

# Struktur Vorlesung

- Kapitel 1: Einleitung
- Kapitel 2: Grundlagen
- Kapitel 3: Ausdrucksstärke und Modellkonstruktionen
- Kapitel 4: Tableau Algorithmen
- Kapitel 5: Komplexität
- Kapitel 6: ABoxen und Anfragebeantwortung
- Kapitel 7: Effiziente Beschreibungslogiken

## Tableau-Algorithmen

# Ziel des Kapitels

Automatisches Schlussfolgern spielt zentrale Rolle für BLen:

- ermöglicht die Entwicklung intelligenter Anwendungen
- die Ausdruckstärke von BLen ist stark darauf zugeschnitten

Wichtig für automatisches Schlussfolgern:

1. Entscheidbarkeit der relevanten Schlussfolgerungsprobleme
2. möglichst geringe Komplexität
3. Algorithmen, die sich in der Praxis performant verhalten

Dieses Kapitel: 1 + 3

# Ziel des Kapitels

Wir konzentrieren uns auf Erfüllbarkeit (vergl. Lemma 2.8)

Dabei betrachten wir

- Die Beschreibungslogik *ALC*
- sowohl Konzepte ohne TBoxen als auch generelle TBoxen;

## Tableau-Algorithmen

Entscheidbarkeit

# Entscheidbarkeit

Kapitel 3:

In  $\mathcal{ALC}$ :

Wenn  $C$  erfüllbar bzgl.  $\mathcal{T}$ , dann haben  $C$  und  $\mathcal{T}$  Modell der Grösse  $2^n$ .

Algorithmus für Erfüllbarkeit:

Gegeben  $C$  und  $\mathcal{T}$  so dass  $|C| + |\mathcal{T}| = n$ ,

- erzeuge alle Interpretation  $\mathcal{I}$  mit  $|\Delta|^{\mathcal{I}} \leq 2^n$   
(es gibt “nur”  $2^{2n^3+2}$  viele davon)

T4.0

- überprüfe, ob  $\mathcal{I}$  Modell von  $C$  und  $\mathcal{T}$   
(in Zeit polynomiell in  $\mathcal{I}$ ,  $C$  und  $\mathcal{T}$ )

$\Rightarrow$  Erfüllbarkeit in  $\mathcal{ALC}$  bzgl. genereller TBoxen entscheidbar

# Entscheidbarkeit

**Lemma 4.1.** Gegeben sei  $\mathcal{ALC}$ -Konzept  $C$ , endl. Interpretation  $\mathcal{I}$  und  $d \in \Delta^{\mathcal{I}}$ . Man kann in polynomieller Zeit (in  $|C| + |\Delta^{\mathcal{I}}|$ ) entscheiden ob  $d \in C^{\mathcal{I}}$ .

T4.1

**Korollar 4.2.** Gegeben sei  $C, \mathcal{T}$  in  $\mathcal{ALC}$  und endl. Interpretation  $\mathcal{I}$ . Man kann in polynomieller Zeit (in  $|C| + |\mathcal{T}| + |\Delta^{\mathcal{I}}|$ ) entscheiden ob  $\mathcal{I}$  ein Modell von  $C$  und  $\mathcal{T}$  ist.

(Model checking in Polynomialzeit)

# Entscheidbarkeit

Komplexität: 2-ExpTime

2-exponentiell viele Interpretationen müssen geprüft werden  
jede Prüfung braucht polynomielle Zeit.

Dieser Ansatz ist kaum tauglich für die Praxis:

- Die Komplexität ist höher als nötig
- Das Aufzählen aller Modelle ist nicht praktikabel:  
zu wenig zielorientiert uns best-case 2-exponentiell!



# Praktikable Algorithmen

In der Praxis haben sich hauptsächlich als effizient herausgestellt:

- Tableau-Algorithmen wie in RACER, FaCT++, Pellet, Hermit
- Resolutionsverfahren wie in KAON2

Tableau Algorithmen:

- werden heute in den meisten BL-Systemen eingesetzt
- wurden ursprünglich für FO und andere Logiken entwickelt
- versuchen, ein (Baum)modell für Eingabe zu konstruieren
- basieren auf der Anwendung von Regeln.

