TECHNISCHE UNIVERSITÄT DRESDEN

# DEDUCTION SYSTEMS

## Tableau Procedures I

**Markus Krötzsch**

**Chair for Knowledge-Based Systems**

**Slides by Sebastian Rudolph**

TU Dresden, 29 April 2018

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

# Agenda

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

# Agenda

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

# Automation

- by now: ad hoc arguments about satisfiability of DL axioms
- wanted: generic algorithm
- considered reasoning task: concept satisfiability
- a concept is satisfiable, if it has a model
  $\rightsquigarrow$ idea: constructive decision procedure that tries to build models

# 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

We want to check satisfiability of the following propositional formula:

$$(p \vee q) \to (\neg p \vee \neg q)$$

# Simple Tableau

We want to check satisfiability of the following propositional formula:

$$(p \vee q) \to (\neg p \vee \neg q)$$

Negation in front of complex expressions and operators like $\to$ and $\leftrightarrow$ difficult to handle, thus reformulate:

# Simple Tableau

We want to check satisfiability of the following propositional formula:

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

Negation in front of complex expressions and operators like $\to$ and $\leftrightarrow$ difficult to handle, thus reformulate:

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

# Simple Tableau

We want to check satisfiability of the following propositional formula:

$$(p \vee q) \to (\neg p \vee \neg q)$$

Negation in front of complex expressions and operators like $\to$ and $\leftrightarrow$ difficult to handle, thus reformulate:

$$\neg(p \vee q) \vee (\neg p \vee \neg q)$$
$$(\neg p \wedge \neg q) \vee (\neg p \vee \neg q)$$

# Simple Tableau

We want to check satisfiability of the following propositional formula:

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

Negation in front of complex expressions and operators like $\to$ and $\leftrightarrow$ difficult to handle, thus reformulate:

$$\begin{aligned}
\neg(p \lor q) \lor (\neg p \lor \neg q) \\
(\neg p \land \neg q) \lor (\neg p \lor \neg q) \\
(\neg p \land \neg q) \lor \neg p \lor \neg q
\end{aligned}$$

# Simple Tableau

$$(\neg p \wedge \neg q) \vee \neg p \vee \neg q$$

# Simple Tableau

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

$$\neg p \land \neg q \qquad\qquad \neg p \qquad\qquad \neg q$$

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

# Simple Tableau

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

| $\neg p \land \neg q$ | $\neg p$ | $\neg q$ |
|---|---|---|
| $\neg p$ | | |
| $\neg q$ | | |

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

# Simple Tableau

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

$\neg p \land \neg q$ | $\neg p$ | $\neg q$

$\neg p$

$\neg q$

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

| $I(p)$ | $I(q)$ | $I(\neg p)$ | $I(\neg q)$ | $I(p \lor q)$ | $I(\neg p \lor \neg q)$ | $I((p \lor q) \to (\neg p \lor \neg q))$ |
|---|---|---|---|---|---|---|
| $t$ | $t$ | $f$ | $f$ | $t$ | $f$ | $f$ |
| $t$ | $f$ | $f$ | $t$ | $t$ | $t$ | $t$ |
| $f$ | $t$ | $t$ | $f$ | $t$ | $t$ | $t$ |
| $f$ | $f$ | $t$ | $t$ | $f$ | $t$ | $t$ |

# Simple Tableau with Contradiction

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

# Simple Tableau with Contradiction

$(\neg p \vee q) \wedge p \wedge \neg q$

$\neg p \vee q$

$p$

$\neg q$

# Simple Tableau with Contradiction

$$(\neg p \vee q) \wedge p \wedge \neg q$$

$$\neg p \vee q$$

$$p$$

$$\neg q$$

$$\neg p \qquad\qquad q$$

# Simple Tableau with Contradiction

$$(\neg p \vee q) \wedge p \wedge \neg q$$

$$\neg p \vee q$$

$$p$$

$$\neg q$$

$$\neg p \qquad\qquad\qquad q$$

# Simple Tableau with Contradiction

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

$$\neg p \lor q$$

$$p$$

$$\neg q$$

$$\neg p \qquad\qquad q$$

$$\bot$$

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

# Simple Tableau with Contradiction

$$(\neg p \vee q) \wedge p \wedge \neg q$$

$$\neg p \vee q$$

$$p$$

$$\neg q$$

$$\begin{array}{ccc} & \neg q & \\ \swarrow & & \searrow \\ \neg p & & q \\ \bot & & \bot \end{array}$$

