RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

# A PSpace-algorithm for $\mathcal{ALCQI}$-satisfiability

Stephan Tobies

LTCS-Report 99-09

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

# A PSpace-algorithm for $\mathcal{ALCQI}$-satisfiability

Stephan Tobies

July 23, 1999

Revised version

**Abstract**

The description logic $\mathcal{ALCQI}$ extends the "standard" description logic $\mathcal{ALC}$ by qualifying number restrictions and converse roles. We show that concept satisfiability for this DL is still decidable in polynomial space. The presented algorithm combines techniques from [Tob99] to deal with qualifying number restrictions and from [HST99] to deal with converse roles.

## 1 The Description Logic $\mathcal{ALCQI}$

Qualifying number restrictions [HB91] are a common generalisation of both role-quantification and standard number restrictions that are present in almost all implementations of 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 [CLN94]. In [Tob99] 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 section we extend this result to converse 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.*

*For an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, we extend the usual semantics of $\mathcal{ALC}$-concepts to qualifying number restrictions as follows:*

$$
\begin{aligned}
(\geqslant n\ R\ C)^\mathcal{I} &:= \{x \in \Delta^\mathcal{I} \mid \sharp\{y \mid (x,y) \in R^\mathcal{I}, y \in C^\mathcal{I}\} \geqslant n\},\\
(\leqslant n\ R\ C)^\mathcal{I} &:= \{x \in \Delta^\mathcal{I} \mid \sharp\{y \mid (x,y) \in R^\mathcal{I}, y \in C^\mathcal{I}\} \leqslant n\},
\end{aligned}
$$

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

In order to avoid considering roles such as $R^{--}$, we define a function $\mathsf{Inv}$ that returns the inverse of a role by setting

$$
\mathsf{Inv}(R) := \begin{cases} R^- & \text{if } R \in N_R \\ S & \text{if } R = S^- \text{ for some } S \in N_R \end{cases}
$$

## 2 Reasoning for $\mathcal{ALCQI}$

In [HB91] a tableaux algorithm is presented that decides SAT($\mathcal{ALCQ}$) in polynomial space, provided that unary coding of numbers in the input is assumed when calculating the size of the input. In [dHR95] it is conjectured that binary coding of numbers would make SAT($\mathcal{ALCQ}$) EXPTIME-complete. 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 $\mathcal{ALCQ}$: They decide the satisfiability of a concept $C$ by trying to explicitly construct a model for it. For a concept of the form $(\geqslant n\ R\ C)$, the algorithm in [HB91] generates $n$ individuals, and the correctness of the algorithms relies on that fact that they are kept in memory *simultaneously*. Assuming unary coding of numbers in the input, this is admissible 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: The number $n$ consumes only $\log_2 n$ bits in the input, making the amount of memory needed for $n$ successors potentially exponential in the size of the input.

In [Tob99] we give an algorithm derived from the one presented in [HB91] that is capable of deciding $\textsc{Sat}(\mathcal{ALCQ})$ in $\textsc{PSpace}$, 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 nodes is used to be able to generate these one after another re-using space. This exactly determines the complexity of $\textsc{Sat}(\mathcal{ALCQ})$ as $\textsc{PSpace}$-complete. This rather surprising result shows that augmenting $\mathcal{ALC}$ with qualifying number restrictions does not increase the (worst-case) complexity of the satisfiability problem.

In this paper we present an extension of the algorithm in [Tob99] that can additionally deal with converse roles and runs in polynomial space. This yields that also $\textsc{Sat}(\mathcal{ALCQI})$ is $\textsc{PSpace}$-complete. The "reset-restart" technique, which is used to deal with concepts moving upwards in the completion tree, has already been used in [HST99] to deal with converse 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 $x$ of the tree is labelled with a set $\mathcal{L}(x) \subseteq clos(D)$ and each edge $\langle x, y\rangle$ is labelled with a role name $\mathcal{L}(\langle x, y\rangle) = R$ for a (possibly inverse) 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 and ancestors 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 $n \in \mathbb{N}$, $(\leqslant n\ R\ C) \in \mathcal{L}(x)$ while $\sharp R^{\mathbf{T}}(x, C) > n$, 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 2 is applicable.*

