Chapter 4

Reasoning in DLs with tableau algorithms

We start with an algorithm for deciding consistency of an ABox without a TBox since this covers most of the inference problems introduced in Chapter 2:

- acyclic TBoxes can be eliminated by expansion
- satisfiability, subsumption, and the instance problem can be reduced to ABox consistency

The tableau-based consistency algorithm tries to generate a finite model for the input ABox $\mathcal{A}_0$:

- applies expansion rules to extend the ABox *one rule per constructor*
- checks for obvious contradictions (clashes)
- an ABox that is complete (no rule applies) and clash-free (no obvious contradictions) describes a model

Tableau algorithm

example

$\mathcal{T}$ $\boxed{\text{GoodStudent} \equiv \text{Smart} \sqcap \text{Studious}}$

Subsumption question:

$$\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqsubseteq^{?}_{\mathcal{T}} \exists\text{attended.GoodStudent}$$

Reduction to satisfiability: is the following concept unsatisfiable w.r.t. $\mathcal{T}$?

$$\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \neg\exists\text{attended.GoodStudent}$$

Reduction to consistency: is the following ABox inconsistent w.r.t. $\mathcal{T}$?

$$\{\, a : (\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \neg\exists\text{attended.GoodStudent}) \,\}$$

Expansion: is the following ABox inconsistent?

$$\{\, a : (\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \neg\exists\text{attended.}(\text{Smart} \sqcap \text{Studious})) \,\}$$

Negation normal form: is the following ABox inconsistent?

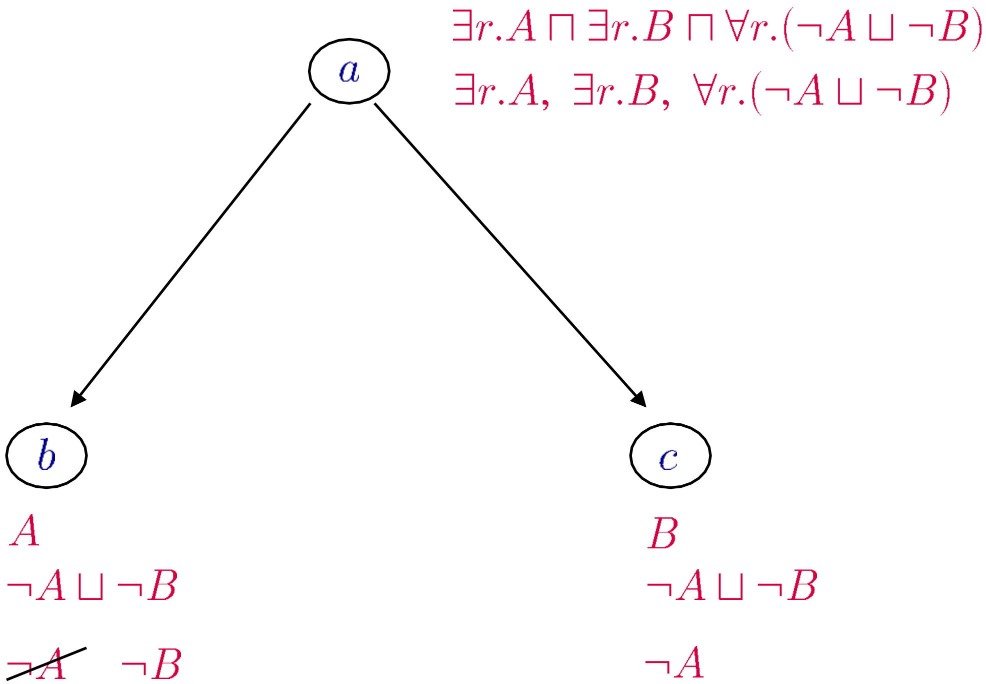
$$\{\, a : (\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \forall\text{attended.}(\neg\text{Smart} \sqcup \neg\text{Studious})) \,\}$$

Tableau algorithm

example continued

Is the following ABox inconsistent?

$$\{\, a : (\exists\text{attended.Smart} \sqcap \exists\text{attended.Studious} \sqcap \forall\text{attended.}(\neg\text{Smart} \sqcup \neg\text{Studious}))\,\}$$



complete and clash-free ABox yields a model for the input ABox

and thus a counterexample to the subsumption relationship

Tableau algorithm

more formal description

Input: An $\mathcal{ALC}$-ABox $\mathcal{A}_0$

Output: "yes" if $\mathcal{A}_0$ is consistent

"no" otherwise

negation only in front of concept names

Preprocessing: normalize the ABox

– transform all concept descriptions in $\mathcal{A}_0$ into negation normal form (NNF) by applying the following equivalence-preserving rules:

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

The NNF can be computed in polynomial time, and it does not change the semantics of the concept.

