2ATAs make DLs easy

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{calvanese,degiacomo,lenzerini}@dis.uniroma1.it

Abstract

In this paper we demonstrate that two-way alternating automata on infinite trees (2ATAs) provide a very elegant and effective formal tool for addressing reasoning in expressive DLs. Indeed, the encoding of a DL concept (to be checked for satisfiability) into an automaton (to be checked for non-emptiness) is: (i) intuitive, indeed, comparable to tableaux rules; (ii) modular, since each construct is dealt with separately; (iii) short, since the encoding is polynomial; and (iv) computationally adequate, i.e., optimally w.r.t. the complexity class of reasoning. To make these claims concrete, we illustrate the use of 2ATAs to decide satisfiability of three expressive DLs of increasing complexity, namely $\mathcal{ALCFI}_{reg}$, which corresponds to *converse*-PDL with local functionality, $\mathcal{ALCFI}b_{reg}$, which extends $\mathcal{ALCFI}_{reg}$ with boolean combinations of roles, and $\mathcal{ALCQI}b_{reg}$, which further extends $\mathcal{ALCFI}b_{reg}$ with qualified number restrictions.

1 Introduction

Vardi introduced in [14] techniques based on *two-way alternating automata on infinite trees* (2ATAs) for reasoning in modal logics of programs. These techniques have been adopted for devising new results for DLs with fixpoints [2, 10]. In this paper we demonstrate that 2ATAs provide indeed a very elegant and effective formal tool for addressing reasoning in expressive DLs. In particular, differently from usual (one-way nondeterministic) tree automata [12], they provide an high level description of the automaton computation that abstracts from the combinatorics and allows one to concentrate on the logical aspects. As a result, the encoding of a DL concept (to be checked for satisfiability) into an automaton (to be checked for non-emptiness) is:

- *intuitive*, indeed, comparable to tableaux rules;
- *modular*, since each construct is dealt with separately;
- *short*, since the encoding is polynomial;
- *computationally adequate*, i.e., optimally w.r.t. the complexity class of reasoning.

Moreover, the typical difficulty of tableau-based techniques for expressive DLs, namely establishing suitable termination conditions, can be dealt with elegantly using appropriate acceptance conditions for the 2ATAs.

To make these claims concrete, we illustrate the use of 2ATAs to decide satisfiability of three expressive DLs of increasing complexity, namely $\mathcal{ALCFI}_{reg}$, which corresponds to *converse*-PDL [5] with local functionality, $\mathcal{ALCFI}b_{reg}$, which extends $\mathcal{ALCFI}_{reg}$ with boolean combinations of roles [13], and $\mathcal{ALCQI}b_{reg}$, which further extends $\mathcal{ALCFI}b_{reg}$ with qualified number restrictions [6], the most expressive form of number restrictions considered in DLs.

2 Automata on infinite trees

Infinite trees are represented as prefix closed (infinite) sets of words over $\mathbb{N}$ (the set of positive natural numbers). Formally, an *infinite tree* is a set of words $T \subseteq \mathbb{N}^*$, such that if $x \cdot c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$. The elements of T are called *nodes*, the empty word ε is the *root* of T, and for every $x \in T$, the nodes $x \cdot c$, with $c \in \mathbb{N}$, are the *successors* of x. By convention we take $x \cdot 0 = x$, and $x \cdot i \cdot -1 = x$. The *branching degree* $d(x)$ of a node x denotes the number of successors of x. If the branching degree of all nodes of a tree is bounded by k, we say that the tree has branching degree k. An *infinite path* P of T is a prefix-closed set $P \subseteq T$ such that for every $i \geq 0$ there exists a unique node $x \in P$ with $|x| = i$. A *labeled tree* over an alphabet Σ is a pair (T, V), where T is a tree and $V : T \to \Sigma$ maps each node of T to an element of Σ.

Alternating automata on infinite trees are a generalization of nondeterministic automata on infinite trees, introduced in [9]. They allow for an elegant reduction of decision problems for temporal and program logics [3, 1]. Let $\mathcal{B}(I)$ be the set of positive boolean formulae over I, built inductively by applying $\wedge$ and $\vee$ starting from **true**, **false**, and elements of I. For a set $J \subseteq I$ and a formula $\varphi \in \mathcal{B}(I)$, we say that J *satisfies* φ if and only if, assigning **true** to the elements in J and **false** to those in $I \setminus J$, makes φ true. For a positive integer k, let $[k] = \{-1, 0, 1, \ldots, k\}$. A *two-way alternating tree automaton* (2ATA) running over infinite trees with branching degree k, is a tuple $\mathbf{A} = \langle \Sigma, Q, \delta, q_0, F \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}([k] \times Q)$ is the transition function, $q_0 \in Q$ is the initial state, and F specifies the acceptance condition.

