TECHNISCHE UNIVERSITÄT DRESDEN

**Dresden University of Technology**
**Institute for Theoretical Computer Science**
**Chair for Automata Theory**

# LTCS–Report

## PSpace Automata with Blocking for Description Logics

Franz Baader Jan Hladik Rafael Peñaloza

LTCS-Report 06-04

Lehrstuhl fur Automatentheorie
Institut fur Theoretische Informatik
TU Dresden
http://lat.inf.tu-dresden.de

Hans-Grundig-Str. 25
01062 Dresden
Germany

# PSpace Automata with Blocking for Description Logics

Franz Baader
Institute for Theoretical Computer Science, TU Dresden, Germany
(franz.baader@tu-dresden.de)

Jan Hladik
Institute for Theoretical Computer Science, TU Dresden, Germany
(jan.hladik@tu-dresden.de)

Rafael Peñaloza
Intelligent Systems Department, University of Leipzig, Germany
(penaloza@informatik.uni-leipzig.de)

February 27, 2007

**Abstract**

In Description Logics (DLs), both tableau-based and automata-based algorithms are frequently used to show decidability and complexity results for basic inference problems such as satis ability of concepts. Whereas tableau-based algorithms usually yield worst-case optimal algorithms in the case of PSpace-complete logics, it is often very hard to design optimal tableau-based algorithms for ExpTime-complete DLs. In contrast, the automata-based approach is usually well-suited to prove ExpTime upper-bounds, but its direct application will usually also yield an ExpTime-algorithm for a PSpace-complete logic since the (tree) automaton constructed for a given concept is usually exponentially large. In the present paper, we formulate conditions under which an on-the- y construction of such an exponentially large automaton can be used to obtain a PSpace-algorithm. We illustrate the usefulness of this approach by proving a new PSpace upper-bound for satis ability of concepts w.r.t. acyclic terminologies in the DL $\mathcal{SI}$, which extends the basic DL $\mathcal{ALC}$ with transitive and inverse roles.

# 1 Introduction

Description Logics (DLs) [2] are a successful family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. DL systems provide their users with inference services that deduce implicit knowledge from the explicitly represented knowledge. For these inference services to be feasible, the underlying inference problems must at least be decidable, and preferably of low complexity. For this reason, investigating the computational complexity of reasoning in DLs of di ering expressive power has been one of the most important research topics in the eld for the last 20 years. Since Description Logics are closely related to Modal Logics (MLs) [16], results and techniques can be transferred between the two areas.

Two of the most prominent methods for showing decidability and complexity results for DLs and MLs are the tableau-based [8, 4] and the automata-based [18, 7] approach. Both approaches basically depend on the tree-model property of the DL/ML under consideration: if a concept/formula is satis able, then it is also satis able in a tree-shaped model. They di er in how they test for the existence of a tree-shaped model. Tableau-based algorithms try to generate such a model in a top-down non-deterministic manner, starting with the root of the tree. Automata-based algorithms construct a tree automaton that accepts exactly the tree-shaped models of the concept/formula, and then test the language accepted by this automaton for emptiness. The usual emptiness test for tree automata is deterministic and works in a bottom-up manner. This di erence between the approaches also leads to di erent behaviour regarding elegance, complexity, and practicability.

If the logic has the *nite* tree model property, then termination of tableau-based algorithms is usually easy to achieve. If, in addition, the tree models these algorithms are trying to construct are of polynomial depth (as is the case for the PSpace-complete problem of satis ability in the basic DL $\mathcal{ALC}$, which corresponds to the multi-modal variant of the ML $\mathsf{K}$), then one can usually modify tableau-based algorithms such that they need only polynomial space: basically, they must only keep one path of the tree in memory [17]. However, the automaton constructed in the automata-based approach is usually exponential, and thus constructing it explicitly before applying the emptiness test requires exponential time and space. In [9], we formulate conditions on the constructed automaton that ensure—in the case of nite tree models of polynomially bounded depth—that an on-the- y construction of the automaton during a non-deterministic top-down emptiness test yields

a PSPACE algorithm.

If the logic does *not* have the *finite* tree model property, then applying the tableau-based approach in a straightforward manner leads to a non-terminating procedure. To ensure termination of tableau-based algorithms in this case, one must apply an appropriate cycle-checking technique, called "blocking" in the DL literature [4]. This is, for example, the case for satisfiability in $\mathcal{ALC}$ w.r.t. so-called general concept inclusions (GCIs) [1]. Since blocking usually occurs only after an exponential number of steps and since tableau-based algorithms are non-deterministic, the best complexity upper-bound that can be obtained this way is NEXPTIME. This is not optimal since satisfiability in $\mathcal{ALC}$ w.r.t. GCIs is "only" EXPTIME-complete. The EXPTIME upper-bound can easily be shown with the automata-based approach: the constructed automaton is of exponential size, and the (bottom-up) emptiness test for tree automata runs in time polynomial in the size of the automaton. Although the automata-based approach yields a worst-case optimal algorithm in this case, the obtained algorithm is not practical since it is also exponential in the best case: before applying the emptiness test, the exponentially large automaton must be constructed. In contrast, optimised implementations of tableau-based algorithms usually behave quite well in practice [10], in spite of the fact that they are not worst-case optimal. There have been some attempts to overcome this mismatch between practical and worst-case optimal algorithms for EXPTIME-complete DLs. In [5] we show that the so-called inverse tableau method [19] can be seen as an on-the-fly implementation of the emptiness test in the automata-based approach, which avoids the a priori construction of the exponentially large automaton. Conversely, we show in [3] that the existence of a sound and complete so-called EXPTIME-admissible tableau-based algorithm for a logic always implies the existence of an EXPTIME automata-based algorithm. This allows us to construct only the (practical, but not worst-case optimal) tableau-based algorithm, and get the optimal EXPTIME upper-bound for free.

In the present paper, we extend the approach from [9] mentioned above such that it can also deal with PSPACE-complete logics that do not have the finite tree model property. A well-known example of such a logic is $\mathcal{ALC}$ extended with transitive roles [14]. To illustrate the power of our approach, we use the more expressive DL $\mathcal{SI}$ as an example, which extends $\mathcal{ALC}$ with transitive and inverse roles. In addition, we also allow for acyclic concept definitions. To the best of our knowledge, the result that satisfiability in $\mathcal{SI}$ w.r.t. acyclic concept definitions is in PSPACE is new.

# 2 The description logic $\mathcal{SI}$

The basic description logic $\mathcal{ALC}$ *(attributive language with complements)* [17] provides concepts (unary relation symbols) and roles (binary relation symbols) and allows for the use of conjunction, disjunction and negation as well as existential and universal quanti cation for concepts. Thus, we can describe a man who has at least one unmarried child and all of whose children are married or happy as:

$$\mathsf{Man} \sqcap \exists \mathsf{has\text{-}child}.\neg\mathsf{Married} \sqcap \forall \mathsf{has\text{-}child}.(\mathsf{Married} \sqcup \mathsf{Happy})$$

$\mathcal{SI}$ is an extension of $\mathcal{ALC}$, where $\mathcal{I}$ stands for inverse roles and $\mathcal{S}$ stands for transitive roles due to the similarity between $\mathcal{ALC}$ with transitive roles and the multi-modal logic $\mathsf{S4_m}$ (see e.g. [6]). Using these constructs, we can stipulate that the relation has-o spring is transitive and describe a king one of whose ancestors was also a king and all of whose descendants are also kings as follows:

$$\mathsf{King} \sqcap \exists \mathsf{has\text{-}o\ \ spring}\ \ .\mathsf{King} \sqcap \forall \mathsf{has\text{-}o\ \ spring}.\mathsf{King}$$

In description logics, a terminology is stored in a knowledge base called *TBox*, which consists of concept de nitions $A \doteq C$ and general concept inclusion axioms $C \sqsubseteq D$, so we can de ne humans as the union of men and women, and we can state that if a man is married, then he has a wife.

$$\begin{aligned} \mathsf{Human} &\doteq \mathsf{Man} \sqcup \mathsf{Woman} \\ \mathsf{Man} \sqcap \mathsf{Married} &\sqsubseteq \exists \mathsf{has\text{-}wife}.\mathsf{Woman} \end{aligned}$$

$\mathcal{SI}$ also contains a top concept $\top$ and a bottom concept $\bot$, standing for a tautology and an unsatis able concept, respectively. Formally, $\mathcal{SI}$ is de ned as follows:

**De nition 1 (Syntax of $\mathcal{SI}$)** Let $N_C$ be a set of *concept names* and $N_R$ be a set of *role names*, where $N_T \quad N_R$ is the set of *transitive role names*. Then the set of $\mathcal{SI}$ *roles* is de ned as $N_R \cup \{r \quad \mid r \in N_R\}$, and the set of $\mathcal{SI}$ *concepts* is the smallest set that satis es the following conditions:

- $\top$, $\bot$, and all concept names are concepts;
- if $C$ and $D$ are concepts, then $\neg C$, $C \sqcup D$ and $C \sqcap D$ are also concepts;
- if $C$ is a concept and $r$ is a role, then $\exists r.C$ and $\forall r.C$ are also concepts.

