Mary Likes all Cats

Carsten Lutz and Ulrike Sattler
LuFG Theoretical Computer Science, RWTH Aachen, Germany
{lutz, sattler}@informatik.rwth-aachen.de

1 Introduction

Most Description Logics (DLs) provide universal value restrictions which allow to make universal statements about domain objects. For example, we can express that an object, say Mary, likes only cats by asserting that Mary is an instance of $\forall$likes.Cat. However, using universal value restrictions, we cannot express the symmetric fact that Mary likes *all* cats. This kind of expressiveness is provided by the DL-equivalent $\forall C.R$ of the modal logic "window" operator [7]: Asserting that Mary is an instance of $\forall$Cat.likes guarantees that each instance of Cat is associated to Mary via the role likes. Such a constructor was suggested in, e.g., [8] for knowledge representation.

The $\forall C.R$ constructor is closely related to *negation* of roles since $\forall \neg R.\neg C$ is equivalent to $\forall C.R$. Thus, negation of roles can also be used to express that Mary likes all cats. Moreover, role negation is of interest since (1) we can internalize general concept inclusion axioms $C \sqsubseteq D$ (see [1, 16]) using the concept $(\forall R.\neg C \sqcup D) \sqcap (\forall \neg R.\neg C \sqcup D)$, and (2) $\mathcal{ALC}$ looses the tree model property when extended with role negation which means that $\mathcal{ALC}$ with role negation offers expressivity which is not provided by most other Description Logics.

Although role negation is a boolean operation on roles and can be viewed as a "standard" DL constructor, there seem to exist only few DLs with role negation. One example of such a logic is given by Hustadt and Schmidt, who show decidability of $\mathcal{ALB}$ (an extension of $\mathcal{ALC}$ with role negation and several other constructors) using resolution techniques [12]. In the DLs $\mathcal{CINB}$ and $\mathcal{DLR}$ [6, 5], a "role negation" constructor is provided, which, however, has a non-standard semantics (i.e., it is role difference rather than role negation) and cannot be used to express $\forall C.R$. In the field of modal logics, several logics have been investigated which are notational variants of DLs with role negation. Examples are **K** extended with an inaccessibility modality [11] and PDL with negation of programs [15].

Surprisingly, to the best of our knowledge, the complexity of DLs with role negation or corresponding modal logics has never been investigated. In this paper, we catch up on this and determine the complexity of (i) the extension of $\mathcal{ALC}$ with role negation, (ii) the extension of $\mathcal{ALC}$ with transitive roles and role negation, and (iii) the extension of $\mathcal{ALC}$ with role negation and all possible combinations of boolean constructors of roles (where role negation may either be unrestricted or restricted to role names). This paper is accompanied by a technical report which contains all proofs and technical details [14]. Note that the technical report is not discussing $\mathcal{ALC}$ with role negation, but the multi-modal logic $K_m^\neg$, which is a notational variant.

2 Preliminaries

In this section, we define syntax and semantics of $\mathcal{ALC}^\neg$ and discuss some model- and complexity-theoretic properties of this Description Logic.

Definition 1 Let $\mathsf{N_C}$ be a set of *concept names* and $\mathsf{N_R}$ a set of *role names*. The set of $\mathcal{ALC}^\neg$*-roles* is $\mathsf{N_R} \cup \{\neg R \mid R \in \mathsf{N_R}\}$. The set of $\mathcal{ALC}^\neg$*-concepts* is the smallest set such that (i) every concept name is an $\mathcal{ALC}^\neg$-concept, and, (ii) if R is an $\mathcal{ALC}^\neg$-role and C and D are $\mathcal{ALC}^\neg$-concepts, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall R.C$, and $\exists R.C$ are $\mathcal{ALC}^\neg$-concepts.

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$ which maps every concept to a subset of $\Delta^\mathcal{I}$ and every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that

$$
\begin{aligned}
(\neg R)^\mathcal{I} &= \Delta^\mathcal{I} \times \Delta^\mathcal{I} \setminus R^\mathcal{I} & \neg C^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I}, \\
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I}, & (C \sqcup D)^\mathcal{I} &= C^\mathcal{I} \cup D^\mathcal{I}, \\
(\exists R.C)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} \mid \text{There exists a } y \in \Delta^\mathcal{I} \text{ with } (x,y) \in R^\mathcal{I} \text{ and } y \in C^\mathcal{I}\}, \\
(\forall R.C)^\mathcal{I} &= \{x \in \Delta^\mathcal{I} \mid \text{For all } y \in \Delta^\mathcal{I}, \text{ if } (x,y) \in R^\mathcal{I}, \text{ then } y \in C^\mathcal{I}\}.
\end{aligned}
$$

A concept C is called *satisfiable* iff there is some interpretation $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$. Such an interpretation is called a *model* of C. A concept D *subsumes* a concept C (written $C \sqsubseteq D$) iff $C^\mathcal{I} \subseteq D^\mathcal{I}$ holds for each interpretation $\mathcal{I}$. Two concepts are said to be equivalent (written $C \equiv D$) if they mutually subsume each other.

It is well-known that, in the presence of concept negation, (un)satisfiability and subsumption can be mutually reduced in constant time, i.e., $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable and a concept C is satisfiable iff not $C \sqsubseteq A \sqcap \neg A$, where A is a concept name. Hence, all complexity results for satisfiability that are obtained in this paper do also apply to subsumption. The semantics of the $\forall C.R$ constructor mentioned in the introduction is

$$(\forall C.R)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \text{For all } y \in \Delta^\mathcal{I},\ y \in C^\mathcal{I} \text{ implies } (x,y) \in R^\mathcal{I}\}.$$

It is easy to see that $\forall C.R \equiv \forall\neg R.\neg C$, and, hence, reasoning in $\mathcal{ALC}$ extended with the $\forall C.R$ constructor can be reduced to reasoning with $\mathcal{ALC}^\neg$.