| | | |
|---|---|---|
| $\sqcap$-rule: | if | 1. $C_1 \sqcap C_2 \in \mathcal{L}(x)$ and<br>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<br>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<br>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. $(\geqslant n\ R\ C) \in \mathcal{L}(x)$, $x$ is not blocked and no non-generating rule is applicable to $x$ or any of its ancestors, and<br>2. $\sharp R^{\mathbf{T}}(x, C) < n$ |
| | then | create a new node $y$ with $\mathcal{L}(\langle x, y\rangle) = R$ and $\mathcal{L}(y) = \mathcal{B}(y) = \{C, E_1, \ldots, E_n\}$ 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 $\mathcal{ALCQI}$

*For an $\mathcal{ALCQI}$-concept $D$, the algorithm starts with a completion tree consisting of a single node $x$ with $\mathcal{L}(x) = \{D\}$. 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.*

## 2.1 Correctness of the Algorithm

In order to prove the correctness of the algorithm we have to show termination, soundness, and completeness.

Before we prove termination of the algorithm we will establish a bound on the size of a completion tree generated by the tableaux algorithm that will also be used in 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 limited by $|D|$.*
3. *The out-degree of $\mathbf{T}$ is limited by $\sharp clos(D) \times 2^{|D|}$.*

*Proof.*

1. The first part of this Lemma can easily be proved by observing that for a concept $D$ in NNF

$$clos(D) = sub(D) \cup \{\sim C \mid C \in sub(D)\}$$

holds, where $sub(D)$ denotes the set of all sub-concepts of $D$. Obviously, $\sharp sub(D) \leq |D|$ and hence $\sharp clos(D) \leq \mathcal{O}(|D|)$.

2. For a node $x$ we define $\ell(x)$ as the maximum depth of nested number restrictions in $\mathcal{L}(x)$. Obviously, for the root $x_0$ of $\mathbf{T}$, $\ell(x_0) \leq |D|$ holds. Also, if $y$ is a successor of $x$ in $\mathbf{T}$, then $\ell(x) > \ell(y)$. Hence each path $x_1, \ldots, x_n$ in $\mathbf{T}$ induces a strictly decreasing sequence $\ell(x_1) > \ell(x_2) > \cdots > \ell(x_k)$ of natural numbers. Thus, the longest path in $\mathbf{T}$ starts at $x_0$ and its length is bounded by $|D|$.

3. Successors in $\mathbf{T}$ are only generated by the $\geqslant$-rule. For a node $x$ this rule will generate at most $n$ successors for each $(\geqslant n\ R\ C) \in \mathcal{L}(x)$. There are at most $\sharp clos(D)$ such formulae in $\mathcal{L}(x)$. Hence the out-degree of $x$ is bounded by $\sharp clos(D) \times 2^{|D|}$, where $2^{|D|}$ is a limit for the biggest number that may appear in $D$ if binary coding is used. ∎

From this we can follow termination of the algorithm.

**Lemma 4 (Termination)** *For any $\mathcal{ALCQI}$-concept $D$ the tableaux algorithm terminates.*

*Proof.* Termination of the algorithm is a consequence of the following facts:

- Each node is labelled with a subset of the finite set $clos(D)$. Concepts are never removed from the labels of the nodes.

- The size of the tree is bounded by Lemma 3.
- The rules either add concepts to the label of a node or add nodes to the tree.
- Whenever a node is deleted from the tree the labels of one of its ancestors grows.

