EXPTIME Tableaux for $\mathcal{ALC}$ Using Sound Global Caching

Rajeev Goré[1] and Linh Anh Nguyen[2]

[1] The Australian National University
Canberra ACT 0200, Australia
[2] Institute of Informatics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
Rajeev.Gore@anu.edu.au nguyen@mimuw.edu.pl

Abstract. We show that global caching can be used with propagation of both satisfiability and unsatisfiability in a sound manner to give an EXPTIME algorithm for checking satisfiability w.r.t. a TBox in the basic description logic $\mathcal{ALC}$. Our algorithm is based on a simple traditional tableau calculus which builds an and-or graph where no two nodes of the graph contain the same formula set. When a duplicate node is about to be created, we use the pre-existing node as a proxy, even if the proxy is from a different branch of the tableau, thereby building global caching into the algorithm from the start. Doing so is important since it allows us to reason explicitly about the correctness of global caching. We then show that propagating both satisfiability and unsatisfiability via the and-or structure of the graph remains sound. In the longer paper, by combining global caching, propagation and cutoffs, our framework reduces the search space more significantly than the framework of [1]. Also, the freedom to use arbitrary search heuristics significantly increases its application potential.
A longer version with all optimisations is currently under review for a journal. An extension for SHI will appear in TABLEAUX 2007.

Keywords: sound caching, decision procedures, optimal complexity.

1 Motivation, Notation and Semantics of $\mathcal{ALC}$

We show that there is a simple way to use global caching and propagation to achieve an EXPTIME decision procedure for $\mathcal{ALC}$. Our algorithm is based on a simple traditional tableau calculus. It builds an and-or graph, where an or-node reflects the application of an "or" branching rule as in a tableau, while an and-node reflects the choice of a tableau rule and possibly many different applications of that rule to a given node of a tableau. We build caching into the construction of the and-or graph by ensuring that no two nodes of the graph have the same content. The status of a non-end-node is computed from the status of its successors using its kind (and-node/or-node) and treating satisfiability w.r.t. the TBox (i.e. `sat`) as true and unsatisfiability w.r.t. the TBox (i.e. `unsat`) as

false. When a node gets status `sat` or `unsat`, the status is propagated to its predecessors in a way appropriate to the graph's and-or structure. With global caching and the assumption that EXPTIME $\neq$ PSPACE, depth-first search has no advantages over other search strategies for our framework. That is, the naive version of our EXPTIME algorithm can accept any systematic search strategy.

By combining global caching, propagation and cutoffs, our framework significantly reduces the search space when compared with the framework of Donini and Massacci [1]. Furthermore, the freedom to use arbitrary search heuristics significantly increases the application potential of our framework.

We use A for atomic concepts, use C and D for arbitrary concepts, and use R for a role name. Concepts in $\mathcal{ALC}$ are formed using the following BNF grammar:

$$C, D ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid C \sqsubseteq D \mid C \doteq D \mid \forall R.C \mid \exists R.C$$

An *interpretation* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$, the *interpretation function* of $\mathcal{I}$, that maps every atomic concept to a subset of $\Delta^{\mathcal{I}}$ and every role name to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to complex concepts as usual.

An interpretation $\mathcal{I}$ *satisfies* a concept C if $C^{\mathcal{I}} \neq \emptyset$, and *validates* a concept C if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$. Clearly, $\mathcal{I}$ *validates* a concept C iff it does not *satisfy* $\neg C$.

A TBox (of global axioms/assumptions) Γ is a finite set of concepts: traditionally, a TBox is defined to be a finite set of terminological axioms of the form $C \doteq D$, where C and D are concepts, but the two definitions are equivalent. An interpretation $\mathcal{I}$ is a *model* of Γ if $\mathcal{I}$ validates all concepts in Γ. We also use X, Y to denote finite sets of concepts. We say that $\mathcal{I}$ *satisfies* X if there exists $d \in \Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$ for all $C \in X$. Note: satisfaction is defined "locally", and $\mathcal{I}$ satisfies X does not mean that $\mathcal{I}$ is a model of X.

We say that Γ *entails* C, and write $\Gamma \models C$, if every model of Γ validates C. We say that C is satisfiable w.r.t. Γ if some model of Γ satisfies $\{C\}$. Similarly, X is satisfiable w.r.t. (a TBox of global axioms/assumptions) Γ if there exists a model of Γ that satisfies X. Observe that $\Gamma \models C$ iff $\neg C$ is unsatisfiable w.r.t. Γ.