It is not hard to show that the satisfiability of $\mathcal{ALC}^\neg$-concepts is ExpTime-hard and in NExpTime: (i) The satisfiability of formulae of the modal logic $\mathbf{K}^u$, i.e., uni-modal $\mathbf{K}$ enriched with the universal modality is known to be ExpTime-hard [17], and can be reduced to the satisfiability of $\mathcal{ALC}^\neg$-concepts. We use the common translation t from $\mathbf{K}$-formulae to $\mathcal{ALC}$-concepts (see, e.g., [16]) and, additionally, replace

- every occurrence of $\Box_u.\varphi$ by $\forall R.t(\varphi) \wedge \forall\neg R.t(\varphi)$ and
- every occurrence of $\Diamond_u.\varphi$ by $\exists R.t(\varphi) \vee \exists\neg R.t(\varphi)$

where $\Box_u$ and $\Diamond_u$ denote the universal modality and R is an arbitrary role name. Now this translation may clearly lead to an exponential blowup in formula/concept size if $\Box_u$ or $\Diamond_u$ are nested in the input concept. However, in the proof of ExpTime-hardness of $\mathbf{K}^u$ [17], $\Box_u$ occurs only once. In this case, the translation is linear, and, thus, satisfiability of $\mathcal{ALC}^\neg$-concepts is ExpTime-hard; (ii) it is well-known that there exists a linear, satisfiability-preserving translation from $\mathcal{ALC}^\neg$ into L^2, the 2-variable fragment of first order logic [4]. Since L^2 is decidable in NExpTime [9], this implies that satisfiability of $\mathcal{ALC}^\neg$-concepts is also in NExpTime. However, these two complexity bounds are obviously not tight. One main contribution of this paper is to give an ExpTime-algorithm for the satisfiability of $\mathcal{ALC}^\neg$-concepts, thus tightening the complexity bounds.

For devising a satisfiability algorithm, it is interesting to know what kind of models need to be considered. In [7], it is proved that the modal logic counterpart of $\mathcal{ALC}^\neg$ has the finite model property. $\mathcal{ALC}^\neg$ does not have the tree model property since, e.g., the concept $A \sqcap \forall\neg R.\neg A$ has no tree model: It is easy to see that, for any $x \in (A \sqcap \forall\neg R.\neg A)^\mathcal{I}$, we must have $(x, x) \in R^\mathcal{I}$. However, we will show that there exists a one-to-one correspondence between models and so-called Hintikka-trees which we then use to decide satisfiability (and thus subsumption) of $\mathcal{ALC}^\neg$-concepts. We do this by building, for each $\mathcal{ALC}^\neg$-concept C, a Büchi-automaton $\mathcal{A}_C$ which accepts the empty (tree-)language iff C is unsatisfiable. Hence we introduce trees, Büchi-automata, and the language they accept here.

Definition 2 Let M be a set and $k \geq 1$. A *k-ary M-tree* is a mapping $T : \{1, \dots, k\}^* \mapsto M$ that labels each node $\alpha \in \{1, \dots, k\}^*$ with $T(\alpha) \in M$. Intuitively, the node αi is the i-th child of α. We use ϵ to denote the empty word (corresponding to the root of the tree). A *path* in a k-ary M-tree is an infinite word over the alphabet $\{1, \dots, k\}$.

A *Büchi-automaton* $\mathcal{A} = (Q, M, I, \Delta, F)$ for k-ary M-trees is defined by a set Q of states, an alphabet M, a subset $I \subseteq Q$ of initial states, a transition relation $\Delta \subseteq Q \times M \times Q^k$, and a subset $F \subseteq Q$ of accepting states.

A *run* of $\mathcal{A}$ on an M-tree T is a mapping $r : \{1, \ldots, k\}^* \mapsto Q$ with

$$(r(\alpha), T(\alpha), r(\alpha 1), \ldots, r(\alpha k)) \in \Delta$$

for each $\alpha \in \{1, \ldots, k\}^*$. A run r on T is *accepting* iff, for each path $i_1 i_2 \cdots$ in T, the set $\{i_j \mid r(i_1 \cdots i_j) \in F\}$ is infinite.

A Büchi-automaton accepts all those M-trees for which an accepting run exists, i.e., the language $\mathfrak{L}(\mathcal{A})$ of M-trees accepted by $\mathcal{A}$ is

$$\mathfrak{L}(\mathcal{A}) = \{T \mid \text{There is an accepting run from } \mathcal{A} \text{ on } T\}.$$

In [18], it is proved that the emptiness problem for Büchi-automata, i.e., the problem to decide whether the language $\mathfrak{L}(\mathcal{A})$ accepted by a given Büchi-automaton $\mathcal{A}$ is empty, is decidable in polynomial time.

3 $\mathcal{ALC}^{\neg}$ is ExpTime-complete

We show that satisfiability of $\mathcal{ALC}^{\neg}$-concepts is decidable in exponential time. For this purpose, we first abstract from models of $\mathcal{ALC}^{\neg}$-concepts to Hintikka-trees, and then show how to construct a Büchi-automaton that accepts exactly Hintikka-trees.

Notation: We assume all concepts to be in negation normal form (NNF), i.e., negation occurs only in front of concept names and role names. Each concept can easily be transformed into an equivalent one in NNF by pushing negation inwards, employing de Morgan's law and the duality between $\forall R.C$ and $\exists R.C$. We use $\bar{C}$ to denote the NNF of $\neg C$.

Since we treat negated and unnegated roles symmetrically, we introduce the notion

$$\exists \bar{R}.C = \begin{cases} \exists \neg R.C & \text{if } R \text{ is atomic,} \\ \exists S.C & \text{if } R = \neg S \text{ for some atomic role } S \end{cases}$$

and analogously $\forall \bar{R}.C$. Let $\mathsf{cl}(C)$ denote the set of C's subconcepts and NNFs of their negations, i.e.,

$$\mathsf{cl}(C) := \{D \mid \begin{array}{l} D \text{ is a subformula of } C \text{ or} \\ D = \bar{E} \text{ for a subformula } E \text{ of } C\}. \end{array}$$

We assume that existential concepts $\exists R.D$ in $\mathsf{cl}(C)$ are linearly ordered, and that $\mathcal{E}(i)$ yields the i-th existential concept in $\mathsf{cl}(C)$.

