Agenda

- Basic Idea of the Tableau Calculus
- Propositional Example
- Transformation into Negation Normal Form
- Satisfiability of $ALC$ Concepts
- Correctness and Termination
- Summary
Agenda

- Basic Idea of the Tableau Calculus
- Propositional Example
- Transformation into Negation Normal Form
- Satisfiability of ALC Concepts
- Correctness and Termination
- Summary
Automation

- by now: ad hoc arguments about satisfiability of DL axioms
- a concept is satisfiable, if it has a model
  ~ idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic
Automation

- by now: ad hoc arguments about satisfiability of DL axioms
- a concept is satisfiable, if it has a model
  \(\sim\) idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic

\[(p \lor q) \rightarrow (\neg p \lor \neg q)\]
Automation

- by now: ad hoc arguments about satisfiability of DL axioms
- a concept is satisfiable, if it has a model
  - idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic

\[(p \lor q) \rightarrow (\neg p \lor \neg q)\]

negation in front of complex expressions and non-atomic operators difficult to handle, thus reformulate:
Automation

- by now: ad hoc arguments about satisfiability of DL axioms
- a concept is satisfiable, if it has a model
  ~ idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic

\[(p \lor q) \rightarrow (\neg p \lor \neg q)\]

negation in front of complex expressions and non-atomic operators difficult to handle, thus reformulate:

\[\neg(p \lor q) \lor (\neg p \lor \neg q)\]
Automation

- by now: ad hoc arguments about satisfiability of DL axioms
- a concept is satisfiable, if it has a model
  ~ idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic

\[(p \lor q) \rightarrow (\neg p \lor \neg q)\]

negation in front of complex expressions and non-atomic operators difficult to handle, thus reformulate:

\[\neg(p \lor q) \lor (\neg p \lor \neg q)\]

\[(\neg p \land \neg q) \lor (\neg p \lor \neg q)\]
Automation

- by now: ad hoc arguments about satisfiability of DL axioms
- a concept is satisfiable, if it has a model
  \(\sim\) idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic

\[(p \lor q) \rightarrow (\neg p \lor \neg q)\]

Negation in front of complex expressions and non-atomic operators difficult to handle, thus reformulate:

\[\neg(p \lor q) \lor (\neg p \lor \neg q)\]
\[\neg(p \land \neg q) \lor (\neg p \lor \neg q)\]
\[(\neg p \land \neg q) \lor \neg p \lor \neg q\]
Agenda

- Basic Idea of the Tableau Calculus
- Propositional Example
- Transformation into Negation Normal Form
- Satisfiability of $\mathcal{ALC}$ Concepts
- Correctness and Termination
- Summary
Simple Tableau

\[(p \land \neg q) \lor \neg p \lor \neg q\]
Simple Tableau

\[ (\neg p \land \neg q) \lor \neg p \lor \neg q \]

- disjunctions lead to branches in the tableau
- tableau: finite set of tableau branches
Simple Tableau

\[(\neg p \land \neg q) \lor \neg p \lor \neg q\]

\[\neg p \land \neg q\]
\[\neg p\]
\[\neg q\]

- disjunctions lead to branches in the tableau
- tableau: finite set of tableau branches
Simple Tableau

$(\neg p \land \neg q) \lor \neg p \lor \neg q$

- $\neg p$
- $\neg q$

- disjunctions lead to branches in the tableau
- tableau: finite set of tableau branches
- compare: truth table

<table>
<thead>
<tr>
<th>$I(p)$</th>
<th>$I(q)$</th>
<th>$I(\neg p)$</th>
<th>$I(\neg q)$</th>
<th>$I(p \lor q)$</th>
<th>$I(\neg p \lor \neg q)$</th>
<th>$I((p \lor q) \rightarrow (\neg p \lor \neg q))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>$t$</td>
<td>$f$</td>
<td>$f$</td>
<td>$t$</td>
<td>$f$</td>
<td>$f$</td>
</tr>
<tr>
<td>$t$</td>
<td>$f$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td>$f$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$t$</td>
<td>$t$</td>
<td>$t$</td>
</tr>
</tbody>
</table>

