that suggest that the coding of numbers in number restrictions is usually irrelevant for the complexity of reasoning, see e.g. [16, 10]. However, for finite model reasoning, similar results do not seem to be known. Indeed, it is a non-trivial problem whether the algorithm used in Section 2 can be adapted to binary coding. We have to leave this problem open and choose an alternative technique: a reduction of finite $\mathcal{ALCQI}$-concept satisfiability to the finite satisfiability of $\mathcal{ALCFI}$-concepts. This reduction is polynomial even for binary coding of numbers, and finite satisfiability in the target logic is in ExpTime due to Theorem 1 (since number restrictions in $\mathcal{ALCFI}$ are restricted to the numbers 0 and 1, the coding is not an issue). Note that we cannot use existing reductions of $\mathcal{ALCQI}$ to $\mathcal{ALCFI}$ such as the one presented in [5] because these work only on potentially infinite, tree-shaped models.

The central idea behind our reduction is to replace counting via qualified number restrictions with counting via concept names: to count up to a number n, we reserve concept names $B_0, \ldots, B_{\lceil \log(n) \rceil}$ representing the bits of the binary coding of numbers between 0 and n. For the actual counting, we can then use well-known (propositional logic) formulas that encode incrementation. We use a TBox involving auxiliary concept names and roles L_R to re-arrange R-neighbors as shown in Figure 1: except for the root, each node on the auxiliary L_R-path attached to x has precisely one R-neighbor. Ignoring the root for a second, this means that we can count via concept names along the auxiliary objects on L_R-paths. However, we cannot gather all original R-neighbors of x on the L_R-path since we only count up to the sum of numbers occurring in the input concept and TBox. Since an object may have more R-neighbors than this, these "unrestricted" R-neighbors are not re-arranged, but attached to the root as shown in the upper right part of Figure 1. To obtain an $\mathcal{ALCFI}$-concept from an $\mathcal{ALCQI}$-concept, we replace, in the input concept and TBox, number restrictions $(\bowtie\ n\, R\, C)$ with new concept names $A_{(\bowtie n\, R\, C)}$, and then use additional TBox axioms to ensure that, e.g., if x is an instance of $A_{(\leqslant n R.C)}$, then there are at most n R-successors in C along the L_R-path starting at x and no R-successor of x is in C. Moreover, we have to take care that the "auxiliary" objects introduced as intermediate points on L_R-paths and the "real" objects are distinguished properly.

The reduction with a proof of its correctness can be found in [11].

Theorem 2 *Finite satisfiability of $\mathcal{ALCQI}$-concepts w.r.t. TBoxes is* ExpTime*-complete if numbers are coded in binary.*

4 ABox Consistency

Finally, we extend the complexity bounds obtained in Sections 2 and 3 to a more general reasoning task: finite $\mathcal{ALCQI}$-ABox consistency. It is well known that (finite) $\mathcal{ALCQI}$-ABox consistency has important applications: whereas finite $\mathcal{ALCQI}$-concept satisfiability algorithms can be used to decide the consistency of conceptual database models and infer implicit IS-A relationships as described in the introduction, $\mathcal{ALCQI}$-ABox consistency can be used as the core component of algorithms deciding containment of conjunctive queries w.r.t. conceptual database models—a task that DLs have succesfully been used for and that calls for finite model reasoning [3, 9].

ABoxes are defined as usual, i.e., for O a countably infinite set of *object names*, an ABox is a finite set of *ABox assertions* of the form $a : C$ or $(a, b) : R$, where a and b are object names, C is a concept, and R a role. Semantics is defined in the usual way. Even though irrelevant for the result, we employ the *unique name assumption*, i.e., $a \neq b$ implies that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. In the following, we will polynomially reduce finite $\mathcal{ALCQI}$-ABox consistency to finite $\mathcal{ALCQI}$-concept satisfiability.

Let $\mathcal{A}$ be an ABox and $\mathcal{T}$ a TBox. For each object name a used in $\mathcal{A}$, $\mathsf{refl}_{\mathcal{A}}(a)$ denotes the set of role names R such that $\{(a,a) : R, (a,a) : R^{-}\} \cap \mathcal{A} \neq \emptyset$. For each object a and role $R \in \mathsf{rol}(\mathcal{A},\mathcal{T})$, $N_{\mathcal{A}}(a,R)$ denotes the set of object names b such that $b \neq a$ and $\{(a,b) : R,\ (b,a) : \mathsf{Inv}(R)\} \cap \mathcal{A} \neq \emptyset$.