Definition 3 (Hintikka-set and Hintikka-tree) Let C be an $\mathcal{ALC}^{\neg}$-concept and k the number of existential concepts in $\mathsf{cl}(C)$. A set $\Psi \subseteq \mathsf{cl}(C)$ is a *Hintikka-set* iff it satisfies the following conditions:

(H1) if $C_1 \sqcap C_2 \in \Psi$, then $\{C_1, C_2\} \subseteq \Psi$,

(H2) if $C_1 \sqcup C_2 \in \Psi$, then $\{C_1, C_2\} \cap \Psi \neq \emptyset$,

(H3) $\{C, \bar{C}\} \not\subseteq \Psi$ for all $\mathcal{ALC}^{\neg}$-concepts C.

A k-ary $2^{\mathsf{cl}(C)}$-tree T is a *Hintikka-tree for* C iff $T(\alpha)$ is a Hintikka-set for each node α in T, and T satisfies, for all nodes $\alpha, \beta \in \{1, \ldots, k\}^*$, the following conditions:

(H4) $C \in T(\epsilon)$,

(H5) if $\{\exists R.D, \forall R.E_1, \ldots, \forall R.E_m\} \subseteq T(\alpha)$ and $\mathcal{E}(i) = \exists R.D$, then $\{D, E_1, \ldots, E_m\} \subseteq T(\alpha i)$

(H6) if $\mathcal{E}(i) \notin T(\alpha)$, then $T(\alpha i) = \emptyset$,

(H7) if $\forall R.D \in T(\alpha)$, then $D \in T(\beta)$, $\bar{D} \in T(\beta)$, or $T(\beta) = \emptyset$,

(H8) if $\{\forall R.D, \forall \bar{R}.E\} \subseteq T(\alpha)$ and $\bar{D} \in T(\beta)$, then $E \in T(\beta)$.

In **(H5)**, **(H7)**, and **(H8)**, R denotes role names and also negations of roles. Obviously, the empty set is also a Hintikka-set. The following lemma shows the connection between models and Hintikka trees.

Lemma 4 An $\mathcal{ALC}^{\neg}$-concept C is satisfiable iff C has a Hintikka-tree.

Proof: Let C be an $\mathcal{ALC}^{\neg}$-concept using role names $R_1, \ldots, R_m$ and let there be k existential concepts in $\mathsf{cl}(C)$.

"$\Leftarrow$" Let T be a Hintikka-tree for C. We define an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows:

$$
\begin{aligned}
\Delta^{\mathcal{I}} &= \{\alpha \in \{1, \ldots, k\}^* \mid T(\alpha) \neq \emptyset\} \\
A^{\mathcal{I}} &= \{\alpha \mid A \in T(\alpha)\} \text{ for all concept names } A \\
R^{\mathcal{I}} &= \{(\alpha, \beta) \mid \beta = \alpha j \text{ and } \mathcal{E}(j) = \exists R.D \in T(\alpha)\} \cup \\
&\quad\ \{(\alpha, \beta) \mid \forall \neg R.D \in T(\alpha) \text{ and } \bar{D} \in T(\beta)\} \text{ for all role names } R
\end{aligned}
$$

To show that there exists an $x \in \Delta^{\mathcal{I}}$ such that $x \in C^{\mathcal{I}}$, we first prove the following claim:

Claim: $D \in T(\alpha)$ implies $\alpha \in D^{\mathcal{I}}$ for all $\alpha \in \Delta^{\mathcal{I}}$ and $D \in \mathsf{cl}(C)$.

The claim is proved by induction over the structure of D. The induction start, i.e., the case that D is a concept name, is an immediate consequence of the definition of $\mathcal{I}$. For the induction step, we make a case distinction according to the topmost constructor in D. Assume $D \in T(\alpha)$.

- $D = \neg E$. Since C is in NNF (by the definition of Hintikka-sets and cl), E is a concept name. By definition of $\mathcal{I}$ and since $T(\alpha)$ is a Hintikka-set and thus satisfies **(H3)**, we have $\alpha \in (\neg E)^{\mathcal{I}}$.

- $D = C_1 \sqcap C_2$ or $D = C_1 \sqcup C_2$. Straightforward by **(H1)** and **(H2)** of Hintikka-sets and by induction hypothesis.

- $D = \exists R.E = \mathcal{E}(j)$ for a j with $1 \leq j \leq k$. First assume that R is a role. By definition of $R^{\mathcal{I}}$, we have $(\alpha, \alpha j) \in R^{\mathcal{I}}$. By **(H5)**, $\exists R.E \in T(\alpha)$ implies $E \in T(\alpha j)$. By induction, $\alpha j \in E^{\mathcal{I}}$, and, hence, $\alpha \in (\exists R.E)^{\mathcal{I}}$.

  Now assume that $R = \neg S$ for a role name S. We show that $(\alpha, \alpha j) \notin S^{\mathcal{I}}$, for, if we have done this, $\alpha \in (\exists R.E)^{\mathcal{I}}$ follows as in the previous case (where R is a concept name). Assume to the contrary that $(\alpha, \alpha j) \in S^{\mathcal{I}}$. Then, by definition of $S^{\mathcal{I}}$, we have either

  1. $\mathcal{E}(j) = \exists S.E' \in T(\alpha)$, or
  2. $\forall \neg S.E' \in T(\alpha)$ and $\bar{E}' \in T(\alpha j)$

  where $E' \in \mathsf{cl}(\varphi)$. In the first case, we have a contradiction to the assumption $\mathcal{E}(j) = \exists \neg S.E$. In the second case, we have $\{\exists \neg S.E, \forall \neg S.E'\} \subseteq T(\alpha)$ which, by **(H5)**, implies $\{E, E'\} \subseteq T(\alpha j)$. Since we also know that $\bar{E}' \in T(\alpha j)$, we obtain a contradiction to **(H3)** of Hintikka-sets and conclude that $(\alpha, \alpha j) \notin S^{\mathcal{I}}$.