TU Dresden Deduction Systems
Simple Tableau with Contradiction

$$(\neg p \lor q) \land p \land \neg q$$
Simple Tableau with Contradiction

$\neg p \lor q \land p \land \neg q$

$\neg p \lor q$

$p$

$\neg q$

• if a branch contains an atomic contradiction (clash), we call this branch closed
• a tableau is closed, if all its branches are
• a complete tableau without open branches shows the formula's unsatisfiability
Simple Tableau with Contradiction

\[(\neg p \lor q) \land p \land \neg q\]

\[\neg p \lor q\]

\[p\]

\[\neg q\]

\[\neg p\]

\[q\]

• if a branch contains an atomic contradiction (clash), we call this branch closed
• a tableau is closed, if all its branches are
• a complete tableau without open branches shows the formula's unsatisfiability
Simple Tableau with Contradiction

\[(\neg p \lor q) \land p \land \neg q\]

\[
\neg p \lor q
\]

\[
p
\]

\[
\neg q
\]

\[
\neg p ~ q
\]
Simple Tableau with Contradiction

\[(\neg p \lor q) \land p \land \neg q\]

- \(\neg p \lor q\)
  - \(p\)
  - \(\neg q\)
    - \(\neg p\)
    - \(q\)

\(\bot\)

- if a branch contains an atomic contradiction (clash), we call this branch closed
Simple Tableau with Contradiction

\[ (\neg p \lor q) \land p \land \neg q \]

\[ \neg p \lor q \]

\[ p \]

\[ \neg q \]

\[ \neg p \quad q \]

\[ \bot \quad \bot \]

- if a branch contains an atomic contradiction (clash), we call this branch closed
- a tableau is closed, if all its branches are
Simple Tableau with Contradiction

\[(\neg p \lor q) \land p \land \neg q\]
\[\neg p \lor q\]
\[p\]
\[\neg q\]
\[\neg p\]
\[q\]
\[\perp\]
\[\perp\]

- if a branch contains an atomic contradiction (clash), we call this branch closed
- a tableau is closed, if all its branches are
- a complete tableau without open branches shows the formula’s unsatisfiability
Constructing a Model from the Tableau

\[
(\neg p \land \neg q) \lor \neg p \lor \neg q
\]

\[
\neg p \land \neg q
\]

\[
\neg p
\]

\[
\neg q
\]

- given an open branch, we can construct a model
Constructing a Model from the Tableau

\[
(\neg p \land \neg q) \lor \neg p \lor \neg q
\]

- \(\neg p\)
- \(\neg q\)

- given an open branch, we can construct a model
- let \(I(p) = \text{false}\) and let \(I(q) = \text{false}\)
Constructing a Model from the Tableau

$$\neg p \land \neg q \lor \neg p \lor \neg q$$

- given an open branch, we can construct a model
- let I(p)=false and let I(q)=false
- let I(p)=false (I(q) is irrelevant since not in the branch, default assignment false)
Constructing a Model from the Tableau

- given an open branch, we can construct a model
- let $I(p) = \text{false}$ and let $I(q) = \text{false}$
- let $I(p) = \text{false}$ ($I(q)$ is irrelevant since not in the branch, default assignment false)
- let $I(q) = \text{false}$ ($I(p)$ is irrelevant since not in the branch, default assignment false)
Propositional Tableau

- not always exponentially many combinations have to be checked (as opposed to truth table method)
- branches can be built one after the other \(\Rightarrow\) only polynomial space needed
- if we care about satisfiability we can stop after constructing the first complete open branch
Construction with only one Branch in Memory

$(\neg p \lor q) \land p \land q$
Construction with only one Branch in Memory

\[(\neg p \lor q) \land p \land q\]

\[\neg p^{1a} \lor q^{1b}\]

\[p\]

\[q\]

