



DEDUCTION SYSTEMS

Tableau Procedures I

Sebastian Rudolph

Dresden, 24 April 2014

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
- a concept is satisfiable, if it has a model
 \rightsquigarrow 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
↔ idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic

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

Automation

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

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

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
 \rightsquigarrow idea: constructive decision procedure that tries to build models
- analog: truth tables in propositional logic

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

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

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

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

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

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

$$\begin{aligned} & \neg(p \vee q) \vee (\neg p \vee \neg q) \\ & (\neg p \wedge \neg q) \vee (\neg p \vee \neg q) \end{aligned}$$

Automation

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

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

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

$$\begin{aligned} & \neg(p \vee q) \vee (\neg p \vee \neg q) \\ & (\neg p \wedge \neg q) \vee (\neg p \vee \neg q) \\ & (\neg p \wedge \neg q) \vee \neg p \vee \neg q \end{aligned}$$

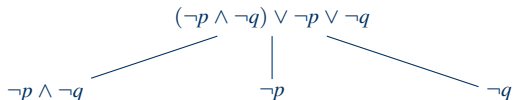
Agenda

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

Simple Tableau

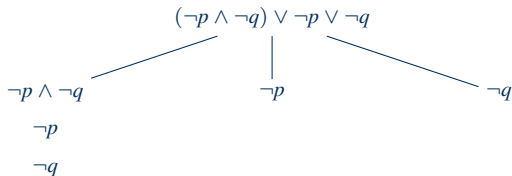
$$(\neg p \wedge \neg q) \vee \neg p \vee \neg q$$

Simple Tableau



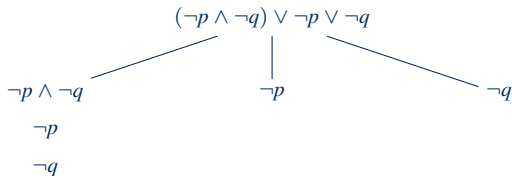
- disjunctions lead to branches in the tableau
- tableau: finite set of tableau branches

Simple Tableau



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

Simple Tableau



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

$I(p)$	$I(q)$	$I(\neg p)$	$I(\neg q)$	$I(p \vee q)$	$I(\neg p \vee \neg q)$	$I((p \vee q) \rightarrow (\neg p \vee \neg q))$
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

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

Simple Tableau with Contradiction

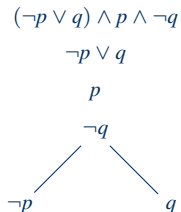
$$(\neg p \vee q) \wedge p \wedge \neg q$$

$$\neg p \vee q$$

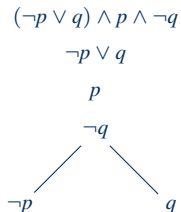
$$p$$

$$\neg q$$

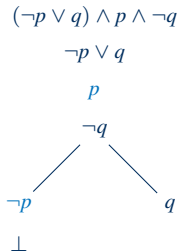
Simple Tableau with Contradiction



Simple Tableau with Contradiction

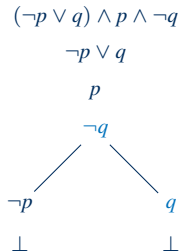


Simple Tableau with Contradiction



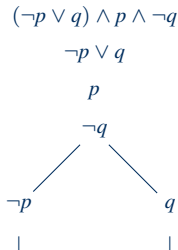
- if a branch contains an atomic contradiction (clash), we call this branch closed

Simple Tableau with Contradiction



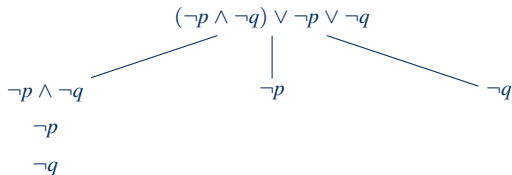
- if a branch contains an atomic contradiction (clash), we call this branch closed
- a tableau is closed, if all its branches are

Simple Tableau with Contradiction



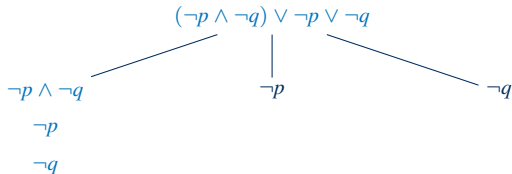
- if a branch contains an atomic contradiction (clash), we call this branch closed
- a tableau is closed, if all its branches are
- a complete tableau without open branches shows the formula's unsatisfiability