- $D = \forall R.E$. First assume that R is a role name and fix a β such that $(\alpha, \beta) \in R^{\mathcal{I}}$. By definition of $R^{\mathcal{I}}$, we have to distinguish two cases:

  1. $\beta = \alpha j$ and $\mathcal{E}(j) = \exists R.E' \in T(\alpha)$, or
  2. $\forall \neg R.E' \in T(\alpha)$ and $\bar{E}' \in T(\beta)$

  In the first case, we have $\{\exists R.E', \forall R.E\} \subseteq T(\alpha)$ which, by **(H5)**, implies $\{E, E'\} \subseteq T(\alpha j)$. By induction, we obtain $\beta \in E^{\mathcal{I}}$. In the second case, we have $\{\forall R.E, \forall \neg R.E'\} \subseteq T(\alpha)$ and $\bar{E}' \in T(\beta)$. By **(H8)**, we have $E \in T(\beta)$, and, by induction, $\beta \in E^{\mathcal{I}}$. Since this holds independently of the choice of β, we conclude $\alpha \in (\forall R.E)^{\mathcal{I}}$.

  Now assume that $R = \neg S$ for a role name S. Fix a β such that $(\alpha, \beta) \notin S^{\mathcal{I}}$. Since $\beta \in \Delta^{\mathcal{I}}$, we have that $T(\beta) \neq \emptyset$. Hence, by **(H7)**, we have $E \in T(\beta)$ or $\bar{E} \in T(\beta)$. However, $\bar{E} \in T(\beta)$ would imply $(\alpha, \beta) \in S^{\mathcal{I}}$ by definition of $S^{\mathcal{I}}$, which is a contradiction to our choice of β. Hence we deduce $E \in T(\beta)$. By induction, we obtain $\beta \in E^{\mathcal{I}}$. Since this holds independently of the choice of β, we conclude $\alpha \in (\forall \neg S.E)^{\mathcal{I}}$.

This completes the proof of the claim. Since $C \in T(\epsilon)$ by **(H4)**, it is an immediate consequence of the claim that $\mathcal{I}$ is a model of C.

"$\Rightarrow$" Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of C, i.e., there exists an $x_0 \in \Delta^{\mathcal{I}}$ with $x_0 \in C^{\mathcal{I}}$. We define a Hintikka-tree for C (i.e., a Hintikka-set label $T(\alpha)$ for each $\alpha \in \{1,\ldots,k\}^*$). To do this, we inductively define a mapping τ from $\{1,\ldots,k\}^*$ to $\Delta^\mathcal{I} \cup \{\bot\}$ in such a way that

$$T(\alpha) = \begin{cases} \{D \in \mathsf{cl}(C) \mid \tau(\alpha) \in D^\mathcal{I}\} & \text{if } \tau(\alpha) \neq \bot \\ \emptyset & \text{otherwise} \end{cases} \qquad (*)$$

For the induction start, set

$$\begin{array}{rcl} \tau(\epsilon) & := & x_0 \\ T(\epsilon) & := & \{D \in \mathsf{cl}(C) \mid x_0 \in D^\mathcal{I}\} \end{array}$$

Now for the induction step. Let $\alpha \in \{1,\ldots,k\}^*$ such that $\tau(\alpha)$ is already defined, and let $i \in \{1,\ldots,k\}$. We make a case distinction as follows:

1. $\tau(\alpha) \neq \bot$ and $\mathcal{E}(i) = \exists R.D \in T(\alpha)$. By $(*)$, we have $\tau(\alpha) \in (\exists R.D)^\mathcal{I}$ which implies the existence of a domain object $x \in \Delta^\mathcal{I}$ such that $(\tau(\alpha), x) \in R^\mathcal{I}$ and $x \in D^\mathcal{I}$. Choose such an x and define $\tau(\alpha i) := x$ and $T(\alpha i) := \{E \in \mathsf{cl}(C) \mid x \in E^\mathcal{I}\}$.

2. if α, i do not match the above case, set $\tau(\alpha i) = \bot$ and $T(\alpha i) = \emptyset$.

By definition, T and τ satisfy $(*)$. We need to prove that the k-ary $2^{\mathsf{cl}(C)}$-tree T just defined is a Hintikka-tree for C. From the semantics of $\mathcal{ALC}^\neg$ and the definition of cl, it is clear that $T(\alpha)$ is a Hintikka-set for each $\alpha \in \{1,\ldots,k\}^*$. Hence, it remains to show that T satisfies **(H4)** to **(H8)**.

(H4) Satisfied by definition of T (see induction start).

(H5) Let $\{\exists R.D, \forall R.E_1, \ldots, \forall R.E_m\} \subseteq T(\alpha)$ and $\mathcal{E}(i) = \exists R.D$. By $(*)$, we have $\tau(\alpha) \neq \bot$ and $\tau(\alpha) \in (\exists R.D \sqcap \forall R.E_1 \sqcap \cdots \sqcap \forall R.E_m)^\mathcal{I}$. By definition of τ (induction step, first case), we have $\tau(\alpha i) = x$ for some $x \in \Delta^\mathcal{I}$, with $(\tau(\alpha), x) \in R^\mathcal{I}$, and $x \in D^\mathcal{I}$. Moreover, the semantics of $\mathcal{ALC}^\neg$ implies $x \in (E_1 \sqcap \cdots \sqcap E_m)^\mathcal{I}$, and, by $(*)$, we thus have $\{D, E_1, \ldots, E_m\} \subseteq T(\alpha i)$.

(H6) Satisfied by definition of T (see induction step, second case).

