

**Description Logics:
a Nice Family of Logics**

Day 2: Tableau Algorithms

ESSLI 2016

Uli Sattler

Today

- relationship between standard DL reasoning problems
- a tableau algorithm to decide consistency of *ALC* ontologies and all other standard DL reasoning problems
- a proof of its correctness
- with some model properties
- some optimisations
- some extensions
 - inverse roles
 - (sketch) number restrictions
- some discussions
- ...loads of stuff: **ask if you have a question!**

Standard DL Reasoning Problems

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$,

- is \mathcal{O} consistent? $\mathcal{O} \models \top \sqsubseteq \perp?$
- is \mathcal{O} coherent? is there concept name A with $\mathcal{O} \models A \sqsubseteq \perp?$
- compute class hierarchy! for all concept names A, B : $\mathcal{O} \models A \sqsubseteq B?$
- classify individuals! for all concept names A , individual names b : $\mathcal{O} \models b : B?$

Theorem 2 Let \mathcal{O} be an ontology and a an individual name **not** in \mathcal{O} . Then

1. C is satisfiable w.r.t. \mathcal{O} iff $\mathcal{O} \cup \{a : C\}$ is consistent
2. \mathcal{O} is coherent iff, for each concept name A ,
 $\mathcal{O} \cup \{a : A\}$ is consistent
3. $\mathcal{O} \models A \sqsubseteq B$ iff $\mathcal{O} \cup \{a : (A \sqcap \neg B)\}$ is **not** consistent
4. $\mathcal{O} \models b : B$ iff $\mathcal{O} \cup \{b : \neg B\}$ is **not** consistent

➔ a decision procedure to solve consistency decides **all** standard DL reasoning problems

Decision Procedure

- A **problem** is a set $P \subseteq M$
 - e.g., M is the set of all \mathcal{ALC} ontologies,
 - $P \subseteq M$ is the set of all **consistent** \mathcal{ALC} ontologies
 - ...and the **problem** P is to decide whether, for a given $m \in M$, we have $m \in P$
- An **algorithm** is a **decision procedure** for a problem $P \subseteq M$ if it is
 - **sound** for P : if it answers " $m \in P$ ", then $m \in P$
 - **complete** for P : if $m \in P$, then it answers " $m \in P$ "
 - **terminating**: it stops after finitely many steps on any input $m \in M$

Why does "sound and complete" not suffice for being a decision procedure?

A tableau algorithm for \mathcal{ALC} ontologies

- For now:
- \mathcal{ALC} : $\sqcap, \sqcup, \neg, \exists r.C, \forall r.C$
 - an algorithm to decide consistency of an ontology

The algorithm decides "Is \mathcal{O} consistent" by trying to construct a model \mathcal{I} for \mathcal{O} :

- if successful, \mathcal{O} is consistent: "look, here is a (description of a) model"
- otherwise, no model exists – provably (we were not simply too lazy to find it)

Algorithm works on a set of **ABoxes**:

- initialised with a singleton set $\mathcal{S} = \{\mathcal{A}\}$ when started with $\mathcal{O} = (\mathcal{T}, \mathcal{A})$
- ABoxes are extended by rules to make constraints on models of \mathcal{O} explicit
- \mathcal{O} is consistent if, for (at least) one of the ABoxes \mathcal{A}' in \mathcal{S} , $(\mathcal{T}, \mathcal{A}')$ is consistent

Negation Normal Form

Technical: we say C and D are **equivalent**, written $C \equiv D$, if they mutually subsume each other.

Technical: all concepts are assumed to be in **Negation Normal Form**
transform all concepts in \mathcal{O} into $\text{NNF}(C)$ by
pushing negation inwards, using

$$\begin{aligned}\neg(C \sqcap D) &\equiv \neg C \sqcup \neg D & \neg(C \sqcup D) &\equiv \neg C \sqcap \neg D \\ \neg(\exists R.C) &\equiv (\forall R.\neg C) & \neg(\forall R.C) &\equiv (\exists R.\neg C)\end{aligned}$$

Lemma: Let C be an \mathcal{ALC} concept. Then $C \equiv \text{NNF}(C)$.

From now on, all concepts in GCI and concept assertions are assumed to be in NNF, and we use $\dot{\neg}C$ to denote the $\text{NNF}(\neg C)$.