## Tableau-Algorithmen

ALC ohne TBoxen

# Tableau Algorithmus

Ziel:

Entwicklung eines Tableau Algorithmus für Erfüllbarkeit  
von *ALC* Konzepten (ohne TBoxen)

# Negationsnormalform

## Definition 4.3. (Negationsnormalform)

Konzept ist in *Negationsnormalform (NNF)* gdw. Negation nur auf atomare Konzepte angewendet wird.

## Lemma 4.4.

Jedes Konzept kann in Linearzeit in ein äquivalentes Konzept in NNF umgewandelt werden.

T4.2

Wir nehmen an, dass Eingabekonzept  $C_0$  in NNF ist.

# I-Baum

Zugrundeliegende Datenstruktur repräsentiert (partielles) Baummodell

## Definition 4.5 (I-Baum)

*I-Baum* für  $C_0$  ist knoten- und kantenbeschrifteter Baum  $(V, E, \mathcal{L})$  mit

- $V$  Knotenmenge;
- $E \subseteq V \times \mathbf{N}_R \times V$  Menge beschrifteter Kanten;
- $\mathcal{L} : V \rightarrow 2^{\text{sub}(C_0)}$  Knotenbeschriftung

T4.3

# Tableau Algorithmus

Tableau Algorithmus berechnet Sequenz

$$M_0, M_1, \dots$$

von Mengen von I-Bäumen.

$M_0 = \{B_{ini}\}$  mit  $B_{ini}$  *initialer I-Baum* für  $C_0$ :

$$V := \{v_{ini}\}$$

$$E := \emptyset$$

$$\mathcal{L}(v_{ini}) := \{C_0\}$$

$M_{i+1}$  aus  $M_i$  durch Anwendung von Tableau Regel

(Transformiert I-Baum in einen oder mehrere neue I-Bäume)

# Tableau Regeln

Sei  $(V, E, \mathcal{L})$  I-Baum.

$\sqcap$ -Regel:

- wähle  $v \in V$  und  $C \sqcap D \in \mathcal{L}(v)$  so dass  $\{C, D\} \not\subseteq \mathcal{L}(v)$
- erweitere  $\mathcal{L}(v)$  um  $C$  und  $D$

$\sqcup$ -Regel:

- wähle  $v \in V$  und  $C \sqcup D \in \mathcal{L}(v)$  so dass  $\{C, D\} \cap \mathcal{L}(v) = \emptyset$
- erweitere  $\mathcal{L}(v)$  um  $C$  **oder** um  $D$  (ergibt zwei I-Bäume)

# Tableau Regeln

$\exists$ -Regel:

- wähle  $v \in V$  und  $\exists r.C \in \mathcal{L}(v)$  so dass  
es kein  $v' \in V$  gibt mit  $(v, r, v') \in E$  und  $C \in \mathcal{L}(v')$
- erweitere  $V$  um neuen Knoten  $v'$  und  $E$  um  $(v, r, v')$ ,  
setze  $\mathcal{L}(v') = \{C\}$

$\forall$ -Regel:

- wähle  $v, v' \in V$  und  $\forall r.C \in \mathcal{L}(v)$  so dass  
 $(v, r, v') \in E$  und  $C \notin \mathcal{L}(v')$
- erweitere  $\mathcal{L}(v')$  um  $C$



# Tableau Algorithmus

Berechnung von  $M_{i+1}$  aus  $M_i$ :

- Auswahl eines  $B \in M_i$  und Anwendung einer der 4 Regeln
- Regelanwendung: ersetzen von  $B$  durch neuen I-Baum bzw. zwei neue I-Bäume ( $\sqcup$ -Regel)

Intuition: Regeln machen implizites Wissen explizit.

I-Baum ist *vollständig*, wenn keine Regel darauf anwendbar ist.