The transition function maps a state $q \in Q$ and an input letter $\sigma \in \Sigma$ to a positive boolean formula over $[k] \times Q$. Intuitively, if $\delta(q, \sigma) = \varphi$, then each pair (c, q') appearing in φ corresponds to a new copy of the automaton going to the direction suggested by c and starting in state q'. For example, if $k = 2$ and $\delta(q_1, \sigma) = (1, q_2) \wedge (1, q_3) \vee (-1, q_1) \wedge (0, q_3)$, when the automaton is in the state q_1 and is reading the node x labeled by the letter σ, it proceeds either by sending off two copies, in the states q_2 and q_3 respectively, to the first successor of x (i.e., $x \cdot 1$), or by sending off one copy in the state q_1 to the predecessor of x (i.e., $x \cdot -1$) and one copy in the state q_3 to x itself (i.e., $x \cdot 0$).

A run of a 2ATA $\mathbf{A}$ over a labeled tree (T, V) is a labeled tree (T_r, r) in which every node is labeled by an element of $T \times Q$. A node in T_r labeled by (x, q) describes a copy of $\mathbf{A}$ that is in the state q and reads the node x of T. The labels of adjacent nodes have to satisfy the transition function of $\mathbf{A}$. Formally, a run (T_r, r) is a $T \times Q$-labeled tree satisfying:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_0)$.
2. Let $y \in T_r$, with $r(y) = (x, q)$ and $\delta(q, V(x)) = \varphi$. Then there is a (possibly empty) set $S = \{(c_1, q_1), \ldots, (c_n, q_n)\} \subseteq [k] \times Q$ such that:
   - S satisfies φ and
   - for all $1 \leq i \leq n$, we have that $y{\cdot}i \in T_r$, $x{\cdot}c_i$ is defined, and $r(y{\cdot}i) = (x{\cdot}c_i, q_i)$.

A run (T_r, r) is *accepting* if all its infinite paths satisfy the acceptance condition[1]. Given an infinite path $P \subseteq T_r$, let $\mathit{inf}(P) \subseteq Q$ be the set of states that appear infinitely often in P (as second components of node labels). We consider here *Büchi* acceptance conditions. A Büchi condition over a state set Q is a subset F of Q, and an infinite path P satisfies F if $\mathit{inf}(P) \cap F \neq \emptyset$.

The nonemptiness problem for 2ATAs consists in determining, for a given 2ATA, whether the set of trees it accepts is nonempty. The results in [14] provide the following complexity characterization of nonemptiness of 2ATAs.

Theorem 2.1 *Given a 2ATA* $\mathbf{A}$ *with* n *states and an input alphabet with* m *elements, deciding nonemptiness of* $\mathbf{A}$ *can be done in time exponential in* n *and polynomial in* m.

3 Reasoning in $\mathcal{ALCFI}_{reg}$

We consider the DL $\mathcal{ALCFI}_{reg}$, defined by the following syntax:

$$\begin{array}{rcl} C, C' & \longrightarrow & A \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \forall R.C \mid \exists R.C \mid (\leq 1\, Q) \\ Q & \longrightarrow & P \mid P^- \\ R, R' & \longrightarrow & Q \mid R \cup R' \mid R \circ R' \mid R^* \mid \mathit{id}(C) \end{array}$$

where A and P denote respectively atomic concepts and atomic roles, and C and R denote respectively arbitrary concepts and roles. We use Q to denote *basic roles*, which for $\mathcal{ALCFI}_{reg}$ are either atomic or inverses of atomic roles. The semantics is the standard one. We concentrate on *concept satisfiability*, i.e., determining whether there exists an interpretation in which the concept has a nonempty extension. By virtue of internalization [11], the results below hold also for reasoning services over knowledge bases (in particular for logical implication).

We show how to decide satisfiability of $\mathcal{ALCFI}_{reg}$ concepts by reducing it to nonemptiness of 2ATAs. To this end we first define the *(syntactic) closure* for $\mathcal{ALCFI}_{reg}$, which extends the standard Fischer-Ladner for *converse*-PDL [4], by treating functional restrictions as atomic concepts. For technical reasons we include in the closure also additional elements representing basic roles and their negations. In particular, the closure $CL_{\mathcal{F}}(C_0)$ of an $\mathcal{ALCFI}_{reg}$ concept C_0 is defined as the smallest set

[1] No condition is imposed on the finite paths of the run.

