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 $\textit{one rule per constructor}$
- checks for obvious contradictions (clashes)
- an ABox that is complete (no rule applies) and clash-free (no obvious contradictions) describes a model
Tableau algorithm

\[ \mathcal{T} \quad \text{GoodStudent} \equiv \text{Smart} \sqcap \text{Studious} \]

Subsumption question:
\[ \exists \text{attended. Smart} \sqcap \exists \text{attended. Studious} \sqsubseteq^2_{\mathcal{T}} \exists \text{attended. GoodStudent} \]

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?
\[ \{ a : (\exists \text{attended. Smart} \sqcap \exists \text{attended. Studious} \sqcap \neg \exists \text{attended. (Smart \sqcap Studious)}) \} \]

Negation normal form: is the following ABox inconsistent?
\[ \{ a : (\exists \text{attended. Smart} \sqcap \exists \text{attended. Studious} \sqcap \forall \text{attended. (\neg Smart \sqcup \neg Studious)}) \} \]
Tableau algorithm

Is the following ABox inconsistent?

\[ \{ a : (\exists \text{attended}.\text{Smart} \land \exists \text{attended}.\text{Studious} \land \forall \text{attended}.(\neg \text{Smart} \cup \neg \text{Studious})) \} \]

\[ \exists r. A \land \exists r. B \land \forall r. (\neg A \cup \neg B) \]
\[ \exists r. A, \exists r. B, \forall r. (\neg A \cup \neg B) \]

\[ a \]

\[ b \]
\[ \neg A \cup \neg B \]
\[ \neg A \]
\[ \neg B \]

\[ c \]
\[ B \]
\[ \neg A \cup \neg B \]
\[ \neg A \]

complete and clash-free ABox yields a model for the input ABox and thus a counterexample to the subsumption relationship
Tableau algorithm

Input: An $\mathcal{ALC}$-ABox $A_0$

Output: “yes” if $A_0$ is consistent
“no” otherwise

Preprocessing: normalize the ABox

- transform all concept descriptions in $A_0$ into negation normal form (NNF) by applying the following equivalence-preserving rules:

\[
\begin{align*}
\neg(C \sqcap D) & \leadsto \neg C \sqcup \neg D \\
\neg(C \sqcup D) & \leadsto \neg C \sqcap \neg D \\
\neg \neg C & \leadsto C \\
\neg (\exists r.C') & \leadsto \forall r. \neg C \\
\neg (\forall r.C') & \leadsto \exists r. \neg C
\end{align*}
\]

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

Exercise!
**Tableau algorithm**

Input: An $\mathcal{ALC}$-ABox $A_0$

Output: “yes” if $A_0$ is consistent  
“no” otherwise

Preprocessing: normalize the ABox
- transform all concept descriptions in $A_0$ 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 $A_0$ 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.

\[
\begin{align*}
A & \quad b \\
\lnot A & \cup \lnot B \\
\Rightarrow A & \quad \lnot B
\end{align*}
\]
Tableau algorithm

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)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqcap$-rule</td>
<td>$\mathcal{A}$ contains $a:(C \sqcap D)$, but not both $a:C$ and $a:D$</td>
<td>$\mathcal{A} \rightarrow \mathcal{A} \cup {a:C, a:D}$</td>
</tr>
<tr>
<td>$\sqcup$-rule</td>
<td>$\mathcal{A}$ contains $a:(C \sqcup D)$, but neither $a:C$ nor $a:D$</td>
<td>$\mathcal{A} \rightarrow \mathcal{A} \cup {a:X}$ for some $X \in {C, D}$</td>
</tr>
<tr>
<td>$\exists$-rule</td>
<td>$\mathcal{A}$ contains $a:(\exists r.C)$, but there is no $b$ with ${(a, b):r, b:C} \subseteq \mathcal{A}$</td>
<td>$\mathcal{A} \rightarrow \mathcal{A} \cup {(a, d):r, d:C}$ where $d$ is new in $\mathcal{A}$</td>
</tr>
<tr>
<td>$\forall$-rule</td>
<td>$\mathcal{A}$ contains $a:(\forall r.C)$ and $(a, b):r$, but not $b:C$</td>
<td>$\mathcal{A} \rightarrow \mathcal{A} \cup {b:C}$</td>
</tr>
</tbody>
</table>
Tableau algorithm

How does it work?

\[ A_0 \]

deterministic rule

nondeterministic rule

complete ABoxes

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.
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 \( \alpha \) such that \( R \) is applicable to \( \alpha \) in \( \mathcal{A} \);

- it returns a set \( \exp(\mathcal{A}, R, \alpha) \) containing each of the ABoxes that can result from applying \( R \) to \( \alpha \) in \( \mathcal{A} \).