- 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

$(\neg p \vee q) \wedge p \wedge \neg q$

$\neg p \vee q$

$p$

$\neg q$

$\neg p$ $\qquad\qquad$ $q$

$\bot$ $\qquad\qquad$ $\bot$

- 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

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

$\neg p \land \neg q$ $\qquad\qquad$ $\neg p$ $\qquad\qquad$ $\neg q$

$\neg p$

$\neg q$

- Given an open branch, we can construct a model:

# Constructing a Model from the Tableau

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

$$\neg p \land \neg q \qquad\qquad \neg p \qquad\qquad \neg q$$

$$\neg p$$

$$\neg q$$

- Given an open branch, we can construct a model:
- Left branch: let I(p)=false and let I(q)=false

# Constructing a Model from the Tableau

$$
\begin{array}{ccc}
& (\neg p \land \neg q) \lor \neg p \lor \neg q & \\
\neg p \land \neg q & \neg p & \neg q \\
\neg p & & \\
\neg q & &
\end{array}
$$

- Given an open branch, we can construct a model:
- Left branch: let I(p)=false and let I(q)=false
- Middle branch: let I(p)=false (I(q) is irrelevant since not in the branch, default assignment false)

# Constructing a Model from the Tableau

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

$\neg p \land \neg q$ $\qquad$ $\neg p$ $\qquad$ $\neg q$

$\neg p$

$\neg q$

- Given an open branch, we can construct a model:
- Left branch: let I(p)=false and let I(q)=false
- Middle branch: let I(p)=false (I(q) is irrelevant since not in the branch, default assignment false)
- Right branch: let I(q)=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 $\rightsquigarrow$ 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

$$(\neg p \vee q) \wedge p \wedge q$$

# Construction with only one Branch in Memory

$$(\neg p \vee q) \wedge p \wedge q$$

$$\neg p^{1a} \vee q^{1b}$$

$$p$$

$$q$$

- 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

$$(\neg p \vee q) \wedge p \wedge q$$

$$\neg p^{1a} \vee q^{1b}$$

$$p$$

$$q$$

$$\neg p^{1a}$$

- 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

$$(\neg p \vee q) \wedge p \wedge q$$

$$\neg p^{1a} \vee q^{1b}$$

$$p$$

$$q$$

$$\neg p^{1a}$$

$$\perp^{1a}$$

- 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

$$(\neg p \vee q) \wedge p \wedge q$$

$$\neg p^{1a} \vee q^{1b}$$

$$p$$

$$q$$

$$\cancel{\neg p^{1a}}$$

$$\cancel{\perp^{1a}}$$

$$q^{1b}$$

- 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

TECHNISCHE UNIVERSITÄT DRESDEN

# NNF Transformation

recursive definition of an NNF transformation:

if $C$ atomic:

$$NNF(C) := C \qquad NNF(\neg C) := \neg C$$

otherwise:

$$
\begin{array}{rlrl}
NNF(\neg\neg C) & := NNF(C) & & \\
NNF(C \sqcap D) & := NNF(C) \sqcap NNF(D) & NNF(\neg(C \sqcap D)) & := NNF(\neg C) \sqcup NNF(\neg D) \\
NNF(C \sqcup D) & := NNF(C) \sqcup NNF(D) & NNF(\neg(C \sqcup D)) & := NNF(\neg C) \sqcap NNF(\neg D) \\
NNF(\forall r.C) & := \forall r.(NNF(C)) & NNF(\neg(\forall r.C)) & := \exists r.(NNF(\neg C)) \\
NNF(\exists r.C) & := \exists r.(NNF(C)) & NNF(\neg(\exists r.C)) & := \forall r.(NNF(\neg C)) \\
NNF(\leqslant n\, s.C) & := \leqslant n\, s.(NNF(C)) & NNF(\neg(\leqslant n\, s.C)) & := \geqslant n+1\, s.(NNF(C)) \\
NNF(\geqslant n\, s.C) & := \geqslant n\, s.(NNF(C)) & NNF(\neg(\geqslant n\, s.C)) & := \leqslant n-1\, s.(NNF(C)) \\
 & & & \quad \text{if } n \geq 1 \\
NNF(\geqslant 0\, s.C) & := \top & NNF(\neg(\geqslant 0\, s.C)) & := \bot \qquad \text{otherwise}
\end{array}
$$