of concepts such that $C_0 \in CL_{\mathcal{F}}(C_0)$ and such that (assuming $\sqcup$ and $\forall$ to be expressed by means of $\sqcap$ and $\exists$, and using Q^- to denote P when $Q = P^-$)[2]:

if	$C \in CL_{\mathcal{F}}(C_0)$	then	$\neg C \in CL_{\mathcal{F}}(C_0)$ (if C is not of the form $\neg C'$)
if	$\neg C \in CL_{\mathcal{F}}(C_0)$	then	$C \in CL_{\mathcal{F}}(C_0)$
if	$C \sqcap C' \in CL_{\mathcal{F}}(C_0)$	then	$C, C' \in CL_{\mathcal{F}}(C_0)$
if	$\exists R.C \in CL_{\mathcal{F}}(C_0)$	then	$C \in CL_{\mathcal{F}}(C_0)$
if	$\exists(R \cup R').C \in CL_{\mathcal{F}}(C_0)$	then	$\exists R.C, \exists R'.C \in CL_{\mathcal{F}}(C_0)$
if	$\exists(R \circ R').C \in CL_{\mathcal{F}}(C_0)$	then	$\exists R.\exists R'.C \in CL_{\mathcal{F}}(C_0)$
if	$\exists R^*.C \in CL_{\mathcal{F}}(C_0)$	then	$\exists R.\exists R^*.C \in CL_{\mathcal{F}}(C_0)$
if	$\exists id(C).C' \in CL_{\mathcal{F}}(C_0)$	then	$C \in CL_{\mathcal{F}}(C_0)$
if	$\exists Q.C \in CL_{\mathcal{F}}(C_0)$	then	$Q, Q^-, \neg Q, \neg Q^- \in CL_{\mathcal{F}}(C_0)$

The cardinality of $CL_{\mathcal{F}}(C_0)$ is linear in the length of C_0.

It can be shown, following the lines of the proof in [15] for converse deterministic PDL, that $\mathcal{ALCFI}_{reg}$ enjoys the *tree-model property*, i.e., every satisfiable concept has a model that has the structure of a (possibly infinite) tree with branching degree linearly bounded by the size of the concept. More precisely, we have the following result.

Theorem 3.1 *Every satisfiable $\mathcal{ALCFI}_{reg}$ concept C_0 has a tree model with branching degree k_{C_0} equal to twice the number of elements of $CL_{\mathcal{F}}(C_0)$.*

This property allows us to check satisfiability of an $\mathcal{ALCFI}_{reg}$ concept C_0 by building a 2ATA that accepts the (labeled) trees that correspond to tree models of C_0. Let $\mathcal{A}$ be the set of atomic concepts appearing in C_0, and $\mathcal{B}$ the set of atomic roles appearing in C_0 and their inverses. We construct from the $\mathcal{ALCFI}_{reg}$ concept C_0 a 2ATA $\mathbf{A}^{\mathcal{F}}_{C_0}$ that checks that C_0 is satisfied at the root of the input tree. We represent in each node of the tree the information about which atomic concepts are true in the node, and about the basic role that connects the predecessor of the node to the node itself (except for the root). More precisely, we label each node x with a set σ of atomic concepts and basic roles. The atomic concepts in σ are those that are true in x, and (for $\mathcal{ALCFI}_{reg}$) σ contains, except for the root, a single basic role, which is the one through which x is reached from its predecessor. That is, if Q stands for an atomic role P, then x is reached from its predecessor through P, and if Q stands for P^-, then the predecessor is reached from x through P. In the root, σ contains no basic role.

Given an $\mathcal{ALCFI}_{reg}$ concept C_0, we construct an automaton $\mathbf{A}^{\mathcal{F}}_{C_0}$ that accepts trees that correspond to tree models of C_0. For technical reasons, it is convenient to consider concepts in *negation normal form* (i.e., negations are pushed inside as much as possible). It is easy to check that the transformation of a concept into an equivalent one in negation normal form can be performed in linear time in the size of the concept. Below, we denote by $nnf(C)$ the negation normal form of C, and with $CL^{nnf}_{\mathcal{F}}(C_0)$ the set $\{nnf(C) \mid C \in CL_{\mathcal{F}}(C_0)\}$. The automaton $\mathbf{A}^{\mathcal{F}}_{C_0} = (\Sigma, S, \delta, s_{ini}, F)$ is defined as follows.

[2] We remind that C and C' stand for arbitrary concepts, and R and R' stand for arbitrary roles.

- The alphabet is $\Sigma = \bigcup_{Q \in \mathcal{B}} 2^{\mathcal{A} \cup \{Q\}}$, i.e., all sets consisting of atomic concepts and at most one basic role. This corresponds to labeling each node of the tree with a truth assignment to the atomic concepts, and with the role used to reach the node from its predecessor.

- The set of states is $S = \{s_{ini}\} \cup CL_{\mathcal{F}}^{nnf}(C_0)$, where s_{ini} is the initial state[3]. Intuitively, when the automaton is in a state $\sigma \in CL_{\mathcal{F}}^{nnf}(C_0)$ and visits a node x of the tree, this means that the automaton has to check that σ holds in x. When σ is an atomic concept A (resp. a basic role Q), this amounts to check that the node label contains A (resp. Q).

- The set F of final states is the set of concepts in $CL_{\mathcal{F}}^{nnf}(C_0)$ of the form $\forall R^*.C$. Observe that concepts of the form $\exists R^*.C$ are not final states, and this is sufficient to guarantee that such concepts are satisfied in all accepting runs of the automaton.

- The transition function δ is defined below.