# Ergebnis

Algorithmus stoppt, wenn keine Regel mehr anwendbar ist.

Rückgabe von

- “erfüllbar” wenn I-Baum gefunden wurde, der keinen *offensichtlichen Widerspruch* enthält:

$$\{A, \neg A\} \subseteq \mathcal{L}(v) \text{ für ein } v \in V, A \in \mathbf{N}_c$$

- “unerfüllbar” sonst

T4.4

# Terminierung

Intuitiv:  $\text{rd}(C)$  ist Schachtelungstiefe von  $\exists/\forall$ -Konstruktoren in  $C$ ,

*Rollentiefe*  $\text{rd}(C)$  von Konzepten  $C \in \text{sub}(C_0)$  induktiv definiert:

- $\text{rd}(A) = \text{rd}(\neg A) = 0$ ;
- $\text{rd}(C \sqcap D) = \text{rd}(C \sqcup D) = \max(\text{rd}(C), \text{rd}(D))$ ;
- $\text{rd}(\exists r.C) = \text{rd}(\forall r.C) = 1 + \text{rd}(C)$ .

Lemma 4.6. Für alle  $C \in \text{sub}(C_0)$  gilt  $\text{rd}(C) \leq |C|$ .

# Terminierung

## Proposition 4.7 (Terminierung)

Der Tableau Algorithmus stoppt nach endlicher Zeit.

T4.5

# Multimengen

Multimengen sind Mengen, in denen Elemente mehrfach vorkommen dürfen:

$$\{a, a, b, b, b\}, \{1, 1, 2, 3, 4, 4, 5, 6, 6, 6\}, \{\emptyset, \emptyset, \{\emptyset, \emptyset\}\}$$

Formal: Multimenge über Menge  $S$  ist Abbildung

$$M : S \rightarrow \mathbb{N}$$

die die Anzahl des Vorkommens der Elemente beschreibt

Die meisten Begriffe übertragen sich von Mengen auf Multimengen:

- Leere Menge  $\emptyset$ :  $s \mapsto 0$  für alle  $s \in S$
- Vereinigung:  $(M_1 \cup M_2)(s) := M_1(s) + M_2(s)$
- Element:  $s \in M$  gdw.  $M(s) > 0$
- Differenz:  $(M_1 \setminus M_2)(s) = \begin{cases} M_1(s) - M_2(s) & \text{wenn } M_1(s) \geq M_2(s) \\ 0 & \text{sonst} \end{cases}$

# Multimengen

Sei  $S$  endlich,  $MM(S)$  die Menge aller Multimengen über  $S$

Gegeben partielle Ordnung  $(S, <)$  ist *Multimengenerweiterung*  $(MM(S), <_{\text{mul}})$  definiert als:

$M_2 <_{\text{mul}} M_1$  gdw.  $\exists X, Y \in MM(S)$  so dass:

- $\emptyset \neq X \subseteq M_1$ ;
- $M_2 = (M_1 \setminus X) \cup Y$ ;
- $\forall y \in Y \exists x \in X : x > y$ .

Also:  $M_2$  erhält man aus  $M_1$  indem man einige Elemente entfernt und durch endlich viele kleinere ersetzt.

$$\{3, 1\} >_{\text{mul}} \{2, 2, 2\} >_{\text{mul}} \{2, 2\} >_{\text{mul}} \{2, 1, 1, 1\}$$

# Multimengen

**Theorem.**

Wenn  $(S, <)$  fundiert ist, dann auch  $(MM(S), <_{\text{mul}})$ .

# Korrektheit und Vollständigkeit

**Proposition 4.10** (Korrektheit)

Wenn der Algorithmus “erfüllbar” zurückgibt, so ist  $C_0$  erfüllbar. **T4.6**



# Korrektheit und Vollständigkeit

**Definition 4.8** (Realisierbarkeit)

Sei  $B = (V, E, \mathcal{L})$  ein I-Baum. Interpretation  $\mathcal{I}$  *realisiert*  $B$  gdw. es gibt Funktion

$$\pi : V \rightarrow \Delta^{\mathcal{I}}$$

so dass

- $(v, r, v') \in E$  impliziert  $(\pi(v), \pi(v')) \in r^{\mathcal{I}}$ ;
- $C \in \mathcal{L}(v)$  impliziert  $\pi(v) \in C^{\mathcal{I}}$ .