# NNF Transformation – Example

$$
\begin{aligned}
& \mathit{NNF}(\neg(\neg C \sqcap (\neg D \sqcup E))) \\
= {} & \mathit{NNF}(\neg\neg C) \sqcup \mathit{NNF}(\neg(\neg D \sqcup E)) \\
= {} & \mathit{NNF}(C) \sqcup \mathit{NNF}(\neg(\neg D \sqcup E)) \\
= {} & C \sqcup \mathit{NNF}(\neg(\neg D \sqcup E)) \\
= {} & C \sqcup (\mathit{NNF}(\neg\neg D) \sqcap \mathit{NNF}(\neg E)) \\
= {} & C \sqcup (\mathit{NNF}(D) \sqcap \mathit{NNF}(\neg E)) \\
= {} & C \sqcup (D \sqcap \mathit{NNF}(\neg E)) \\
= {} & C \sqcup (D \sqcap \neg E)
\end{aligned}
$$

# 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 node 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, $\mathsf{SF}(C)$ the set of all subformulae of $C$ and $\mathsf{Rol}(C)$ 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 $L \colon V \to 2^{\mathsf{SF}(C)}$ and $L \colon V \times V \to 2^{\mathsf{Rol}(C)}$.

# 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 $v \in V$ mit $C \sqcap D \in L(v)$ 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 $V = V \cup \{v'\}$, $E = E \cup \{\langle v, v' \rangle\}$, $L(v') := \{C\}$ and $L(v, v') := \{r\}$ for $v'$ a new node.

$\forall$-rule: For arbitrary $v, v' \in V$, $v'$ $r$-successor 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 $r$-successor of a node $v$ if $\langle v, v' \rangle \in E$ and $r \in L(v, v')$

# Tableau Rules for $\mathcal{ALC}$ Concepts

$\sqcap$-rule: For an arbitrary $v \in V$ mit $C \sqcap D \in L(v)$ 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 $V = V \cup \{v'\}$, $E = E \cup \{\langle v, v' \rangle\}$, $L(v') := \{C\}$ and $L(v, v') := \{r\}$ for $v'$ a new node.

$\forall$-rule: For arbitrary $v, v' \in V$, $v'$ $r$-successor 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 $r$-successor 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

$\sqcap$-rule: For an arbitrary $v \in V$ mit $C \sqcap D \in L(v)$ 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 $V = V \cup \{v'\}$, $E = E \cup \{\langle v, v' \rangle\}$, $L(v') := \{C\}$ and $L(v, v') := \{r\}$ for $v'$ a new node.

$\forall$-rule: For arbitrary $v, v' \in V$, $v'$ $r$-successor 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 $r$-successor 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 Algorithm Example

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

$u$

$$L(u) = \{C\}$$

# Tableau Algorithm Example

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

$u$

$$\begin{aligned} L(u) = \{&C, \exists r.(A \sqcup \exists r.B), \\ &\exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \end{aligned}$$

# Tableau Algorithm Example

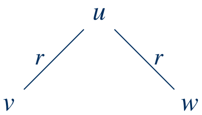
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$

$$\begin{tikzcd} & u \arrow[dl, dash, "r"'] \\ v & \end{tikzcd}$$

$$\begin{aligned} L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\ &\quad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\ L(v) &= \{A \sqcup \exists r.B\} \end{aligned}$$

# Tableau Algorithm Example

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




$$\begin{aligned} L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\ &\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\ L(v) &= \{A \sqcup \exists r.B\} \\ L(w) &= \{\neg A\} \end{aligned}$$

# Tableau Algorithm Example

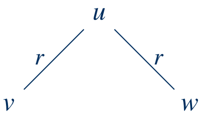
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$

$$\begin{tikzcd} & u \arrow[dl, dash, "r"'] \arrow[dr, dash, "r"] & \\ v & & w \end{tikzcd}$$

$$\begin{aligned} L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\ &\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\ L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A)\} \\ L(w) &= \{\neg A\} \end{aligned}$$