1. For each $\sigma \in 2^{\mathcal{A}}$, i.e., containing no basic role, there is a transition from the initial state
$$\delta(s_{ini}, \sigma) \quad = \quad (0, nnf(C_0))$$
Such a transition checks that the root of the tree is not labeled with any basic role, and moves to the state that verifies C_0 in the root itself.

2. For each $\sigma \in \Sigma$, and each atomic concept or basic role $s \in \mathcal{A} \cup \mathcal{B}$ there are transitions
$$\begin{aligned} \delta(s, \sigma) &= \begin{cases} \textbf{true} & \text{if } s \in \sigma \\ \textbf{false} & \text{if } s \notin \sigma \end{cases} \\ \delta(\neg s, \sigma) &= \begin{cases} \textbf{true} & \text{if } s \notin \sigma \\ \textbf{false} & \text{if } s \in \sigma \end{cases} \end{aligned}$$
For $s \in \mathcal{A}$, such transitions check the truth value of atomic concepts and their negations in the current node of the tree. For $s \in \mathcal{B}$, such transitions check through which role the current node is reached.

[3]Recall that $CL_{\mathcal{F}}^{nnf}(C_0)$ contains also the atomic roles appearing in C_0 and their inverses.

3. For the concepts in $CL^{nnf}_{\mathcal{F}}(C_0)$ and each $\sigma \in \Sigma$ there are transitions

$$
\begin{array}{rcl}
\delta(C \sqcap C', \sigma) & = & (0, C) \ \wedge \ (0, C') \\
\delta(C \sqcup C', \sigma) & = & (0, C) \ \vee \ (0, C') \\
\delta(\forall Q.C, \sigma) & = & ((0, \neg Q^-) \vee (-1, C)) \ \wedge \ \bigwedge_{1 \leq i \leq k_{C_0}} ((i, \neg Q) \vee (i, C)) \\
\delta(\forall (R \cup R').C, \sigma) & = & (0, \forall R.C) \ \wedge \ (0, \forall R'.C) \\
\delta(\forall (R \circ R').C, \sigma) & = & (0, \forall R.\forall R'.C) \\
\delta(\forall R^*.C, \sigma) & = & (0, C) \ \wedge \ (0, \forall R.\forall R^*.C) \\
\delta(\forall id(C).C', \sigma) & = & (0, nnf(\neg C)) \ \vee \ (0, C') \\
\delta(\exists Q.C, \sigma) & = & ((0, Q^-) \wedge (-1, C)) \ \vee \ \bigvee_{1 \leq i \leq k_{C_0}} ((i, Q) \wedge (i, C)) \\
\delta(\exists (R \cup R').C, \sigma) & = & (0, \exists R.C) \ \vee \ (0, \exists R'.C) \\
\delta(\exists (R \circ R').C, \sigma) & = & (0, \exists R.\exists R'.C) \\
\delta(\exists R^*.C, \sigma) & = & (0, C) \ \vee \ (0, \exists R.\exists R^*.C) \\
\delta(\exists id(C).C', \sigma) & = & (0, C) \ \wedge \ (0, C')
\end{array}
$$

All such transitions, except for those involving $\forall R^*.C$ and $\exists R^*.C$, inductively decompose concepts and roles, and move to appropriate states of the automaton and nodes of the tree. The transitions involving $\forall R^*.C$ treat $\forall R^*.C$ as the equivalent concept $C \sqcap \forall R.\forall R^*.C$, and the transitions involving $\exists R^*.C$ treat $\exists R^*.C$ as the equivalent concept $C \sqcup \exists R.\exists R^*.C$.

4. For each concept of the form $(\leq 1\, Q)$ in $CL^{nnf}_{\mathcal{F}}(C)$ and each $\sigma \in \Sigma$ there is a transition

$$
\begin{array}{rcl}
\delta((\leq 1\, Q), \sigma) & = & ((0, Q^-) \ \wedge \ \bigwedge_{1 \leq i \leq k_{C_0}} (i, \neg Q)) \ \vee \\
 & & ((0, \neg Q^-) \ \wedge \ \bigwedge_{1 \leq i < j \leq k_{C_0}} ((i, \neg Q) \vee (j, \neg Q)))
\end{array}
$$

Such transitions check that, for a node x labeled with $(\leq 1\, Q)$, there exists at most one node (among the predecessor and the successors of x) reachable from x through Q.

5. For each concept of the form $\neg(\leq 1\, Q)$ in $CL^{nnf}_{\mathcal{F}}(C)$ and each $\sigma \in \Sigma$ there is a transition

$$
\begin{array}{rcl}
\delta(\neg(\leq 1\, Q), \sigma) & = & ((0, Q^-) \ \wedge \ \bigvee_{1 \leq i \leq k_{C_0}} (i, Q)) \ \vee \\
 & & \bigvee_{1 \leq i < j \leq k_{C_0}} ((i, Q) \wedge (j, Q))
\end{array}
$$

Such transitions check that, for a node x labeled with $\neg(\leq 1\, Q)$, there exist at least two nodes (among the predecessor and the successors of x) reachable from x through Q.