Examples:

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

\[
\exp(\{b : \neg D, a : \forall r.D, (a, b) : r\}, \forall\text{-rule}, (a : \forall r.D, (a, b) : r))
\]
Algorithm consistent()
Input: a normalised \( \mathcal{ALC} \) ABox \( \mathcal{A} \)
if \( \text{expand}(\mathcal{A}) \neq \emptyset \) then
  return “consistent”
else
  return “inconsistent”

Algorithm 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 \( \alpha \) in \( \mathcal{A} \) to which \( R \) is applicable
  if there is \( \mathcal{A}' \in \text{exp}(\mathcal{A}, R, \alpha) \) with \( \text{expand}(\mathcal{A}') \neq \emptyset \) then
    return \( \text{expand}(\mathcal{A}') \)
  else
    return \( \emptyset \)
else
  if \( \mathcal{A} \) contains a clash then
    return \( \emptyset \)
  else
    return \( \mathcal{A} \)
Tableau algorithm

\[ A_{\text{ex}} = \{ a : A \sqcap \exists s. F, \quad (a, b) : s, \]
\[ a : \forall s. (\neg F \sqcup \neg B), \quad (a, c) : r, \]
\[ b : B, \quad c : C \sqcap \exists s. D \} \]
## Expansion rules

One for every constructor (except for negation)

### The $\square$-rule

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

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

### 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:** $\mathcal{A} \rightarrow \mathcal{A} \cup \{(a, d) : r, d : C\}$ 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:** $\mathcal{A} \rightarrow \mathcal{A} \cup \{ b : C \}$
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.
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.

- **Root individual**: individual occurring in the input ABox
- **Tree individual**: individual generated by the application of the ∃-rule
- **If the ∃-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:
consistent(\(\mathcal{A}\)) terminates for all normalised \(\mathcal{ALC}\) ABoxes \(\mathcal{A}\)

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

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

auxiliary definitions and results

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

\[ \text{sub}(\mathcal{A}) = \bigcup_{a : C \in \mathcal{A}} \text{sub}(C) \]

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

\[ \text{sub}(\mathcal{K}) = \text{sub}(\mathcal{T}) \cup \text{sub}(\mathcal{A}). \]

Set of concepts occurring in a concept assertion:

\[ \text{con}_\mathcal{A}(a) = \{ C' \mid a : C \in \mathcal{A} \}. \]

**Lemma 4.3**

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

linear in the size of \( \mathcal{A} \)
Lemma 4.4 (Termination)

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

Proof: blackboard
**Soundness**

**Lemma 4.5** (Soundness)

If \(\text{consistent}(\mathcal{A})\) returns “consistent”, then \(\mathcal{A}\) is consistent.

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

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 \(\mathcal{A}'\).

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

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

\[
\begin{align*}
\Delta^I &= \{ a \mid a : C \in \mathcal{A}' \} \\
\alpha^I &= \text{a for each individual name a occurring in } \mathcal{A}' \\
\lambda^I &= \{ a \mid A \in \text{con}_{\mathcal{A}}(a) \} \text{ for each concept name } A \text{ in } \text{sub}(\mathcal{A}') \\
\tau^I &= \{ (a, b) \mid (a, b) : r \in \mathcal{A}' \} \text{ for each role } r \text{ occurring in } \mathcal{A}'
\end{align*}
\]

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

Proof: blackboard
Completeness

Lemma 4.6 (Completeness)

If $\mathcal{A}$ is consistent, then $\text{consistent}(\mathcal{A})$ 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 $\text{consistent}(\mathcal{A})$ 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.

Proof: blackboard
Tableau algorithm

Why is it a decision procedure for consistency?

We have shown:

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

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

- **Completeness:**
  if \(\mathcal{A}\) is consistent, then 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.
We will see in Chapter 5 that the complexity of the $\mathcal{ALC}$ ABox consistency problem is $\text{PSPACE}$-complete.

However, the tableau algorithm as described until now needs exponential time and space for two reasons:

- Due to the nondeterministic $\sqcap$-rule, exponentially many complete ABoxes may be generated.

\[ \mathcal{A}_0 \]

\[ \text{deterministic rule} \]

\[ \text{nondeterministic rule} \]

\[ \text{complete ABoxes} \]
Tableau algorithm

We will see in Chapter 5 that the complexity of the \( \mathcal{ALC} \) ABox consistency problem is \( \text{PSPACE-complete} \).

However, the tableau algorithm as described until now needs exponential time and space for two reasons:

- Due to the nondeterministic \( \sqcup \)-rule, exponentially many complete ABoxes may be generated.

- Due to the interaction of \( \forall \) and \( \exists \), complete ABoxes may be exponentially large.