We use $\mathsf{cl}(\mathcal{A},\mathcal{T})$ to denote the smallest set containing all sub-concepts of concepts appearing in $\mathcal{A}$ and $\mathcal{T}$ that is closed under $\dot{\neg}$. It can be easily shown that the cardinality of $\mathsf{cl}(\mathcal{A},\mathcal{T})$ is linear in the size of $\mathcal{A}$ and $\mathcal{T}$. Moreover, $\mathsf{rol}(\mathcal{A},\mathcal{T})$ denotes the set of all roles (i.e., role names or inverses of role names) used in $\mathcal{A}$ or $\mathcal{T}$.

A *type* T for an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ is defined as in Definition 1 with the only exception that $\mathsf{cl}(C_0,\mathcal{T})$ is replaced with $\mathsf{cl}(\mathcal{A},\mathcal{T})$. In what follows, we will sometimes identify types T with the conjunction $\bigsqcap_{C \in T} C$ and write, e.g., $d \in T^{\mathcal{I}}$ for $d \in (\bigsqcap_{C \in T} C)^{\mathcal{I}}$. It is easily seen that the number of types for an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ is exponential in the size of $\mathcal{A}$ and $\mathcal{T}$.

A central notion for the reduction of finite $\mathcal{ALCQI}$-ABox consistency to finite $\mathcal{ALCQI}$-concept satisfiability is that of a *reduction candidate*: a mapping t that associates a type $t(a)$ with each object name a occurring in $\mathcal{A}$ such that $a : C \in \mathcal{A}$ implies $C \in t(a)$. For each reduction candidate t, object name a, role $R \in \mathsf{rol}(\mathcal{A},\mathcal{T})$, and type $T \in \text{range}(t)$, we use $\#_t^{\mathcal{A}}(a,R,T)$ to denote the number of objects b such that $b \in N_{\mathcal{A}}(a,R)$ and $t(b) = T$. Then, for each object name a used in $\mathcal{A}$, we define its *t-reduction concept* $C_t^{\mathcal{A}}(a)$ as follows:

$$\begin{aligned} C_t^{\mathcal{A}}(a) \quad := \quad & t(a) \sqcap X \sqcap \bigsqcap_{R \in \mathsf{refl}_{\mathcal{A}}(a)} \exists R.(t(a) \sqcap X) \sqcap \\ & \bigsqcap_{R \in \mathsf{rol}(\mathcal{A},\mathcal{T})} \bigsqcap_{T \in \text{range}(t)} (\geqslant \#_t^{\mathcal{A}}(a,R,T)\ R\ (T \sqcap \neg X)), \end{aligned}$$

where X is a fresh concept name not used in $\mathcal{A}$ and $\mathcal{T}$. Finally, the reduction candidate t is called *realizable* iff, for every object a used in $\mathcal{A}$, the reduction concept $C_t^{\mathcal{A}}(a)$ is finitely satisfiable w.r.t. $\mathcal{T}$. The following lemma describes the relationship between ABoxes and reduction candidates:

Lemma 3 *Let $\mathcal{A}$ be an ABox and $\mathcal{T}$ a TBox. $\mathcal{A}$ is finitely consistent w.r.t. $\mathcal{T}$ iff there exists a realizable reduction candidate for $\mathcal{A}$ and $\mathcal{T}$.*

Since the number of types for $\mathcal{A}$ and $\mathcal{T}$ is exponential in the size of $\mathcal{A}$ and $\mathcal{T}$, and the number of object names used in $\mathcal{A}$ is linear in the size of $\mathcal{A}$, the number of reduction candidates for $\mathcal{A}$ and $\mathcal{T}$ is exponential in the size of $\mathcal{A}$ and $\mathcal{T}$. Thus, to decide finite consistency of $\mathcal{A}$ w.r.t. $\mathcal{T}$, we may simply enumerate all reduction candidates for $\mathcal{A}$ and $\mathcal{T}$ and check them for realizability: by Lemma 3, $\mathcal{A}$ is finitely consistent w.r.t. $\mathcal{T}$ iff we find a realizable reduction type. Since the size of the reduction concepts is clearly polynomial in the size of $\mathcal{A}$ and $\mathcal{T}$, by Theorem 2 the resulting algorithm can be executed in deterministic time exponential in $\mathcal{A}$ and $\mathcal{T}$.