To avoid roles like $r^{--}$, we define, for a role $r$, the *inverse of* $r$ *(*$\overline{r}$*)* as $r^-$, if $r$ is a role name, and as $s$ if $r$ is an inverse role $s^-$. Since a role is transitive iff its inverse is transitive, we use a predicate $\mathsf{trans}(r)$ which is defined as true iff $r$ or $\overline{r}$ belongs to $N_T$.

A *concept definition* has the form $A \doteq C$, where $A$ is a concept name and $C$ is a concept. If there exists a concept definition for $A$, it is called a *defined* concept name, otherwise it is called *primitive*. A *general concept inclusion axiom (GCI)* has the form $C \sqsubseteq D$, where $C$ and $D$ are concepts. An *acyclic TBox* is a set of concept definitions $A_i \doteq C_i$, where there is at most one definition for a concept name and there is no sequence of concept definitions $A_1 \doteq C_1, \ldots, A_n \doteq C_n$ such that $C_i$ contains $A_{i+1}$ for $1 \leq i < n$ and $C_n$ contains $A_1$. A *general TBox* can additionally contain GCIs.

The definition of acyclic TBoxes ensures that no concepts are defined recursively (as in $\mathsf{Male} \doteq \neg\mathsf{Female}$ and $\mathsf{Female} \doteq \neg\mathsf{Male}$) and that concept definitions are not abused to define equivalence between complex concepts (as in $A \doteq \forall r.C$ and $A \doteq \exists r.D$). Thus, an acyclic TBox is essentially a set of *macro definitions*, which could in principle be completely expanded. However, acyclic TBoxes are still useful because a concept with expanded definitions may be significantly harder to read for a human and, more importantly, it may be exponentially larger: the size of the TBox $\mathcal{T}_n := \{A_i \doteq (\exists r.A_{i+1}) \sqcap (\exists r.\neg A_{i+1}) \mid 0 \leq i \leq n\}$ is linear in $n$, whereas the length of the expanded concept $A_0$ is exponential.

**Definition 2 (Semantics of $\mathcal{SI}$)** An *interpretation* $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a set (called the *domain* of $\mathcal{I}$) and $\cdot^{\mathcal{I}}$ is a function which assigns to every concept name $A$ a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to every role name $r$ a subset $r^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This function is extended to complex concepts and roles as follows:

$$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}, \quad \bot^{\mathcal{I}} = \emptyset,$$

$$(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}, \quad (\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},$$

$$(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \text{ there is a } y \in \Delta^{\mathcal{I}} \text{ with } (x,y) \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\},$$

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

$$(r^-)^{\mathcal{I}} := \{(y,x) \mid (x,y) \in r^{\mathcal{I}}\}.$$

Moreover, we demand that for a transitive role name $r$ it holds that $r^{\mathcal{I}} = (r^{\mathcal{I}})^{\mathsf{tr}}$, where $^{\mathsf{tr}}$ denotes transitive closure of a relation.

An interpretation $\mathcal{I}$ is called a *model* for a concept $C$ if $C^{\mathcal{I}} \neq \emptyset$, and a *model for a TBox* $\mathcal{T}$ if, for every definition $A \doteq C \in \mathcal{T}$, $A^{\mathcal{I}} = C^{\mathcal{I}}$ and for

every GCI $C \sqsubseteq D$, $C^{\mathcal{I}}$ $D^{\mathcal{I}}$. A model of $C$ *with respect to* $\mathcal{T}$ is a model of $C$ and $\mathcal{T}$. A concept $C$ is called *satis able (with respect to* $\mathcal{T}$*)* if there is a model of $C$ (and $\mathcal{T}$). Similarly, we say that a concept $C$ is *subsumed* by a concept $D$, written as $C \sqsubseteq D$ (w.r.t. $\mathcal{T}$) if, in every interpretation (every model for $\mathcal{T}$), $C^{\mathcal{I}}$ $D^{\mathcal{I}}$.

The subsumption problem and the satis ability problem are the standard inference problems for DLs. In the presence of negation and conjunction, they can be mutually reduced to each other: $C$ is satis able i $C$ is not subsumed by $\bot$, and $C$ is subsumed by $D$ i $C \sqcap \neg D$ is unsatis able. In the following, we will therefore only consider the satis ability problem.

## 2.1 Hintikka trees

Tree models of satis able $\mathcal{SI}$ concepts can be obtained by applying the well-known technique of unravelling [6]. For example, the $\mathcal{SI}$ concept $A$ is satis able w.r.t. the general TBox $\{A \sqsubseteq \exists r.A\}$ in a one-element model whose single element belongs to $A$ and is related to itself via $r$. The corresponding unravelled model consists of a sequence $d_0, d_1, d_2, \ldots$ of elements, all belonging to $A$, where $d_i$ is related to $d_{i+1}$ via $r$. Intuitively, Hintikka trees are tree models where every node is labelled with the appropriate concepts for the element it represents. These concepts are taken from the set of subconcepts of the concept to be tested for satis ability and of the concepts occurring in the TBox. In our example, the nodes $d_i$ would be labelled by $A$ and $\exists r.A$ since each $d_i$ belongs to these concepts.

To simplify the formal de nitions, we assume in the following that *all* concepts are in *negation normal form (NNF)*, i.e. negation appears only directly in front of concept names. An $\mathcal{SI}$ concept can be transformed into NNF in linear time using de Morgan's laws, duality of quanti ers, and elimination of double negation. We denote the NNF of a concept $C$ by $\mathsf{nnf}(C)$ and $\mathsf{nnf}(\neg C)$ by $\backsim C$.

**De nition 3 (Subconcepts, Hintikka sets)** The set of *subconcepts* of an $\mathcal{SI}$ concept $C$ ($\mathsf{sub}(C)$) is the least set $S$ that contains $C$ and has the following properties: if $S$ contains $\neg A$ for a concept name $A$, then $A \in S$; if $S$ contains $D \sqcup E$ or $D \sqcap E$, then $\{D, E\}$ $S$; if $S$ contains $\exists r.D$ or $\forall r.D$, then $D \in S$. For a TBox $\mathcal{T}$, $\mathsf{sub}(C, \mathcal{T})$ is de ned as follows:

$$\mathsf{sub}(C) \quad \cup \bigcup_{A \doteq D \in \mathcal{T}} (\{A, \neg A\} \cup \mathsf{sub}(D) \cup \mathsf{sub}(\backsim D)) \quad \cup \bigcup_{D \sqsubseteq E \in \mathcal{T}} \mathsf{sub}(\backsim D \sqcup E)$$

A set $H$ $\mathsf{sub}(C, \mathcal{T})$ is called a *Hintikka set for* $C$ if the following three conditions are satis ed: if $D \sqcap E \in H$, then $\{D, E\}$ $H$; if $D \sqcup E \in H$, then

$\{D, E\} \cap H \neq \emptyset$; and there is no concept name $A$ with $\{A, \neg A\}$ $H$. For a TBox $\mathcal{T}$, a Hintikka set $H$ is called $\mathcal{T}$-*expanded* if for every GCI $D \sqsubseteq E \in \mathcal{T}$, it holds that $\backsim D \sqcup E \in H$, and for every concept de nition $A \doteq D \in \mathcal{T}$, it holds that, if $A \in H$ then $D \in H$, and if $\neg A \in H$ then $\backsim D \in H$.[1]

Hintikka trees for $C$ and $\mathcal{T}$ are in nite trees of a xed arity $k$, which is determined by the number of existential restrictions, i.e. concepts of the form $\exists r.D$, in $\mathsf{sub}(C, \mathcal{T})$. For a positive integer $k$, we denote the set $\{1, \ldots, k\}$ by $K$. The nodes of a $k$-ary tree can be denoted by the elements of $K$ , with the empty word $\varepsilon$ denoting the root, and $ui$ the $i$th successor of $u$. In the case of labelled trees, we will refer to the label of the node $u$ in the tree $t$ by $t(u)$. In the de nition of Hintikka trees, we need to know which successor in the tree corresponds to which existential restriction. For this purpose, we x a linear order on the existential restrictions in $\mathsf{sub}(C, \mathcal{T})$. Let $\varphi : \{\exists r.D \in \mathsf{sub}(C, \mathcal{T})\} \to K$ be the corresponding ordering function, i.e. $\varphi(\exists r.D)$ determines the successor node corresponding to $\exists r.D$. In general, such a successor node need not exist in a tree model. To obtain a full $k$-ary tree, Hintikka trees contain appropriate dummy nodes. For technical reasons, which will become clear later on, the nodes of the Hintikka trees de ned below are not simply labelled by Hintikka sets, but by quadruples ( , , , $\varrho$), where $\varrho$ is the role which connects the node with the father node, is the complete Hintikka set for the node, consists of the unique concept $D$ contained in because of an existential restriction $\exists \varrho.D$ in the father node, and contains only those concepts that are contained in because of universal restrictions $\forall \varrho E$ in the father node. We will use a special new role name for nodes that are not connected to the father node by a role, i.e. the root node and those (dummy) nodes which are labelled with an empty set of concepts.