T4.7

$B$  ist *realisierbar* wenn es Interpretation  $\mathcal{I}$  gibt, die  $B$  realisiert.

Menge  $M$  von I-Bäumen ist *realisierbar* gdw. ein  $B \in M$  realisierbar.

Beachte: realisierbarer I-Baum enthält keinen offensichtlichen Widerspruch!

# Korrektheit und Vollständigkeit

**Proposition 4.11** (Vollständigkeit)

Wenn  $C_0$  erfüllbar, so gibt der Algorithmus “erfüllbar” zurück.

T4.8

# Komplexitätsanalyse

Beobachtung:

die I-Bäume können höchstens exponentiell gross werden  
(Beweis von Proposition 4.7)

Dieser Worst Case kann tatsächlich eintreffen:

Erfüllbarkeitstest von

$$\prod_{i < n} \forall r^i. (\exists r. B \sqcap \exists r. \neg B) \quad \forall r^i. C = \underbrace{\forall r. \dots \forall r. C}_{i \text{ mal}}$$

generiert Baum der Größe  $2^n$ .

T4.9

Also: exponentieller Zeit- und Platzverbrauch (sogar 2-exponentiell!).

# Praktikabilität

Naive Implementierung ist nicht effizient.

Implementierungsgrundlagen:

- Es wird nur ein Baum zur Zeit generiert, keine Menge
- bei der  $\sqcup$ -Regel muss man sich also entscheiden (Heuristik)  
ggf. Entscheidung revidieren (Backtracking)
- es wird nur ein Teil des Baumes (Pfad) im Speicher gehalten

Darüber hinaus gibt es zahlreiche effektive Optimierungstechniken.

# Optimierungen: Backjumping

Beispiel für Optimierung: Backjumping

Form von dependenzbasiertem Backtracking:

- führe Buch über die “Herkunft” von Knotenbeschriftungen und Kanten mittels Dependenzmenge
- wenn Backtracking nötig (offens. Widerspruch), springe direkt zu einer der Ursachen des Widerspruches zurück.

Hat dramatische Effekte in der Praxis.

T4.10

## Tableau-Algorithmen

ALC mit generellen TBoxen

# Tableau-Algorithmus

Ziel:

Entwicklung eines Tableau Algorithmus für Erfüllbarkeit  
in  $\mathcal{ALC}$  bzgl. genereller TBoxen

Jede generelle TBox  $\mathcal{T}$  ist äquivalent zu einer TBox der Form  $\{\top \sqsubseteq C_{\mathcal{T}}\}$ :

$$\text{setze } C_{\mathcal{T}} := \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} C \rightarrow D. \quad \text{T4.11}$$

Wir nehmen an, dass

- Eingabe  $C_0$  in NNF;
- Eingabe  $\mathcal{T}$  hat Form  $\{\top \sqsubseteq C_{\mathcal{T}}\}$  mit  $C_{\mathcal{T}}$  in NNF.

# Tableau-Algorithmus

Modifiziere vorigen Algorithmus durch Hinzufügen folgender Regel:

TBox-Regel:

- wähle  $v \in V$  so dass  $C_{\mathcal{T}} \notin \mathcal{L}(v)$
- erweitere  $\mathcal{L}(v)$  um  $C_{\mathcal{T}}$

Problem: Algorithmus terminiert nicht!

T4.12



# Blockieren

Problem:

Die Tiefe von Baummodellen kann unendlich sein!

Lösung:

Konstruiere nur endliches Anfangsstück eines Baummodelles,  
mittels dessen sich die Existenz eines vollst. Modelles entscheiden lässt.