Tableau algorithm

more formal description

Input: An $\mathcal{ALC}$-ABox $\mathcal{A}_0$

Output: "yes" if $\mathcal{A}_0$ is consistent

"no" otherwise

negation only in front of concept names

Preprocessing: normalize the ABox

- transform all concept descriptions in $\mathcal{A}_0$ into negation normal form (NNF)
- ensure that the ABox is non-empty
  by adding $a : \top$ for an arbitrary individual name a if needed
- ensure that every individual name a occurring in the ABox
  occurs in a concept assertion by adding $a : \top$ if needed

We assume in the following that the input ABox $\mathcal{A}_0$ is normalized in this sense.

Tableau algorithm

more formal description

Application of expansion rules:

- The rules are triggered by the presence of certain assertions in the current ABox,
- and extend the ABox by new assertions.
- Deterministic rule: only one option for how to extend the ABox.
- Nondeterministic rule: several options for how to extend the ABox, where at least one of them must lead to success.

A b

$\neg A \sqcup \neg B$

$\cancel{\neg A}$ $\neg B$

Tableau algorithm

more formal description

Application of expansion rules:

- The rules are triggered by the presence of certain assertions in the current ABox,
- and extend the ABox by new assertion.
- Deterministic rule: only one option for how to extend the ABox.
- Nondeterministic rule: several options for how to extend the ABox, where at least one of them must lead to success.
  - Nondeterministic algorithm: always "guesses" the "right" option.
  - Deterministic realization: try options consecutively and backtrack in case of failure.

Expansion rules

one for every constructor (except for negation)

The $\sqcap$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcap D)$, but not both $a:C$ and $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:C, a:D\}$

The $\sqcup$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcup D)$, but neither $a:C$ nor $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:X\}$ for some $X \in \{C, D\}$

The $\exists$-rule

Condition: $\mathcal{A}$ contains $a:(\exists r.C)$, but there is no b with $\{(a,b):r, b:C\} \subseteq \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a,d):r, d:C\}$ where d is new in $\mathcal{A}$