**De nition 4 (Hintikka trees)** The tuple (( $_0$, $_0$, $_0$, $\varrho_0$), ( $_1$, $_1$, $_1$, $\varrho_1$), ..., ( $_k$, $_k$, $_k$, $\varrho_k$)) is called $C, \mathcal{T}$-*compatible* if, for all $i, 0$ $i$ $k$, $_i \cup$ $_i$ $_i$, $_i$ is a $\mathcal{T}$-expanded Hintikka set, and the following holds for every existential concept $\exists r.D \in \mathsf{sub}(C, \mathcal{T})$:

if $\exists r.D \in$ $_0$, then

1. $_{\varphi(\exists r.D)}$ consists of $D$;
2. $_{\varphi(\exists r.D)}$ consists of all concepts $E$ for which there is a universal restriction $\forall r.E \in$ $_0$, and additionally $\forall r.E$ if $\mathsf{trans}(r)$;

[1] We will refer to this technique of handling concept de nitions as *lazy unfolding*. Note that, in contrast to GCIs, concept de nitions are only applied if $A$ or $\neg A$ is explicitly present in $H$.

3. for every concept $\forall \overline{r}.F \in \quad_{\varphi(\exists r.D)}$, $\quad_0$ contains $F$, and additionally $\forall \overline{r}.F$ if $\mathsf{trans}(r)$;
4. $\varrho_{\varphi(\exists r.D)} = r$;

if $\exists r.D \notin \quad_0$, then $\quad_{\varphi(\exists r.D)} = \quad_{\varphi(\exists r.D)} = \quad_{\varphi(\exists r.D)} = \emptyset$ and $\varrho_{\varphi(\exists r.D)} = \quad$.

A $k$-ary tree $t$ is called a *Hintikka tree for $C$ and $\mathcal{T}$* if, for every node $v \in K$ , the tuple $(t(v), t(v1), \ldots, t(vk))$ is $C, \mathcal{T}$-compatible, and $t(\varepsilon)$ has empty - and -components, an -component containing $C$, and as its $\varrho$-component. For a role $r$, we say that a node $w$ is an *$r$-neighbour* of a node $v$ if $w = v \quad \varphi(\exists r.D)$ for some concept $D$ and $\exists r.D \in \quad (v)$ or if $v = w \quad \varphi(\exists \overline{r}.D)$ and $\exists \overline{r}.D \in \quad (w)$.

Our de nition of a Hintikka tree ensures that the existence of such a tree characterises satis ability of $\mathcal{SI}$ concepts. It basically combines the technique for handling transitive and inverse roles introduced in [11][2] with the technique for dealing with acyclic TBoxes employed in [9].

**Theorem 5** The $\mathcal{SI}$ concept $C$ is satis able w.r.t. the general TBox $\mathcal{T}$ i there exists a Hintikka tree for $C$ and $\mathcal{T}$.

**Proof.** For a node $v$ with $t(v) = ( \quad , \quad , \quad , \varrho)$, we will refer to the components as $\quad(v)$, $\quad(v)$ etc.

For the "if" direction, we will show how to construct a model $( \quad^{\mathcal{I}}, \quad^{\mathcal{I}})$ from a Hintikka tree $t$. Let $\quad^{\mathcal{I}} = \{v \in K \quad \mid t(v) \neq (\emptyset, \emptyset, \emptyset, \quad)\}$. For a role name $r \in N_R \setminus N_T$, we de ne $r^{\mathcal{I}} = \{(v, w) \mid w \text{ is an } r\text{-neighbour of } v\}$. If $r \in N_T$, we de ne $r^{\mathcal{I}}$ as the transitive closure of this relation.

For a primitive concept name $A$, we de ne $A^{\mathcal{I}} = \{v \in \quad^{\mathcal{I}} \mid A \in \quad (v)\}$. In order to show that this interpretation can be extended to de ned concept names and that it interprets complex concepts correctly, we de ne a weight function $o(C)$ for concept terms $C$ as follows:

$o(A) = 0$ for a primitive concept name $A$;

$o(C \sqcap D) = o(C \sqcup D) = \max\{o(C), o(D)\} + 1$;

$o(\exists r.C) = o(\forall r.C) = o(C) + 1$;

$o(B) = o(C) + 1$ for a de ned concept name $B \doteq C$.

Notice that $o$ is well-founded because $\mathcal{T}$ is acyclic. We can now show by induction over the weight of the appearing concepts that if $D \in \quad (v)$, then $v \in D^{\mathcal{I}}$:

[2]there used in the context of tableau-based algorithms.

if $A \in \quad (v)$ for a primitive concept name $A$, $v \in A^{\mathcal{I}}$ holds by de nition;

if $E \sqcap F \in \quad (v)$ then, since $\quad (v)$ is a Hintikka set, it contains $E$ and $F$, and by induction $v \in E^{\mathcal{I}} \cap F^{\mathcal{I}}$ holds;

if $E \sqcup F \in \quad (v)$ then $v \in E^{\mathcal{I}} \cup F^{\mathcal{I}}$ follows from an analogous argument;

if $\exists r.E \in \quad (v)$ for a role name $r$ then, since $t$ is a Hintikka tree, $(v, v \quad \varphi(\exists r.E)) \in r^{\mathcal{I}}$ and $E \in \quad (v \quad \varphi(\exists r.E))$ (inverse roles can be treated analogously), thus by induction $v \in (\exists r.E)^{\mathcal{I}}$ holds;

if $\forall r.E \in \quad (v)$ for a role $r$ and $(v, w) \in r^{\mathcal{I}}$, then $(v, w) \in r^{\mathcal{I}}$ holds either because $w$ is an $r$-neighbour of $v$ in the Hintikka tree, in which case $E \in \quad (w)$ holds by de nition of $C, \mathcal{T}$-compatible, or $r$ is a transitive role and $(v, w)$ is in the transitive closure of the relation de ned above. In this case, there exists a sequence of tree nodes $v = v_0, v_1, \ldots, v_{f \ 1}, v_f = w$ such that for every $i < f$, $v_{i+1}$ is an $r$-neighbour of $v_i$. Since $\mathsf{trans}(r)$ holds, every node label $t(v_i)$ for $1 \quad i \quad t$ contains $\forall r.E$ and $E$ because of the de nition of $C, \mathcal{T}$-compatible, thus it follows by induction that $w \in E^{\mathcal{I}}$ and $v \in (\forall r.E)^{\mathcal{I}}$;

if $B \in \quad (v)$ for a de ned concept name $B \doteq C$, we know that $C \in \quad (v)$ because $\quad (v)$ is $\mathcal{T}$-expanded. Since $o(C) < o(B)$, it follows by induction that $v \in C^{\mathcal{I}}$ holds. Thus we can de ne $B^{\mathcal{I}} = C^{\mathcal{I}}$ and obtain $v \in B^{\mathcal{I}}$.

For a GCI $E \sqsubseteq F$, $\quad (v)$ contains $\backsim E \sqcup F$ for every node $v$. As $\quad (v)$ is a Hintikka set, it contains $F$ or $\backsim E$. If it contains $F$ then, as we have just shown, $v$ belongs to $F^{\mathcal{I}}$. Otherwise, $\quad (v)$ contains $\backsim E$, and $v \in (\backsim E)^{\mathcal{I}} = \quad ^{\mathcal{I}} \setminus E^{\mathcal{I}}$ holds, which implies $v \notin E^{\mathcal{I}}$. Therefore every node $v \in E^{\mathcal{I}}$ is also contained in $F^{\mathcal{I}}$.