The call \( \text{consistent}(\{C_n(a)\}) \) generates a single complete ABox of size exponential in \( n \).

\[
\begin{align*}
C_1 & : = \exists r.A \sqcap \exists r.B \\
C_{i+1} & : = \exists r.A \sqcap \exists r.B \sqcap \forall r.C_i
\end{align*}
\]

size of \( C_n \) is linear in \( n \)
The tableau algorithm can be modified such that it uses only polynomial space:

- Due to the nondeterministic □-rule, exponentially many complete ABoxes may be generated.
  
  - use a nondeterministic algorithm, which always chooses the correct alternative (if possible);
  - thus only one complete ABox is generated;
  - use Savitch’s theorem, which says that PSpace = NPSpace.
The tableau algorithm can be modified such that it uses only polynomial space:

- Due to the nondeterministic □-rule, exponentially many complete ABoxes may be generated.

- Due to the interaction of ∀- and ∃, complete ABoxes may be exponentially large.

Idea:

generate/explore the tree in a depth-first manner while keeping only one path in memory
Tableau algorithm w.r.t. acyclic TBoxes

In principle, consistency of ABoxes w.r.t. acyclic TBoxes can be reduced to consistency of ABoxes without TBox by unfolding.

Problem: unfolding of an acyclic TBox may result in an exponential blow-up.

Idea: unfolding only “on demand” (lazy unfolding)

The $\equiv_1$-rule

**Condition:** $a : A \in \mathcal{A}, A \equiv C \in \mathcal{T}$, and $a : C \not\in \mathcal{A}$

**Action:** $\mathcal{A} \rightarrow \mathcal{A} \cup \{a : C\}$

The $\equiv_2$-rule

**Condition:** $a : \neg A \in \mathcal{A}, A \equiv C \in \mathcal{T}$, and $a : \neg C \not\in \mathcal{A}$

**Action:** $\mathcal{A} \rightarrow \mathcal{A} \cup \{a : \neg C\}$

Negation normal form of $\neg C$

Termination, soundness, and completeness can be shown similarly to the case without TBox (Exercise).
Tableau algorithm w.r.t. general TBoxes

Preprocessing: also normalize the TBox

- transform all GCIs in $\mathcal{T}$ into the form $\top \sqsubseteq E$

$$\mathcal{I} \text{ satisfies } C \sqsubseteq D \quad \text{iff} \quad \mathcal{I} \text{ satisfies } \top \sqsubseteq D \sqcup \neg C$$

- transform the right-hand sides $E$ of GCIs $\top \sqsubseteq E$ in $\mathcal{T}$ into NNF

We assume in the following that the input TBox $\mathcal{T}$ is normalized in this sense.
Tableau algorithm w.r.t. general TBoxes

Add a new expansion rule that takes the semantics of normalized GCIs into account:

The \( \sqsubseteq \)-rule

- **Condition:** \( a : C \in A, \top \sqsubseteq D \in T, a : D \notin A \)
- **Action:** \( A \rightarrow A \cup \{a : C\} \)

Note: since the input ABox is normalized, all individuals occur in a concept assertion.
Tableau algorithm w.r.t. general TBoxes

Add a new expansion rule that takes the semantics of normalized GCIs into account:

The $\sqsubseteq$-rule

**Condition:** $a : C \in \mathcal{A}$, $\top \sqsubseteq D \in \mathcal{T}$, $a : D \notin \mathcal{A}$

**Action:** $\mathcal{A} \rightarrow \mathcal{A} \cup \{a : C\}$

Soundness and completeness of the tableau algorithm extended with this rule is easy to show.

Termination? Need not hold!

Example: $(\{A \sqsubseteq \exists r.A\}, \{a : A\})$
How can we regain termination.

**Definition 4.8 (\(\mathcal{ALC}\) blocking)**

An individual name \(b\) in an \(\mathcal{ALC}\) ABox \(\mathcal{A}\) is blocked by an individual name \(a\) if

- \(a\) is an ancestor of \(b\) and
- \(\text{con}_{\mathcal{A}}(a) \supseteq \text{con}_{\mathcal{A}}(b)\).

An individual name \(b\) is blocked in \(\mathcal{A}\) if

- it is blocked by some individual name \(a\), or
- if one or more of its ancestors is blocked in \(\mathcal{A}\).

When it is clear from the context, we may not mention the ABox explicitly; e.g., we may simply say that \(b\) is blocked.
Tableau algorithm w.r.t. general TBoxes

The tableau algorithm for $\mathcal{ALC}$ knowledge base consistency uses

- the $\cap$-rule, the $\sqcup$-rule, the $\forall$-rule without changes,
- the new $\sqsubseteq$-rule,
- the following modified $\exists$-rule:

The modified $\exists$-rule

**Condition:** $\mathcal{A}$ contains $a : (\exists r. C)$, but there is no $b$ with $\{(a, b) : r, b : C\} \subseteq \mathcal{A}$ and $a$ is not blocked

**Action:** $\mathcal{A} \rightarrow \mathcal{A} \cup \{(a, d) : r, d : C\}$, where $d$ is new in $\mathcal{A}$
Algorithm consistent()
Input: a normalised $\mathcal{ALC}$ KB $(\mathcal{T}, \mathcal{A})$  
if expand$(\mathcal{T}, \mathcal{A}) \neq \emptyset$ then  
    return “consistent”  
else  
    return “inconsistent”

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

Definition 4.9

deterministic version of the tableau algorithm for KB consistency
Termination

Lemma 4.10 (Termination)

For each normalized $\mathcal{ALC}$ KB $\mathcal{K}$, $\text{consistent}(\mathcal{K})$ terminates.