Assume that algorithm does not terminate. Due to the mentioned facts this can only be because of an infinite number of deletions of subtrees. Each node can of course only be deleted once, but the successors of a single node may be deleted several times. The root of the completion tree cannot be deleted because it has no predecessor. Hence there are nodes which are never deleted. Choose one of these nodes $x$ with maximum distance from the root, i.e., which has a maximum number of predecessors. Suppose that $x$'s successors are deleted only finitely many times. This cannot be the case because, after the last deletion of $x$'s successors, the "new" successors were never deleted and thus $x$ would not have maximum distance from the root. Hence $x$ triggers the deletion of its successors infinitely many times. However, the *choose*-rule is the only rules that leads to a deletion, and it simultaneously leads to an increase of $\mathcal{L}(x)$, namely by the missing concept which caused the deletion of $x$'s successors. Since we never remove any concepts from the labels, this implies the existence of an infinitely increasing chain of subsets of $clos(D)$, which is clearly impossible. ∎

**Lemma 5 (Soundness)** *If the expansion 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 such a completion tree for $D$. A model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for $D$ can be defined by setting $\Delta^{\mathcal{I}}$ to be the nodes of $\mathbf{T}$ and by defining:

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

Inductively we will show for all $x \in \Delta^{\mathcal{I}}$ and all $C \in clos(D)$ that $C \in \mathcal{L}(x)$ implies $x \in C^{\mathcal{I}}$. We cannot use induction over the structure of concepts due to the $\geqslant$-rule that adds negated concepts to the tree. Instead we will use the following *norm* $\|\cdot\|$ of a concept $C$. The norm $\|C\|$ for concept in NNF is inductively defined by:

$$\begin{array}{lclcl} \|A\| & := & \|\neg A\| & := & 0 \quad \text{for } A \in N_C \\ \|C_1 \sqcap C_2\| & := & \|C_1 \sqcup C_2\| & := & 1 + \|C_1\| + \|C_2\| \\ \|(\bowtie\ n\, S\, C)\| & & & := & 1 + \|C\| \end{array}$$

The two base cases of the induction are $C = A$ or $C = \neg A$. If $A \in \mathcal{L}(x)$, then by definition $x \in A^{\mathcal{I}}$. If $\neg A \in \mathcal{L}(x)$, then $A \notin \mathcal{L}(x)$ because $\mathbf{T}$ is clash-free and hence $x \notin A^{\mathcal{I}}$. For the induction step we have to distinguish several cases:

- $C = C_1 \sqcap C_2$. Since $\mathbf{T}$ is complete $C \in \mathcal{L}(x)$ implies $C_1 \in \mathcal{L}(x)$ and $C_2 \in \mathcal{L}(x)$. Hence, by induction, we have $x \in C_1^{\mathcal{I}}$ and $x \in C_2^{\mathcal{I}}$ which yields $x \in (C_1 \sqcap C_2)^{\mathcal{I}}$.
- $C = C_1 \sqcup C_2$. Similar to the previous case.
- $C = (\geqslant\ n\ R\ E)$. For an $x$ with $C \in \mathcal{L}(x)$ we have $\sharp R^{\mathbf{T}}(x, E) \geqslant n$ because $\mathbf{T}$ is complete. Hence there are $n$ distinct $R$-neighbours $y_1, \ldots, y_n$ with $E \in \mathcal{L}(y_i)$ for all $i$. By induction, we have $y_i \in E^{\mathcal{I}}$ and, since, for each $R$-neighbour $y_j$, $\langle x, y_j\rangle \in R^{\mathcal{I}}$ holds, also $x \in C^{\mathcal{I}}$.
- $C = (\leqslant\ n\ R\ E)$. Let $x$ be an individual with $C \in \mathcal{L}(x)$. For any $R$-neighbour $y$ of $x$ either $E \in \mathcal{L}(y)$ of ${\sim}E \in \mathcal{L}(y)$. This is guaranteed by the *choose*-rule (for an $R$-predecessor of $x$) and by the $\geqslant$-rule which is suspended until no other rules can applied to $x$ or any predecessor of $x$ together with the reset-restart mechanism that is triggered by concepts "moving upwards" in the tree.

