

Vorlesungsübersicht

Beschreibungslogik

Kapitel 4: Tableau-Algorithmen

Sommersemester 2017 Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

<http://tinyurl.com/ss17-bl>

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

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: **Punkt 3**

Wir konzentrieren uns auf **Erfüllbarkeit** (vgl. 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 Logik erster Stufe 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.

Kapitel 4: Tableau-Algorithmen

- 1 ALC ohne TBoxen
- 2 ALC mit TBoxen
- 3 Erweiterungen von ALC

Negationsnormalform

Definition 4.1 (Negationsnormalform)

Ein Konzept ist in **Negationsnormalform (NNF)**, wenn 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.

T 4.1

Wir nehmen ab jetzt an, dass das Eingabekonzept C_0 in NNF ist.

Kapitel 4: Tableau-Algorithmen

- 1 ALC ohne TBoxen
- 2 ALC mit TBoxen
- 3 Erweiterungen von ALC

I-Bäume

... bilden eine Datenstruktur, die ein (partielles) Baummodell repräsentiert

Definition 4.3 (I-Baum)

Ein **I-Baum** für C_0 ist ein 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
- $\mathcal{L} : V \rightarrow 2^{\text{sub}(C_0)}$ ist die Knotenbeschriftung

T 4.2

Tableau-Algorithmus

Der Tableau-Algorithmus berechnet eine Folge

$$M_0, M_1, \dots$$

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} entsteht aus M_i durch Anwendung von Tableau-Regel (transformiert I-Baum in einen oder mehrere neue I-Bäume)

Tableau-Regeln

∃-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 $\mathcal{L}(v') = \{C\}$

∀-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-Regeln

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

∩-Regel:

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

∪-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-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 (∪-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.

Dann 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

T 4.3

Terminierung

Für Terminierung benötigen wir die Schachtelungstiefe von \exists -/ \forall -Konstruktoren in C :

Die **Rollentiefe** $rd(C)$ von Konzepten $C \in \text{sub}(C_0)$ ist induktiv wie folgt definiert.

$$\begin{aligned} rd(A) &= rd(\neg A) &= 0 \\ rd(C \sqcap D) &= rd(C \sqcup D) &= \max(rd(C), rd(D)) \\ rd(\exists r.C) &= rd(\forall r.C) &= 1 + rd(C) \end{aligned}$$

Lemma 4.4

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

Analyse des Tableau-Algorithmus

Wir müssen nun zeigen, dass der Algorithmus ...

- 1 **terminiert:**
Nach endlicher Zeit sind alle erzeugten I-Bäume vollständig, und der Algo. kann stoppen und „(un)erfüllbar“ zurückgeben.
- 2 **korrekt ist:**
Wenn der Algorithmus „erfüllbar“ zurückgibt, dann ist das Eingabekonzept auch tatsächlich erfüllbar.
- 3 **vollständig ist:**
Wenn das Eingabekonzept erfüllbar ist, dann gibt der Algorithmus auch „erfüllbar“ zurück.

Wir beginnen mit Terminierung.

Terminierung

Theorem 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. T 4.4
- **Behauptung 2.** Es werden nur I-Bäume mit einer **Tiefe von maximal $|C_0|$** generiert. T 4.5
- **Behauptung 3.** Sei M_0, M_1, \dots die erzeugte Folge und $B \in M_i$ für ein $i \geq 0$. Dann ist B durch die Anwendung von maximal

$$\underbrace{|C_0|^{|\underbrace{C_0}|}}_{\# \text{ Knoten im Baum}} \cdot \underbrace{|C_0|}_{\text{Größe Knotenbeschriftungen}} \leq 2^{2|C_0|^2} =: n$$
 Regeln entstanden. (folgt kombinatorisch aus Beh. 1 und 2)
- **Schritt 4** benötigt spezielle **Ordnungsrelation** auf den $M_i \dots$

Multimengen

Mengen, in denen Elemente **mehrfach** vorkommen dürfen:

Eine **Multimenge über einer Menge S** ist eine 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

Leicht zu zeigen:

Fakt 4.6

Auch $(MM(S), <_{mul})$ ist strikte partielle Ordnung.

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

Bsp.: $(\mathbb{N}, <)$ ist wohlfundiert, aber $(\mathbb{Z}, <)$ und $([0, 1]_{\mathbb{R}}, <)$ nicht.

Theorem 4.7

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

Multimengen

$MM(S)$ bezeichnet die Menge aller Multimengen über der Menge S .

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

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

- $\emptyset \neq X \subseteq M_1$
- $M_2 = (M_1 \setminus X) \cup Y$
- Für alle $y \in Y$ gibt es ein $x \in X$ mit $x > y$.

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

Z. B.: $\{3, 1\} >_{mul} \{2, 2, 2\} >_{mul} \{2, 2\} >_{mul} \{2, 1, 1, 1\}$

Zurück zur Terminierung

Theorem 4.5 (wiederholt)

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, \dots 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:** Terminierung folgt nun mit Behauptung 3. **T4.6** \square

Zurück zu: Analyse des Tableau-Algorithmus



Wir müssen nun zeigen, dass der Algorithmus ...