Dazu müssen wir die Anwendung der  $\exists$ -Regel einschränken.

# Blockieren

**Definition 4.12.** (Blockiert)

Sei  $(V, E, \mathcal{L})$  ein I-Baum und  $u, v \in V$ .

Dann ist  $v$  *direkt blockiert von  $u$*  wenn

1.  $u$  Vorgänger von  $v$  in  $B$  ist und
2.  $\mathcal{L}(v) \subseteq \mathcal{L}(u)$

$v$  ist *blockiert* wenn  $v$  direkt blockiert ist oder einen direkt blockierten Vorgänger hat.

T4.13

"Vorgänger" meint nicht unbedingt direkten Vorgänger!

# Blockieren

Neue  $\exists$ -Regel:

$\exists'$ -Regel:

- wähle  $v \in V$  und  $\exists r.C \in \mathcal{L}(v)$  so dass

$v$  nicht blockiert ist und

es kein  $v' \in V$  gibt mit  $(v, r, v') \in E$  und  $C \in \mathcal{L}(v')$

- erweitere  $V$  um neuen Knoten  $v'$  und  $E$  um  $(v, r, v')$ ,  
setze  $\mathcal{L}(v') = \{C\}$

T4.14

# Tableau-Algorithmus

## Proposition 4.14 (Vollständigkeit)

Wenn  $C_0$  erfüllbar bzgl.  $\mathcal{T}$ , so gibt der Algorithmus “erfüllbar” zurück.

Beweis wie ohne TBoxen:

alle  $M_0, \dots, M_n$  sind realisierbar bzgl.  $\mathcal{T}$  (Induktion),  
also enthält  $M_n$  keinen offensichtlichen Widerspruch.

Unterschiede:

- Realisierbarkeit wird bzgl. Modellen von  $\mathcal{T}$  definiert;
- neuer Fall für TBox-Regel.

# Tableau-Algorithmus

Es bleibt also zu zeigen:

## Proposition 4.15 (Korrektheit)

Wenn der Algorithmus “erfüllbar” zurückgibt, so ist  $C_0$  erfüllbar bzgl.  $\mathcal{T}$ .

T4.15

## Proposition 4.16 (Terminierung)

Der Tableau Algorithmus stoppt nach endlicher Zeit.

T4.16

# Komplexitätsanalyse

Beobachtung:

die I-Bäume können höchstens doppelt exponentiell gross werden  
(Beweis von Proposition 4.15)

Dieser Worst Case kann tatsächlich eintreffen:

**Lemma 4.17.**

Es gibt Eingabe  $C_0, \mathcal{T}$ , für die der Tableau Algorithmus einen Baum von doppelt exponentieller Größe generiert.

T4.17

Also: 2-exponentieller Zeit- und Platzverbrauch (sogar 3-exponentiell!).

## Bemerkung zur TBox-Regel

Generelle TBoxen führen zu Backtracking:

Normalisierung  $\mathcal{T}$  zu  $\{\top \sqsubseteq \bigsqcup_{C \sqsubseteq D \in \mathcal{T}} C \rightarrow D\}$ , also

$$\{\top \sqsubseteq \bigsqcup_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D\}$$

$\sqcup$ -Regel für *jede* Konzeptinklusion auf *jeden* Knoten angewendet!

Für effiziente Implementierung braucht man Optimierungstechniken, die die Disjunktionen soweit möglich eliminieren (“Absorption”).

## Tableau-Algorithmen

Erweiterungen von ALC



# Erweiterungen

Algorithmus kann auf *ALCT*, *ALCQ* und *ALCQT* erweitert werden.

Das ist teilweise subtiler als erwartet, z.B.:

- *ALCT*

Offensichtlich: Hinzufügen von Regeln für  $\exists r^-.C$  und  $\forall r^-.C$

Weniger offensichtlich: Blockierungsbedingung muss verschärft werden,  
sonst ist Algorithmus nicht korrekt!

Für *ALCQT* ist noch aufwendigere Blockierungsbedingung nötig.