Constructing a Model from the Tableau



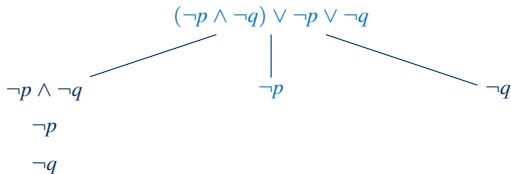
- given an open branch, we can construct a model

Constructing a Model from the Tableau



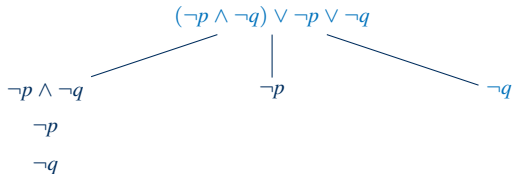
- given an open branch, we can construct a model
- let $I(p)=\text{false}$ and let $I(q)=\text{false}$

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)

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 \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

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

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

$$p$$

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

$$q^{1b}$$

- 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 ALC Concepts
- Correctness and Termination
- Summary

Propositional Logic – Some Logical Equivalences

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

$$\varphi \wedge \psi \equiv \psi \wedge \varphi$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \vee \psi \equiv \psi \vee \varphi$$

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

$$\varphi \wedge (\psi \wedge \omega) \equiv (\varphi \wedge \psi) \wedge \omega$$

$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$$

$$\varphi \vee (\psi \vee \omega) \equiv (\varphi \vee \psi) \vee \omega$$

$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$$

$$\varphi \wedge \varphi \equiv \varphi$$

$$\neg\neg\varphi \equiv \varphi$$

$$\varphi \vee \varphi \equiv \varphi$$

$$\varphi \wedge (\psi \vee \varphi) \equiv \varphi$$

$$\varphi \vee (\psi \wedge \omega) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \omega)$$

$$\varphi \vee (\psi \wedge \varphi) \equiv \varphi$$

$$\varphi \wedge (\psi \vee \omega) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \omega)$$

Further Logical Equivalences

$$\neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D$$

$$\neg(D \sqcup D) \rightsquigarrow \neg C \sqcap \neg D$$

$$\neg\neg C \rightsquigarrow C$$

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

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

$$\neg(\leq n s. C) \rightsquigarrow \geq n + 1 s. C$$

$$\neg(\geq n s. C) \rightsquigarrow \leq n - 1 s. C, \quad n \geq 1$$

$$\neg(\geq 0 s. C) \rightsquigarrow \perp$$

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

NNF Transformation

recursive definition of an NNF transformation:

if C atomic:

$$NNF(C) := C$$

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

otherwise:

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

$$NNF(C \sqcap D) := NNF(C) \sqcap NNF(D) \quad NNF(\neg(C \sqcap D)) := NNF(\neg C) \sqcup NNF(\neg D)$$

$$NNF(C \sqcup D) := NNF(C) \sqcup NNF(D) \quad NNF(\neg(C \sqcup D)) := NNF(\neg C) \sqcap NNF(\neg D)$$

$$NNF(\forall r.C) := \forall r.(NNF(C)) \quad NNF(\neg(\forall r.C)) := \exists r.(NNF(\neg C))$$

$$NNF(\exists r.C) := \exists r.(NNF(C)) \quad NNF(\neg(\exists r.C)) := \forall r.(NNF(\neg C))$$

$$NNF(\leq n s.C) := \leq n s.(NNF(C)) \quad NNF(\neg(\leq n s.C)) := \geq n + 1 s.(NNF(C))$$

$$NNF(\geq n s.C) := \geq n s.(NNF(C)) \quad NNF(\neg(\geq n s.C)) := \leq n - 1 s.(NNF(C))$$

if $n \geq 1$

$$NNF(\geq 0 s.C) := \top$$

$$NNF(\neg(\geq 0 s.C)) := \perp \quad \text{otherwise}$$

NNF Transformation – Example

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

Agenda

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

Tableau for \mathcal{ALC} Concepts

- tableau for a propositional formula α : one element, labeled with subformulae of α
- 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, $SF(C)$ the set of all subformulae of C and $Roll(C)$ 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 $L: V \rightarrow 2^{SF(C)}$ and $L: V \times V \rightarrow 2^{Roll(C)}$.