  We show that $\sharp R^{\mathcal{I}}(x, E) \leqslant \sharp R^{\mathbf{T}}(x, E)$: Assume $\sharp R^{\mathcal{I}}(x, E) > \sharp R^{\mathbf{T}}(x, E)$. This implies the existence of some $y$ with $\langle x, y\rangle \in R^{\mathcal{I}}$ with $y \in E^{\mathcal{I}}$ but $E \notin \mathcal{L}(y)$. This implies ${\sim}E \in \mathcal{L}(y)$, which, by induction yields $y \in ({\sim}E)^{\mathcal{I}}$ in contradiction to $x \in E^{\mathcal{I}}$.

Since $D \in \mathcal{L}(x_0)$ for the root $x_0$ of $\mathbf{T}$ this implies $D^{\mathcal{I}} \neq \emptyset$ and hence $\mathcal{I}$ is a model for $D$. ∎

**Lemma 6 (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 this model to guide the application of the non-deterministic completion rules. For this we will incremently define a function $\pi$ mapping the nodes in $\mathbf{T}$ to elements of $\Delta^{\mathcal{I}}$ such that at any given stage the following holds:

$$\left.\begin{array}{l} 1.\ \mathcal{L}(x) \Rightarrow \pi(x) \in C^{\mathcal{I}} \\ 2.\ \text{if } \mathcal{L}(\langle x, y\rangle) = R \text{ then } \langle \pi(x), \pi(y)\rangle \in R^{\mathcal{I}} \\ 3.\ \text{if } y, z \text{ are two } R\text{-neighbours of } x \text{ then } \pi(y) \neq \pi(z) \end{array}\right\} \quad (*)$$

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

- The $\sqcap$-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, then $\pi(x) \in (C_1 \sqcap C_2)^{\mathcal{I}}$. This implies $\pi(x) \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$, and hence the rule can be applied without violating $(*)$.

- The $\sqcup$-rule: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, then $\pi(x) \in (C_1 \sqcup C_2)^{\mathcal{I}}$. This implies $\pi(x) \in C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$. Hence the $\sqcup$-rule can add a concept $E \in \{C_1, C_2\}$ to $\mathcal{L}(x)$ such that $(*)$ still holds.

- The *choose*-rule: obviously, either $\pi(y) \in E^{\mathcal{I}}$ or $\pi(y) \notin E^{\mathcal{I}}$ for any node $y$ of the tree. Since $({\sim}E)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus E^{\mathcal{I}}$ the rule can always be applied in a way that maintains $(*)$. Deletion of nodes does not violate $(*)$

- The $\geqslant$-rule: if $(\geqslant n\ R\ C) \in \mathcal{L}(x)$, then $\pi(x) \in (\geqslant n\ R\ C)^{\mathcal{I}}$. This implies $\sharp S^{\mathcal{I}}(\pi(x), C) \geqslant n$. We claim that there is an element $t \in \Delta^{\mathcal{I}}$ such that

  $$\langle \pi(x), t\rangle \in R^{\mathcal{I}}, t \in C^{\mathcal{I}}, \text{ and } t \notin \{\pi(y) \mid y \text{ is an } R\text{-neighbour of } x\ \} \quad (**)$$

  We will come back to this claim later. Let $D_1, \ldots, D_n$ be an enumeration of the set $\{D \mid (\bowtie n\ R\ D) \in \mathcal{L}(x)\}$. The $\geqslant$-rule can add a new node $y$ with $\mathcal{L}(\langle x, y\rangle) = R$ and $\mathcal{L}(y) = \{C\} \cup \{D_i \mid t \in D_i^{\mathcal{I}}\} \cup \{{\sim}D_i \mid$

$t \notin D_i^{\mathcal{I}}\}$. If we set $\pi' := \pi[y \mapsto t]$, then the modified tree together with $\pi'$ satisfies $(*)$.

Why does there exists an element $t$ that satisfies $(**)$? It is obvious that there exists an element $t$ with $\langle \pi(x), t\rangle \in R^I$ and $t \in C^{\mathcal{I}}$ such that $t \notin \{\pi(y) \mid y$ is an $R$-neighbour of $x$ and $C \in \mathfrak{L}(y)\}$ because $\sharp R^T(\pi(x), C) \geq n > \sharp R^{\mathbf{T}}(x, C)$.