✓ **terminiert:**

Nach endlicher Zeit sind alle erzeugten I-Bäume vollständig, und der Algo. kann stoppen und „(un)erfüllbar“ zurückgeben.

② **korrekt** ist:

Wenn der Algorithmus „erfüllbar“ zurückgibt, dann ist das Eingabekonzept auch tatsächlich erfüllbar.

③ **vollständig** ist:

Wenn das Eingabekonzept erfüllbar ist, dann gibt der Algorithmus auch „erfüllbar“ zurück.

Wir machen weiter mit Korrektheit und Vollständigkeit.

Vollständigkeit: Vorbereitungen

Definition 4.9 (Realisierbarkeit)

Sei $B = (V, E, \mathcal{L})$ ein I-Baum und \mathcal{I} eine Interpretation. \mathcal{I} realisiert B , wenn es eine Funktion

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

gibt, so dass gilt:

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

T 4.8

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:

Ein realisierbarer I-Baum enthält **keinen offensichtlichen Widerspruch**.

Korrektheit

Theorem 4.8

Wenn der Tableau-Algorithmus „erfüllbar“ zurückgibt, dann ist C_0 erfüllbar.

Beweis. Wenn der Algorithmus „erfüllbar“ zurückgibt, so hat er einen **vollständigen** I-Baum $B = (V, E, \mathcal{L})$ **ohne offens. Widerspruch** gefunden. Aus B konstruieren wir Interpretation \mathcal{I} wie folgt.

$$\Delta^{\mathcal{I}} = V$$

$$r^{\mathcal{I}} = \{(v, v') \mid (v, r, v') \in E\} \quad \text{für alle Rollennamen } r$$

$$A^{\mathcal{I}} = \{v \mid A \in \mathcal{L}(v)\} \quad \text{für alle Konzeptnamen } A$$

Die Erfüllbarkeit von C_0 folgt dann aus:

Behauptung. Für alle Konzepte C und Knoten $v \in V$ gilt:

$$C \in \mathcal{L}(v) \quad \text{impliziert} \quad v \in C^{\mathcal{I}}$$

T 4.7



Vollständigkeit

Theorem 4.10

Wenn C_0 erfüllbar ist, dann gibt der Tableau-Algorithmus „erfüllbar“ zurück.

Beweis. Sei C_0 erfüllbar. Nach Theorem 4.5 berechnet der Algorithmus eine **endliche** Folge

$$M_0, \dots, M_n$$

von Mengen von I-Bäumen. Wir zeigen:

Behauptung. Für alle $i \leq n$ ist M_i realisierbar.

T 4.9

Somit gibt es ein realisierbares $B \in M_n$.

Damit ist B vollständig und ohne offensichtlichen Widerspruch.

Also gibt der Algorithmus „erfüllbar“ zurück.



Komplexitätsanalyse

Beobachtung:

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

Dieser Worst Case kann tatsächlich eintreten:

- Sei $\forall r^i.C$ eine Abkürzung für $\underbrace{\forall r. \dots \forall r}_i.C$.
- Dann generiert der Erfüllbarkeitstest von

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

einen Baum der Größe 2^n .

T 4.10

Also: mindestens **exponentieller** Zeit- und Platzverbrauch
(wegen der Baum**mengen** sogar **doppelt** exponentiell, also 2^{2^n})

Beispiel für Optimierung: Backjumping

... ist eine 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 Widerspruchs zurück.

Hat dramatische Effekte in der Praxis.

T 4.11

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 wirksame **Optimierungstechniken**, z. B. Backjumping.

Kapitel 4: Tableau-Algorithmen

① ALC ohne TBoxen

② ALC mit TBoxen

③ Erweiterungen von ALC

Vorbetrachtungen

Ziel:

Entwicklung eines Tableau-Algorithmus für Erfüllbarkeit in ALC

bzgl. TBoxen

Beobachtung:

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

$$\text{setze dazu } C_{\mathcal{T}} := \bigsqcup_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D \quad \text{T 4.12}$$

Wir nehmen ab jetzt an, dass

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

Blockieren

Problem:

Algorithmus terminiert nicht,
weil die Tiefe von Baummodellen unendlich sein kann!

Lösung:

Konstruiere nur ein **endliches Anfangsstück** eines Baummodells,
anhand dessen sich die Existenz eines vollständigen Modells
entscheiden lässt.

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

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!

T 4.13

Blockieren

Definition 4.11 (Blockieren)

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

Dann ist v **direkt blockiert durch** u , wenn

- 1 u ein Vorgänger^a 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.

^a**Beachte:** „Vorgänger“ bedeutet nicht unbedingt „direkter Vorgänger“!

T 4.14

Blockieren

Geänderte \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 $\mathcal{L}(v') = \{C\}$

T 4.15

Analyse des Tableau-Algorithmus

Wir müssen wieder zeigen, dass der Algorithmus . . .