- when encountering a disjunction we assign so-called choice points
- all extensions of the branch based on such a choice are also marked
Construction with only one Branch in Memory

\[(\neg p \lor q) \land p \land q\]

\[\neg p^{1a} \lor q^{1b}\]

\[p\]

\[q\]

\[\neg p^{1a}\]

- when encountering a disjunction we assign so-called choice points
- all extensions of the branch based on such a choice are also marked
Construction with only one Branch in Memory

\[(\neg p \lor q) \land p \land q\]

- when encountering a disjunction we assign so-called choice points
- all extensions of the branch based on such a choice are also marked
- when encountering a contradiction caused by a choice, remove marked formulae and try next choice
Construction with only one Branch in Memory

\[
(\neg p \lor q) \land p \land q \\
\neg p^{1a} \lor q^{1b} \\
p \\
q \\
\neg p^{1a} \\
\neg p^{1a} \\
q^{1b}
\]

- when encountering a disjunction we assign so-called choice points
- all extensions of the branch based on such a choice are also marked
- when encountering a contradiction caused by a choice, remove marked formulae and try next choice
From Propositional Tableau to Tableau for DLs

How can the tableaux be extended for checking satisfiability of $\mathcal{ALC}$ concepts? NB: initially, we assume no underlying knowledge base, thus unsatisfiability means that the concept is contradictory “by itself”.

- tableau represents an element of the domain (plus its “environment”)
From Propositional Tableau to Tableau for DLs

How can the tableaux be extended for checking satisfiability of \(\mathcal{ALC}\) concepts?

NB: initially, we assume no underlying knowledge base, thus unsatisfiability means that the concept is contradictory “by itself”.

- tableau represents an element of the domain (plus its “environment”)
- tableau branch: finite set of propositions of the form \(C(a), r(a, b)\)
- for existential quantifiers, new domain elements are introduced
- universal quantifiers propagate formulae (=concept expressions) to neighboring elements
Agenda

- Basic Idea of the Tableau Calculus
- Propositional Example
- Transformation into Negation Normal Form
- Satisfiability of $\mathcal{ALC}$ Concepts
- Correctness and Termination
- Summary
Propositional Logic – Some Logical Equivalences

- We aim at negations being present only in front of atomic concepts

\[
\begin{align*}
\varphi \land \psi & \equiv \psi \land \varphi \\
\varphi \lor \psi & \equiv \psi \lor \varphi \\
\varphi \land (\psi \land \omega) & \equiv (\varphi \land \psi) \land \omega \\
\varphi \lor (\psi \lor \omega) & \equiv (\varphi \lor \psi) \lor \omega \\
\varphi \land \varphi & \equiv \varphi \\
\varphi \lor \varphi & \equiv \varphi \\
\varphi \land (\psi \lor \varphi) & \equiv \varphi \\
\varphi \lor (\psi \land \varphi) & \equiv \varphi \\
\varphi \lor (\psi \land \omega) & \equiv (\varphi \land \psi) \lor (\varphi \land \omega) \\
\varphi \land (\psi \lor \omega) & \equiv (\varphi \lor \psi) \land (\varphi \lor \omega) \\
\end{align*}
\]
Further Logical Equivalences

\[ \neg(C \cap D) \iff \neg C \cup \neg D \]
\[ \neg(D \cup D) \iff \neg C \cap \neg D \]
\[ \neg\neg C \iff C \]
\[ \neg(\forall r. C) \iff \exists r. (\neg C) \]
\[ \neg(\exists r. C) \iff \forall r. (\neg C) \]
\[ \neg(\leq n s. C) \iff \geq n + 1 s. C \]
\[ \neg(\geq n s. C) \iff \leq n - 1 s. C, \quad n \geq 1 \]
\[ \neg(\geq 0 s. C) \iff \bot \]

- apply these rules iteratively until none can be applied any more
- \( \iff \) equivalent concept in negation normal form