Note: We now assume that concepts are in *negation normal form*, where $\doteq$ and $\sqsubseteq$ are translated away and $\neg$ occurs only directly before atomic concepts.

2 A Tableau Calculus for $\mathcal{ALC}$

We consider tableaux with a fixed TBox of global axioms/assumptions Γ. The numerator of each tableau rule contains one or more distinguished concepts called the *principal concepts*. We write $X; Y$ for $X \cup Y$, and $X; C$ for $X \cup \{C\}$. The calculus $C\mathcal{ALC}$ for $\mathcal{ALC}$ consists of the tableau rules below:

$$(\bot)\ \frac{X\ ;\ A\ ;\ \neg A}{\bot} \qquad (\sqcap)\ \frac{X\ ;\ C \sqcap D}{X\ ;\ C\ ;\ D} \qquad (\sqcup)\ \frac{X\ ;\ C \sqcup D}{X\ ;\ C \mid X\ ;\ D}$$

$$(\exists R)\ \Gamma:\ \frac{X\ ;\ \exists R.C}{\{D : \forall R.D \in X\}\ ;\ C\ ;\ \Gamma}$$

The rules ($\bot$), ($\sqcap$), and ($\sqcup$) are *static* rules, while ($\exists R$) is a *transitional rule.*

A $\mathcal{CALC}$-tableau (tableau, for short) w.r.t. a TBox Γ for a finite set X of concepts is a tree with root $(\Gamma; X)$ whose nodes carry finite sets of concepts obtained from their parent nodes by instantiating a $\mathcal{CALC}$-tableau rule with the proviso that: if a child s carries a set Y and no rule is applicable to Y or Y has already appeared on the branch from the root to s then s is an *end node.*

A branch in a tableau is *closed* if its end node carries only $\bot$. A tableau is *closed* if every one of its branches is closed. A tableau is *open* if it is not closed.

A finite set X of concepts is *consistent* w.r.t. a TBox Γ if every tableau w.r.t. Γ for X is open. If some tableau w.r.t. Γ for X is closed then X is *inconsistent* w.r.t. Γ. Calculus $\mathcal{CALC}$ is *sound* if for all finite sets Γ and X of concepts, X is satisfiable w.r.t. Γ implies X is consistent w.r.t. Γ. It is *complete* if for all finite sets Γ and X of concepts, X is consistent w.r.t. Γ implies X is satisfiable w.r.t. Γ. A tableau rule is *sound* if, whenever the numerator is $\mathcal{ALC}$-satisfiable w.r.t. the TBox then one of the denominators is $\mathcal{ALC}$-satisfiable w.r.t. the TBox.

Lemma 1. *The calculus $\mathcal{CALC}$ is sound because all rules of $\mathcal{CALC}$ are sound.*

Observe that every concept appearing in a tableau w.r.t. Γ for X is a subformula of $\Gamma \cup X \cup \{\bot\}$. Thus $\mathcal{CALC}$ thus has the *analytic subformula property.*

3 Completeness

A model graph is a tuple $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$, where: Δ is a finite set; τ is a distinguished element of Δ; $\mathcal{C}$ is a function that maps each element of Δ to a set of concepts; and $\mathcal{E}$ is a function that maps each role name to a binary relation on Δ.

A model graph $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ is *saturated* if every $x \in \Delta$ satisfies:

1. if $C \sqcap D \in \mathcal{C}(x)$ then $\{C, D\} \subseteq \mathcal{C}(x)$
2. if $C \sqcup D \in \mathcal{C}(x)$ then $C \in \mathcal{C}(x)$ or $D \in \mathcal{C}(x)$
3. if $\forall R.C \in \mathcal{C}(x)$ and $\mathcal{E}(R)(x, y)$ holds then $C \in \mathcal{C}(y)$
4. if $\exists R.C \in \mathcal{C}(x)$ then there exists $y \in \Delta$ with $\mathcal{E}(R)(x, y)$ and $C \in \mathcal{C}(y)$.

A saturated model graph $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ is *consistent* if no $x \in \Delta$ has a $\mathcal{C}(x)$ containing $\bot$ or containing a pair $A, \neg A$ for some atomic concept A.