Proof: blackboard
Soundness

**Lemma 4.11** (Soundness)

If \( \text{consistent}(\mathcal{K}) \) returns “consistent”, then \( \mathcal{K} \) is consistent.

**Proof.** Let \( \mathcal{A}' \) be the set returned by \( \text{expand}(\mathcal{K}) \).

We use \( \mathcal{A}' \) to construct a suitable model \( \mathcal{I} = (\Delta^I, \cdot^I) \) of \( \mathcal{K} \) in two steps:

- Construct a new ABox \( \mathcal{A}'' \) that contains
  - those axioms in \( \mathcal{A}' \) that do not involve blocked individual names
  - new “loop-back” role assertions:

\[
\begin{array}{c}
\alpha \\
r \\
\downarrow \\
b
\end{array} \quad r \\
\begin{array}{c}
b' \\
b \\
\end{array}
\]

blocks \( b \)
Soundness

Lemma 4.11 (Soundness)

If consistent(\mathcal{K}) returns “consistent”, then \mathcal{K} is consistent.

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

We use \mathcal{A}' to construct a suitable model \mathcal{I} = (\Delta^\mathcal{I}, .^\mathcal{I}) of \mathcal{K} in two steps:

- Construct a new ABox \mathcal{A}'' that contains
  - those axioms in \mathcal{A}' that do not involve blocked individual names
  - new “loop-back” role assertions:

- Use \mathcal{A}'' to construct a model of \mathcal{K}. 
Soundness

- Construct a new ABox $A''$ that contains
  - those axioms in $A'$ that do not involve blocked individual names
  - new “loop-back” role assertions:

\[
A'' = \{ a : C \mid a : C \in A' \text{ and } a \text{ is not blocked} \} \cup \\
\{(a, b) : r \mid (a, b) : r \in A' \text{ and } b \text{ is not blocked} \} \cup \\
\{(a, b') : r \mid (a, b) : r \in A', a \text{ is not blocked and } b \text{ is blocked by } b' \}
\]
Soundness

- Construct a new ABox $\mathcal{A}''$ that contains
  - those axioms in $\mathcal{A}'$ that do not involve blocked individual names
  - new “loop-back” role assertions:

\[
\mathcal{A}'' = \{ a : C \mid a : C \in \mathcal{A}' \text{ and } a \text{ is not blocked} \} \cup \\
\{ (a, b) : r \mid (a, b) : r \in \mathcal{A}' \text{ and } b \text{ is not blocked} \} \cup \\
\{ (a, b') : r \mid (a, b) : r \in \mathcal{A}', a \text{ is not blocked and } b \text{ is blocked by } b' \}
\]

The following holds:

- $\mathcal{A} \subseteq \mathcal{A}''$ and none of the individual names occurring in $\mathcal{A}''$ is blocked
- $\text{con}_{\mathcal{A}''}(a) = \text{con}_{\mathcal{A}'}(a)$ for all individuals $a$ occurring in $\mathcal{A}''$
- Since $\mathcal{A}'$ is clash-free, and complete, $\mathcal{A}''$ is also clash-free and complete
Soundness

- Use $A''$ to construct a model of $K$.

We construct an interpretation $I$ from $A''$ exactly as in the proof of Lemma 4.5:

- $\Delta^I = \{a \mid a \text{ is an individual name occurring in } A''\}$
- $a^I = a \text{ for each individual name } a \text{ occurring in } A''$
- $A^I = \{A \mid A \in \text{con}_{A''}(a)\} \text{ for each concept name } A \text{ occurring in } A''$
- $r^I = \{(a, b) \mid (a, b) : r \in A''\} \text{ for each role } r \text{ occurring in } A''$

- $I$ is a model of $A''$ and hence of $A$
- $I$ is a model of $T$

Proof: blackboard
Completeness

Lemma 4.12 (Completeness)

If \( \mathcal{K} \) is consistent, then consistent \((\mathcal{K})\) returns “consistent”.

Proof. It only remains to show that the \( \Box \)-rule preserves KB consistency.

Blackboard

Theorem 4.13

The tableau algorithm presented in Definition 4.9 is a decision procedure for the consistency of \( ALC \) knowledge bases