An Introduction to Description Logics:

Techniques, Properties, and Applications

NASSLLI, Day 2, Part 2

Reasoning via Tableau Algorithms

Uli Sattler

Today

- relationship between standard DL reasoning problems
- a tableau algorithm to decide consistency of $\mathcal{ALC}$ ontologies and all other standard DL reasoning problems
- a proof of its correctness
- with some model properties
- some optimisations

As a result, we should have

- a deep understanding the semantics
- a deep understanding of the reasoning problems
- some understanding of the sources of complexity

Standard DL Reasoning Problems

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

- is $\mathcal{O}$ consistent? $\mathcal{O} \models \top \sqsubseteq \bot$?
- is $\mathcal{O}$ coherent? is there concept name A with $\mathcal{O} \models A \sqsubseteq \bot$?
- 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: $\mathcal{O} \models b\colon B$?

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

- intialised 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 $\mathsf{NNF}(C)$ by
pushing negation inwards, using

$$\neg(C \sqcap D) \equiv \neg C \sqcup \neg D \qquad \neg(C \sqcup D) \equiv \neg C \sqcap \neg D$$
$$\neg(\exists R.C) \equiv (\forall R.\neg C) \qquad \neg(\forall R.C) \equiv (\exists R.\neg C)$$

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

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

A tableau algorithm for $\mathcal{ALC}$ ontologies

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., $\{a\colon A,\ a\colon \neg A\} \not\subseteq \mathcal{A}$, for any a, A
- for optimisation, we can avoid applying rules to ABoxes containing a clash

Using the tableau algorithm for $\mathcal{ALC}$ ontologies

Following Theorem 2, we can use the algorithm to test

- satisfiability of a concept C by starting it with $\{a\colon C\}$
- satisfiability of a concept C wr.t. $\mathcal{O}$ by starting it with $\mathcal{O} \cup \{a\colon C\}$ (a not in $\mathcal{O}$)
- subsumption $C \sqsubseteq D$ by starting it with $\{a\colon (C \sqcap \neg D)\}$
- subsumption $C \sqsubseteq D$ wr.t. $\mathcal{O}$ by starting it with $\mathcal{O} \cup \{a\colon (C \sqcap \neg D)\}$ (a not in $\mathcal{O}$)
- whether b is an instance of C w.r.t. $\mathcal{O}$ by starting it with $\mathcal{O} \cup \{b\colon \neg C\}$

- ...and interpreting the results according to Theorem 2.

Preliminary Tableau Expansion Rules for $\mathcal{ALC}$

$\sqcap$-rule: if $a\colon C_1 \sqcap C_2 \in \mathcal{A}$ and $\{a\colon C_1, a\colon C_2\} \not\subseteq \mathcal{A}$

then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon C_1, a\colon C_2\}$

$\sqcup$-rule: if $a\colon C_1 \sqcup C_2 \in \mathcal{A}$ and $\{a\colon C_1, a\colon C_2\} \cap \mathcal{A} = \emptyset$

then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon C_1\}$ and $\mathcal{A} \cup \{a\colon C_2\}$

$\exists$-rule: if $a\colon \exists s.C \in \mathcal{A}$ and there is no b with $\{(a,b)\colon s,\ b\colon C\} \subseteq \mathcal{A}$

then create a new individual name c and
replace $\mathcal{A}$ with $\mathcal{A} \cup \{(a,c)\colon s,\ c\colon C\}$

$\forall$-rule: if $\{a\colon \forall s.C,\ (a,b)\colon s\} \subseteq \mathcal{A}$ and $b\colon C \notin \mathcal{A}$

then replace $\mathcal{A}$ with $\mathcal{A} \cup \{b\colon C\}$

GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$ and $a\colon (\dot{\neg} C \sqcup D) \notin \mathcal{A}$ for a in $\mathcal{A}$,

then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon (\dot{\neg} C \sqcup D)\}$

Tableau Algorithm for $\mathcal{ALC}$: Observations

- We only apply rules if their application does **"something new"**
- 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, $a\colon F \notin \mathcal{A}$ for $a\colon B \in \mathcal{A}$ and $B \sqsubseteq F \in \mathcal{T}$,
then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon F\}$

- If $\mathcal{A}$ is replaced with $\mathcal{A}'$, then $\mathcal{A} \subseteq \mathcal{A}'$

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

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

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 $A = \{a\colon A\}$.