The algorithm

- works on sets of **ABoxes** \mathcal{S}
- starts with a singleton set $\mathcal{S} = \{\mathcal{A}\}$ when started with $\mathcal{O} = (\mathcal{T}, \mathcal{A})$
- applies **rules** that infer constraints on models of \mathcal{O}
- a rule is applied to **some** $\mathcal{A} \in \mathcal{S}$; its application replaces \mathcal{A} with one or two **ABoxes**
- answers " \mathcal{O} is consistent" if rule application leads to an **ABox** \mathcal{A} that is
 - **complete**, i.e., to which no more rules apply and
 - **clash-free**, i.e., $\{a: A, a: \neg A\} \not\subseteq \mathcal{A}$, for any a, A
- for optimisation, we can avoid applying rules to **ABoxes** containing a clash

Preliminary Tableau Expansion Rules for \mathcal{ALC}

- \sqcap -rule: if $a : C_1 \sqcap C_2 \in \mathcal{A}$ and $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1, a : C_2\}$
- \sqcup -rule: if $a : C_1 \sqcup C_2 \in \mathcal{A}$ and $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a : C_1\}$ and $\mathcal{A} \cup \{a : C_2\}$
- \exists -rule: if $a : \exists s.C \in \mathcal{A}$ and there is no b with $\{(a, b) : s, b : C\} \subseteq \mathcal{A}$
then create a new individual name c and
replace \mathcal{A} with $\mathcal{A} \cup \{(a, c) : s, c : C\}$
- \forall -rule: if $\{a : \forall s.C, (a, b) : s\} \subseteq \mathcal{A}$ and $b : C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b : C\}$
- GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$ and $a : (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : (\neg C \sqcup D)\}$

Tableau Algorithm for \mathcal{ALC}

Example: apply the tableau algorithm to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ with

$$\begin{aligned} \mathcal{T} = \{ & A \sqsubseteq B \sqcap \exists r.G \sqcap \forall r.C, & \mathcal{A} = \{ & a: A, b: E, \\ & E \sqsubseteq \forall r.(F \sqcup H), & & (a, c): r, (b, c): r, \\ & F \sqsubseteq \neg C, & & c: G \} \\ & G \sqsubseteq \exists s.B \} \end{aligned}$$

Is \mathcal{O} consistent?

Tableau Algorithm for \mathcal{ALC} : Observations

- We only apply rules if their application does “something new”
- If \mathcal{A} is replaced with \mathcal{A}' , then $\mathcal{A} \subseteq \mathcal{A}'$
- The \sqcup -rule is the only one to replace an ABox with more than one other
- To understand the GCI-rule, convince yourself that

\mathcal{I} satisfies a GCI $C \sqsubseteq D$ iff, for each $e \in \Delta^{\mathcal{I}}$, we have $e \notin C^{\mathcal{I}}$ or $e \in D^{\mathcal{I}}$

– and $e \notin C^{\mathcal{I}}$ is the case iff $e \in (\neg C)^{\mathcal{I}}$

- The GCI-rule adds a disjunction per individual and GCI \Rightarrow this is
 - **bad**, and
 - **stupid** for GCIs with a concept name on its left hand side (why?) \Rightarrow we add an **abbreviated GCI rule**:

GCI-2-rule: if B is a concept name, $a : F \notin \mathcal{A}$ for $a : B \in \mathcal{A}$ and $B \sqsubseteq F \in \mathcal{T}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a : F\}$

Termination of our Tableau Algorithm for \mathcal{ALC}

As is, the tableau algorithm does not terminate:

Example: apply the tableau algorithm to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$ and $\mathcal{A} = \{a: A\}$.

To ensure termination, use **blocking**: each rule is only applicable to an individual a in an ABox \mathcal{A} if there is no other individual b with

$$\{C \mid a: C \in \mathcal{A}\} \subseteq \{C \mid b: C \in \mathcal{A}\}.$$

In case we have

- a freshly introduced individual (i.e., not present in input ontology) a ,
- an individual b with
 - $\{C \mid a: C \in \mathcal{A}\} \subseteq \{C \mid b: C \in \mathcal{A}\}$,
 - b is older than a (i.e., was created earlier than a)

we say b **blocks** a and we say a is **blocked**.