For the "only-if" direction, we show how a model $( \quad ^{\mathcal{I}}, \ ^{\mathcal{I}})$ for $C$ w.r.t. $\mathcal{T}$ can be used to de ne a $C, \mathcal{T}$-compatible Hintikka tree $t$ with $C \in \quad (\varepsilon)$. Let $k$ be the number of existential concepts in $\mathsf{sub}(C, \mathcal{T})$ and $\varphi$ be a function as in De nition 4. We inductively de ne a function $\vartheta : K \quad \rightarrow \quad ^{\mathcal{I}} \cup \{ \ \}$ for a new individual $\quad$ such that $\vartheta(v)$ satis es all concepts in $\quad (v)$.

Since $( \quad ^{\mathcal{I}}, \ ^{\mathcal{I}})$ is a model, there exists an element $d_0 \in \quad ^{\mathcal{I}}$ with $d_0 \in C^{\mathcal{I}}$. So we de ne $\vartheta(\varepsilon) = d_0$ and set $\quad (\varepsilon) = \quad (\varepsilon) = \emptyset$, $\quad (\varepsilon) = \{E \in \mathsf{sub}(C, \mathcal{T}) \mid d_0 \in E^{\mathcal{I}}\}$, and $\varrho(\varepsilon) = \quad$ . Then we inductively de ne, for every node $v$ for which $\vartheta$ is already de ned, the labels of $v \quad i, 1 \quad i \quad k$, as follows: if $\quad (v)$ contains the existential concept $\exists r.E$ with $i = \varphi(\exists r.E)$ then, since $\vartheta(v)$ satis es $\exists r.E$, there exists a $d \in \quad ^{\mathcal{I}}$ with $(\vartheta(v), d) \in r^{\mathcal{I}}$ and $d \in E^{\mathcal{I}}$, and thus we set $\vartheta(v \quad i) = d$, $\quad (v \quad i) = \{F \in \mathsf{sub}(C, \mathcal{T}) \mid d \in F^{\mathcal{I}}\}$, $\varrho(v \quad i) = r$, $\quad (v \quad i) = \{\mathrm{E}\}$, and $\quad (v \quad i)$ contains every $F$ with $\forall r.F \in \quad (v)$ and, if $r$ is

transitive, additionally $\forall r.F$. If $\vartheta(v)$ does not belong to $(\exists r.E)^{\mathcal{I}}$, we de ne $\vartheta(v\ \ i) =$ and $(\ \ (v\ \ i),\ \ (v\ \ i),\ \ (v\ \ i), \varrho(v\ \ i)) = (\emptyset, \emptyset, \emptyset,\ \ )$.

It follows by construction that $(v\ \ i)$ and $(v\ \ i)$ are subsets of $(v\ \ i)$ and that the tuple $((\ \ (v),\ \ (v),\ \ (v), \varrho(v)), (\ \ (v\ \ 1),\ \ (v\ \ i),\ \ (v\ \ 1), \varrho(v\ \ 1)), \ldots, (\ \ (v\ \ k),\ \ (v\ \ k),\ \ (v\ \ k), \varrho(v\ \ k)))$ is $C, \mathcal{T}$-compatible. Note that for every $v \in K$ , $(v)$ is a Hintikka set since it follows from the fact that $(\ \ ^{\mathcal{I}}, ^{\mathcal{I}})$ is a model that if $d \in (E \sqcup [\sqcap]F)^{\mathcal{I}}$, then $d \in E^{\mathcal{I}} \cup [\cap]F^{\mathcal{I}}$, and that $d \in E^{\mathcal{I}}$ i $d \notin (\neg E)^{\mathcal{I}}$. ⧈

# 3 Tree automata

The existence of a Hintikka tree can be decided with the help of so-called looping automata, i.e. automata on in nite trees without a special acceptance condition. After introducing these automata, we will rst show how they can be used to decide satis ability in $\mathcal{SI}$ w.r.t. general TBoxes in exponential time. Then we will introduce a restricted class of looping automata and use it to show that satis ability in $\mathcal{SI}$ w.r.t. acyclic TBoxes can be decided in polynomial space.

## 3.1 Looping automata

The following de nition of looping tree automata does not include an alphabet for labelling the nodes of the trees. In fact, when deciding the emptiness problem for such automata, only the *existence* of a tree accepted by the automaton is relevant, and not the labels of its nodes. For our reduction this implies that the automaton we construct for a given input $C, \mathcal{T}$ has as its *successful runs* all Hintikka trees for $C, \mathcal{T}$ rather than actually accepting all Hintikka trees for $C, \mathcal{T}$.

**De nition 6 (Automaton, run)** A *looping tree automaton* over $k$-ary trees is a tuple $(Q,\ \ , I)$, where $Q$ is a nite set of states, $Q^{k+1}$ is the transition relation, and $I \quad Q$ is the set of initial states. A *run* of this automaton on the (unique) unlabelled $k$-ary tree $t$ is a labelled $k$-ary tree $r : K \ \to Q$ such that $(r(v), r(v1), \ldots, r(vk)) \in$ for all $v \in K$ . The run is *successful* if $r(\varepsilon) \in I$. The *emptiness problem for looping tree automata* is the problem of deciding whether a given looping tree automaton has a successful run or not.

In order to *decide the emptiness problem* in time polynomial in the size of $\mathcal{A}$, one computes the set of all bad states, i.e. states that do not occur in

any run, in a *bottom-up* manner [18, 5]: states that do not occur as rst component in the transition relation are bad, and if all transitions that have the state $q$ as rst component contain a state already known to be bad, then $q$ is also bad. The automaton has a successful run i there is an initial state that is not bad.

For an $\mathcal{SI}$ concept $C$ and a general TBox $\mathcal{T}$, we can construct a looping tree automaton whose successful runs are exactly the Hintikka trees for $C$ and $\mathcal{T}$.

**De nition 7 (Automaton $\mathcal{A}_{C,\mathcal{T}}$)** For an $\mathcal{SI}$ concept $C$ and a TBox $\mathcal{T}$, let $k$ be the number of existential restrictions in $\mathsf{sub}(C,\mathcal{T})$. Then the looping automaton $\mathcal{A}_{C,\mathcal{T}} = (Q, \quad , I)$ is de ned as follows:

$Q$ consists of all 4-tuples $( \quad , \quad , \quad , \varrho)$ such that $\quad \cup \qquad\qquad \mathsf{sub}(C,\mathcal{T})$, is a singleton set, is a $\mathcal{T}$-expanded Hintikka set for $C$, and $\varrho$ occurs in $C$ or $\mathcal{T}$ or is equal to ;

consists of all $C,\mathcal{T}$-compatible tuples $(( \ _0, \ _0, \ _0, \varrho_0), ( \ _1, \ _1, \ _1, \varrho_1), \ldots, ( \ _k, \ _k, \ _k, \varrho_k))$;

$I := \{(\emptyset, \emptyset, \quad , \quad ) \in Q \mid C \in \quad \}$.

**Lemma 8** $\mathcal{A}_{C,\mathcal{T}}$ has a successful run i $C$ is satis able w.r.t. $\mathcal{T}$.

**Proof.** Since the transition relation is de ned exactly as the relation $C,\mathcal{T}$-compatible, this follows by simple induction. ⊡

Since the cardinality of $\mathsf{sub}(C,\mathcal{T})$ and the size of each of its elements is linear in the size of $C,\mathcal{T}$, the size of the automaton $\mathcal{A}_{C,\mathcal{T}}$ is exponential in the size of $C,\mathcal{T}$. Together with the fact that the emptiness problem for looping tree automata can be decided in polynomial time, this yields:

**Theorem 9** Satis ability in $\mathcal{SI}$ w.r.t. general TBoxes is in ExpTime.

This complexity upper-bound is optimal since ExpTime-hardness follows from the known hardness result for $\mathcal{ALC}$ with general TBoxes [16].

One could also try to solve the emptiness problem by constructing a successful run in a *top-down manner*: label the root with an element $q_0$ of $I$, then apply a transition with rst component $q_0$ to label the successor nodes, etc. There are, however, two problems with this approach. First, it yields a *non-deterministic* algorithm since $I$ may contain more than one element, and in each step more than one transition may be applicable. Second, one must employ an appropriate cycle-checking technique (similar to blocking

in tableau-based algorithms) to obtain a terminating algorithm. Applied to the automaton $\mathcal{A}_{C,\mathcal{T}}$, this approach would at best yield a (non-optimal) NExpTime satis ability test.

## 3.2 Blocking-invariant automata

In order to obtain a PSpace result for satis ability w.r.t. *acyclic* TBoxes, we use the top-down emptiness test sketched above. In fact, in this case non-determinism is unproblematic since NPSpace is equal to PSpace by Savitch's theorem [15]. The advantage of the top-down over the bottom-up emptiness test is that it is not necessary to construct the whole automaton before applying the emptiness test. Instead, the automaton can be constructed on-the- y. However, we still need to deal with the termination problem. For this purpose, we adapt the blocking technique known from the tableau-based approach. In the following, when we speak about a *path* in a $k$-ary tree, we mean a sequence of nodes $v_1, \ldots, v_m$ such that $v_1$ is the root $\varepsilon$ and $v_{i+1}$ is a direct successor of $v_i$.