A run of the automaton $\mathbf{A}^{\mathcal{F}}_{C_0}$ on an infinite tree starts in the root checking that C_0 holds there (item 1 above). It does so by inductively decomposing $nnf(C_0)$ while appropriately navigating the tree (item 3) until it arrives to atomic concepts, functional restrictions, and their negations. These are checked locally (items 2, 4 and 5). Concepts of the form $\forall R^*.C$ and $\exists R^*.C$ are propagated using the equivalent concepts

$C \sqcap \forall R.\forall R^*.C$ and $C \sqcup \exists R.\exists R^*.C$, respectively. It is only the propagation of such concepts that may generate infinite branches in a run. Now, a run of the automaton may contain an infinite branch in which $\exists R^*.C$ is always resolved by choosing the disjunct $\exists R.\exists R^*.C$, without ever choosing the disjunct C. This infinite branch in the run corresponds to an infinite path in the tree where R is iterated forever and in which C is never fulfilled. However, the semantics of $\exists R^*.C$ requires that C is fulfilled after a finite number of iterations of R. Hence such an infinite path cannot be used to satisfy $\exists R^*.C$. The acceptance condition of the automaton, which requires that each infinite branch in a run contains a state of the form $\forall R^*.C$, rules out such infinite branches in accepting runs. Indeed, a run always deferring the fulfillment of C will contain an infinite branch where all states have the form $\exists R_1.\cdots\exists R_n.\exists R^*.C$, with $n \geq 0$ and $R_1 \circ \cdots \circ R_n$ a postfix of R. Observe that the only remaining infinite branches in a run are those that arise by propagating concepts of the form $\forall R^*.C$ indefinitely often. The acceptance condition allows for such branches.

Given a labeled tree $\mathcal{T} = (T, V)$ accepted by $\mathbf{A}^{\mathcal{F}}_{C_0}$, we define an interpretation $\mathcal{I}_{\mathcal{T}} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows. First, we define for each atomic role P, a relation $\mathcal{R}_P$ as follows: $\mathcal{R}_P = \{(x, xi) \mid P \in V(xi)\} \cup \{(xi, x) \mid P^- \in V(xi)\}$. Then, using such relations, we define:

$$
\begin{array}{rcll}
\Delta^{\mathcal{I}} & = & \{\, x \mid (\varepsilon, x) \in (\bigcup_P (\mathcal{R}_P \cup \mathcal{R}_P^-))^* \,\} & \\
A^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \cap \{\, x \mid A \in V(x) \,\}, & \text{for each atomic concept } A \\
P^{\mathcal{I}} & = & (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \cap \mathcal{R}_P, & \text{for each atomic role } P
\end{array}
$$

Lemma 3.2 *If a labeled tree $\mathcal{T}$ is accepted by $\mathbf{A}^{\mathcal{F}}_{C_0}$, then $\mathcal{I}_{\mathcal{T}}$ is a model of C_0.*

Conversely, given a tree model $\mathcal{I}$ of C_0 with branching degree k_{C_0}, we can obtain a labeled tree $\mathcal{T}_{\mathcal{I}} = (T, V)$ (with branching degree k_{C_0}) as follows:

$$
\begin{array}{rcll}
T & = & \Delta^{\mathcal{I}} & \\
V(\varepsilon) & = & \{A \mid \varepsilon \in A^{\mathcal{I}}\} & \\
V(xi) & = & \{A \mid xi \in A^{\mathcal{I}}\} \cup \{Q \mid (x, xi) \in Q^{\mathcal{I}}\}, & \text{for each node } xi
\end{array}
$$

Lemma 3.3 *If $\mathcal{I}$ is a tree model of C_0 with branching degree k_{C_0}, then $\mathcal{T}_{\mathcal{I}}$ is a labeled tree accepted by $\mathbf{A}^{\mathcal{F}}_{C_0}$.*

From the lemmas above and the tree model property of $\mathcal{ALCFI}_{reg}$ (Theorem 3.1), we get the following result.

Theorem 3.4 *An $\mathcal{ALCFI}_{reg}$ concept C_0 is satisfiable if and only if the set of trees accepted by $\mathbf{A}^{\mathcal{F}}_{C_0}$ is not empty.*

From this theorem, it follows that we can use algorithms for nonemptiness of 2ATAs to check satisfiability in $\mathcal{ALCFI}_{reg}$. It turns out that such a decision procedure is indeed optimal w.r.t. the computational complexity. The 2ATA $\mathbf{A}^{\mathcal{F}}_{C_0}$ has a number of states that is linear in the size of C_0, while the alphabet is exponential in the number of atomic concepts occurring in C_0. By Theorem 2.1 we get an upper bound for reasoning in $\mathcal{ALCFI}_{reg}$ that matches the ExpTime lower bound.