Given a model graph $M = \langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$, the *interpretation corresponding to M* is the interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ where $A^{\mathcal{I}} = \{x \in \Delta \mid A \in \mathcal{C}(x)\}$ for every atomic concept A and $R^{\mathcal{I}} = \mathcal{E}(R)$ for every role name R.

Lemma 2. *By induction on the structure of C we can show that if $\mathcal{I}$ is the interpretation corresponding to a consistent saturated model graph $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$, then for every $x \in \Delta$ and $C \in \mathcal{C}(x)$ we have $x \in C^{\mathcal{I}}$.*

Given finite sets X and Γ of concepts, where X is consistent w.r.t. Γ, we construct a model of Γ that satisfies X by constructing a consistent saturated model graph $\langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ with $X \subseteq \mathcal{C}(\tau)$ and $\Gamma \subseteq \mathcal{C}(x)$ for every $x \in \Delta$.

Algorithm 1
Input: a TBox Γ and a finite set X of concepts, where X is consistent w.r.t. Γ.
Output: a model graph $M = \langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$.

1. For an arbitrary node name τ, let $\Delta := \{\tau\}$, and $\mathcal{E}(R) := \emptyset$ for every role name R. Let $\mathcal{C}(\tau)$ be a saturation of $\Gamma \cup X$ and mark τ as unexpanded.
2. While Δ contains unexpanded elements, take one, say x, and do:
   (a) For every concept $\exists R.C \in \mathcal{C}(x)$:
      i. Let $Y = \{D \mid \forall R.D \in \mathcal{C}(x)\} \cup \{C\} \cup \Gamma$ be the result of applying rule $(\exists R)$ to $\mathcal{C}(x)$, and let Z be a saturation of Y.
      ii. If there exists a (proxy) $y \in \Delta$ with $\mathcal{C}(y) = Z$ then add pair (x, y) to $\mathcal{E}(R)$;
      iii. Else add a new element y with $\mathcal{C}(y) := Z$ to Δ, mark y as unexpanded, and add the pair (x, y) to $\mathcal{E}(R)$.
   (b) Mark x as expanded.

Fig. 1. Constructing a Model Graph

Saturation The rules $(\sqcap)$ and $(\sqcup)$ do not carry their principal concept into their denominators. For these rules, let (ρ') be the version that carries the principal concept into each of its denominators. Each new rule is clearly sound for $\mathcal{ALC}$.

For a finite set X of concepts that is consistent w.r.t. a TBox Γ, a set Y of concepts is called a *saturation of* X w.r.t. Γ if Y is a maximal set consistent w.r.t. Γ that is obtainable from X (as a leaf node in a tableau) by applications of the rules $(\sqcap')$ and $(\sqcup')$. A set X is *closed* w.r.t. a tableau rule if applying that rule to X gives back X as one of the denominators.

Lemma 3. *Let X be a finite set of concepts consistent w.r.t. a TBox Γ, and Y a saturation of X w.r.t. Γ. Then $X \subseteq Y \subseteq Sf(\Gamma \cup X)$ and Y is closed w.r.t. the rules $(\sqcap')$ and $(\sqcup')$. Furthermore, there is an effective procedure that constructs such a set Y from Γ and X.*

Constructing Model Graphs Figure 1 contains an algorithm for constructing a model graph. Algorithm 1 assumes that X is consistent w.r.t. Γ and constructs a model of Γ that satisfies X. Algorithm 1 terminates because each $x \in \Delta$ has a unique finite set $\mathcal{C}(x) \subseteq Sf(\Gamma \cup X)$, so eventually Step 2(a)ii always finds a proxy. Note that Step 2(a)ii builds caching into the algorithm.

Lemma 4. *Let Γ be a TBox, X be a finite set of concepts consistent w.r.t. Γ, $M = \langle \Delta, \tau, \mathcal{C}, \mathcal{E} \rangle$ be the model graph constructed by Algorithm 1 for Γ and X, and $\mathcal{I}$ be the interpretation corresponding to M. Then $\mathcal{I}$ validates Γ and satisfies X.*

Theorem 1. *The calculus $C\mathcal{ALC}$ is sound and complete.*

4 A Simple EXPTIME Decision Procedure for $\mathcal{ALC}$

In Figure 2 we present an EXPTIME decision procedure for $\mathcal{ALC}$ which directly uses the tableau rules of $C\mathcal{ALC}$ to create an and-or graph as follows.

