Beschreibungslogik
Kapitel 7: ABoxen und Anfragebeantwortung

Sommersemester 2017 Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

http://tinyurl.com/ss17-bl

Vorlesungsübersicht

Ziel des Kapitels

TBoxen repräsentieren allgemeines, begriffliches Wissen.

Um **konkrete Situationen** zu repräsentieren, braucht man **Instanzdaten.**

Für medizinische Ontologien wie SNOMED z. B. Patientendaten:

$\text{Patient}(p_1)$	$\text{Patient}(p_2)$
$\text{Medikament}(m)$	$\text{Krankheit}(k)$
$\text{erhält}(p_1, m)$	$\text{erhält}(p_2, m)$
$\text{heilt}(m, k)$	$\text{hat}(p_1, k)$

Konzeptnamen Patient, Medikament etc. können in TBox definiert sein.

Ziel des Kapitels

Ziel des Kapitels

- Einführen eines Formalismus für Instanzdaten **(ABox)**
- Studium von Schlussfolgerungsproblemen mit Instanzdaten (insb. verschiedene Varianten von **Anfragebeantwortung**)
- Fokus: Anfragebeantw. mit Datenbanksystemen **und** Ontologien **(Query Rewriting)**

Kapitel 7: ABoxen und Anfragebeantwortung

Kapitel 7: Effiziente Beschreibungslogiken

ABoxen: Syntax

Wir reservieren eine unendliche Menge von **Individuen** $a, b, \ldots$

Diese entsprechen Konstanten im Sinne der Prädikatenlogik.

Definition 7.1 (ABoxen, Syntax)

- Eine **Konzeptassertion** hat die Form $A(a)$, A Konzeptname.
- Eine **Rollenassertion** hat die Form $r(a, b)$, r Rollenname.

Eine **ABox** ist eine endliche Menge von (Konzept- und Rollen-)assertionen.

T 7.1

Ind($\mathcal{A}$) bezeichnet die Menge der in $\mathcal{A}$ verwendeten Individuen.

ABoxen: Semantik

Definition 7.2 (ABoxen, Semantik)

Interpretation $\mathcal{I}$

- **erfüllt** $A(a)$, wenn $a \in A^{\mathcal{I}}$;
- **erfüllt** $r(a, b)$, wenn $(a, b) \in r^{\mathcal{I}}$.

$\mathcal{I}$ ist **Modell** von $\mathcal{A}$, wenn $\mathcal{I}$ alle Assertionen in $\mathcal{A}$ erfüllt.

T 7.1 Forts.

Beachte: Modell $\mathcal{I}$ darf zusätzlich Assertionen wahr machen, die in $\mathcal{A}$ **nicht** vorkommen.

Das in ABoxen repräsentierte Wissen ist also **unvollständiges Wissen**.

Wissensbasen

ABox und TBox fasst man manchmal zu einer **Wissensbasis** zusammen:

Definition 7.3 (Wissensbasis)

Wissensbasis (WB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ besteht aus TBox $\mathcal{T}$ und ABox $\mathcal{A}$.

Interpretation $\mathcal{I}$ ist **Modell** von $\mathcal{K}$, wenn $\mathcal{I}$ Modell von $\mathcal{T}$ und von $\mathcal{A}$.

In vielen Anwendungen haben $\mathcal{T}$ und $\mathcal{A}$ **unterschiedlichen Status:**

- $\mathcal{T}$ wird einmal erstellt;
  ändert sich danach üblicherweise nicht mehr.
- $\mathcal{A}$ ändert sich häufig (wie Datenbank).

Grundlegende Schlussfolgerungsprobleme

Definition 7.4 (Konsistenz, Instanz)

Sei $\mathcal{K}$ Wissensbasis, $A(a)$ Konzeptassertion. Dann ist

- $\mathcal{K}$ **konsistent**, wenn $\mathcal{K}$ Modell hat;
- a eine **Instanz von A bzgl.** $\mathcal{K}$, wenn jedes Modell von $\mathcal{K}$ auch $A(a)$ erfüllt.
  Wir schreiben dann $\mathcal{K} \models A(a)$.

T 7.2

Konsistenzproblem:
Gegeben $\mathcal{K}$, entscheide ob $\mathcal{K}$ konsistent ist.

Instanzproblem:
Gegeben $\mathcal{K}$ und $A(a)$, entscheide ob $\mathcal{K} \models A(a)$.

Reduktionen

Konsistenz- und (Nicht-)Instanzproblem sind wechselseitig polynomiell reduzierbar:

Lemma 7.5

1. $\mathcal{K}$ ist konsistent **gdw.** $\mathcal{K} \not\models A(a)$, A neuer Konzeptname.
2. $(\mathcal{T}, \mathcal{A}) \models A(a)$ **gdw.** $(\mathcal{T} \cup \{\overline{A} \equiv \neg A\},\ \mathcal{A} \cup \{\overline{A}(a)\})$ inkonsistent.

T 7.3

Konzept-Erfüllbarkeit ist polynomiell reduzierbar auf Konsistenz:

Lemma 7.6

A erfüllbar bzgl. $\mathcal{T}$ **gdw.** $(\mathcal{T}, \{A(a)\})$ konsistent.

Intuitiv: Erfüllbarkeit entspricht genau WB-Konsistenz mit ABoxen der Form $\{A(a)\}$.

Anfragebeantwortung

Viele Anwendungen verwenden ABoxen wie (semantische) Datenbanken.

Verschiedene Anfragesprachen möglich:

- **Instanzanfrage**:
  gegeben $\mathcal{K}$ und A, ermittle alle a mit $\mathcal{K} \models A(a)$.
- **Konjunktive Anfragen**:
  mächtigere Anfragesprache, Definition später

Anfragebeantwortung ist ein **Berechnungsproblem,**
kein Entscheidungsproblem! **T 7.2 Forts.**

Instanzanfragen in $\mathcal{ALC}$

Beantworten einer Instanzanfrage A zur Wissensbasis $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

- Antworten berechenbar durch mehrfaches Entscheiden des Instanzproblems:

  Überprüfe, ob $\mathcal{K} \models A(a)$ für alle Individuen a in $\mathcal{A}$.
- Instanzproblem kann auf Konsistenzproblem reduziert werden.

Zur Beantwortung von Instanzanfragen ist es also im Prinzip ausreichend, **Konsistenz** von Wissensbasen entscheiden zu können.

In der Praxis ist das allerdings **nicht sehr effizient.**

Mögliche Algorithmen für Konsistenz:

- Erweiterung von $\mathcal{ALC}$-`Elim` (bzw. $\mathcal{ALC}$-`Worlds`, falls $\mathcal{T} = \emptyset$)
- Erweiterung von Tableau-Algorithmen
- Reduktion auf Konzept-Erfüllbarkeit bzgl. $\mathcal{T}$ *(Precompletion)*

Komplexität

Mit wenigen Ausnahmen:

Konsistenzproblem hat dieselbe Komplexität wie Erfüllbarkeit.

Für $\mathcal{ALC}$, $\mathcal{ALCI}$ also ExpTime-vollständig.

Konjunktive Anfragen führen zu noch **höheren Komplexitäten:**

- in $\mathcal{ALC}$ immer noch ExpTime-vollständig
- in $\mathcal{ALCI}$ 2ExpTime-vollständig

Fragen:

- