Struktur Vorlesung

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

Tableau-Algorithmen

Ziel des Kapitels

Automatisches Schlussfolgern spielt zentrale Rolle für BLen:

- ermöglicht die Entwicklung intelligenter Anwendungen
- die Ausdrucksstärke 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: 3

Wir konzentrieren uns auf Erfüllbarkeit (vergl. Lemma 2.9)

Praktikable Algorithmen

In der Praxis haben sich hauptsächlich als effizient herausgestellt:

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

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.

Wir betrachten zunächst Erfüllbarkeit ohne TBoxen, dann mit TBoxen

Tableau-Algorithmen

ALC ohne TBoxen

Negationsnormalform

Definition 4.1 (Negationsnormalform)
Konzept ist in *Negationsnormalform (NNF)* gdw. Negation nur auf Konzeptnamen (also nicht auf zusammengesetzte Konzepte) angewendet wird.

Lemma 4.2
Jedes Konzept kann in Linearzeit in ein äquivalentes Konzept in NNF umgewandelt werden.

T4.1

Wir nehmen an, dass Eingabekonzept C_0 in NNF ist.

I-Baum

Zugrunde liegende Datenstruktur repräsentiert (partielles) Baummodell

Definition 4.3 (I-Baum)
I-Baum für C_0 ist knoten- und kantenbeschrifteter Baum $(V, E, \mathcal{L})$ mit

- V Knotenmenge
- E ist Menge beschrifteter Kanten (v, r, v') mit $v, v' \in V$, r Rollenname
- $\mathcal{L} : V \to 2^{\mathsf{sub}(C_0)}$ Knotenbeschriftung

T4.2

Tableau-Algorithmus

Tableau-Algorithmus berechnet Folge

$$\boldsymbol{M_0, M_1, \ldots}$$

von Mengen von I-Bäumen.

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

$$\begin{aligned} \boldsymbol{V} &:= \{\boldsymbol{v_{\mathsf{ini}}}\} \\ \boldsymbol{E} &:= \emptyset \\ \boldsymbol{\mathcal{L}(v_{\mathsf{ini}})} &:= \{\boldsymbol{C_0}\} \end{aligned}$$

$\boldsymbol{M_{i+1}}$ entsteht aus $\boldsymbol{M_i}$ 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 $\boldsymbol{M_{i+1}}$ aus $\boldsymbol{M_i}$:

- Auswahl eines $\boldsymbol{B} \in \boldsymbol{M_i}$ und Anwendung einer der 4 Regeln
- Regelanwendung: ersetzen von $\boldsymbol{B}$ 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 alle I-Bäume vollständig sind.

Rückgabe von

- „erfüllbar", wenn I-Baum gefunden wurde, der keinen *offensichtlichen Widerspruch* enthält:

  $\{A, \neg A\} \subseteq \mathcal{L}(v)$ für einen Knoten v und Konzeptnamen A

- „unerfüllbar" sonst

T4.3

Terminierung

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

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

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

Lemma 4.4 Für alle $C \in \mathsf{sub}(C_0)$ gilt $\mathsf{rd}(C) \leq |C|$.

Terminierung

Proposition 4.5 (Terminierung)
Der Tableau-Algorithmus stoppt nach endlicher Zeit.

Beweis in 4 Schritten:

Behauptung 1 Es werden nur I-Bäume mit einem Verzweigungsgrad von maximal $|C_0|$ generiert. T4.5a

Behauptung 2 Es werden nur I-Bäume mit einer Tiefe von maximal $|C_0|$ generiert. T4.5b

Behauptung 3 Sei $M_0, M_1, \ldots$ die erzeugte Folge und $B \in M_i$ für ein $i \geq 0$. Dann ist B durch die Anwendung von maximal

$$\underbrace{|C_0|^{|C_0|}}_{\#\text{ Knoten im Baum}} \cdot \underbrace{|C_0|}_{\#\text{ Größe Knotenbeschriftungen}} \leq 2^{2|C_0|^2} =: n$$

Regeln entstanden. (folgt kombinatorisch aus Beh. 1 und 2)

Schritt 4: benötigt spezielle Ordnungsrelation auf den M_i:

Multimengen

Multimengen sind Mengen, in denen Elemente mehrfach vorkommen dürfen:

z. B.: $\{a, a, b, b, b\}$ $\{1, 1, 2, 3, 4, 4, 5, 6, 6, 6\}$ $\{\emptyset, \emptyset, \{\emptyset, \emptyset\}\}$

Formal: Multimenge über Menge S ist Abbildung

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

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

$\mathrm{MM}(S)$ bezeichnet die Menge aller Multimengen über der Menge S.

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

$M_2 <_{\mathsf{mul}} M_1$ gdw. $\exists X, Y \in \mathrm{MM}(S)$, 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.

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

Multimengen

Leicht zu zeigen:
Auch $(\mathbf{MM}(S), <_{\mathsf{mul}})$ ist strikte partielle Ordnung.

Zur Erinnerung:
Partielle Ordnung $<$ heißt *wohlfundiert*,
wenn $<$ keine unendlich absteigenden Ketten hat.

Also ist z.B. $(\mathbb{N}, <)$ wohlfundiert, aber $(\mathbb{Z}, <)$ und $([\mathbf{0}, \mathbf{1}]_{\mathbb{R}}, <)$ nicht.

Theorem 4.6
Wenn $(S, <)$ wohlfundiert ist, dann ist auch $(\mathbf{MM}(S), <_{\mathsf{mul}})$ wohlfundiert.