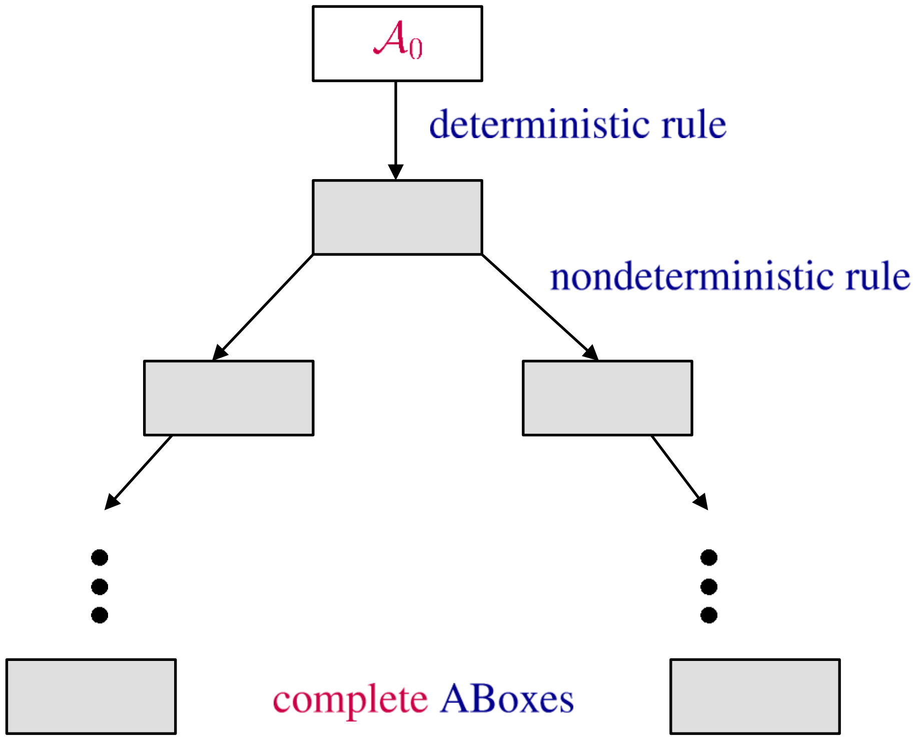
The $\forall$-rule

Condition: $\mathcal{A}$ contains $a:(\forall r.C)$ and $(a,b):r$, but not $b:C$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{b:C\}$

Tableau algorithm

How does it work?



Return “consistent” iff one of these complete ABoxes is clash-free.

Tableau algorithm

more formally

Definition 4.1 (Complete and clash-free ABox)

- An ABox $\mathcal{A}$ contains a clash if

$$\{a : C, a : \neg C\} \subseteq \mathcal{A}$$

  for some individual name a, and for some concept C.

- $\mathcal{A}$ is complete if it contains a clash, or
  if none of the expansion rules is applicable.

Tableau algorithm

more formally

The procedure exp:

- takes as input a normalised and clash-free $\mathcal{ALC}$ ABox $\mathcal{A}$, a rule R and an assertion or pair of assertions α such that R is applicable to α in $\mathcal{A}$;
- it returns a set $\mathsf{exp}(\mathcal{A}, R, \alpha)$ containing each of the ABoxes that can result from applying R to α in $\mathcal{A}$.

Examples:

$\mathsf{exp}(\{a : \neg D, a : C \sqcup D\}, \sqcup\text{-rule}, a : C \sqcup D)$

$\mathsf{exp}(\{b : \neg D, a : \forall r.D, (a, b) : r\}, \forall\text{-rule}, (a : \forall r.D, (a, b) : r))$

Definition 4.2

deterministic version of the tableau algorithm

Algorithm $\mathsf{consistent}()$
Input: a normalised $\mathcal{ALC}$ ABox $\mathcal{A}$
if $\mathsf{expand}(\mathcal{A}) \neq \emptyset$ **then**
    return "consistent"
else
    return "inconsistent"

Algorithm $\mathsf{expand}()$
Input: a normalised $\mathcal{ALC}$ ABox $\mathcal{A}$
if $\mathcal{A}$ is not complete **then**
    select a rule R that is applicable to $\mathcal{A}$ and an assertion
    or pair of assertions α in $\mathcal{A}$ to which R is applicable
    if there is $\mathcal{A}' \in \mathsf{exp}(\mathcal{A}, R, \alpha)$ with $\mathsf{expand}(\mathcal{A}') \neq \emptyset$ **then**
        return $\mathsf{expand}(\mathcal{A}')$
    else
        return $\emptyset$
else
    if $\mathcal{A}$ contains a clash **then**
        return $\emptyset$
    else
        return $\mathcal{A}$

Tableau algorithm

example

$$\begin{aligned}\mathcal{A}_{ex} = \{ & a : A \sqcap \exists s.F, & & (a,b) : s, \\ & a : \forall s.(\neg F \sqcup \neg B), & & (a,c) : r, \\ & b : B, & & c : C \sqcap \exists s.D\}\end{aligned}$$

Expansion rules

one for every constructor (except for negation)

The $\sqcap$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcap D)$, but not both $a:C$ and $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:C, a:D\}$

The $\sqcup$-rule

Condition: $\mathcal{A}$ contains $a:(C \sqcup D)$, but neither $a:C$ nor $a:D$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{a:X\}$ for some $X \in \{C, D\}$

The $\exists$-rule

Condition: $\mathcal{A}$ contains $a:(\exists r.C)$, but there is no b with $\{(a,b):r, b:C\} \subseteq \mathcal{A}$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{(a,d):r, d:C\}$ where d is new in $\mathcal{A}$