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

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

In case we have

- a freshly introduced individual (i.e., not present in input ontology) a,
- an individual b with
  - $\{C \mid a\colon C \in \mathcal{A}\} \subseteq \{C \mid b\colon C \in \mathcal{A}\}$,
  - 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 $a\colon C_1 \sqcap C_2 \in \mathcal{A}$, a **is not blocked**, and $\{a\colon C_1, a\colon C_2\} \not\subseteq \mathcal{A}$
then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon C_1, a\colon C_2\}$

$\sqcup$-rule: if $a\colon C_1 \sqcup C_2 \in \mathcal{A}$, a **is not blocked**, and $\{a\colon C_1, a\colon C_2\} \cap \mathcal{A} = \emptyset$
then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon C_1\}$ and $\mathcal{A} \cup \{a\colon C_2\}$

$\exists$-rule: if $a\colon \exists s.C \in \mathcal{A}$, a **is not blocked**, and there is no b with
$\{(a,b)\colon s,\ b\colon C\} \subseteq \mathcal{A}$
then create a new individual c and replace $\mathcal{A}$ with $\mathcal{A} \cup \{(a,c)\colon s,\ c\colon C\}$

$\forall$-rule: if $\{a\colon \forall s.C,\ (a,b)\colon s\} \subseteq \mathcal{A}$, a **is not blocked**, and $b\colon C \notin \mathcal{A}$
then replace $\mathcal{A}$ with $\mathcal{A} \cup \{b\colon C\}$

GCI-rule: if $C \sqsubseteq D \in \mathcal{T}$, a **is not blocked**, and
if C is a concept name, $a\colon C \in \mathcal{A}$ but $a\colon D \notin \mathcal{A}$,
then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon D\}$

else if $a\colon (\dot{\neg} C \sqcup D) \notin \mathcal{A}$ for a in $\mathcal{A}$,
then replace $\mathcal{A}$ with $\mathcal{A} \cup \{a\colon (\dot{\neg} C \sqcup D)\}$

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 $A = \{a\colon A\}$.

...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. Satisfiability (and subsumption) of $\mathcal{ALC}$ concepts is decidable in **PSpace**.
3. Consistency of $\mathcal{ALC}$ ontologies is decidable in **ExpSpace**.
4. $\mathcal{ALC}$ ontologies have the **finite model property**
   i.e., every consistent ontology has a **finite** model.
5. $\mathcal{ALC}$ ontologies have the **tree model property**
   i.e., every consistent ontology has a **tree** model.

Proof of Lemma 3.1: Termination

Let **sub**$(\mathcal{O})$ be the set of all subconcepts of concepts occurring in $\mathcal{A}$ together with all subconcepts of $\dot{\neg}C \sqcup D$ 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 **sub**$(\mathcal{O})$ and

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

   because, at each position in a concept, at most one sub-concept starts
4. due to blocking, there can be at most $2^{\#\,\mathsf{sub}(\mathcal{O})}$ individuals in each ABox: if $\{C \mid a\colon C \in \mathcal{A}\} \subseteq \{C \mid b\colon C \in \mathcal{A}\}$, 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.

Proof of Lemma 3.2: Soundness

(2) Let $\mathcal{A}_f$ be a complete & clash-free ABox generated for $\mathcal{O} = (\mathcal{T}, \mathcal{A})$,
and let $\mathcal{B}_f$ be $\mathcal{A}_f$ without assertions involving blocked individuals.
Define an interpretation $\mathcal{I}$ as follows:

$$\begin{array}{rcll}
\Delta^\mathcal{I} & := & \{x \mid x \text{ is an individual in } \mathcal{B}_f\} & \\
A^\mathcal{I} & := & \{x \in \Delta^\mathcal{I} \mid x\colon A \in \mathcal{B}_f\} & \text{for concept names } A \\
r^\mathcal{I} & := & \{(x, y) \in \Delta^\mathcal{I} \times \in \Delta^\mathcal{I} \mid & (x, y)\colon r \in \mathcal{B}_f \text{ or} \\
 & & & (x, y')\colon r \in \mathcal{A}_f \text{ and } y \text{ blocks } y' \text{ in } \mathcal{A}_f\}
\end{array}$$

and show, by induction on structure of concepts:

(C1) $x\colon D \in \mathcal{B}_f$ implies $x \in D^\mathcal{I}$

(C2) $C \sqsubseteq D \in \mathcal{T}$ implies $C^\mathcal{I} \subseteq D^\mathcal{I}$

➙ $\mathcal{I}$ is a model of $(\mathcal{T}, \mathcal{B}_f)$ ($\mathcal{I}$ satisfies all role assertions by definition)
➙ $\mathcal{I}$ is a model of $(\mathcal{T}, \mathcal{A})$ because $\mathcal{A} \subseteq \mathcal{B}_f$
➙ $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is consistent