Assume $t$ appears as an image of an $R$-neighbour $y$ of $x$ with $C \notin \mathfrak{L}(y)$. This implies ${\sim}C \in \mathfrak{L}(y)$ as follows: Either $y$ is an $R$-predecessor of $x$, then in order for the $\geqslant$-rule to be applicable, no non-generating rules and especially the *choose*-rule is not applicable to $x$ and its ancestor which implies $\{C, {\sim}C\} \cap \mathfrak{L}(y) \neq \emptyset$. If $y$ is an $R$-successor of $x$ then it must have been generated by an application of the $\geqslant$-rule to $x$. In order for this rule to be applicable no non-generating rule may have been applicable to $x$ or any of its ancestors. This implies that at the time of the generation of $y$ already $(\geqslant n\ R\ C) \in \mathfrak{L}(x)$ held and hence the $\geqslant$-rule ensures $\{C, {\sim}C\} \cap \mathfrak{L}(y) \neq \emptyset$.

In any case ${\sim}C \in \mathfrak{L}(y)$ holds and together with $(*)$ this implies $t \notin C^{\mathcal{I}}$ which contradicts $t \in C^{\mathcal{I}}$. Hence $C \in \mathfrak{L}(y)$ must hold which is a contradiction to the assumption $C \notin \mathfrak{L}(y)$ and thus there must be an element that satisfies $(**)$.

This concludes the proof of the claim. The claim yields the lemma as follows: Obviously, $(*)$ holds for the initial tree with only a single 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 $D$). The claim yields that whenenver a rule is applicable then it can be applied in a manner that maintains $(*)$. Lemma 4 yields that each sequence of rule applications must terminate, and also each tree for which $(*)$ holds is necessarily clash-free. It cannot contain a clash of the form $\{A, \neg A\} \subseteq \mathfrak{L}(x)$ because this would imply $\pi(x) \in A^{\mathcal{I}}$ and $\pi(x) \notin A^{\mathcal{I}}$. It can neither contain a clash of the form $(\leqslant n\ R\ C) \in \mathfrak{L}(x)$ and $\sharp R^{\mathbf{T}}(x, C) > n$ because $\pi$ is an injective function on the set of all $R$-neighbours of $y$ and hence $\sharp R^{\mathbf{T}}(x, C) > n$ implies $\sharp R^{\mathcal{I}}(x, C) > n$ and which cannot be the case since $\pi(x) \in (\leqslant n\ R\ C)^{\mathcal{I}}$. ∎

Summing up, from Lemmas 4, 5, and 6 we get the following:

**Theorem 7** *The tableaux algorithm is a non-deterministic decision procedure for $\mathcal{ALCQI}$-satisfiability.*

## 2.2 Complexity of $\mathcal{ALCQI}$

What remains to show is that the algorithm can be implemented to run in polynomial space. This is stated in the following lemma.

Due to Savitch's theorem [Sav70] that states that PSPACE coincides with NPSPACE we don't have to deal with the non-determinism in the rules. Nevertheless, 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 8** *The tableaux algorithm can be implemented in* PSPACE.

*Proof.* Let $D$ be the $\mathcal{ALCQI}$-concept to be tested for satisfiability. We can 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*[SSS91], 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 has already been the key to a PSPACE upper bound for the propositional modal logic $\mathsf{K}_m$ and $\mathcal{ALC}$ in [Lad77, SSS91, HM92]. To do this we need to store the values for $\sharp R^{\mathbf{T}}(x, C)$ for each node $x$ in the path, 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.

Consider the algorithm in Fig. 2, where $\overline{\mathcal{R}_D}$ denotes the set of role names that appear in $clos(D)$ together with their inverses. It re-uses the space needed to check the satisfiability of a successor $y$ of $x$ once the existence of a complete and clash-free "subtree" for the constraints on $y$ has been established. This is admissible since the tableaux rules can delete but will never modify this subtree once is it completed. This deletion is necessary because the *choose*-rule pushes concepts upwards in the tree which might have an influence of the subtrees of the effected node. Since these have already 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 role 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 these numbers.

