Technical detail: the tableau algorithm

- works on a tree (semantics through viewing tree as an ABox):
  - **nodes** represent elements of $\Delta^T$, labelled with sub-concepts of $C_0$
  - **edges** represent role-successorships between elements of $\Delta^T$
- works on concepts in negation normal form: push negation inside using de Morgan’ laws and
  
  $\neg(\exists R. C) \leadsto \forall R. \neg C$
  $\neg(\forall R. C) \leadsto \exists R. \neg C$
  $\neg(\leq n R) \leadsto (\geq (n + 1) R)$
  $\neg(\geq n R) \leadsto (\leq (n - 1) R)$ (n ≥ 0)
  $\neg(\geq 0 R) \leadsto A \sqcap \neg A$
- is initialised with a tree consisting of a single (root) node $x_0$ with $\mathcal{L}(x_0) = \{ C_0 \}$:
  
  $x_0 \bullet \{ C_0 \}$
- a tree $T$ contains a clash if, for a node $x$ in $T$,
  
  $\{ A, \neg A \} \subseteq \mathcal{L}(x)$ or
  
  $\{ (\geq m R), (\leq n R) \} \subseteq \mathcal{L}(x)$ for $n < m$
### Reasoning Procedures: $\mathcal{ALC}$ Tableau Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \bullet {C_1 \sqcap C_2, \ldots}$</td>
<td>$\rightarrow \sqcap$</td>
<td>$x \bullet {C_1 \sqcap C_2, C_1, C_2, \ldots}$</td>
</tr>
<tr>
<td>$x \bullet {C_1 \sqcup C_2, \ldots}$</td>
<td>$\rightarrow \sqcup$</td>
<td>$x \bullet {C_1 \sqcap C_2, C, \ldots}$ for $C \in {C_1, C_2}$</td>
</tr>
<tr>
<td>$x \bullet {\exists R.C, \ldots}$</td>
<td>$\rightarrow \exists$</td>
<td>$x \bullet {\exists R.C, \ldots}$</td>
</tr>
<tr>
<td>$\rightarrow \forall$</td>
<td>$x \bullet {\forall R.C, \ldots}$</td>
<td></td>
</tr>
<tr>
<td>$x \bullet {C}$</td>
<td>$\rightarrow \exists$</td>
<td>$x \bullet {\exists R.C, \ldots}$</td>
</tr>
<tr>
<td>$y \bullet {C}$</td>
<td>$\rightarrow \forall$</td>
<td>$y \bullet {C, \ldots}$</td>
</tr>
</tbody>
</table>
Reasoning Procedures: $\mathcal{N}$ Tableau Rules

$x \cdot \{ (\geq n \ R), \ldots \}$

$x$ has no $R$-succ.

$x \cdot \{ (\leq n \ R), \ldots \}$

$R$

$y \bullet \{ \}$

merge two $R$-succs.
Lemma

Let $C_0$ be an $\mathcal{ALCN}$ concept and $T$ obtained by applying the tableau rules to $C_0$. Then

1. the rule application terminates,
2. if $T$ is consistent and $\rightarrow$ is applicable to $T$, then $\rightarrow$ can be applied such that it yields consistent $T'$,
3. if $T$ contains a clash, then $T$ has no model, and
4. if no more rules apply to $T$, then $T$ defines (canonical) model for $C_0$.

Corollary

(1) The tableau algorithm is a PSpace decision procedure for consistency (and subsumption) of $\mathcal{ALCN}$ concepts

(2) $\mathcal{ALCN}$ has the tree model property
Proof of the Lemma

1. (Termination) The algorithm “monotonically” constructs a tree whose
   depth is linear in $|C_0|$: quantifier depth decreases from node to succs.
   breadth is linear in $|C_0|$ (even if number in NRs are coded binarily)

2. (Local Consistency) Easy to prove (by definition of the semantics) that
   if $\mathcal{I}$ is a model of $T$, then $\rightarrow$ can be applied to $T$ such that
   $\mathcal{I}$ is a model of $T' := \rightarrow(T)$

3. Obvious: $T$ with a clash has no model—recall definition of a clash:
   \[
   \{A, \neg A\} \subseteq L(x) \text{ or }
   \{(\geq m \cdot R), (\leq n \cdot R)\} \subseteq L(x) \text{ for } n < m
   \]
Proof of the Lemma (ctd.)

4. (Canonical model) “Complete” tree $T$ defines a (tree) pre-model $I$:
   - nodes correspond to elements of $\Delta^I$
   - edges define role-relationship
   - $x \in A^I$ iff $A \in \mathcal{L}(x)$ for concept names $A$

   Check that $C \in \mathcal{L}(x)$ implies $x \in C^I$—if $C$ is no number restriction.

For NRs, if $(\geq n \ R) \in \mathcal{L}(x)$ and $x$ has less than $n \ R$-successors,
   copy some $\ R$-successors (including sub-trees) to obtain $n \ R$-successors

$\implies$ canonical tree model for input concept
To make the tableau algorithm run in PSpace:

Recall Savitch: \( \text{PSPACE} = \text{NPSPACE} \)

① observe that branches are independent from each other
② observe that each node (label) requires linear space only
③ recall that paths are of length \( \leq |C_0| \)
   \( \sim \) each path can be stored in \( \mathcal{O}(|C_0|^2) \)
④ construct/search the tree \textbf{depth first}
⑤ re-use space from already constructed branches
This tableau algorithm can be modified to a PSpace decision procedure for

- $\mathcal{ALC}$ with qualifying number restrictions $(\geq n \ R \ C)$ and $(\leq n \ R \ C)$
- $\mathcal{ALC}$ with inverse roles (e.g. $\text{has-child}^-$)
- $\mathcal{ALC}$ with role conjunction
  $\exists (R \sqcap S).C$ and $\forall (R \sqcap S).C$
- TBoxes with acyclic concept definitions $A \equiv C$:  
  unfolding (macro expansion) is easy, but suboptimal: may yield exponential blow-up
  lazy unfolding (unfolding on demand) is optimal, consistency in PSpace decidable
Language extensions that require more elaborate techniques include

- **TBoxes with general axioms** $C_i \sqsubseteq D_i$:
  - each node must be labelled with $\neg C_i \sqcup D_i$
  - quantifier depth no longer decreases
  - $\leadsto$ termination not guaranteed

- **Transitive closure of roles**:
  - node labels $(\forall R^*.C)$ yields $C$ in all $R^n$-successor labels
  - quantifier depth no longer decreases
  - $\leadsto$ termination not guaranteed

Use **blocking** (cycle detection) to ensure termination
(but the right blocking to not destroy soundness or completeness)