Algorithm 2

Input: two finite sets of concepts Γ and X
Output: an and-or graph $G = \langle V, E\rangle$ with $\tau \in V$ as the initial node such that $\tau.status = \texttt{sat}$ iff X is satisfiable w.r.t. Γ

1. create a new node τ with $\tau.content := \Gamma \cup X$ and $\tau.status := \texttt{unexpanded}$; let $V := \{\tau\}$ and $E := \emptyset$;
2. while $\tau.status \notin \{\texttt{sat}, \texttt{unsat}\}$ and we can choose an unexpanded node $v \in V$ do:
   (a) $\mathcal{D} := \emptyset$;
   (b) if no $\mathcal{CALC}$-tableau rule is applicable to $v.content$ then $v.status := \texttt{sat}$
   (c) else if $(\bot)$ is applicable to $v.content$ then $v.status := \texttt{unsat}$
   (d) else if $(\sqcap)$ is applicable to $v.content$ giving denominator Y then $v.kind := \texttt{and-node}$, $\mathcal{D} := \{Y\}$
   (e) else if $(\sqcup)$ is applicable to $v.content$ giving denominators Y_1 and Y_2 then $v.kind := \texttt{or-node}$, $\mathcal{D} := \{Y_1, Y_2\}$
   (f) else
      i. $v.kind := \texttt{and-node}$,
      ii. for every $\exists R.C \in v.content$, apply $(\exists R)$ to $v.content$ giving denominator $\{D \mid \forall R.D \in v.content\} \cup \{C\} \cup \Gamma$ and add this denominator to $\mathcal{D}$;
   (g) for every denominator $Y \in \mathcal{D}$ do
      i. if some (proxy) $w \in V$ has $w.content = Y$ then add edge (v, w) to E
      ii. else let w be a new node, set $w.content := Y$, $w.status := \texttt{unexpanded}$, add w to V, and add edge (v, w) to E;
   (h) if ($v.kind = \texttt{or-node}$ and one of the successors of v has status $\texttt{sat}$) or ($v.kind = \texttt{and-node}$ and all the successors of v have status $\texttt{sat}$) then $v.status := \texttt{sat}$, $propagate(G, v)$
   (i) else if ($v.kind = \texttt{and-node}$ and one of the successors of v has status $\texttt{unsat}$) or ($v.kind = \texttt{or-node}$ and all the successors of v have status $\texttt{unsat}$) then $v.status := \texttt{unsat}$, $propagate(G, v)$
   (j) else $v.status := \texttt{expanded}$;
3. if $\tau.status \notin \{\texttt{sat}, \texttt{unsat}\}$ then for every node $v \in V$ with $v.status \neq \texttt{unsat}$, set $v.status := \texttt{sat}$;

Fig. 2. A Simple EXPTIME Decision Procedure for $\mathcal{ALC}$

A node in the constructed and-or graph is a record with three attributes:

content: the set of concepts carried by the node
status: $\{\texttt{unexpanded}, \texttt{expanded}, \texttt{sat}, \texttt{unsat}\}$
kind: $\{\texttt{and-node}, \texttt{or-node}\}$

To check whether a given finite set X is satisfiable w.r.t. the given TBox Γ, the content of the initial node τ with status `unexpanded` is $\Gamma \cup X$. The main while-loop continues processing nodes until the status of τ is determined to be in $\{\texttt{sat}, \texttt{unsat}\}$, or until every node is expanded, whichever happens first.

Inside the main loop, Steps (2b) to (2f) try to apply one and only one of the tableau rules in the order $(\bot)$, $(\sqcap)$, $(\sqcup)$, $(\exists R)$ to the current node v. The set $\mathcal{D}$

Procedure *propagate*(G, v)
Parameters: an and-or graph $G = \langle V, E \rangle$ and $v \in V$ with $v.status \in \{\texttt{sat}, \texttt{unsat}\}$
Returns: a modified and-or graph $G = \langle V, E \rangle$

1. $queue := \{v\}$;
2. while $queue$ is not empty do
3. (a) extract x from $queue$;
   (b) for every $u \in V$ with $(u, x) \in E$ and $u.status = \texttt{expanded}$ do
      i. if ($u.kind = \texttt{or-node}$ and one of the successors of u has status $\texttt{sat}$)
         or ($u.kind = \texttt{and-node}$ and all the successors of u have status $\texttt{sat}$) then
            $u.status := \texttt{sat}$, $queue := queue \cup \{u\}$
      ii. else if ($u.kind = \texttt{and-node}$ and one of the successors of u has status $\texttt{unsat}$)
         or ($u.kind = \texttt{or-node}$ and all the successors of u have status $\texttt{unsat}$) then
            $u.status := \texttt{unsat}$, $queue := queue \cup \{u\}$;