**De nition 10 ($\leftarrowtail$-invariant, $m$-blocking)** Let $\mathcal{A} = (Q, \quad, I)$ be a looping tree automaton and $\leftarrowtail$ be a binary relation over $Q$, called the *blocking relation*. If $q \leftarrowtail p$, then we say that $q$ is *blocked* by $p$. The automaton $\mathcal{A}$ is called $\leftarrowtail$*-invariant* if, for every $q \leftarrowtail p$, and $(q_0, q_1, \ldots, q_{i-1}, q, q_{i+1}, \ldots, q_k) \in$ , it holds that $(q_0, q_1, \ldots, q_{i-1}, p, q_{i+1}, \ldots, q_k) \in$ . A $\leftarrowtail$-invariant automaton $\mathcal{A}$ is called $m$*-blocking* if, for every successful run $r$ of $\mathcal{A}$ and every path $v_1, \ldots, v_m$ of length $m$ in $r$, there are $1 \quad i < j \quad m$ such that $r(v_j) \leftarrowtail r(v_i)$.

Obviously, any looping automaton $\mathcal{A} = (Q, \quad, I)$ is $=$-invariant and $m$-blocking for every $m > \#Q$ (where $\#Q$ denotes the cardinality of $Q$). However, we are interested in automata and blocking relations where blocking occurs earlier than after a linear number of transitions.

To test an $m$-blocking automaton for emptiness, it is su cient to construct partial runs of depth $m$. More formally, we de ne $K^{n} := \bigcup_{i=0}^{n} K^i$. A *partial run of depth* $m$ is a mapping $r : K^{m-1} \to Q$ such that $(r(v), r(v1), \ldots, r(vk)) \in$ for all $v \in K^{m-2}$. It is *successful* if $r(\varepsilon) \in I$.

**Lemma 11** An $m$-blocking automaton $\mathcal{A} = (Q, \quad, I)$ has a successful run i it has a successful partial run of depth $m$.

**Proof.** The "only if" direction is trivial, so only the "if" direction will be proved. For this purpose, we will show how to construct a complete successful

run from a partial one by replacing, for every blocked node $v \leftarrowtail w$, the subtree starting at $v$ with the subtree starting at $w$.

Suppose there is a successful partial run $r$ of depth $m$. This run will be used to construct a function $\quad : K \ \rightarrow K^{\ m}$ inductively as de ned below. The intuitive meaning of $\ (v) = w$ is "$w$ stands for $v$", i.e. we will use the labels of $w$ and $w$'s successors in the partial run also for $v$ and $v$'s successors in the complete run.

$(\varepsilon) := \varepsilon$,

for a node $v\ i$, if there is a predecessor $w$ of $\ (v)\ i$ such that $r(\ (v)\ i) \leftarrowtail r(w)$, then $\ (v\ \ i) := w$; and $\ (v\ \ i) := \ (v)\ \ i$ otherwise.

In the following, we will refer to (direct or indirect) successors of blocked nodes as *indirectly blocked*. Notice that the range of $\quad$ does not contain any blocked or indirectly blocked nodes, since we start with an non-blocked node and, whenever we encounter a blocked node, we replace it and its successors with the blocking one and its successors. Moreover, for every node $v$ with $\ (v) \neq v$, the depth of $v$, $|v|$, is larger than $|\ (v)|$, because $\quad$ maps a blocked node to a predecessor and the child of a blocked node to a child of the predecessor etc.

We will now show by induction over $|v|$ that the function $\quad$ is well-de ned, more precisely that $|\ (v)| < m$ for all $v \in K$ , and that we can use $\quad$ to construct a successful run $s$ from the successful partial run $r$ by setting, for every node $v$, $s(v) := r(\ (v))$. For the root, $s(\varepsilon) = r(\varepsilon)$ holds, thus both $s$ and $r$ start with the same label. If, for any node $v$, the successors of $v$ are not blocked, then the transition $(s(v), s(v\ \ 1), \ldots, s(v\ \ k))$ is contained in $\quad$ because $(r(\ (v)), r(\ (v)\ \ 1), \ldots, r(\ (v)\ \ k))$ is a transition in the run $r$. In this case, since $\ (v)$ is not blocked or indirectly blocked, $|\ (v)\ \ i| < m$ for all $1 \quad i \quad k$, because otherwise the path to $\ (v)\ \ i$ would have length at least $m$ without containing a blocked node, in contradiction with the induction hypothesis that the part of $s$ constructed so far is part of a successful run and that neither $\ (v)$ nor any of its predecessors is blocked.

If any successors of $v$ are blocked, i.e. $r(v\ \ i) \leftarrowtail r(w)$ then $(r(\ (v))$, $r(\ (v)\ \ 1)$, ..., $r(\ (v)\ \ i)$, ..., $r(\ (v)\ \ k)) \in \quad$ implies $(r(\ (v)), r(\ (v)\ \ 1)$, ..., $r(\ (w)), \ldots, r(\ (v)\ \ k)) \in \quad$ because of the de nition of $\leftarrowtail$-invariance. Hence, $(s(v), s(v\ \ 1), \ldots, s(v\ \ k)) \in \quad$ , and $s$ is a successful run of $\mathcal{A}$. In this case, since $w$ is a predecessor of $\ (v)\ \ i$ and $|\ (v)| < m$, it holds that $|w| < m$, and thus $|\ (v\ \ i)| < m$. Observe that $w$ cannot be blocked itself because $\ (v)$ is a successor of $w$ or equal to $w$ and the range of $\quad$ does not contain blocked or indirectly blocked nodes, thus the range of $\quad$ only contains non-blocked nodes. ⊡

```
1: if I ≠ ∅ then
2:    guess an initial state q ∈ I
3: else
4:    return "empty"
5: end if
6: if there is a transition from q then
7:    guess such a transition (q, q_1, ..., q_k) ∈
8:    push(SQ, (q_1, ..., q_k)), push(SN, 0)
9: else
10:   return "empty"
11: end if
12: while SN is not empty do
13:   (q_1, ..., q_k) := pop(SQ), n := pop(SN) + 1
14:   if n    k then
15:      push(SQ, (q_1, ..., q_k)), push(SN, n)
16:      if length(SN) < m    1 then
17:         if there is a transition from q_n then
18:            guess a transition (q_n, q'_1, ..., q'_k) ∈
19:            push(SQ, (q'_1, ..., q'_k)), push(SN, 0)
20:         else
21:            return "empty"
22:         end if
23:      end if
24:   end if
25: end while
26: return "not empty"
```

Figure 1: The top-down emptiness test for $m$-blocking automata.

For $k > 1$, the size of a successful partial run of depth $m$ is still exponential in $m$. However, when checking for the existence of such a run, one can perform a depth- rst traversal of the run while constructing it. To do this, it is basically enough to have at most one path of length up to $m$ in memory.[3] The algorithm that realizes this idea is shown in Figure 1. It uses two stacks: the stack SQ stores, for every node in the current path, the right-hand side of the transition which led to this node, and the stack SN stores, for every node in the current path, on which component of this right-hand side we are currently working. If we refer to the depth of SN by $d$ and to the elements in SN by SN(1)[the bottom element], ..., SN($d$)[the top element], the next node to be checked is SN(1) SN(2) ... SN($d$) + 1. The entries of SQ and SN are elements of $Q^k$ and $K \cup \{0\}$, respectively, and the number of entries is bounded by $m$ for each stack.

Figure 2 shows the values stored in each of the stacks SQ and SN at the beginning of an iteration, and their relation with the traversal of the run. The circled nodes represent the path followed to reach the node about to be checked. The values of the elements of the stack are shown next to the depth in the run to which they correspond. For this reason, the stacks appear backwards, with their bottom element at the top of the gure, and vice versa.

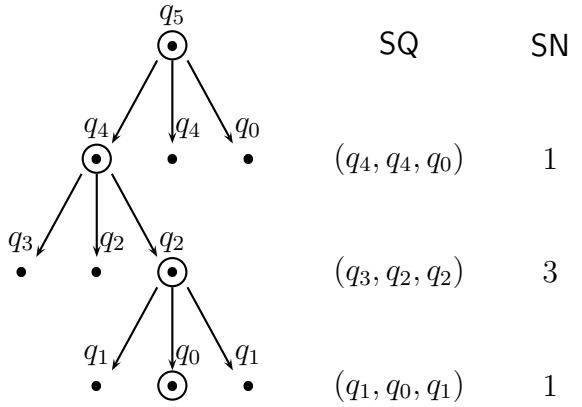


Figure 2: A run and the corresponding data structures.