- terminiert,
- korrekt ist,
- vollständig ist.

Wir beginnen diesmal mit Vollständigkeit.

Vollständigkeit

Theorem 4.12

Wenn C_0 **bezüglich \mathcal{T}** erfüllbar ist, dann gibt der Tableau-Algorithmus „erfüllbar“ zurück.

Beweis wie ohne TBoxen:

- Alle M_0, \dots, 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.

Korrektheit

Theorem 4.13

Wenn der Tableau-Algorithmus „erfüllbar“ zurückgibt, dann ist C_0 **bezüglich \mathcal{T}** erfüllbar.

Beweis. Analog zu Theorem 4.8: konstruieren aus vollständigem I-Baum $B = (V, E, \mathcal{L})$ ohne offens. Widerspruch ein Modell \mathcal{I} :

$$\Delta^{\mathcal{I}} = \{v \in V \mid v \text{ unblockiert}\}$$

$$r^{\mathcal{I}} = \{(v, v') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (v, r, v') \in E\}$$

$$\cup \{(v, u) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists (v, r, v') \in E, v' \text{ direkt blockiert von } u\}$$

$$A^{\mathcal{I}} = \{v \mid A \in \mathcal{L}(v)\}$$

Die Erfüllbarkeit von C_0 **bzgl. \mathcal{T}** folgt dann aus:

Behauptung. Für alle Konzepte C und Knoten $v \in \Delta^{\mathcal{I}}$ gilt:

$$C \in \mathcal{L}(v) \text{ impliziert } v \in C^{\mathcal{I}}$$

T 4.16 □

Terminierung

Theorem 4.14 (Terminierung)

Der Tableau-Algorithmus stoppt nach endlicher Zeit.

Beweis. Dieselben Schritte wie im Fall ohne TBoxen (Thm. 4.5):

- **Behauptung 1.** Es werden nur I-Bäume mit **Verzweigungsgrad** $\leq k := |C_0| + |\mathcal{T}|$ generiert. (Beweis wie in Thm. 4.5)
- **Behauptung 2.** Es werden nur I-Bäume mit einer **Tiefe von maximal 2^k** generiert. T 4.17
- **Behauptung 3.** Sei M_0, M_1, \dots die erzeugte Folge und $B \in M_i$ für ein $i \geq 0$. Dann ist B durch die Anwendung von maximal $\underbrace{k^{2^k}}_{\# \text{ Knoten im Baum}} \cdot \underbrace{k}_{\text{Größe Knotenbeschriftungen}} \leq 2^{2^{3k}}$ Regeln entstanden. (folgt kombinatorisch aus Beh. 1 und 2)
- **Schritt 4:** Terminierung wie gehabt mit Beh. 3 (MM-Ordnung). \square

Bemerkung zur TBox-Regel

TBoxen verursachen erheblich mehr Backtracking:

$$\text{Normalisierung von } \mathcal{T} \text{ zu } \{T \sqsubseteq \prod_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D\}$$

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

Für effiziente Implementierung braucht man Optimierungstechniken, die die „TBox-Disjunktionen“ eliminieren, soweit möglich (**Absorption**). T 4.18

Komplexitätsanalyse

Beobachtung:

Die I-Bäume können höchstens **doppelt** exponentiell groß werden. (Beweis von Theorem 4.14)

Dieser Worst Case kann tatsächlich eintreten:

Lemma 4.15

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

(ohne Beweis)

Also: mindestens **2-exponentieller** Zeit- und Platzverbrauch (wegen der Baum**mengen** sogar **3-exponentiell**, also $2^{2^{2^n}}$)

Kapitel 4: Tableau-Algorithmen

1 ALC ohne TBoxen

2 ALC mit TBoxen

3 Erweiterungen von ALC

Erweiterungen von ALC

Algorithmus kann auf $ALCI$, $ALCQ$ und $ALCQI$ erweitert werden.

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

- $ALCI$

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

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

Für $ALCQI$ ist noch aufwendigere Blockierungsbedingung nötig.

Literatur für dieses Kapitel



Franz Baader, Ian Horrocks, Carsten Lutz, Uli Sattler.

[Basis]

An Introduction to Description Logic.

Cambridge University Press, 2007.

Kapitel 4: Reasoning in DLs with Tableau Algorithms

Demnächst in SUUB verfügbar: <http://tinyurl.com/suub-intro-dl>

Bestellung beim Verlag: <http://www.cambridge.org/9780521695428>

Ich habe ein Exemplar und lasse Euch auf Nachfrage gern reinschauen.



Franz Baader, Uli Sattler.

[Weiterführend]

An Overview of Tableau Algorithms for Description Logics.

Studia Logica, 69:5–40, 2001. <http://dx.doi.org/10.1023/A:1013882326814>



Dmitry Tsarkov, Ian Horrocks, Peter F. Patel-Schneider. [Weiterführ.]

Optimizing Terminological Reasoning for Expressive Description Logics.

J. Autom. Reasoning 39:277–316, 2007.

<http://dx.doi.org/10.1007/s10817-007-9077-y>