Tableau Expansion Rules for \mathcal{ALC}

- \sqcap -rule: if $a: C_1 \sqcap C_2 \in \mathcal{A}$, a is not blocked, and $\{a: C_1, a: C_2\} \not\subseteq \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{a: C_1, a: C_2\}$
- \sqcup -rule: if $a: C_1 \sqcup C_2 \in \mathcal{A}$, a is not blocked, and $\{a: C_1, a: C_2\} \cap \mathcal{A} = \emptyset$
then replace \mathcal{A} with $\mathcal{A} \cup \{a: C_1\}$ and $\mathcal{A} \cup \{a: C_2\}$
- \exists -rule: if $a: \exists s.C \in \mathcal{A}$, a is not blocked, and there is no b with
 $\{(a, b): s, b: C\} \subseteq \mathcal{A}$
then create a new individual c and replace \mathcal{A} with $\mathcal{A} \cup \{(a, c): s, c: C\}$
- \forall -rule: if $\{a: \forall s.C, (a, b): s\} \subseteq \mathcal{A}$, a is not blocked, and $b: C \notin \mathcal{A}$
then replace \mathcal{A} with $\mathcal{A} \cup \{b: C\}$
- GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, a is not blocked, and
if C is a concept name, $a: C \in \mathcal{A}$ but $a: D \notin \mathcal{A}$,
then replace \mathcal{A} with $\mathcal{A} \cup \{a: D\}$
else if $a: (\neg C \sqcup D) \notin \mathcal{A}$ for a in \mathcal{A} ,
then replace \mathcal{A} with $\mathcal{A} \cup \{a: (\neg C \sqcup D)\}$

Termination of our Tableau Algorithm for \mathcal{ALC}

Convince yourself that, for the given example, the tableau algorithm terminates:

Example: apply the tableau algorithm to $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ with $\mathcal{T} = \{A \sqsubseteq \exists r.A\}$
and $\mathcal{A} = \{a : A\}$.

...now for the general case!

Properties of our tableau algorithm

Lemma 3: Let \mathcal{O} an \mathcal{ALC} ontology in NNF. Then

1. the algorithm terminates when applied to \mathcal{O}
2. if the rules generate a complete & clash-free ABox, then \mathcal{O} is consistent
3. if \mathcal{O} is consistent, then the rules generate a clash-free & complete ABox

- Corollary 1:**
1. Our tableau algorithm **decides consistency** of \mathcal{ALC} ontologies.
 2. (with Theorem 2) all other \mathcal{ALC} reasoning problems are decidable.
 3. Satisfiability (and subsumption) of \mathcal{ALC} concepts is decidable in **PSpace**.
 4. Consistency of \mathcal{ALC} ontologies is decidable in **ExpSpace**.
 5. \mathcal{ALC} ontologies have the **finite model property**
i.e., every consistent ontology has a **finite model**.
 6. \mathcal{ALC} ontologies have the **tree model property**
i.e., every consistent ontology has a **tree model**.

Proof of Lemma 3.1: Termination

Let $\text{sub}(\mathcal{O})$ be the set of all subconcepts of concepts occurring in \mathcal{A} together with all subconcepts of $\neg C \sqcup D$ for each $C \sqsubseteq D \in \mathcal{T}$.

(1) **Termination** is a consequence of these observations:

1. a rule replaces one ABox with at most two ABoxes
2. the ABoxes are constructed in a **monotonic way**,
i.e., each rule adds assertions, nothing is removed
3. concept assertions added are restricted to $\text{sub}(\mathcal{O})$ and

$$\#\text{sub}(\mathcal{O}) \leq \sum_{C \sqsubseteq D \in \mathcal{O}} (2 + |C| + |D|) + \sum_{a: C \in \mathcal{O}} |C|$$

because, at each position in a concept, at most one sub-concept starts

4. due to blocking, there can be at most $2^{\#\text{sub}(\mathcal{O})}$ individuals in each ABox: if $\{C \mid a: C \in \mathcal{A}\} \subseteq \{C \mid b: C \in \mathcal{A}\}$, a is blocked and no rules are applied to a .

Eventually, all ABoxes will be complete (and possibly have a clash), and the algorithm terminates.

Thank you for your attention!