TU Dresden Deduction Systems
recursive definition of an NNF transformation:

if $C$ atomic:

$$\text{NNF}(C) := C$$

$$\text{NNF}(\neg C) := \neg C$$

otherwise:

$$\text{NNF}(\neg\neg C) := \text{NNF}(C)$$

$$\text{NNF}(C \land D) := \text{NNF}(C) \land \text{NNF}(D)$$

$$\text{NNF}(\neg(C \land D)) := \text{NNF}(\neg C) \lor \text{NNF}(\neg D)$$

$$\text{NNF}(C \lor D) := \text{NNF}(C) \lor \text{NNF}(D)$$

$$\text{NNF}(\neg(C \lor D)) := \text{NNF}(\neg C) \land \text{NNF}(\neg D)$$

$$\text{NNF}(\forall r. C) := \forall r. (\text{NNF}(C))$$

$$\text{NNF}(\neg(\forall r. C)) := \exists r. (\text{NNF}(\neg C))$$

$$\text{NNF}(\exists r. C) := \exists r. (\text{NNF}(C))$$

$$\text{NNF}(\neg(\exists r. C)) := \forall r. (\text{NNF}(\neg C))$$

$$\text{NNF}(\leq n \, s. C) := \leq n \, s. (\text{NNF}(C))$$

$$\text{NNF}(\neg(\leq n \, s. C)) := \geq n + 1 \, s. (\text{NNF}(C))$$

$$\text{NNF}(\geq n \, s. C) := \geq n \, s. (\text{NNF}(C))$$

$$\text{NNF}(\neg(\geq n \, s. C)) := \leq n - 1 \, s. (\text{NNF}(C))$$

if $n \geq 1$

$$\text{NNF}(\geq 0 \, s. C) := \top$$

$$\text{NNF}(\neg(\geq 0 \, s. C)) := \bot$$

otherwise
NNF Transformation – Example

\[ \text{NNF} \left( \neg \left( \neg C \sqcap \left( \neg D \sqcup E \right) \right) \right) \]

\[ = \text{NNF} \left( \neg \neg C \right) \sqcup \text{NNF} \left( \neg \left( \neg D \sqcup E \right) \right) \]

\[ = \text{NNF} \left( C \right) \sqcup \text{NNF} \left( \neg \left( \neg D \sqcup E \right) \right) \]

\[ = C \sqcup \text{NNF} \left( \neg \left( \neg D \sqcup E \right) \right) \]

\[ = C \sqcup \left( \text{NNF} \left( \neg \neg D \right) \sqcap \text{NNF} \left( \neg E \right) \right) \]

\[ = C \sqcup \left( \text{NNF} \left( D \right) \sqcap \text{NNF} \left( \neg E \right) \right) \]

\[ = C \sqcup \left( D \sqcap \text{NNF} \left( \neg E \right) \right) \]

\[ = C \sqcup \left( D \sqcap \neg E \right) \]
Agenda

• Basic Idea of the Tableau Calculus
• Propositional Example
• Transformation into Negation Normal Form
• Satisfiability of $\mathcal{ALC}$ Concepts
• Correctness and Termination
• Summary
Tableau for $\mathcal{ALC}$ Concepts

- tableau for a propositional formula $\alpha$: one element, labeled with subformulae of $\alpha$
- tableau for an $\mathcal{ALC}$ concept $C$: graph (more precisely: tree) where the nodes are labeled with subformulae of $C$
- root labeled with $C$
- represents model for $C$ (if complete and clash-free)
- non-root nodes are enforced by existential quantifiers

Definition

Let $C$ be an $\mathcal{ALC}$ concept, $\text{SF}(C)$ the set of all subformulae of $C$ and $\text{Rol}(C)$ the set of all roles occurring in $C$. A tableau for $C$ is a tree $G = \langle V, E, L \rangle$, with nodes $V$, edges $E \subseteq V \times V$ and a labeling function $L$ with $L: V \rightarrow 2^{\text{SF}(C)}$ and $L: V \times V \rightarrow 2^{\text{Rol}(C)}$. 
Properties of the $\mathcal{ALC}$ Tableau Algorithm

- the algorithm is specified as a set of rules
- every rule breaks down a complex concept into its parts
- rules applicable in any order
- the algorithm is non-deterministic (due to disjunction)
- check for atomic contradictions