After starting the algorithm, we rst guess an initial state and transition. If we can nd one, we push the labels of the nodes $1, \ldots, k$ onto SQ and the number 0 onto SN. Then we enter the while loop. As long as the stacks are not empty, we take the top elements of both stacks. If $n > k$ in line 14, this indicates that we have checked all nodes on this level, and we backtrack

[3]This is similar to the so-called trace technique for tableau-based algorithms [17].

without pushing anything on the stacks, which means that we will continue at the next upper level in the next loop. Otherwise, we store the information that we have to check our next sibling by pushing the same tuple of states onto SQ and the incremented number $n$ onto SN. Next, we test if we have already reached the maximum depth in line 16. If the answer is yes, we backtrack, otherwise we try to nd a transition from the current node and if there is one, we push the relevant information on the stacks, which means that we will descend into the next lower level in the next loop. If there is no transition, we reject the input.

Note that the algorithm does not require the automaton $\mathcal{A}$ to be explicitly given. It can be constructed on-the- y during the run of the algorithm.

**De nition 12** Assume that we have a set of inputs $\mathfrak{I}$ and a construction that yields, for every $\mathfrak{i} \in \mathfrak{I}$, an $m_\mathfrak{i}$-blocking automaton $\mathcal{A}_\mathfrak{i} = (Q_\mathfrak{i}, \ _\mathfrak{i}, I_\mathfrak{i})$ working on $k_\mathfrak{i}$-ary trees. We say that this construction is a PSPACE *on-the- y construction* if there is a polynomial $P$ such that, for every input $\mathfrak{i}$ of size $n$ we have

$m_\mathfrak{i} \quad P(n)$ and $k_\mathfrak{i} \quad P(n)$;

every element of $Q_\mathfrak{i}$ is of a size bounded by $P(n)$;

there is a $P(n)$-space bounded non-deterministic algorithm for guessing an element of $I_\mathfrak{i}$;

there is a $P(n)$-space bounded non-deterministic algorithm for guessing, on input $q \in Q_\mathfrak{i}$, a transition from $\ _\mathfrak{i}$ with rst component $q$.

The algorithms guessing an initial state (a transition starting with $q$) are assumed to yield the answer "no" if there is no initial state (no such transition).

The following theorem shows that the conditions in De nition 12 are su cient to ensure a PSPACE result.

**Theorem 13** If the automata $\mathcal{A}_\mathfrak{i}$ are obtained from the inputs $\mathfrak{i} \in \mathfrak{I}$ by a PSPACE on-the- y construction, then the emptiness problem for $\mathcal{A}_\mathfrak{i}$ can be decided by a deterministic algorithm in space polynomial in the size of $\mathfrak{i}$.

**Proof.** We will rst show by induction that if the algorithm described in Figure 1 answers "not empty", then the we can de ne a successful partial run $r$ from the $q_i$ values used by the algorithm. Since the algorithm answers "not empty", there is an initial transition $(q, q_1, \ldots, q_k)$. Then set $r(\varepsilon) = q$

and $r(i) = q_i$ for all $1 \quad i \quad k$. Suppose now that the algorithm visits a node $v = a_0 \ \ldots \ a_\ell \in K$ . Then, by induction hypothesis, $r$ is de ned for the previously visited nodes. If $\mathsf{length}(SP) < k$, then the algorithm guesses a transition, and $r(v \ \ i) = q'_i$ de nes a transition in the run. Otherwise, the algorithm has reached depth $m$, so we have reached the maximum depth of the partial run.

Conversely, if there is a successful partial run $r$, then it is possible to guess the initial state, and initial transition $(r(\varepsilon), r(1), \ldots, r(k))$. By De nition 12, the space required for guessing the initial state $r(\varepsilon)$ and the transition from $r(\varepsilon)$ is bounded by $P(n)$. When the algorithm visits one of these initial nodes, they have the same labels as in $r$. Now suppose the algorithm visits a node $v$ with $r(v) = q$. If the length of $v$ is smaller than $m$, then there is a transition on $r, (r(v), r(v \ \ 1), \ldots, r(v \ \ k))$ which the algorithm can guess (using space bounded by $P(n)$) and so it will not return "empty". At any time, the stack $\mathsf{SQ}$ contains at most $m_{\mathfrak{i}}$ tuples of $k_{\mathfrak{i}}$ states and $\mathsf{SN}$ contains at most $m_{\mathfrak{i}}$ numbers between 0 and $k_{\mathfrak{i}}$. Since $m_{\mathfrak{i}}$, $k_{\mathfrak{i}}$ and the size of each state are bounded by $P(m)$, the space used by these stacks is polynomial in the size of $\mathfrak{i}$.

It follows from Lemma 11 that this emptiness test is sound and complete. From Savitch's theorem [15] we obtain the deterministic complexity class.

□

## 3.3 Satis ability in $\mathcal{SI}$ w.r.t. acyclic TBoxes

It is easy to see that the construction of the automaton $\mathcal{A}_{C,\mathcal{T}}$ from a given $\mathcal{SI}$ concept $C$ and a general TBox $\mathcal{T}$ satis es all but one of the conditions of a PSpace on-the- y construction. The condition that is violated is the one requiring that blocking must occur after a polynomial number of steps. In the case of general TBoxes, this is not surprising since we know that the satis ability problem is ExpTime-hard. Unfortunately, this condition is also violated if $\mathcal{T}$ is an acyclic TBox. The reason is that successor states may contain new concepts that are not really required by the de nition of $C, \mathcal{T}$-compatible tuples, but are also not prevented by this de nition. In the case of acyclic TBoxes, we can construct a subautomaton that avoids such unnecessary concepts. It has less runs than $\mathcal{A}_{C,\mathcal{T}}$, but it does have a successful run whenever $\mathcal{A}_{C,\mathcal{T}}$ has one. The construction of this subautomaton follows the following general pattern.

**De nition 14 (Faithful)** Let $\mathcal{A} = (Q, \quad , I)$ be a looping tree automaton on $k$-ary trees. The family of functions $f_q : Q \to Q^{\mathrm{S}}$ for $q \in Q^{\mathrm{S}}$ is *faithful*

w.r.t. $\mathcal{A}$ if $I$ $Q^{\mathrm{S}}$ $Q$, and the following two conditions are satis ed for every $q \in Q^{\mathrm{S}}$:

1. if $(q, q_1, \ldots, q_k) \in$ , then $(q, f_q(q_1), \ldots, f_q(q_k)) \in$ ;
2. if $(q_0, q_1, \ldots, q_k) \in$ , then $(f_q(q_0), f_q(q_1), \ldots, f_q(q_k)) \in$ . [4]

The *subautomaton* $\mathcal{A}^S = (Q^S, \ ^S, I)$ *of* $\mathcal{A}$ *induced by this family* has the transition relation $\ ^{\mathrm{S}} := \{(q, f_q(q_1), \ldots, f_q(q_k)) \mid (q, q_1, \ldots, q_k) \in$ and $q \in Q^{\mathrm{S}}\}$.

Instead of testing $\mathcal{A}$ for emptiness, we can equivalently test $\mathcal{A}^{\mathrm{S}}$.

**Lemma 15** Let $\mathcal{A}$ be a looping tree automaton and $\mathcal{A}^{\mathrm{S}}$ its subautomaton induced by the faithful family of functions $f_q : Q \to Q^{\mathrm{S}}$ for $q \in Q^{\mathrm{S}}$. Then $\mathcal{A}$ has a successful run i $\mathcal{A}^{\mathrm{S}}$ has a successful run.