Fig. 3. Propagating Satisfiability and Unsatisfiability Through an And-Or Graph

contains the contents of the resulting denominators of v. If the applied tableau rule is ($\sqcap$) then v has one denominator in $\mathcal{D}$; if the applied rule is ($\sqcup$) then v has two denominators in $\mathcal{D}$; otherwise, each concept $\exists R.C \in v.content$ contributes one appropriate denominator to $\mathcal{D}$. At Step (2g), for every denominator in $\mathcal{D}$, we create the required successor in the graph G only if it does not yet exist in the graph: this step merely mimics Algorithm 1 and therefore uses global caching.

In Algorithm 2, a node that contains both A and $\neg A$ for some atomic concept A becomes an end-node with status `unsat` (i.e. unsatisfiable w.r.t. Γ). A node to which no tableau rule is applicable becomes an end-node with status `sat` (i.e. satisfiable w.r.t. Γ). Both conclusions are **irrevocable** because each relies only on classical propositional principles and not on modal principles. That is, we do not need to undo either of these at any stage.

On the other hand, an application of ($\sqcup$) to a node v causes v to be an *or-node*, while an application of ($\sqcap$) or ($\exists R$) to a node v causes v to be an *and-node*. Steps (2h) and (2i) try to compute the status of such a non-end-node v using the kind (or-node/and-node) of v and the status of the successors of v, treating `unsat` as irrevocably `false` and `sat` as irrevocably `true`.

If these steps cannot determine the status of v as `sat` or `unsat`, then its status is set to `expanded`. But if these steps do determine the status of a node v to be `sat` or `unsat`, this information is itself propagated to the predecessors of v in the and-or graph G via the routine *propagate*(G, v), explained shortly.

The main loop ends when the status of the initial node τ becomes `sat` or `unsat` or all nodes of the graph have been expanded. In the latter case, all nodes with status $\neq$ `unsat` are given status `sat` (effectively giving the status *open* to tableau branches which loop). Again, caching is present at Step 2(g)i.

The procedure *propagate* used in the above algorithm is specified in Figure 3. As parameters, it accepts an and-or graph G and a node v with (irrevocable)

status `sat` or `unsat`. The purpose is to propagate the status of v through the and-or graph and alter G to reflect the new information.

Initially, the queue of nodes to be processed contains only v. Then while the queue is not empty: a node x is extracted; the status of x is propagated to each predecessor u of x; and if the status of a predecessor u becomes (irrevocably) `sat` or `unsat` then u is inserted into the queue for further propagation.

This construction thus uses both caching and propagation techniques.

Proposition 1. *Algorithm 2 runs in EXPTIME.*

Proof. Let $G = \langle V, E \rangle$ be the graph constructed by Algorithm 2 for Γ and X and n be the size of input, i.e. the sum of the lengths of the concepts of $\Gamma \cup X$.

Each $v \in V$ has $v.content \subseteq Sf(\Gamma \cup X)$, hence $v.content$ has size $2^{O(n)}$. For all $v, w \in V$, if $v \neq w$ then $v.content \neq w.content$. Hence V contains $2^{O(n)}$ nodes.

Every $v \in V$ is expanded (by Steps (2a)–(2j)) only once and every expansion takes $2^{O(n)}$ time units not counting the execution time of procedure *propagate* since $v.content$ contains $2^{O(n)}$ concepts and we still have to search for proxies amongst possibly $2^{O(n)}$ nodes in V. When $v.status$ becomes `sat` or `unsat`, the procedure *propagate* executes $2^{O(n)}$ basic steps directly involved with v, so the total time of the executions of *propagate* is of rank $2^{2.O(n)}$. Hence Algorithm 2 runs in exponential time.