Tableau algorithm for checking satisfiability of $\mathcal{ALC}$ concepts

**Input:** an $\mathcal{ALC}$ concept in NNF

**Output:**
- true if there is a clash-free tableau where no more rules can be applied
- false otherwise (tableau closed)
Tableau Rules for $\mathcal{ALC}$ Concepts

$\sqcap$-rule: For an arbitrary $v \in V$ with $C \sqcap D \in L(v)$ and $
\{C, D\} \not\subseteq L(v)$, let $L(v) := L(v) \cup \{C, D\}$.

$\sqcup$-rule: For an arbitrary $v \in V$ with $C \sqcup D \in L(v)$ and
$\{C, D\} \cap L(v) = \emptyset$, choose $X \in \{C, D\}$ and let
$L(v) := L(v) \cup \{X\}$.

$\exists$-rule: For an arbitrary $v \in V$ with $\exists r.C \in L(v)$ such that
there is no $r$-successor $v'$ of $v$ with $C \in L(v')$,
let $V = V \cup \{v'\}$, $E = E \cup \{(v, v')\}$, $L(v') := \{C\}$ and
$L(v, v') := \{r\}$ for $v'$ a new node.

$\forall$-rule: For arbitrary $v, v' \in V$, $v'$ $r$-neighbor of $v$,
$\forall r.C \in L(v)$ and $C \notin L(v')$, let $L(v') := L(v') \cup \{C\}$.

- a node $v'$ is an $r$-neighbor of a node $v$ if $(v, v') \in E$ and $r \in L(v, v')$
Tableau Rules for $\mathcal{ALC}$ Concepts

$\Box$-rule: For an arbitrary $v \in V$ with $C \cap D \in L(v)$ and $
\{C, D\} \nsubseteq L(v)$, let $L(v) := L(v) \cup \{C, D\}$.

$\square$-rule: For an arbitrary $v \in V$ with $C \sqcup D \in L(v)$ and $\{C, D\} \cap L(v) = \emptyset$, choose $X \in \{C, D\}$ and let $L(v) := L(v) \cup \{X\}$.

$\exists$-rule: For an arbitrary $v \in V$ with $\exists r.C \in L(v)$ such that there is no $r$-successor $v'$ of $v$ with $C \in L(v')$, let $V = V \cup \{v'\}$, $E = E \cup \{(v, v')\}$, $L(v') := \{C\}$ and $L(v, v') := \{r\}$ for $v'$ a new node.

$\forall$-rule: For arbitrary $v, v' \in V$, $v'$ $r$-neighbor of $v$,
$\forall r.C \in L(v)$ and $C \not\in L(v')$, let $L(v') := L(v') \cup \{C\}$.

- a node $v'$ is an $r$-neighbor of a node $v$ if $\langle v, v' \rangle \in E$ and $r \in L(v, v')$
- rule application order: “don’t care” non-determinism
Tableau Rules for $\mathcal{ALC}$ Concepts

\-rule: For an arbitrary $v \in V$ mit $C \cap D \in L(v)$ and
\{C, D\} \not\subseteq L(v)$, let $L(v) := L(v) \cup \{C, D\}$.

\-rule: For an arbitrary $v \in V$ with $C \cup D \in L(v)$ and
\{C, D\} \cap L(v) = \emptyset$, choose $X \in \{C, D\}$ and let
$L(v) := L(v) \cup \{X\}$.

\-rule: For an arbitrary $v \in V$ with $\exists r.C \in L(v)$ such that
there is no $r$-successor $v'$ of $v$ with $C \in L(v')$,
let $V = V \cup \{v'\}$, $E = E \cup \{(v, v')\}$, $L(v') := \{C\}$ and
$L(v, v') := \{r\}$ for $v'$ a new node.

\-rule: For arbitrary $v, v' \in V$, $v'$ $r$-neighbor of $v$,
\forall r.C \in L(v)$ and $C \not\in L(v')$, let $L(v') := L(v') \cup \{C\}$.