(H7) Let $\forall R.D \in T(\alpha)$ and fix a $\beta \in \{1,\ldots,k\}^*$. If $\tau(\beta) = \bot$, then we have $T(\beta) = \emptyset$ by $(*)$ and **(H7)** is satisfied. If $\tau(\beta) \neq \bot$, then $\tau(\beta) \in \Delta^\mathcal{I}$ and we have either $\tau(\beta) \in D^\mathcal{I}$ or $\tau(\beta) \in \bar{D}^\mathcal{I}$. Again, $(*)$ implies that **(H7)** is satisfied.

(H8) Assume $\{\forall R.E, \forall \bar{R}.D\} \subseteq T(\alpha)$ and $\bar{E} \in T(\beta)$. By $(*)$, we have $\tau(\alpha) \in (\forall R.E \sqcap \forall \bar{R}.D)^\mathcal{I}$ and $\tau(\beta) \in \bar{E}^\mathcal{I}$. This implies $(\tau(\alpha), \tau(\beta)) \in \bar{R}^\mathcal{I}$ since

1. we have either $(\tau(\alpha), \tau(\beta)) \in R^\mathcal{I}$ or $(\tau(\alpha), \tau(\beta)) \in \bar{R}^\mathcal{I}$ and
2. $(\tau(\alpha), \tau(\beta)) \in R^\mathcal{I}$ is impossible since $\tau(\alpha) \in (\forall R.E)^\mathcal{I}$ and $\tau(\beta) \in \bar{E}^\mathcal{I}$.

Hence, due to the semantics of $\mathcal{ALC}^\neg$, we have $\tau(\beta) \in D^\mathcal{I}$, which, by $(*)$, implies $D \in T(\beta)$.

❏

The lemma shows that Hintikka-trees are appropriate abstractions of models of $\mathcal{ALC}^\neg$-concepts. Hintikka-trees enjoy the nice property that they are trees, and we can thus define, for an $\mathcal{ALC}^\neg$-concept C, a tree-automaton $\mathcal{A}_C$ that accepts exactly the Hintikka-trees for C.

Definition 5 For an $\mathcal{ALC}^\neg$-concept C with k existential concepts in $\mathsf{cl}(C)$, the Büchi-automaton $\mathcal{A}_C = (Q, 2^{\mathsf{cl}(C)}, \Delta, I, Q)$ is defined as follows:

- $Q \subseteq \{\Psi \in 2^{\mathsf{cl}(C)} \mid \Psi \text{ is a Hintikka-set}\} \times 2^P \times 2^S$ where

$$\begin{array}{rcl} P & = & \{\{\forall R.D, \forall \bar{R}.E\} \mid \forall R.D, \forall \bar{R}.E \in \mathsf{cl}(C)\}, \\ S & = & \{\forall R.D \mid \forall R.D \in \mathsf{cl}(C)\}, \end{array}$$

  and each $(\Psi, p, s) \in Q$ satisfies

  1. if $\{\forall R.D, \forall \bar{R}.E\} \in p$ and $\bar{D} \in \Psi$, then $E \in \Psi$,
  2. if $\forall R.D \in s$, then $\Psi = \emptyset$ or $\{D, \bar{D}\} \cap \Psi \neq \emptyset$,
  3. if $\forall R.D \in \Psi$, then $\forall R.D \in s$, and
  4. if $\{\forall R.D, \forall \bar{R}.E\} \subseteq \Psi$, then $\{\forall R.D, \forall \bar{R}.E\} \in p$.

- $I = \{(\Psi, p, s) \mid C \in \Psi\}$.

- $((\Psi, p, s), \Psi', (\Psi_1, p_1, s_1), \ldots, (\Psi_k, p_k, s_k)) \in \Delta$ iff

  $\Psi = \Psi', \quad p_i = p, \quad s_i = s$ for all $1 \leq i \leq k$, and
  if $\mathcal{E}(i) = \exists R.D \in \Psi$, then $D \in \Psi_i$ and $E \in \Psi_i$ for each $\forall R.E \in \Psi$ and
  if $\mathcal{E}(i) = \exists R.D \notin \Psi$, then $\Psi_i = \emptyset$.

As a consequence of the following lemma and Lemma 4, we can reduce satisfiability of $\mathcal{ALC}^\neg$-concepts to the emptiness problem for Büchi-automata.

Lemma 6 T is a Hintikka-tree for an $\mathcal{ALC}^\neg$-concept C iff $T \in \mathcal{L}(\mathcal{A}_C)$.

Proof sketch: Let C be an $\mathcal{ALC}^\neg$-concept and k, $\mathcal{A}_C$ as in Definition 5.

"⇒" Let T be Hintikka-tree for C. We prove that there is an accepting run of $\mathcal{A}_C$ on T. First, define

$$\begin{array}{rcl} p & := & \{\{\forall R.D, \forall \bar{R}.E\} \mid \text{ There is a node } \alpha \text{ in } T \text{ with } \{\forall R.D, \forall \bar{R}.E\} \subseteq T(\alpha)\} \\ s & := & \{\forall R.D \mid \text{ There is a node } \alpha \text{ in } T \text{ with } \forall R.D \in T(\alpha)\} \end{array}$$

We can show that $r(\alpha) = (T(\alpha), p, s)$ is an accepting run of $\mathcal{A}_C$ on T. By definition, r is defined for each $\alpha \in \{1, \ldots, k\}^*$. Furthermore, using the Properties of Hintikka trees, it is straightforward to show that the following three conditions hold:

1. $r(\alpha) \in Q$,

2. $r(\epsilon) \in I$, and

3. $((T(\alpha), p, s), T(\alpha), (T(\alpha 1), p, s), \ldots, (T(\alpha_k), p, s)) \in \Delta$.

"$\Leftarrow$" Let $T \in \mathcal{L}(\mathcal{A}_C)$ and r be an accepting run of $\mathcal{A}_C$ on T. It can be proved that T is a Hintikka-tree for C. By definition of $\mathcal{A}_C$, T is a k-ary $2^{\mathsf{cl}(C)}$-tree, and $r(\alpha) = (\Psi_\alpha, p_\alpha, s_\alpha)$ implies $\Psi_\alpha = T(\alpha)$ by definition of Δ. Hence, by definition of Q, each node in T is labelled with a Hintikka-set. It remains to prove that T satisfied **(H4)** to **(H8)**. This is straightforward using the definitions of Büchi-automata, $\mathcal{A}_C$, and accepting runs. ❑

What is the size of Büchi-automata $\mathcal{A}_C = (Q_C, M_C, I_C, \Delta_C, F_C)$? Obviously, the cardinality of $\mathsf{cl}(C)$ is linear in the length of C. Hence, by definition of $\mathcal{A}_C$, the cardinality of Q_C and M_C are exponential in the length of C. Again by definition of $\mathcal{A}_C$, this implies that the cardinalities of I_C, Δ_C, and F_C are also exponential in the length of C. Hence the size of $\mathcal{A}_C$ is exponential in the length of C. This fact together with Lemma 4, Lemma 6, and the fact that emptiness of the language accepted by a Büchi-automaton $\mathcal{A}_C$ can be tested in time polynomial in the size of $\mathcal{A}_C$ [18], we have that satisfiability of $\mathcal{ALC}^\neg$-concepts is in ExpTime. We already noted in Section 2 that satisfiability of $\mathcal{ALC}^\neg$-concepts is also ExpTime-hard. Hence, we obtain the following theorem:

Theorem 7 Satisfiability of $\mathcal{ALC}^\neg$-concepts is ExpTime-complete.

Many Description Logics provide roles with certain properties, the most prominent property being transitivity. It is hence an interesting question whether the obtained result generalises to $\mathcal{ALC}^\neg$ with transitive roles. More precisely, the DL $\mathcal{S}^\neg$ (generalizing the DL $\mathcal{S}$ defined in [10]) is obtained from $\mathcal{ALC}^\neg$ as follows: Assume that N_t is a subset of the set of role names N_R. Elements of N_t are called *transitive roles*. An $\mathcal{S}^\neg$ interpretation is an $\mathcal{ALC}^\neg$ interpretation which interprets each $R \in \mathsf{N}_t$ as a transitive relation $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. To define Hintikka trees for $\mathcal{S}^\neg$, we need to introduce counterparts for **(H5)** and **(H8)** which deal with transitive roles. These additional two properties need then to be reflected in the definition of the corresponding Büchi-automata. Using these modified definitions, the following result can be proved analogously to the proof of Theorem 7 (see [14] for details).

Theorem 8 Satisfiability of $\mathcal{S}^\neg$-concepts is ExpTime-complete.

4 Adding Intersection and Union of Roles

In this section, we investigate the complexity of adding the standard DL role constructors intersection and union of roles to the logic $\mathcal{ALC}^\neg$. In doing this, one has the choice to either restrict the applicability of negation to atomic roles or allowing for full negation w.r.t. roles. In the latter case, adding union is obviously equivalent to adding intersection or both. We start with the smallest extension, i.e., we add either intersection or union of roles while restricting negation to role names.

Definition 9 An $\mathcal{ALC}^{(\neg),\sqcup}$-concept is an $\mathcal{ALC}^{\neg}$-concept which, additionally, allows for roles of the form $R_1 \sqcup \cdots \sqcup R_k$, where each R_i is an $\mathcal{ALC}^{\neg}$-role. An $\mathcal{ALC}^{(\neg),\sqcap}$-concept is an $\mathcal{ALC}^{\neg}$-concept which, additionally, allows for roles of the form $R_1 \sqcap \cdots \sqcap R_k$, where each R_i is an $\mathcal{ALC}^{\neg}$-role. The semantics of the new roles is defined as follows:

$$\begin{array}{rcl}(R_1 \sqcup \cdots \sqcup R_k)^{\mathcal{I}} & = & R_1^{\mathcal{I}} \cup \cdots \cup R_k^{\mathcal{I}} \\ (R_1 \sqcap \cdots \sqcap R_k)^{\mathcal{I}} & = & R_1^{\mathcal{I}} \cap \cdots \cap R_k^{\mathcal{I}}\end{array}$$

Let us first investigate the logic $\mathcal{ALC}^{(\neg),\sqcup}$. It is not hard to see that

$$\begin{array}{rcl}\forall R_1 \sqcup \cdots \sqcup R_k.C & \equiv & \forall R_1.C \sqcap \cdots \sqcap \forall R_k.C \text{ and} \\ \exists R_1 \sqcup \cdots \sqcup R_k.C & \equiv & \exists R_1.C \sqcup \cdots \sqcup \forall R_k.C,\end{array}$$

i.e., satisfiability of $\mathcal{ALC}^{(\neg),\sqcup}$-concepts can be linearly reduced to the satisfiability of $\mathcal{ALC}^{\neg}$-concepts which gives the following result:

Theorem 10 Satisfiability of $\mathcal{ALC}^{(\neg),\sqcup}$-concepts is ExpTime-complete.

Next, we will show that the satisfiability of $\mathcal{ALC}^{(\neg),\sqcap}$-concepts is NExpTime-hard. The proof is given by a reduction of a NExpTime-complete variant of the well-known, undecidable domino problem.

A domino problem [2, 13] is given by a finite set of *domino types*. All domino types are of the same size, each type has a quadratic shape and colored edges. Of each type, an unlimited number of dominoe is available. The problem in the original domino problem is to arrange these dominoe to cover the plane without holes or overlapping, such that adjacent dominoe have identical colors on their touching edges (rotation of the dominoe is not allowed). In the NExpTime-complete variant of the domino problem that we use, the task is not to tile the whole plane, but to tile a $2^{n+1} \times 2^{n+1}$-torus, i.e., a $2^{n+1} \times 2^{n+1}$-rectangle whose edges are "glued" together. See, e.g., [2, 13] for undecidable versions of the domino problem and [3] for bounded variants.

Definition 11 Let $\mathcal{D} = (D, H, V)$ be a *domino system*, where D is a finite set of *domino types* and $H, V \subseteq D \times D$ represent the horizontal and vertical matching conditions. For $s, t \in \mathbb{N}$, let $U(s,t)$ be the torus $\mathbb{Z}_s \times \mathbb{Z}_t$, where $\mathbb{Z}_n$ denotes the set $\{0, \ldots, n-1\}$. Let $a = a_0, \ldots, a_{n-1}$ be an n-tuple of dominoe (with $n \leq s$). We say that $\mathcal{D}$ *tiles* $U(s,t)$ *with initial condition* a 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 $(d, d') \in H$
- if $\tau(x, y) = d$ and $\tau(x, y \oplus_t 1) = d'$, then $(d, d') \in V$
- $\tau(i, 0) = a_i$ for $0 \leq i < n$.

where $\oplus_n$ denotes addition modulo n. Such a mapping τ is called a *solution* for $\mathcal{D}$ w.r.t. a.

The following is a consequence of Theorem 6.1.2 in [3] (see also [14]).

Theorem 12 There exists a domino system $\mathcal{D}$ such that the following is a NExpTime-hard problem: Given an initial condition $a = a_0 \cdots a_{n-1}$ of length n, does $\mathcal{D}$ tile the torus $U(2^{n+1}, 2^{n+1})$ with initial condition a?

We reduce the NExpTime-complete variant of the domino problem from Theorem 12 to the satisfiability of $\mathcal{ALC}^{(\neg),\sqcap}$-concepts. Given a domino system $\mathcal{D} = (D, H, V)$ and an initial condition $a = a_0, \ldots, a_{n-1}$, we define a reduction concept $C_{(\mathcal{D},a)}$ such that $C_{(\mathcal{D},a)}$ is satisfiable iff $\mathcal{D}$ tiles the torus $U(2^{n+1}, 2^{n+1})$ with initial condition a. The reduction concept $C_{(\mathcal{D},a)}$ can be found in Figure 2. In this figure, $\forall C$ is an abbreviation for $\forall R.C \wedge \forall\neg R.C$, where R is an arbitrary role name. Obviously, in each model $\mathcal{I}$ of $\forall C$, we have $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. Furthermore, we write $\forall R^n.C$ to denote $\underbrace{\forall R \ldots \forall R}_{n \text{ times}}.C$.

The strategy of the reduction is to define the reduction concept $C_{(\mathcal{D},a)}$ such that, for every model $\mathcal{I}$ of $C_{(\mathcal{D},a)}$ with domain $\Delta^{\mathcal{I}}$,

1. there exists a concept name A_d for every domino type $d \in D$ such that each $x \in \Delta^{\mathcal{I}}$ is in the extension of A_d for exactly one $d \in D$ (first line of *Tiling*),
2. for each point (i, j) in the torus $U(2^{n+1}, 2^{n+1})$, there exists a corresponding set of domain objects $\{x_1, \ldots, x_k\} \subseteq \Delta^{\mathcal{I}}$ with $k \geq 1$ and a $d \in D$ such that all $x_1, \ldots, x_k$ are in the extension of A_d ($Count_x$, $Count_y$, *Stable*, and *Unique* concepts)
3. the horizontal and vertical conditions V and H are satisfied w.r.t. sets of worlds representing points in the plane (second and third line of *Tiling*), and
4. the initial condition is satisfied (*Init*).

Properties 1, 3, and 4 are enforced in a standard way using $\mathcal{ALC}$ concepts together with the $\forall C$ constructor introduced above. Property 2, however, needs some explanation. Usually, domino-reductions axiomatize a "grid" in order to capture the structure of the torus. As Property 2 indicates, we employ a different strategy: Each world in each model of $C_{(\mathcal{D},a)}$ corresponds to a point (i, j) in

	atomic negation	full negation
$-$	ExpTime-complete	
$\sqcup$	ExpTime-complete	NExpTime-complete
$\sqcap$	NExpTime-complete	NExpTime-complete
$\sqcap$ and $\sqcup$	NExpTime-complete	NExpTime-complete

Figure 1: Complexity of $\mathcal{ALC}$ extended with various role constructors.

the torus. The number i is binarily encoded by the concept names $X_0, \ldots, X_n$ while the number j is encoded by the concept names $Y_0, \ldots, Y_n$. We use standard binary incrementation modulo 2^{n+1} to ensure that, for every domain object x corresponding to a position (i, j), there exists an object y_1 such that y_1 corresponds to $(i \oplus_{2^{n+1}} 1, j)$ and $(x, y_1) \in R_x^{\mathcal{I}}$, and an object y_2 such that y_2 corresponds to $(i, j \oplus_{2^{n+1}} 1)$ and $(y, y_2) \in R_y^{\mathcal{I}}$. The $Count_x$ and $Count_y$ concepts encode the incrementation of the one dimension while the *Stable* concept ensures that the other dimension does not change. It remains to guarantee that every two domain objects corresponding to the same position are labeled with the same domino. This task is accomplished by the *Unique* concept which is the only one to use negation and conjunction of roles (not nested, though). In order to understand the *Unique* concept, it may be helpful to read subconcepts of the form $\forall \neg R. \neg C$ as $\forall C.R$.

Proposition 13 A domino system $\mathcal{D}$ tiles the torus $U(2^{n+1}, 2^{n+1})$ with initial condition $a = a_0, \ldots, a_{n-1}$ iff $C_{(\mathcal{D},a)}$ is satisfiable.

Together with Theorem 12, we obtain the desired result.

Theorem 14 Satisfiability of $\mathcal{ALC}^{(\neg),\sqcap}$-concepts is NExpTime-hard.

Instead of giving an upper bound for $\mathcal{ALC}^{(\neg),\sqcap}$, we give an upper bound for the logic $\mathcal{ALC}^{\neg,\sqcap,\sqcup}$, i.e., the extension of $\mathcal{ALC}^{\neg}$ by union and intersection allowing for full negation of roles with the obvious syntax and semantics: The translation of $\mathcal{ALC}^{\neg}$-concepts to L^2-formulae mentioned in Section 2 can also be applied to $\mathcal{ALC}^{\neg,\sqcap,\sqcup}$-concepts.

Corollary 15 Satisfiability of $\mathcal{ALC}^{\neg,\sqcap,\sqcup}$-concepts is in NExpTime.

Figure 1 summarizes the complexity results obtained in this paper (omitting the results for $\mathcal{S}^{\neg}$).

Acknowledgments

The authors would like to thank Franz Baader for fruitful discussions. The first author was supported by the DFG Project BA1122/3-1 "Combinations of Modal and Description Logics".

$$
\begin{aligned}
Count_x &= \forall\Big[\bigsqcap_{k=0}^{n}\Big((\bigsqcap_{j=0}^{k-1} X_j) \rightarrow (X_k \leftrightarrow \forall R_x.\neg X_k)\Big) \sqcap \\
&\qquad \bigsqcap_{k=0}^{n}\Big((\bigsqcup_{j=0}^{k-1} \neg X_j) \rightarrow (X_k \leftrightarrow \forall R_x.X_k)\Big) \sqcap \exists R_x.\top\Big] \\
Count_y &\quad \text{like } Count_x, \text{ replace } R_x \text{ by } R_y,\ X_j \text{ by } Y_j, \text{ and } X_k \text{ by } Y_k \\
Stable &= \forall\Big[\bigsqcap_{k=0}^{n}(X_k \rightarrow \forall R_y.X_k) \sqcap \bigsqcap_{k=0}^{n}(\neg X_k \rightarrow \forall R_y.\neg X_k) \sqcap \\
&\qquad \bigsqcap_{k=0}^{n}(Y_k \rightarrow \forall R_x.Y_k) \sqcap \bigsqcap_{k=0}^{n}(\neg Y_k \rightarrow \forall R_x.\neg Y_k)\Big] \\
Unique &= \forall\Big[\bigsqcap_{k=0}^{n}\Big((X_k \rightarrow \forall\neg R_k.\neg X_k) \sqcap (\neg X_k \rightarrow \forall\neg R_k.X_k)\Big) \sqcap \\
&\qquad \bigsqcap_{k=0}^{n}\Big((Y_k \rightarrow \forall\neg S_k.\neg Y_k) \sqcap (\neg Y_k \rightarrow \forall\neg S_k.Y_k)\Big) \sqcap \\
&\qquad \bigsqcap_{d\in D} A_d \rightarrow \forall R_0 \sqcap \cdots \sqcap R_n \sqcap S_0 \sqcap \cdots \sqcap S_n.A_d\Big] \\
Tiling &= \forall\Big[(\bigsqcup_{d\in D} A_d) \sqcap \bigsqcap_{d\in D}\ \bigsqcap_{d'\in D\setminus\{d\}} \neg(A_d \sqcap A_{d'}) \sqcap \\
&\qquad \bigsqcap_{d\in D} A_d \rightarrow \Big(\forall R_x.\bigsqcup_{(d,d')\in H} p_{d'}\Big) \sqcap \\
&\qquad \bigsqcap_{d\in D} A_d \rightarrow \Big(\forall R_y.\bigsqcup_{(d,d')\in G} A_{d'}\Big)\Big] \\
Init &= \bigsqcap_{k=0}^{n}(\neg X_i \sqcap \neg Y_i) \sqcap A_{w_0} \sqcap \forall R_x.A_{w_1} \sqcap \cdots \sqcap \forall R_x^{n-1}.A_{w_{n-1}} \\
C_{(\mathcal{D},a)} &= Count_x \sqcap Count_y \sqcap Stable \sqcap Unique \sqcap Tiling \sqcap Init
\end{aligned}
$$

Figure 2: The $\mathcal{ALC}^{(\neg),\sqcap}$ concept $C_{(\mathcal{D},a)}$ for $\mathcal{D} = (D, H, V)$ and $a = a_0, \ldots, a_{n-1}$.

References

[1] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of IJCAI-91*, 1991.

[2] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66, 1966.

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

[4] A. Borgida. On the relative expressivness of description logics and predicate logics. *Artificial Intelligence*, 82(1 - 2):353–367, 1996.

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. Conjunctive query containment in description logics with n-ary relations. In M.-C. Rousset, R. Brachmann, F. Donini, E. Franconi, I. Horrocks, and A. Levy, editors, *Proceedings of the International Workshop on Description Logics*, Gif sur Yvette, France, 1997. Université Paris-Sud.

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

[7] G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In D. Skordev, editor, *Mathematical Logic and Applications*, pages 253–263, New York, 1987. Plenum Press.

[8] R. Givan, D. McAllester, and S. Shalaby. Natural language based inference procedures applied to schubert's steamroller. In Kathleen Dean, Thomas L.; McKeown, editor, *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 915–922. MIT Press, July 1991.

[9] E. Grädel, P. Kolaitis, and M. Vardi. On the Decision Problem for Two-Variable First-Order Logic. *Bulletin of Symbolic Logic*, 3:53–69, 1997.

[10] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proc. of LPAR'99*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[11] I. L. Humberstone. Inaccessible worlds. *Notre Dame Journal of Formal Logic*, 24(3):346–352, 1983.

[12] U. Hustadt and R. Schmidt. Issues of decidability for description logics in the framework of resolution. In R. Caterra and G. Salzer, editors, *Automated Deduction in classical and non-classical logic*, volume 1761 of *LNAI*, pages 191–205. Springer-Verlag, 1996.

[13] D.E. Knuth. *The Art of computer programming*, volume 1. Addison Wesley Publ. Co., Reading, Massachussetts, 1968.

[14] C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logics. LTCS-Report 00-02, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2000. See http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html.

[15] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93(2), 1991.

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

[17] E. Spaan. *Complexity of Modal Logics.* PhD thesis, University of Amsterdam, 1993.

[18] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.