Chapter 4

Reasoning with Tableaux Algorithms
We start with an algorithm for deciding **consistency of an ABox without a TBox**. This covers most of the inference problems introduced in Chapter 2:

- acyclic TBoxes can be eliminated by **expansion**
- satisfiability, subsumption, equivalence, and instance checking can be reduced to ABox consistency

**Idea**

The **tableau-based consistency algorithm** tries to generate a **finite model** for the input ABox $\mathcal{A}_0$ (one rule per constructor)

- **tableau rules** extend the ABox
- **obvious contradictions** show inconsistency
- if an ABox is:
  - complete (no rule applicable) and
  - open (no contradictions found),
  then it describes a model
Example

\[ \mathcal{T} : \ \text{Heroine} \equiv \text{Hero} \sqcap \text{Female} \]

Subsumption:
\[ \exists \text{helps.} \text{Hero} \sqcap \exists \text{helps.} \text{Female} \sqsubseteq \mathcal{T} \exists \text{helps.} \text{Heroine} \]

Reduction to satisfiability: is the following concept unsatisfiable w.r.t. \( \mathcal{T} \)?
\[ \exists \text{helps.} \text{Hero} \sqcap \exists \text{helps.} \text{Female} \sqcap \neg \exists \text{helps.} \text{Heroine} \]

Reduction to consistency: is the following ABox inconsistent w.r.t. \( \mathcal{T} \)?
\[ \{ (\exists \text{helps.} \text{Hero} \sqcap \exists \text{helps.} \text{Female} \sqcap \neg \exists \text{helps.} \text{Heroine}) (a) \} \]

Expansion: is the following ABox inconsistent?
\[ \{ (\exists \text{helps.} \text{Hero} \sqcap \exists \text{helps.} \text{Female} \sqcap \neg \exists \text{helps.} (\text{Hero} \sqcap \text{Female})) (a) \} \]

NNF: is the following ABox inconsistent?
\[ \{ (\exists \text{helps.} \text{Hero} \sqcap \exists \text{helps.} \text{Female} \sqcap \forall \text{helps.} (\neg \text{Hero} \sqcup \neg \text{Female})) (a) \} \]
Expansion of the ABox

Deciding inconsistency of

\{ (\exists \text{helps.Hero} \land \exists \text{helps.Female} \land \forall \text{helps.}(\neg \text{Hero} \sqcup \neg \text{Female})) (a) \}\n
This is a complete, open ABox → model of input ABox

It is consistent, subsumption does not hold
Formal Algorithm

Input An ABox $\mathcal{A}_0$

Output “yes” if $\mathcal{A}_0$ is consistent, “no” otherwise

Preprocessing: transform all concept descriptions of $\mathcal{A}_0$ to negation normal form

$\neg (C \cap D) \leadsto \neg C \cup \neg D$

$\neg (C \cup D) \leadsto \neg C \cap \neg D$

$\neg \neg C \leadsto C$

$\neg (\exists r. C) \leadsto \forall r. \neg C$

$\neg (\forall r. C) \leadsto \exists r. \neg C$

NNF transformation in polynomial time, is semantics invariant
Formal Algorithm (2)

Data Structure:
finite set of ABoxes. Initialized to \( \mathcal{A}_0 \) (in NNF)

Rule applications:
tableau rules replace one ABox from the set by finitely many new ABoxes

Termination:
when no rule can be applied to any ABox in the set
ABox is complete if no rule applies to it

Return:

“yes” if the set contains an open ABox
\( \mathcal{A} \) is open if contains no obvious contradiction of the form \( A(a), \neg A(a) \)
“no” otherwise (i.e., if all ABoxes are closed)
## Tableau Rules

There is one rule for each constructor (except negation)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\cap)-rule</td>
<td>(\mathcal{A}) contains ((C \cap D)(a)) but not both (C(a)) and (D(a))</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {C(a), D(a)})</td>
</tr>
<tr>
<td>(\sqcup)-rule</td>
<td>(\mathcal{A}) contains ((C \sqcup D)(a)) but neither (C(a)) nor (D(a))</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {C(a)}) and (\mathcal{A}'' := \mathcal{A} \cup {D(a)})</td>
</tr>
<tr>
<td>(\exists)-rule</td>
<td>(\mathcal{A}) contains ((\exists r.C)(a)) but there is no (b) with ({r(a, b), C(b)} \subseteq \mathcal{A})</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {r(a, b), C(b)}) where (b) is a new individual name</td>
</tr>
<tr>
<td>(\forall)-rule</td>
<td>(\mathcal{A}) contains ((\forall r.C)(a)) and (r(a, b)) but not (C(b))</td>
<td>(\mathcal{A}' := \mathcal{A} \cup {C(b)})</td>
</tr>
</tbody>
</table>
Tableau Algorithm is a Decision Procedure

Lemma 4.1
rules preserve consistency

Lemma 4.10
termination

Lemma 4.4
soundness complete and open = consistent
completeness closed = inconsistent

A₀

deterministic rule

non-deterministic rule

complete ABoxes

...