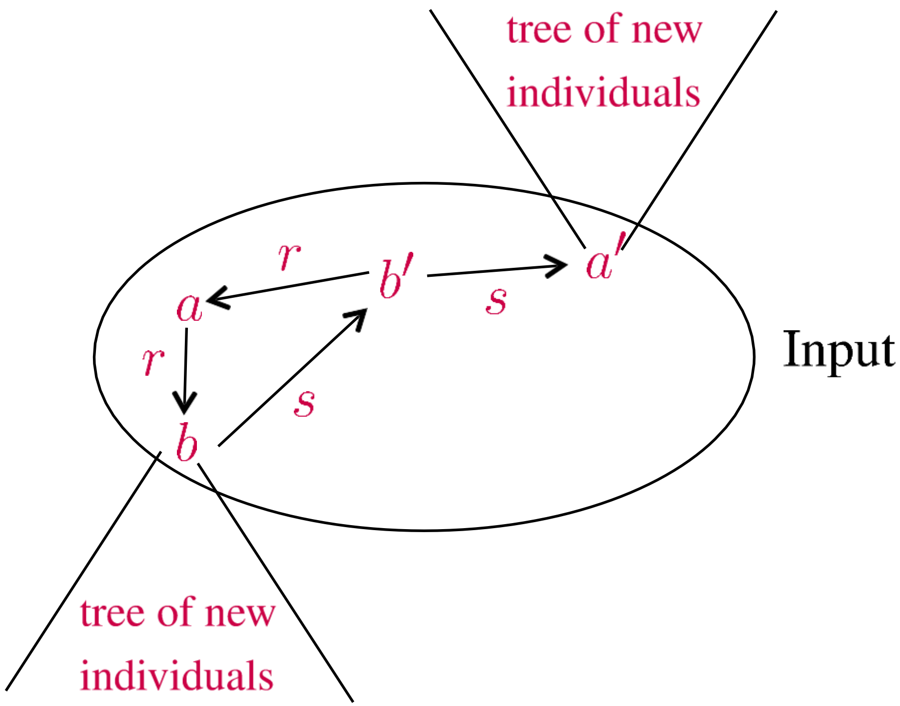
The $\forall$-rule

Condition: $\mathcal{A}$ contains $a:(\forall r.C)$ and $(a,b):r$, but not $b:C$

Action: $\mathcal{A} \longrightarrow \mathcal{A} \cup \{b:C\}$

Trees and forests

In an ABox generated by the algorithm, the individuals generated by the ∃-rule form a tree whose root is an individual from the input ABox.



Trees and forests

In an ABox generated by the algorithm, the individuals generated by the $\exists$-rule form a tree whose root is an individual from the input ABox.

- Root individual: individual occurring in the input ABox
- Tree individual: individual generated by the application of the $\exists$-rule
- If the $\exists$-rule adds a tree individual b and a role assertion $(a, b) : r$, then b is a (r-) successor of a and a is a predecessor of b
- We use ancestor and and descendant for the transitive closure of predecessor and successor, respectively

Note: root individuals may have successors and hence descendants, but they have no predecessor or ancestors.

Tableau algorithm

Why is it a decision procedure for consistency?

We need to show:

Termination:
$\mathsf{consistent}(\mathcal{A})$ terminates for all normalised $\mathcal{ALC}$ ABoxes $\mathcal{A}$

Soundness:
if $\mathsf{consistent}(\mathcal{A})$ returns "consistent", then $\mathcal{A}$ is consistent

Completeness:
if $\mathcal{A}$ is consistent, then $\mathsf{consistent}(\mathcal{A})$ returns "consistent"

Termination

auxiliary definitions and results

Extend the definition of subconcept to ABoxes and to knowledge bases:

$$\mathsf{sub}(\mathcal{A}) = \bigcup_{a:C\in\mathcal{A}} \mathsf{sub}(C)$$

and for $\mathcal{K} = (\mathcal{T}, \mathcal{A})$,

$$\mathsf{sub}(\mathcal{K}) = \mathsf{sub}(\mathcal{T}) \cup \mathsf{sub}(\mathcal{A}).$$

Set of concepts occurring in a concept assertion:

$$\mathsf{con}_{\mathcal{A}}(a) = \{C \mid a : C \in \mathcal{A}\}.$$

Lemma 4.3

For each $\mathcal{ALC}$ ABox $\mathcal{A}$, we have that $|\,\mathsf{sub}(\mathcal{A})| \leq \sum_{a:C\in\mathcal{A}} \mathsf{size}(C)$.

linear in the size of $\mathcal{A}$

Termination

Lemma 4.4 (Termination)

For each normalized $\mathcal{ALC}$ ABox $\mathcal{A}$, $\mathsf{consistent}(\mathcal{A})$ terminates.