An algorithm that works as previously described is shown in Fig. 2. Let us examine the memoryusage of this algorithm. Let $n = |D|$. The algorithm is designed to keep only a single path of $\mathbf{T}$ in memory at a given stage. For each node $x$ on a path, $\mathcal{L}(x) \subseteq \mathit{clos}(D)$ and hence its size is bounded by $2n$. Thus, for a single variable $x$, $\mathcal{L}(x)$ can be stored in $\mathcal{O}(n)$ bits. For each variable, there are at most $\sharp \overline{\mathcal{R}_D} \times \sharp \mathit{clos}(D) = \mathcal{O}(n^2)$ counters to be stored. The numbers to be stored in these counters do not exceed the out-degree of the tree, which, by Lemma 3, is bounded by $\sharp \mathit{clos}(D) \times 2^{|D|}$. Hence each counter can be stored using $\mathcal{O}(n^2)$ bits when binary coding is used to represent the counters, and all counters for a single variable require $\mathcal{O}(n^4)$ bits. Due to Lemma 3, the length of a path is limited by $n$, which yields an overall memory consumption of $\mathcal{O}(n^5 + n^2)$ bits. ∎

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

**Theorem 9** SAT*($\mathcal{ALCQI}$) is* PSPACE*-complete, even if numbers in the input are represented in binary coding.*

# References


[CLN94] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proceedings of KR-94*, 1994.

[dHR95] Wiebe Van der Hoek and Maarten De Rijke. Counting objects. *J. of Logic and Computation*, 5(3):325–345, June 1995.

[HB91] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning, KR-91*, pages 335–346, Boston (USA), 1991.

```
𝒜ℒ𝒞𝒬ℐ − SAT(D) := sat(x_0, [x_0 ↦ {D}])
sat(x, 𝔏):
    allocate counters ♯R^T(x, C) for all R ∈ overline{ℛ_D} and C ∈ clos(D).
  restart:
    for each counter ♯R^T(x, C):
        if x has a predecessor y and 𝔏(⟨y, x⟩) = Inv(R) and C ∈ 𝔏(y)
            then ♯R^T(x, C) := 1 else ♯R^T(x, C) := 0
    while (the ⊓- or the ⊔-rule can be applied at x) and (T is clash-free)
    do
        apply the ⊓- or the ⊔-rule to x.
    od
    if T contains a clash then return "not satisfiable".
    if the choose-rule is applicable to x for (⋈ n R C) ∈ 𝔏(x)
        then return "restart with C"
    while (the ⩾-rule applies to a concept (⩾ n R C) ∈ 𝔏(x))
    do
        𝒞_new := {C, E_1, ..., E_k}
        where
            {D_1, ..., D_k} = {D | (⋈ m R D) ∈ 𝔏(x), and
            E_i is chosen non-deterministically from {D_i, ∼D_i}
        for each D ∈ 𝒞_new do increase ♯R^T(x, D)
        if (⩽ m R D) ∈ 𝔏(x) and ♯R^T(x, D) > m
            then return "not satisfiable".
        result := sat(y, 𝔏[y ↦ 𝒞_new, ⟨x, y⟩ ↦ R])
        where y is a fresh node
        if result = "not satisfiable" then return "not satisfiable"
        if result = "restart with D" then
            𝔏(x) := 𝔏(x) ∪ {E}
            where E is chose non-deterministically from {D, ∼D}
            goto restart
    od
    remove the counters for x from memory.
    return "satisfiable"
```

**Figure 2:** A NPSPACE decision procedure for $\mathcal{ALCQI}$-satisfiability.

[HM92] 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.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan 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.

[Lad77] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, September 1977.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, April 1970.

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[Tob99] S. Tobies. A pspace algorithm for graded modal logic. In *Proceedings of the 16th International Conference on Automated Deduction (CADE-99)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1999. To appear.