Zurück zur Terminierung

Proposition 4.5 (Terminierung)
Der Tableau-Algorithmus stoppt nach endlicher Zeit.

Beweis in 4 Schritten:

✔ **Behauptung 1** Es werden nur I-Bäume mit einem Verzweigungsgrad von maximal $|C_0|$ generiert.

✔ **Behauptung 2** Es werden nur I-Bäume mit einer Tiefe von maximal $|C_0|$ generiert.

✔ **Behauptung 3** Sei $M_0, M_1, \ldots$ die erzeugte Folge und $B \in M_i$ für ein $i \geq 0$. Dann ist B durch die Anwendung von maximal

$$\underbrace{|C_0|^{|C_0|}}_{\#\text{ Knoten im Baum}} \cdot \underbrace{|C_0|}_{\#\text{ Größe Knotenbeschriftungen}} \leq 2^{2|C_0|^2} =: n$$

Regeln entstanden. (folgt kombinatorisch aus Beh. 1 und 2)

Terminierung kann nun mittels Behauptung 3 bewiesen werden. T4.5d

Korrektheit und Vollständigkeit

Proposition 4.7 (Korrektheit)
Wenn der Tableau-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

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

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 4.9 (Vollständigkeit)
Wenn C_0 erfüllbar,
so gibt der Tableau-Algorithmus „erfüllbar“ zurück. T4.8

Komplexitätsanalyse

Beobachtung:

Die I-Bäume können höchstens exponentiell groß werden.
(Beweis von Proposition 4.5)

Dieser Worst Case kann tatsächlich eintreffen:

Erfüllbarkeitstest von

$$\forall r^i.C = \forall \underbrace{r. \cdots .\forall r}_{i \text{ mal}}.C$$

$$\bigsqcap_{i<n} \forall r^i.(\exists r.B \sqcap \exists r.\neg B)$$

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 (offensichtlicher 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. TBoxen

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

setze $C_\mathcal{T} := \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D$.

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 Baummodells, anhand dessen sich die Existenz eines vollständigen Modells entscheiden lässt.

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

Blockieren

Definition 4.10 (Blockiert)

Sei $(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{\mathcal{L}})$ ein I-Baum und $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}$.
Dann ist $\boldsymbol{v}$ *direkt blockiert durch* $\boldsymbol{u}$, wenn

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

$\boldsymbol{v}$ ist *blockiert*, wenn $\boldsymbol{v}$ direkt blockiert ist oder einen direkt blockierten Vorgänger hat.

T4.13

„Vorgänger“ bedeutet nicht unbedingt „direkter 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 4.11 (Vollständigkeit)
Wenn C_0 erfüllbar bzgl. $\mathcal{T}$, so gibt der Algorithmus „erfüllbar“ zurück.

Beweis wie ohne TBoxen:

alle $M_0, \ldots, M_n$ sind realisierbar bzgl. $\mathcal{T}$ (Induktion),
also enthält M_n einen Baum ohne offensichtlichen Widerspruch.

Unterschiede:

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

Tableau-Algorithmus

Es bleibt also zu zeigen:

Proposition 4.12 (Korrektheit)
Wenn der Algorithmus „erfüllbar" zurückgibt, so ist C_0 erfüllbar bzgl. $\mathcal{T}$.

T4.15

Proposition 4.13 (Terminierung)
Der Tableau-Algorithmus stoppt nach endlicher Zeit.

Dies zeigen wir mit denselben Schritten wie im Fall ohne TBoxen (Prop. 4.5):

Terminierung

Beweis in 4 Schritten:

Behauptung 1 Es werden nur I-Bäume mit einem Verzweigungsgrad von maximal $k := |C_0| + |\mathcal{T}|$ generiert. (Beweis wie in Prop. 4.5)

Behauptung 2 Es werden nur I-Bäume mit einer Tiefe von maximal 2^k generiert. T4.16

Behauptung 3 Sei $M_0, M_1, \ldots$ die erzeugte Folge und $B \in M_i$ für ein $i \geq 0$. Dann ist B durch die Anwendung von maximal

$$\underbrace{k^{2^k}}_{\#\text{ Knoten im Baum}} \cdot \underbrace{k}_{\#\text{ Größe Knotenbeschriftungen}} \leq 2^{2^{3k}} =: n$$

Regeln entstanden. (folgt kombinatorisch aus Beh. 1 und 2)

Terminierung kann nun wie gehabt mittels Behauptung 3 bewiesen werden (per Multimengenordnung).

Komplexitätsanalyse

Beobachtung:

Die I-Bäume können höchstens doppelt exponentiell groß werden.
(Beweis von Proposition 4.15)

Dieser Worst Case kann tatsächlich eintreten:

Lemma 4.14
Es gibt Eingabe C_0, $\mathcal{T}$, für die der Tableau-Algorithmus einen Baum von exponentieller Tiefe generiert.

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

Bemerkung zur TBox-Regel

TBoxen führen zu Backtracking:

Normalisierung von $\mathcal{T}$ zu $\{\top \sqsubseteq \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D\}$

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

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

Tableau-Algorithmen

Erweiterungen von ALC

Erweiterungen

Algorithmus kann auf $\mathcal{ALCI}$, $\mathcal{ALCQ}$ und $\mathcal{ALCQI}$ erweitert werden.

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

- $\mathcal{ALCI}$

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

  Weniger offensichtlich: Blockierungsbedingung muss verschärft werden,

  sonst ist Algorithmus nicht korrekt!

Für $\mathcal{ALCQI}$ ist noch aufwendigere Blockierungsbedingung nötig.