Soundness

Lemma 4.5 (Soundness)

If $\mathsf{consistent}(\mathcal{A})$ returns "consistent", then $\mathcal{A}$ is consistent.

Proof. Let $\mathcal{A}'$ be the set returned by $\mathsf{expand}(\mathcal{A})$.

Since the algorithm returns "consistent", $\mathcal{A}'$ is a complete and clash-free ABox.

We use $\mathcal{A}'$ to define an interpretation $\mathcal{I}$ and show that it is a model of $\mathcal{A}'$.

$$
\begin{aligned}
\Delta^{\mathcal{I}} &= \{a \mid a : C \in \mathcal{A}'\} \\
a^{\mathcal{I}} &= a \text{ for each individual name } a \text{ occurring in } \mathcal{A}' \\
A^{\mathcal{I}} &= \{a \mid A \in \mathsf{con}_{\mathcal{A}'}(a)\} \text{ for each concept name } A \text{ in } \mathsf{sub}(\mathcal{A}') \\
r^{\mathcal{I}} &= \{(a,b) \mid (a,b) : r \in \mathcal{A}'\} \text{ for each role } r \text{ occurring in } \mathcal{A}'
\end{aligned}
$$

Since the expansion rules never delete assertions, we have $\mathcal{A} \subseteq \mathcal{A}'$, so $\mathcal{I}$ is also a model of $\mathcal{A}$.

Soundness

proof continued

$$
\begin{array}{rcl}
\Delta^{\mathcal{I}} & = & \{a \mid a : C \in \mathcal{A}'\} \\
a^{\mathcal{I}} & = & a \text{ for each individual name } a \text{ occurring in } \mathcal{A}' \\
A^{\mathcal{I}} & = & \{a \mid A \in \mathsf{con}_{\mathcal{A}'}(a)\} \text{ for each concept name } A \text{ in } \mathsf{sub}(\mathcal{A}') \\
r^{\mathcal{I}} & = & \{(a,b) \mid (a,b) : r \in \mathcal{A}'\} \text{ for each role } r \text{ occurring in } \mathcal{A}'
\end{array}
$$

The interpretation $\mathcal{I}$ it is a model of $\mathcal{A}'$.

Completeness

Lemma 4.6 (Completeness)

If $\mathcal{A}$ is consistent, then $\mathsf{consistent}(\mathcal{A})$ returns "consistent".

Proof. Let $\mathcal{A}$ be consistent, and consider a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ of $\mathcal{A}$.

Since $\mathcal{A}$ is consistent, it cannot contain a clash.

Thus, if $\mathcal{A}$ is complete, then expand simply returns $\mathcal{A}$ and $\mathsf{consistent}(\mathcal{A})$ returns "consistent".

If $\mathcal{A}$ is not complete, then expand calls itself recursively until $\mathcal{A}$ is complete; each call selects a rule and applies it.

It is thus sufficient to show that rule application preserves consistency.

Tableau algorithm

Why is it a decision procedure for consistency?

We have shown:

Termination:
$\mathsf{consistent}(\mathcal{A})$ terminates for all normalised $\mathcal{ALC}$ ABoxes $\mathcal{A}$

Soundness:
if $\mathsf{consistent}(\mathcal{A})$ returns “consistent”, then $\mathcal{A}$ is consistent

Completeness:
if $\mathcal{A}$ is consistent, then $\mathsf{consistent}(\mathcal{A})$ returns “consistent”

Theorem 4.7

The tableau algorithm presented in Definition 4.2 is a decision procedure for the consistency of $\mathcal{ALC}$ ABoxes.