Theorem 3.5 *Concept satisfiability (and hence logical implication) in $\mathcal{ALCFI}_{reg}$ is* ExpTime-*complete.*

4 Boolean combinations of roles

The technique for reasoning in $\mathcal{ALCFI}_{reg}$ based on 2ATAs can be extended in a straightforward way to deal also with boolean combinations of atomic roles and their inverses [13]. More precisely, we consider the logic $\mathcal{ALCFI}b_{reg}$, which extends $\mathcal{ALCFI}_{reg}$ by allowing as basic roles boolean combinations (with negation restricted to be difference) of atomic roles and their inverses. Hence, to obtain $\mathcal{ALCFI}b_{reg}$, we can replace the syntax rule for basic roles of $\mathcal{ALCFI}_{reg}$ with the following:

$$Q, Q' \quad\longrightarrow\quad P \mid P^- \mid Q \cap Q' \mid Q \cup Q' \mid Q \setminus Q'$$

W.l.o.g., we assume that difference is applied only to atomic and/or inverse atomic roles.

We first extend the definition of closure to take into account the additional constructs. The closure $CL_b(C_0)$ of an $\mathcal{ALCFI}b_{reg}$ concept C_0 is defined as for $CL_{\mathcal{F}}$, by replacing the last rule dealing with basic roles with the following ones:

$$\begin{array}{lll}
\text{if} & \exists Q.C \in CL_b(C_0) & \text{then} \quad Q \in CL_b(C_0) \\
\text{if} & P \in CL_b(C_0) & \text{then} \quad \neg P,\, P^-,\, \neg P^- \in CL_b(C_0) \\
\text{if} & P^- \in CL_b(C_0) & \text{then} \quad \neg P,\, P^-,\, \neg P^- \in CL_b(C_0) \\
\text{if} & P \setminus P' \in CL_b(C_0) & \text{then} \quad P,\, P' \in CL_b(C_0) \\
\text{if} & Q \cap Q' \in CL_b(C_0) & \text{then} \quad Q,\, Q' \in CL_b(C_0) \\
\text{if} & Q \cup Q' \in CL_b(C_0) & \text{then} \quad Q,\, Q' \in CL_b(C_0)
\end{array}$$

CL_b^{nnf} is defined similarly to $CL_{\mathcal{F}}^{nnf}$.

Following [13], one can show that $\mathcal{ALCFI}b_{reg}$ enjoys the tree model property. More precisely, every satisfiable $\mathcal{ALCFI}b_{reg}$ concept C_0 has a tree model with branching degree k_{C_0} equal to twice the number of elements of $CL_b(C_0)$ (notice that $CL_b(C_0)$ takes into account boolean combinations of roles).

Then, the 2ATA $\mathbf{A}^b_{C_0} = (\Sigma, S, \delta, s_{ini}, F)$ used to check satisfiability of an $\mathcal{ALCFI}b_{reg}$ concept C_0 is defined, similarly as for $\mathcal{ALCFI}_{reg}$, as follows:

- The alphabet is $\Sigma = 2^{\mathcal{A} \cup \mathcal{B}}$.
- The set of states is $S = \{s_{ini}\} \cup CL_b^{nnf}(C_0)$.
- The set F of final states is the set of concepts in $CL_b^{nnf}(C_0)$ of the form $\forall R^*.C$.
- The transition function δ is defined as for the automaton for $\mathcal{ALCFI}_{reg}$, with the following additional transitions used to handle boolean combinations of atomic and inverse atomic roles. For each $\sigma \in \Sigma$:

$$\begin{array}{rcl}
\delta(Q \cap Q', \sigma) & = & (0, Q) \;\wedge\; (0, Q') \\
\delta(Q \cup Q', \sigma) & = & (0, Q) \;\vee\; (0, Q') \\
\delta(Q \setminus Q', \sigma) & = & (0, Q) \;\wedge\; (0, \neg Q')
\end{array}$$

(Remember that in $Q \setminus Q'$, the roles Q and Q' are either atomic or inverse of atomic, and hence $\neg Q'$ is indeed a state of the automaton.)

One can show the following result.

Theorem 4.1 *An $\mathcal{ALCFI}b_{reg}$ concept C_0 is satisfiable if and only if the set of trees accepted by $\mathbf{A}^b_{C_0}$ is not empty.*

Since the number of states of $\mathbf{A}^b_{C_0}$ is linear in the size of C_0, by Theorem 2.1 we get the following result.

Theorem 4.2 *Concept satisfiability (and hence logical implication) in $\mathcal{ALCFI}b_{reg}$ is* ExpTime*-complete.*

5 Qualified number restrictions

Next we extend our investigation to qualified number restrictions. More precisely, we consider the logic $\mathcal{ALCQI}b_{reg}$, which extends $\mathcal{ALCFI}b_{reg}$ by allowing as concepts also qualified number restrictions of the form:

$$\geqslant n\, Q.C \qquad\qquad \leqslant n\, Q.C$$

where n is a non-negative integer, Q is a boolean combination (with negation restricted to be difference) of atomic roles and their inverses, and C is an arbitrary $\mathcal{ALCQI}b_{reg}$ concept.