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 -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 $\langle v, v'\rangle \in E$ 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 -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 $\langle v, v'\rangle \in E$ 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 -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 $\langle v, v' \rangle \in E$ 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

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

u

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

Tableau Algorithmus Example

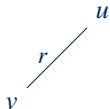
$$C = \exists r.(A \sqcup \exists r.B) \sqcap \exists r.\neg A \sqcap \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))$$

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

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

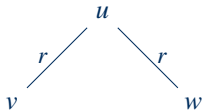


$$L(u) = \{C, \exists r.(A \sqcup \exists r.B), \\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}$$

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

Tableau Algorithmus Example

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



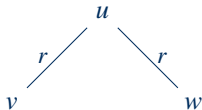
$$L(u) = \{C, \exists r.(A \sqcup \exists r.B), \\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}$$

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

$$L(w) = \{\neg A\}$$

Tableau Algorithmus Example

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



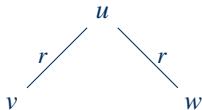
$$L(u) = \{C, \exists r.(A \sqcup \exists r.B), \\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}$$

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

$$L(w) = \{\neg A\}$$

Tableau Algorithmus Example

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



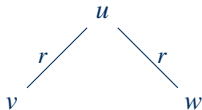
$$L(u) = \{C, \exists r.(A \sqcup \exists r.B), \\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}$$

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

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

Tableau Algorithmus Example

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



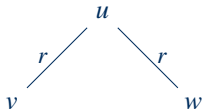
$$L(u) = \{C, \exists r.(A \sqcup \exists r.B), \\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}$$

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

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

Tableau Algorithmus Example

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



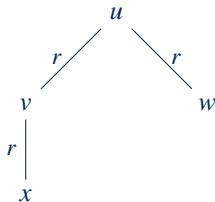
$$L(u) = \{C, \exists r.(A \sqcup \exists r.B), \\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}$$

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

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

Tableau Algorithmus Example

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



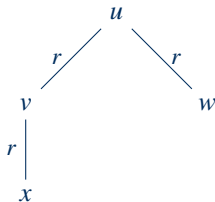
$$L(u) = \{C, \exists r.(A \sqcup \exists r.B), \\ \exists r.\neg A, \forall r.(\neg A \sqcap \forall r.(\neg B \sqcup A))\}$$

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

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

Tableau Algorithmus Example

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



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

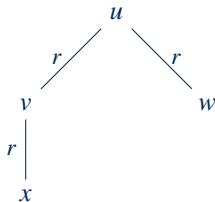
$$L(v) = \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{\exists r.B}\}$$

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

$$L(x) = \{B\}$$

Tableau Algorithmus Example

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



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

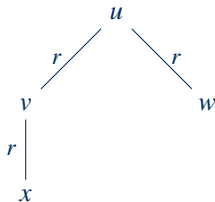
$$L(v) = \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{\exists r.B}\}$$

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

$$L(x) = \{B, \neg B \sqcup A\}$$

Tableau Algorithmus Example

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



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

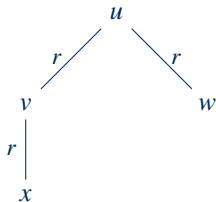
$$L(v) = \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \cancel{\exists r.B}\}$$

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

$$L(x) = \{B, \neg B \sqcup A, \neg B\}$$

Tableau Algorithmus Example

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



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

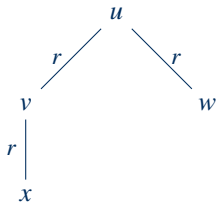
$$L(v) = \{A \sqcup \exists r.B, \neg A, \forall r.(\neg B \sqcup A), \exists r.B\}$$

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

$$L(x) = \{B, \neg B \sqcup A, B\}$$

Tableau Algorithmus Example

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



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

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

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

$$L(x) = \{B, \neg B \sqcup A, A\}$$

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 konstruieren

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*.

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