Lemma 5. *It is an invariant of Algorithm 2 that for every $v \in V$:*

1. *if $v.status =$ `unsat` then*
   - *$v.content$ contains both A and $\neg A$ for some atomic concept A,*
   - *or $v.kind =$ `and-node` and there exists $(v, w) \in E$ such that $w \neq v$ and $w.status =$ `unsat`,*
   - *or $v.kind =$ `or-node` and for every $(v, w) \in E$, $w.status =$ `unsat`;*
2. *if $v.status =$ `sat` then*
   - *no $\mathcal{CALC}$-tableau rule is applicable to $v.content$,*
   - *or $v.kind =$ `or-node` and there exists $(v, w) \in E$ with $w.status =$ `sat`,*
   - *or $v.kind =$ `and-node` and for every $(v, w) \in E$, $w.status =$ `sat`.*

(If $v.kind =$ `or-node` and $(v, w) \in E$ then $w \neq v$ since $w.content \neq v.content$.)

Lemma 6. *Let $G = \langle V, E \rangle$ be the graph constructed by Algorithm 2 for Γ and X. For every $v \in V$, if $v.status =$ `unsat` then $v.content$ is inconsistent w.r.t. Γ.*

Proof. Using Lemma 5, we can construct a closed tableau w.r.t. Γ for $v.content$ by induction on the way v depends on its successors and by copying nodes to ensure that the resulting structure is a (tree) tableau rather than a graph.

Let $G = \langle V, E \rangle$ be the graph constructed by Algorithm 2 for Γ and X. For $v \in V$ with $v.status =$ `sat`, we say that $v_0 = v, v_1, \ldots, v_k$ with $k \geq 0$ is a *saturation path of v in G* if for each $1 \leq i \leq k$, we have $v_i.status =$ `sat`, the edge $E(v_{i-1}, v_i)$ was created by an application of $(\sqcap)$ or $(\sqcup)$, and $v_k.content$ contains no concepts of the form $C \sqcap D$ nor $C \sqcup D$. By Lemma 5, if $v.status =$ `sat` then there exists a saturation path of v in G.

Lemma 7. *Let $G = \langle V, E\rangle$ be the graph constructed by Algorithm 2 for Γ and X. For all $v \in V$, if $v.status = \texttt{sat}$ then every tableau w.r.t. Γ for $v.content$ is open.*

Proof. Choose any $v \in V$ with $v.status = \texttt{sat}$ and let T be an arbitrary tableau (tree) w.r.t. Γ for $v.content$.

We maintain a *current node cn* of T that will follow edges of T to pin-point an open branch of T. Initially we set $cn := v$. We also keep a (finite) saturation path σ of the form $\sigma_0, \ldots, \sigma_k$ for some $\sigma_0 \in V$ and call σ the *current saturation path in G*. At the beginning, set $\sigma_0 := v$, so v is a node of both T and G and let σ be a saturation path for σ_0 in G: we know σ exists since $v.status = \texttt{sat}$.

We maintain the following invariant where $cn.content$ is the set carried by cn:

Invariant: $\forall C \in cn.content. \exists i. 0 \leq i \leq k,\ C \in \sigma_i.content$.

Remark 1. Observe that if $C \in \sigma_i.content$ for some $0 \leq i \leq k$ and C is of the form A, $\neg A$, $\exists R.D$, or $\forall R.D$ then $C \in \sigma_k.content$ since the saturation process does not affect concepts of these forms. By the definition of saturation path, we know that $\sigma_k.status = \texttt{sat}$, hence the $(\bot)$-rule is not applicable to $\sigma_k.content$. Hence, the invariant implies that $cn.content$ does not contain a pair A, $\neg A$ for any atomic concept A, and thus the rule $(\bot)$ is not applicable to cn. Also, note that the universal quantification over C encompasses the existential quantification over i, so each C can have a different σ_i in the invariant.

Clearly, the invariant holds at the beginning with $i = 0$ since $\sigma_0 = v = cn$ is in σ. Depending upon the rule applied to cn in the tableau T, we maintain the invariant by changing the value of the current node cn of T and possibly also the current saturation path σ in G. By Remark 1, the branch formed by the instances of cn is an open branch of T.

Theorem 2. *Let $G = \langle V, E\rangle$ be constructed by Algorithm 2 for Γ and X, with $\tau \in V$ as the initial node. Then X is satisfiable w.r.t. Γ iff $\tau.status = \texttt{sat}$.*

Corollary 1. *Algorithm 2 is an EXPTIME decision procedure for $\mathcal{ALC}$.*

We have extended our method to SHI and also to regular RBoxes. It can also be extended for checking consistency of an ABox w.r.t. a TBox in $\mathcal{ALC}$.

References

1. F. Donini and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124:87–138, 2000.