We first extend the definition of closure to take into account qualified number restriction. The closure $CL_Q(C_0)$ of an $\mathcal{ALCQI}b_{reg}$ concept C_0 is defined by adding to the rules for CL_b the following ones:

$$\begin{array}{llll} \text{if} & \geqslant n\, Q.C \in CL_b(C_0) & \text{then} & Q, C \in CL_b(C_0) \\ \text{if} & \leqslant n\, Q.C \in CL_b(C_0) & \text{then} & Q, C \in CL_b(C_0) \end{array}$$

CL_Q^{nnf} is defined as for the other logics.

One can show that $\mathcal{ALCQI}b_{reg}$ enjoys again the tree-model property. More precisely, every satisfiable $\mathcal{ALCQI}b_{reg}$ concept C_0 has a tree model with branching degree k_{C_0} equal to $(n_{max}+1)\cdot|CL_Q(C_0)|$, where n_{max} is the maximal number appearing inside a qualified number restriction, and $|CL_Q(C_0)|$ is the number of elements of $CL_Q(C_0)$.

Then, the 2ATA $\mathbf{A}^{\mathcal{Q}}_{C_0} = (\Sigma, S, \delta, s_{ini}, F)$ used to check satisfiability of an $\mathcal{ALCQI}b_{reg}$ concept C_0 is defined, similarly as for $\mathcal{ALCFI}b_{reg}$, as follows:

- The alphabet is $\Sigma = 2^{\mathcal{A}\cup\mathcal{B}}$.
- The set of states is $S = \{s_{ini}\} \cup CL_Q^{nnf}(C_0) \cup \mathcal{S}_Q$, where

  $$\begin{array}{rcl} S_Q &=& \{\langle \geqslant n\, Q.C, i, j\rangle \mid\ \geqslant n\, Q.C \in CL_Q^{nnf}(C_0),\ 0 \le i \le k_{C_0}{+}1,\ 0 \le j \le n\} \cup \\ && \{\langle \leqslant n\, Q.C, i, j\rangle \mid\ \leqslant n\, Q.C \in CL_Q^{nnf}(C_0),\ 0 \le i \le k_{C_0}{+}1,\ 0 \le j \le n{+}1\} \end{array}$$

  Intuitively, the states $\langle \geqslant n\, Q.C, i, j\rangle$ are used to check whether a qualified number restriction $\geqslant n\, Q.C$ is satisfied in a node x by counting the number of nodes (among the predecessor and the successors of x) reached from x through Q in which C holds. More precisely, the automaton will be in a state $\langle \geqslant n\, Q.C, i, j\rangle$ if among the first $i-1$ successors of x (the 0-th successor is the predecessor) there are j nodes reached from x through Q in which C holds.

- The set F of final states is the set of concepts in $CL_Q^{nnf}(C_0)$ of the form $\forall R^*.C$.
- The transition function δ is defined as for the automaton for $\mathcal{ALCFIb}_{reg}$, with the following additional transitions used to handle qualified number restrictions (we use $\gtrless$ to denote either $\geqslant$ or $\leqslant$). For each $\sigma \in \Sigma$:

$$
\begin{array}{rcl}
\delta(\gtrless n\, Q.C, \sigma) & = & (0, \langle \gtrless n\, Q.C, 0, 0\rangle) \\
\delta(\langle \gtrless n\, Q.C, 0, 0\rangle, \sigma) & = & (((0, \neg Q^-) \vee (-1, nnf(\neg C))) \wedge (0, \langle \gtrless n\, Q.C, 1, 0\rangle)) \vee \\
 & & ((0, Q^-) \wedge (-1, C) \wedge (0, \langle \gtrless n\, Q.C, 1, 1\rangle)) \\
\delta(\langle \geqslant n\, Q.C, i, j\rangle, \sigma) & = & (((i, \neg Q) \vee (i, nnf(\neg C))) \wedge (0, \langle \geqslant n\, Q.C, i{+}1, j\rangle)) \vee \\
 & & ((i, Q) \wedge (i, C) \wedge (0, \langle \geqslant n\, Q.C, i{+}1, j{+}1\rangle)), \\
 & & \quad \text{for } 1 \leq i \leq k_{C_0},\ 0 \leq j \leq n{-}1 \\
\delta(\langle \geqslant n\, Q.C, i, n\rangle, \sigma) & = & \textbf{true}, \qquad \text{for } 0 \leq i \leq k_{C_0}{+}1 \\
\delta(\langle \geqslant n\, Q.C, k_{C_0}{+}1, j\rangle, \sigma) & = & \textbf{false}, \qquad \text{for } 0 \leq j \leq n{-}1 \\
\delta(\langle \leqslant n\, Q.C, i, j\rangle, \sigma) & = & (((i, \neg Q) \vee (i, nnf(\neg C))) \wedge (0, \langle \leqslant n\, Q.C, i{+}1, j\rangle)) \vee \\
 & & ((i, Q) \wedge (i, C) \wedge (0, \langle \leqslant n\, Q.C, i{+}1, j{+}1\rangle)), \\
 & & \quad \text{for } 1 \leq i \leq k_{C_0},\ 0 \leq j \leq n \\
\delta(\langle \leqslant n\, Q.C, i, n{+}1\rangle, \sigma) & = & \textbf{false}, \qquad \text{for } 0 \leq i \leq k_{C_0}{+}1 \\
\delta(\langle \leqslant n\, Q.C, k_{C_0}{+}1, j\rangle, \sigma) & = & \textbf{true}, \qquad \text{for } 0 \leq j \leq n
\end{array}
$$