**Proof.** Since every successful run of $\mathcal{A}^{\mathrm{S}}$ is also a successful run of $\mathcal{A}$, the "if" direction is obvious. For the "only if" direction, we will show how to transform a successful run $r$ of $\mathcal{A}$ into a successful run $s$ of $\mathcal{A}^{\mathrm{S}}$. To do this, we traverse $r$ breadth- rst, creating an intermediate run $\hat{r}$, which initially is equal to $r$. At every node $v \in K$ , we replace the labels of the direct and indirect successors of $v$ with their respective $f_{\hat{r}(v)}$ values (see De nition 14). More formally, at node $v$, we replace $\hat{r}(w)$ with $f_{\hat{r}(v)}(\hat{r}(w))$ for all $w \in \{v \ \ u \mid u \in K^+\}$. By De nition 14, $\hat{r}$ is still a successful run after the replacement (note that condition 2 is necessary to ensure transitions from the successors of $v$). We de ne $s$ as the value of $\hat{r}$ "in the limit", i.e. for every node $v$, $s(v)$ has the value of $\hat{r}(v)$ after $v$ has been processed. ⧈

Before we can de ne an appropriate family of functions for $\mathcal{A}_{C,\mathcal{T}}$, we must introduce some notation. For an $\mathcal{SI}$ concept $C$ and an acyclic TBox $\mathcal{T}$, the *role depth* $\mathsf{rd}_{\mathcal{T}}(C)$ of $C$ w.r.t. $\mathcal{T}$ is the maximal nesting of (universal and existential) role restrictions in the concept obtained by expanding $C$ w.r.t. $\mathcal{T}$:

$$\mathsf{rd}_{\mathcal{T}}(A) = 0 \text{ for a primitive concept name } A,$$

$$\mathsf{rd}_{\mathcal{T}}(A) = \mathsf{rd}_{\mathcal{T}}(C) \text{ for a de ned concept } A \doteq C,$$

$$\mathsf{rd}_{\mathcal{T}}(\neg C) = \mathsf{rd}_{\mathcal{T}}(C),\ \mathsf{rd}_{\mathcal{T}}(C \sqcup D) = \mathsf{rd}_{\mathcal{T}}(C \sqcap D) = \max\{\mathsf{rd}_{\mathcal{T}}(C), \mathsf{rd}_{\mathcal{T}}(D)\},$$

$$\mathsf{rd}_{\mathcal{T}}(\exists r.C) = \mathsf{rd}_{\mathcal{T}}(\forall r.C) = \mathsf{rd}_{\mathcal{T}}(C) + 1,$$

[4] Note that this condition does neither imply nor follow from condition 1, since $q_0$ need not be equal to $q$, and it is not required that $f_q(q)$ equals $q$.

$\mathsf{rd}_{\mathcal{T}}(S) = \max\{\mathsf{rd}_{\mathcal{T}}(D) \mid D \in S\}$ for a set of concepts $S$.

Obviously, $\mathsf{rd}_{\mathcal{T}}(C)$ is polynomially bounded by the size of $C, \mathcal{T}$. For a set of $\mathcal{SI}$ concepts $S$, its role depth $\mathsf{rd}_{\mathcal{T}}(S)$ w.r.t. $\mathcal{T}$ is the maximal role depth w.r.t. $\mathcal{T}$ of the elements of $S$. We de ne $\mathsf{sub}_{\leqslant n}(C, \mathcal{T}) := \{D \mid D \in \mathsf{sub}(C, \mathcal{T}) \wedge \mathsf{rd}_{\mathcal{T}}(D) \quad n\}$, and $S/r := \{D \in S \mid \text{there is an } E \text{ such that } D = \forall r.E\}$.

The main idea underlying the next de nition is the following. If $\mathcal{T}$ is acyclic, then the de nition of $C, \mathcal{T}$-compatibility requires, for a transition $(q, q_1, \ldots, q_k)$ of $\mathcal{A}_{C,\mathcal{T}}$, only the existence of concepts in $q_i = (\ _i, \ _i, \ _i, \varrho_i)$ that are of a smaller depth than the maximal depth $n$ of concepts in $q$ if $\varrho_i$ is not transitive. If $\varrho_i$ is transitive, then $\ _i$ may also contain universal restrictions of depth $n$. We can therefore remove from the states $q_i$ all concepts with a higher depth and still maintain $C, \mathcal{T}$-compatibility.

**De nition 16 (Functions $f_q$)** For two states $q = (\ , \ , \ , \varrho)$ and $q' = (\ ', \ ', \ ', \varrho')$ of $\mathcal{A}_{C,\mathcal{T}}$ with $\mathsf{rd}_{\mathcal{T}}(\ ) = n$, we de ne the function $f_q(q')$ as follows:

if $\mathsf{rd}_{\mathcal{T}}(\ ') \quad \mathsf{rd}_{\mathcal{T}}(\ )$, then $f_q(q') := (\emptyset, \emptyset, \emptyset, \ )$;

otherwise, $f_q(q') := (\ ', \ '', \ '', \varrho')$, where

- $P = \mathsf{sub}_{\leqslant n}(C, \mathcal{T})/\varrho'$, if $\mathsf{trans}(\varrho')$; otherwise $P = \emptyset$;
- $'' = \ ' \cap (\mathsf{sub}_{\leqslant n \ 1}(C, \mathcal{T}) \cup P)$;
- $'' = \ ' \cap (\mathsf{sub}_{\leqslant n \ 1}(C, \mathcal{T}) \cup \ '')$.

If $\mathcal{T}$ is acyclic, then the set $\ ''$ de ned above is still a $\mathcal{T}$-expanded Hintikka set.

**Lemma 17** The family of mappings $f_q$ (for states $q$ of $\mathcal{A}_{C,\mathcal{T}}$) introduced in De nition 16 is faithful w.r.t. $\mathcal{A}_{C,\mathcal{T}}$.

**Proof.** We have to show that both conditions of De nition 14 are satis ed.

*Condition 1.* The case that a successor is replaced by $(\emptyset, \emptyset, \emptyset, \ )$ cannot occur because in every successor $q_i$ of $q$, the role depth of $\ _i$ is strictly smaller than the maximum depth of $\ $. Assume that $(q, q_1, \ldots, q_k) \in \ $. To prove that $(q, f_q(q_1), \ldots, f_q(q_k))$ is also contained in $\ $, we have to show that this transition satis es the conditions for $C, \mathcal{T}$-compatibility in De nition 4. Number 1 and 4 are obvious. Number 3 holds because we do not remove anything from $\ $. Finally, we do not remove any concepts from the $\ _i$ sets, because these concepts have a maximum depth of $\mathsf{rd}_{\mathcal{T}}(\ )$, if $\varrho_i$ is transitive, or $\mathsf{rd}_{\mathcal{T}}(\ ) \quad 1$, otherwise. Thus, we only remove concepts from $\ _i$, and none of the removed concepts is required.

*Condition 2.* Let $(q_0, q_1, \ldots, q_k) \in$ . If for some $i > 0$ with $\varphi(\exists r.D) = i$, $q_i$ is replaced by $(\emptyset, \emptyset, \emptyset,\ )$, this means that for the concept $D \in\ _i$, $\mathsf{rd}_{\mathcal{T}}(D)$ $\mathsf{rd}_{\mathcal{T}}(\ )$. This implies that the corresponding existential concept $\exists r.D$ in $\ _0$ has a depth which is strictly larger than $\mathsf{rd}_{\mathcal{T}}(\ )$, and therefore will be removed from $f_q(q_0)$. Otherwise, we again have to show the four conditions from De nition 4. Number 1 and 4 are again obvious. For number 3, observe that if $\forall \overline{r}.F \in f_q(\ _i)$ with $\varrho_i = r$, then $\mathsf{rd}_{\mathcal{T}}(\forall \overline{r}.F) < n$ because $\overline{r} \neq \varrho_i$, and thus neither $F$ nor $\forall \overline{r}.F$ will be removed from $\ _0$. For number 2, if $\forall r.E \in f_q(\ _0)$, then it holds either that $\mathsf{rd}_{\mathcal{T}}(\forall r.E) < n$ or $\mathsf{rd}_{\mathcal{T}}(\forall r.E) = n$ and $\mathsf{trans}(r)$. In the former case, neither $E$ nor $\forall r.E$ will be removed from $\ _i$. In the latter case, $\forall r.E$ will not be removed because $\varrho_i = r$ and $\mathsf{trans}(r)$ holds. ⧈

Consequently, $\mathcal{A}_{C,\mathcal{T}}$ has a successful run i the induced subautomaton $\mathcal{A}^{\mathrm{S}}_{C,\mathcal{T}}$ has a successful run.

**Lemma 18** The construction of $\mathcal{A}^{\mathrm{S}}_{C,\mathcal{T}}$ from an input consisting of an $\mathcal{SI}$ concept $C$ and an acyclic TBox $\mathcal{T}$ is a PSPACE on-the- y construction.

**Proof.** Let $\mathfrak{i} = (C, \mathcal{T})$ be an input, i.e. an $\mathcal{SI}$ concept and TBox. We de ne $a(\mathfrak{i}) := \#\mathsf{sub}(C, \mathcal{T})$, which means that $a(\mathfrak{i})$ is at most quadratic in the size of $\mathfrak{i}$. The blocking relation $\hookleftarrow_{\mathcal{SI}}$ is de ned as follows: $(\ _1,\ \ _1,\ \ _1, \varrho_1) \hookleftarrow_{\mathcal{SI}} (\ _2,\ \ _2,\ \ _2, \varrho_2)$ if $\ _1 = \ _2,\ \ _1 = \ \ _2,\ \ _1/\overline{\varrho}_1 = \ _2/\overline{\varrho}_2$, and $\varrho_1 = \varrho_2$. We have to show that there is a polynomial $P(n)$ satisfying the conditions in De nition 12.