# Tableau Algorithm Example

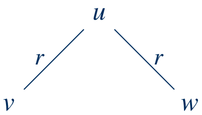
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$




$$\begin{aligned} L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\ &\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\ L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A)\} \\ L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\} \end{aligned}$$

# Tableau Algorithm Example

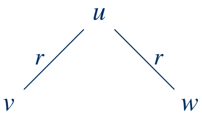
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$




$$\begin{aligned}
L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\
&\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\
L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), A\} \\
L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\}
\end{aligned}$$

# Tableau Algorithm Example

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




$$\begin{aligned} L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\ &\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\ L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{A}\} \\ L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\} \end{aligned}$$

# Tableau Algorithm Example

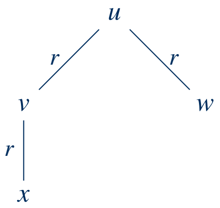
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$

$$\begin{tikzcd} & u \arrow[dl, dash, "r"'] \arrow[dr, dash, "r"] & \\ v & & w \end{tikzcd}$$

$$\begin{aligned} L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\ &\quad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\ L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{A}, \exists r.B\} \\ L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\} \end{aligned}$$

# Tableau Algorithm Example

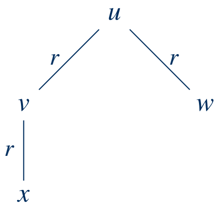
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$




$$\begin{aligned}
L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\
&\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\
L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{A}, \exists r.B\} \\
L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\} \\
L(x) &= \{B\}
\end{aligned}$$

# Tableau Algorithm Example

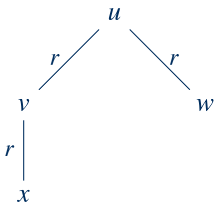
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$




$$\begin{aligned}
L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\
&\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\
L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{A}, \exists r.B\} \\
L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\} \\
L(x) &= \{B, \neg B \sqcup A\}
\end{aligned}$$

# Tableau Algorithm Example

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




$$\begin{aligned}
L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\
&\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\
L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{A}, \exists r.B\} \\
L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\} \\
L(x) &= \{B, \neg B \sqcup A, \neg B\}
\end{aligned}$$

# Tableau Algorithm Example

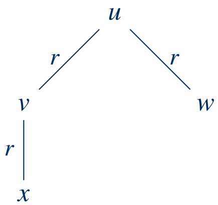
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$

$$\begin{tikzcd} & u \arrow[dl, dash, "r"'] \arrow[dr, dash, "r"] & \\ v \arrow[d, dash, "r"'] & & w \\ x & & \end{tikzcd}$$

$$\begin{aligned} L(u) &= \{C, \exists r.(A \sqcup \exists r.B), \\ &\qquad \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\} \\ L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{A}, \exists r.B\} \\ L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\} \\ L(x) &= \{B, \neg B \sqcup A, \cancel{\neg B}\} \end{aligned}$$

# Tableau Algorithm Example

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




$$\begin{aligned}
L(u) &= \{C, \exists r.(A \sqcup \exists r.B),\\
&\quad\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}\\
L(v) &= \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{A}, \exists r.B\}\\
L(w) &= \{\neg A, \forall r.(\neg B \sqcup A)\}\\
L(x) &= \{B, \neg B \sqcup A, \cancel{\neg B}, A\}
\end{aligned}$$

# Tableau Algorithm Example

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

$$\begin{aligned}
\Delta^{\mathcal{I}} &= \{u, v, w, x\}, \\
A^{\mathcal{I}} &= \{x\}, \\
B^{\mathcal{I}} &= \{x\}, \\
r^{\mathcal{I}} &= \{\langle u, v\rangle, \langle u, w\rangle, \langle v, x\rangle\}.
\end{aligned}$$

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

# Tableau Algorithm Properties

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

If the algorithm is successful, the constructed clash-free complete tableau will give rise to a model with the following 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.

# 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 ouput 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 $\mathtt{true}$, then the input concept is satisfiable
3. if the input concept is satisfiable, then the output is $\mathtt{true}$.

# 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 tree model that we can construct
- the algorithm is correct, complete and terminating
- serves as basis for practically implemented algorithms
- next: extension to knowledge bases