- a node $v'$ is an $r$-neighbor of a node $v$ if $\langle v, v' \rangle \in E$ and $r \in L(v, v')$
- rule application order: “don’t care” non-determinism
- choice of disjunction: “don’t know” non-determinism
Tableau Algorithmus Example

\[ C = \exists r. (A \cup \exists r. B) \sqcap \exists r. \neg A \sqcap \forall r. (\neg A \sqcap \forall r. (\neg B \cup A)) \]

\[ L(u) = \{ C \} \]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \sqcap \exists r. \neg A \sqcap \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \]

\[ u \]

\[ L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \} \]
Tableau Algorithmus Example

\[ C = \exists r.(A \cup \exists r.B) \cap \exists r.\neg A \cap \forall r.((\neg A \cap \forall r.(\neg B \cup A))) \]

\[
L(u) = \{ C, \exists r.(A \cup \exists r.B), \\
\exists r.\neg A, \forall r.((\neg A \cap \forall r.(\neg B \cup A))) \}
\]

\[
L(v) = \{ A \cup \exists r.B \}
\]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \cap \exists r. \neg A \cap \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \]

\[ L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \} \]

\[ L(v) = \{ A \sqcup \exists r. B \} \]

\[ L(w) = \{ \neg A \} \]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \land \exists r. \neg A \land \forall r. (\neg A \land \forall r. (\neg B \sqcup A)) \]

\[ L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \land \forall r. (\neg B \sqcup A)) \} \]

\[ L(v) = \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A) \} \]

\[ L(w) = \{ \neg A \} \]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \sqcap \exists r. \neg A \sqcap \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \]

\[
L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \\
\exists r. \neg A, \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \}
\]

\[
L(v) = \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A) \}
\]

\[
L(w) = \{ \neg A, \forall r. (\neg B \sqcup A) \}
\]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \cap \exists r. \neg A \cap \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \]

\[
\begin{align*}
L(u) &= \{ C, \exists r. (A \sqcup \exists r. B), \\
&\quad \exists r. \neg A, \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \} \\
L(v) &= \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A), A \} \\
L(w) &= \{ \neg A, \forall r. (\neg B \sqcup A) \}
\end{align*}
\]
Tableau Algorithmus Example

\[
C = \exists r. (A \sqcup \exists r. B) \sqcap \exists r. \neg A \sqcap \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A))
\]

\[
L(u) = \{ C, \exists r. (A \sqcup \exists r. B),
\exists r. \neg A, \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \}
\]

\[
L(v) = \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A), \}
\]

\[
L(w) = \{ \neg A, \forall r. (\neg B \sqcup A) \}
\]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \cap \exists r. \neg A \cap \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \]

\[
L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \}
\]

\[
L(v) = \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A), \square, \exists r. B \}
\]

\[
L(w) = \{ \neg A, \forall r. (\neg B \sqcup A) \}
\]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \cap \exists r. \neg A \cap \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \]

\[ L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \cap \forall r. (\neg B \sqcup A)) \} \]

\[ L(v) = \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A), \blacksquare, \exists r. B \} \]

\[ L(w) = \{ \neg A, \forall r. (\neg B \sqcup A) \} \]

\[ L(x) = \{ B \} \]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \sqcap \exists r. \neg A \sqcap \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \]

\[ L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \} \]

\[ L(v) = \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A), X, \exists r. B \} \]

\[ L(w) = \{ \neg A, \forall r. (\neg B \sqcup A) \} \]

\[ L(x) = \{ B, \neg B \sqcup A \} \]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \sqcap \exists r. \neg A \sqcap \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \]

\[
\begin{align*}
L(u) &= \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \} \\
L(v) &= \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A) \} \\
L(w) &= \{ \neg A, \forall r. (\neg B \sqcup A) \} \\
L(x) &= \{ B, \neg B \sqcup A, \neg B \}
\end{align*}
\]
Tableau Algorithmus Example

\[ C = \exists r. (A \sqcup \exists r. B) \sqcap \exists r. \neg A \sqcap \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \]