*Every element of $Q_{\mathfrak{i}}$ is of a size bounded by $P(n)$.* Every state label is a subset of $\mathsf{sub}(C, \mathcal{T})$ and therefore bounded by $a(\mathfrak{i})$.

*There is a $P(n)$-space bounded non-deterministic algorithm for guessing an initial state or successor states for a given state.* This is obvious, since the size of every state is bounded by $a(\mathfrak{i})$ and all necessary information for the successor states can be obtained from the current state.

*The automaton $\mathcal{A}^{S}_{C,\mathcal{T}}$ is operating on $k_{\mathfrak{i}}$-ary trees and $m_{\mathfrak{i}}$-blocking, with $m_{\mathfrak{i}}$ $P(n)$ and $k_{\mathfrak{i}}$ $P(m)$.* The tree width $k_{\mathfrak{i}}$ is bounded by the number of existential subconcepts of $\mathfrak{i}$ and therefore by $a(\mathfrak{i})$. In order to show a polynomial bound for $m_{\mathfrak{i}}$, we rst have to show that $\mathcal{A}^{\mathrm{S}}_{C,\mathcal{T}}$ is $\hookleftarrow_{\mathcal{SI}}$-invariant. For states $\{q, q_i\}$ $Q^{\mathrm{S}}$ with $q = (\ ,\ ,\ , \varrho)$ and $q_i = (\ _i,\ \ _i,\ \ _i, \varrho_i)$ let $(q_0, \ldots, q_j, \ldots, q_k)$ be a transition and $q_j \hookleftarrow_{\mathcal{SI}} q_i$. Then the tuple $(q_0, \ldots, q_i, \ldots, q_k)$ is also $C, \mathcal{T}$-compatible since $\ _j = \ \ _i,\ \ _j = \ \ _i$, $\varrho_j = \varrho_i$ and $\ _j$ contains the same universal concepts involving $\overline{\varrho}_j$ as $\ _i$.

What is the maximum depth of a blocked node in a successful run? Firstly, observe that transitions $(q, q_1, \ldots, q', \ldots, q_k)$ with $q = (\ ,\ ,\ , \varrho)$ and $q' = (\ ',\ \ ',\ \ ', \varrho')$ where $\varrho'$ is di erent from $\varrho$ or not transitive decrease

the maximum depth of concepts contained in the state: if $\varrho'$ is not transitive, then $\mathsf{rd}_{\mathcal{T}}(\ \ ')$ is smaller than $\mathsf{rd}_{\mathcal{T}}(\ \ )$ by de nition. If $\varrho'$ is transitive, but di erent from $\varrho$, then $\ \ '$ can only have concepts of depth $\mathsf{rd}_{\mathcal{T}}(\ \ )$ if these start with $\forall\varrho'$. Similarly, $\ \ $ can only contain concepts of the same depth as its predecessor state if they begin with $\forall\varrho$, which implies that the role depth decreases after two transitions. (This is the key to obtaining a polynomial bound, and it does not hold for general TBoxes, where the GCIs maintain the same role depth in every node.) This depth is bounded by the maximum depth in $\mathsf{sub}(\mathrm{C},\mathcal{T})$ and therefore by $a(\mathfrak{i})$, and thus there are only $a(\mathfrak{i})$ such steps possible before depth 0 is reached. After this point, the path will contain a blocked node, since all further nodes are labelled with $(\emptyset,\emptyset,\emptyset,\ \ )$.

So the role depth can only remain the same along a subpath (a subpath is a path which does not need to begin at $\varepsilon$) where every transition involves the same transitive role $r$. From the de nition of $\ \ $, it follows for any subpath with labels $(\ _0,\ \ _0,\ \ _0,r)$, $(\ _1,\ \ _1,\ \ _1,r)$, $\ldots$, $(\ _\ell,\ \ _\ell,\ \ _\ell,r)$, that $\ _i\ \ \ _{i+1}$, for all $1\ \ i\ \ \ell\ \ 1$, so there are at most $a(\mathfrak{i})$ di erent sets $\ _i$ possible. By the same argument, it also holds on this subpath that $\ _{i+1}/\overline{r}\ \ \ _i/\overline{r}, 1\ \ i\ \ \ell\ \ 1$. Once again, it is only possible to have a subpath of length $m$ with di erent sets. Finally, since $\ _i$ contains only one concept, there are also only $a(\mathfrak{i})$ possibilities for this set. In total, every $r$-subpath of length larger than $a(\mathfrak{i})^3$ must have $i<j$ such that $\ _j=\ \ _i$, $\ _j=\ \ _i$ and $\ _j/\overline{r}=\ \ _i/\overline{r}$, and hence $(\ _j,\ \ _j,\ \ _j,r)\leftarrowtail(\ _i,\ \ _j,\ \ _i,r)$. Thus, an $r$-subpath for a transitive role $r$ either contains a blocked node or is shorter than $a(\mathfrak{i})^3$ and therefore followed by a transition with a role other than $r$, which decreases the maximum depth of concepts contained in $\ \ $. Altogether, we obtain that every path which is longer than $a(\mathfrak{i})^4$ contains a blocked node.

This concludes the proof that the construction of $\mathcal{A}^{\mathrm{S}}_{C,\mathcal{T}}$ is a PSPACE on-the- y construction with $P(n)=n^8$. ⊡

Since we know that $C$ is satis able w.r.t. $\mathcal{T}$ i $\mathcal{A}_{C,\mathcal{T}}$ has a successful run i $\mathcal{A}^{\mathrm{S}}_{C,\mathcal{T}}$ has a successful run, Theorem 13 yields the desired PSPACE upper-bound.

**Theorem 19** Satis ability in $\mathcal{SI}$ w.r.t. acyclic TBoxes is in PSPACE.

PSPACE-hardness for this problem follows directly from the known PSPACE-hardness of satis ability w.r.t. the empty TBox in $\mathcal{ALC}$ [17].

# 4 Conclusion

We have identi ed a class of automata for which emptiness can be tested in a manner that is more e cient than the standard deterministic bottom-up

emptiness test. The key to obtaining this result is the employment of the *blocking* technique known from tableau algorithms, which allows us to use instead a a nondeterministic top-down emptiness test which can be aborted after a "blocked" state is reached. If the number of transitions before this happens is polynomial in the size of the input, emptiness of the automaton can be tested on-the- y using space polynomial in the size of the input rather than time exponential in the size of the input.

As an example for the application of this method, we have shown how blocking automata can be used to decide satis ability of $\mathcal{SI}$ concepts w.r.t. acyclic TBoxes in PSpace.

# References


[1] F. Baader, H.-J. Burckert, B. Hollunder, W. Nutt, and J.H. Siekmann. Concept logics. In *Computational Logics, Symposium Proceedings*, Springer-Verlag, 1990.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider (eds). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[3] F. Baader, J. Hladik, C. Lutz, and F. Wolter. From tableaux to automata for description logics. *Fundamenta Informaticae*, 57(2–4):247–279, 2003.

[4] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[5] F. Baader and S. Tobies. The inverse method implements the automata approach for modal satis ability. In *Proc. IJCAR 2001*, Sprincer LNCS 2083, 2001.

[6] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[7] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive DLs with xpoints based on automata on in nite trees. In *Proc. IJCAI'99*, 1999.

[8] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel, 1983.

[9] J. Hladik and R. Peñaloza. PSPACE automata for description logics. In *Proceedings of DL 2006*, CEUR Workshop Proceedings, 2006.

[10] I. Horrocks and P.F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.

[11] I. Horrocks, U. Sattler, and S. Tobies. A PSpace-algorithm for deciding $\mathcal{ALCNI}_{R^+}$-satis ability. LTCS-Report 98-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1998.

[12] C. Lutz. Complexity of terminological reasoning revisited. In *Proc. LPAR'99*, Springer LNAI 1705, 1999.

[13] B. Nebel. Terminological reasoning is inherently intractable. *Arti cial Intelligence*, 43:235–249, 1990.

[14] U. Sattler. A concept language extended with di erent kinds of transitive roles. In *Proc. KI'96*, Springer LNAI 1137, 1996.

[15] W.J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. of Computer and System Sciences*, 4:177–192, 1970.

[16] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. IJCAI'91*, 1991.

[17] M. Schmidt-Schau and G. Smolka. Attributive concept descriptions with complements. *Arti cial Intelligence*, 48(1):1–26, 1991.

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

[19] A. Voronkov. How to optimize proof-search in modal logics: new methods of proving reduncancy criteria for sequent calculi. *ACM Transactions on Computational Logic*, 2(2), 2001.