Theorem 3 *Finite $\mathcal{ALCQI}$-ABox consistency w.r.t. TBoxes is* ExpTime-*complete if numbers are coded in binary.*

5 Discussion

In this paper, we have determined finite model reasoning in the description logic $\mathcal{ALCQI}$ to be ExpTime-complete. This shows that reasoning w.r.t. finite models is not harder than reasoning w.r.t. arbitrary models, which is known to be also ExpTime-complete [6, 5]. We hope that, ultimately, this research will lead to the development of finite model reasoning systems that behave equally well as existing DL reasoners doing reasoning w.r.t. arbitrary models. Note, however, that the current algorithm is *best-case* ExpTime since it constructs an exponentially large equation system. It can thus not be expected to have an acceptable runtime behaviour if implemented in a naive way. Nevertheless, we believe that the use of equation systems and linear programming is indispensable for finite model reasoning in $\mathcal{ALCQI}$. Thus, efforts to obtain efficient reasoners should perhaps concentrate on methods to avoid best-case exponentiality such as on-the-fly construction of equation systems. Finally, the reductions presented in [11] can also not be expected to exhibit an acceptable run-time behaviour and it would thus be interesting to try to replace them by more "direct" methods for dealing with binary coding of numbers and with ABoxes.

It is interesting to relate our results to formalisms from other areas where finite model reasoning is an issue. Take for example the full μ-calculus, i.e. the extension of $\mathcal{ALC}$ with fixpoints and inverse roles. For the $\nu\mu$-fragment of this logic, satisfiability in arbitrary models is ExpTime-complete [17] while the best known upper bound for finite satisfiability is 2-ExpTime [1]. The situation is similar for the two-variable fragment of first-order logic with counting quantifiers [8]. Whereas reasoning in arbitrary models is NExpTime-complete [12] (with unary coding), the best known upper bound for finite model reasoning is 2-NExpTime (this follows from results in [8]). As future work, it would be interesting to try and push our results to more expressive logics such as the ones mentioned above, in order to obtain tight complexity results for reasoning in finite models.

References

[1] M. Bojanczyk. Two-way alternating automata and finite models. In *Proc. of ICALP2002*, volume 2380 of *LNCS*. Springer-Verlag, 2002.

[2] D. Calvanese. Finite model reasoning in description logics. In *Proc. of KR-96*. Morgan Kaufmann, 1996.

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

[4] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[5] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms.* PhD thesis, Università degli Studi di Roma "La Sapienza", 1995.

[6] G. De Giacomo and M. Lenzerini. Tbox and Abox reasoning in expressive description logics. In *Proc. of KR-96*, pages 316–327. Morgan Kaufmann, 1996.

[7] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modelling. In *Working Notes of the ECAI2000 Workshop KRDB2000.* CEUR (`http://ceur-ws.org/`), 2000.

[8] E. Grädel, M. Otto, and E. Rosen. Two-Variable Logic with Counting is Decidable. In *Proceedings of Twelfth IEEE Symposium on Logic in Computer Science (LICS'97)*, 1997.

[9] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In A. Voronkov, editor, *Proc. of LPAR 2000*, number 1955 in LNAI. Springer-Verlag, 2000.

[10] O. Kupferman, U. Sattler, and M. Y. Vardi. The complexity of the graded mu-calculus. In *Proc. of CADE-18*, volume 2392 of *LNAI.* Springer-Verlag, 2002.

[11] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. LTCS-Report 02-05, Technical University Dresden, 2002. Available from http://lat.inf.tu-dresden.de/research/reports.html.

[12] L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM Journal on Computing*, 29(4):1083–1117, August 2000.

[13] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, Sydney, 1991.

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

[15] B. Thalheim. Fundamentals of cardinality constraints. In *Proceedings of the Conference on Entity-Relationship-Approaches 1992 (ER92)*, number 645 in LNCS, pages 7–23. Springer Verlag, 1992.

[16] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, RWTH Aachen, 2001. electronically available at `http://www.bth.rwth-aachen.de/ediss/ediss.html`.

[17] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. of ICALP'98*, volume 1443 of *LNCS*, pages 628–641. Springer-Verlag, 1998.