\[ L(u) = \{ C, \exists r. (A \sqcup \exists r. B), \exists r. \neg A, \forall r. (\neg A \sqcap \forall r. (\neg B \sqcup A)) \} \]

\[ L(v) = \{ A \sqcup \exists r. B, \neg A, \forall r. (\neg B \sqcup A), \xmark, \exists r. B \} \]

\[ L(w) = \{ \neg A, \forall r. (\neg B \sqcup A) \} \]

\[ L(x) = \{ B, \neg B \sqcup A, \xmark \} \]
Tableau Algorithmus Example

\[ C = \exists r. (A \cup \exists r. B) \cap \exists r. \neg A \cap \forall r. (\neg A \cap \forall r. (\neg B \cup A)) \]

\[ L(u) = \{ C, \exists r. (A \cup \exists r. B), \exists r. \neg A, \forall r. (\neg A \cap \forall r. (\neg B \cup A)) \} \]

\[ L(v) = \{ A \cup \exists r. B, \neg A, \forall r. (\neg B \cup A), \times, \exists r. B \} \]

\[ L(w) = \{ \neg A, \forall r. (\neg B \cup A) \} \]

\[ L(x) = \{ B, \neg B \cup A, \times, B, A \} \]
Tableau Algorithm Example

the model $\mathcal{I}$ constructed by the algorithm is the following:

$\Delta^\mathcal{I} = \{ u, v, w, x \}$

$A^\mathcal{I} = \{ x \}$

$B^\mathcal{I} = \{ x \}$

$r^\mathcal{I} = \{ \langle u, v \rangle, \langle u, w \rangle, \langle v, x \rangle \}$

Check that indeed $C^\mathcal{I} = \{ u \}$, given the defined semantics of $\mathcal{ALC}$
Tableau Algorithm Properties

1. the model is **finite**: only finitely many elements in the domain
2. the model is **tree-shaped**: the tableau is a labeled tree

The algorithm will always construct finite trees

- from a clash-free tableau, we can construct a finite model
- if there is no clash-free tableau, there is no model
Agenda

- Basic Idea of the Tableau Calculus
- Propositional Example
- Transformation into Negation Normal Form
- Satisfiability of $\mathcal{ALC}$ Concepts
- Correctness and Termination
- Summary
Tableau Properties

- the depth (number of nested quantifiers) decreases in every node
- every node is labeled only with subformulae of $C$
- $C$ has only polynomially many subformulae
- if the output is true we can build a model out of the constructed tableau
- on the other hand, we can use a model of a satisfiable concept to construct a clash-free tableau for this concept
Tableau Algorithm for $\mathcal{ALC}$ Concepts

**Theorem**

1. The algorithm terminates for every input.
2. If the output is $true$, then the input concept is satisfiable.
3. If the input concept is satisfiable, then the output is $true$. 

**Corollary**

Every $\mathcal{ALC}$ concept $C$ has the following properties:

1. Finite model property: If $C$ has a model, then it has a finite one.
2. Tree model property: If $C$ has a model, then it has a tree-shaped one.
Tableau Algorithm for $\mathcal{ALC}$ Concepts

**Theorem**
1. the algorithm terminates for every input
2. if the output is $true$, then the input concept is satisfiable
3. if the input concept is satisfiable, then the output is $true$.

**Corollary**
Every $\mathcal{ALC}$ concept $C$ has the following properties:
1. finite model property: If $C$ has a model, then it has a finite one.
2. tree model property If $C$ has a model, then it has a tree-shaped one.
Agenda

- Basic Idea of the Tableau Calculus
- Propositional Example
- Transformation into Negation Normal Form
- Satisfiability of $\mathcal{ALC}$ Concepts
- Correctness and Termination
- Summary
Summary

- we now have a constructive method for building model abstractions
- satisfiable $\mathcal{ALC}$ concepts always have a finite model that we can construct
- the algorithm is correct, complete and terminating
- serves as basis for practically implemented algorithms
- next: extension to knowledge bases