One can show the following result.

Theorem 5.1 *An $\mathcal{ALCQIb}_{reg}$ concept C_0 is satisfiable if and only if the set of trees accepted by $\mathbf{A}_{C_0}^{\mathcal{Q}}$ is not empty.*

If the numbers inside qualified number restrictions are coded in unary, then $\mathbf{A}_{C_0}^{\mathcal{Q}}$ has a number of states that is polynomial (actually cubic) in the size of C_0. Applying again Theorem 2.1, we get the following result.

Theorem 5.2 *Concept satisfiability (and hence logical implication) in $\mathcal{ALCQIb}_{reg}$ is* ExpTime*-complete, assuming that numbers inside qualified number restrictions are coded in unary.*

Obviously, if one assumes numbers to be coded in binary, Theorem 5.1 does not provide us with an ExpTime upper bound for concept satisfiability. However, one can also deal with numbers coded in binary by adopting a more complex reduction from concept satisfiability in $\mathcal{ALCQIb}_{reg}$ to nonemptiness of 2ATAs. Observe that, with binary coding, the exponential blowup in the size of $\mathbf{A}_{C_0}^{\mathcal{Q}}$ has two reasons:

1. the necessity to explicitly have transitions to the k_{C_0} successors of a node (with binary coding k_{C_0} is exponential in the size of C_0);

2. the necessity to count up to the maximum number n_{max} appearing inside qualified number restrictions (which again is exponential).

To cope with point (1), one can construct a 2ATA that, instead of accepting trees of branching degree k_{C_0}, accepts binary trees that represent them (using the standard encoding of k-ary trees in binary trees, where the first child of a node becomes its left successor, and the next sibling of a node becomes its right successor). Observe that, a node y connected to a node x via a basic role Q in a k_{C_0}-ary tree, may be far away from x in the corresponding binary tree. It is possible to take this into account by introducing a linear number of additional states and suitable transitions in the automaton that works on the binary tree.

To cope with point (2), one can make use of transitions that simulate in the automaton a "binary counter", instead of a unary counter as done for $\mathbf{A}^{\mathcal{Q}}_{C_0}$, thus using a number of states that is polynomial in n_{max}, even when it is coded in binary.

Hence, it is possible to show that, using techniques based on 2ATAs, concept satisfiability (and hence logical implication) in $\mathcal{ALCQI}b_{reg}$ can be decided in ExpTime, even when numbers inside qualified number restrictions are coded in binary.

6 Conclusions

We have shown that using 2ATAs one can provide an intuitive, short, modular, and computationally adequate encoding of reasoning services in expressive DLs into nonemptiness of automata on infinite trees.

There are two main research directions that can be pursued to exploit 2ATAs for reasoning in DLs. First, 2ATAs can be used for characterizing reasoning for new combinations of constructs and/or for new reasoning services. With respect to this, many new results in DLs exploit the power of tree automata [7, 8], and can be rephrased using 2ATAs to abstract away the "combinatorial noise" and put forward the main argument. Second, it is of great interest to study practical methods for testing nonemptiness of 2ATAs (which is ExpTime-complete) in light of the optimization techniques developed for the current state-of-the-art tableau-based DLs systems.

References

[1] O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Proc. of the 6th Int. Conf. on Computer Aided Verification (CAV'94)*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155. Springer, 1994.

[2] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 84–89, 1999.

[3] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of the 32nd Annual Symp. on the Foundations of Computer Science (FOCS'91)*, pages 368–377, 1991.

[4] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and System Sciences*, 18:194–211, 1979.

[5] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press, 2000.

[6] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 335–346, 1991.

[7] C. Lutz. Interval-based temporal reasoning with general TBoxes. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 89–94, 2001.

[8] C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyashev, editors, *Advances in Modal Logics*, volume 3. CSLI Publications, 2001.

[9] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[10] U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, pages 76–91, 2001.

[11] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[12] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 133–192. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[13] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[14] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. of the 25th Int. Coll. on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998.

[15] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Sciences*, 32:183–221, 1986. A preliminary version appeared in *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC'84)*.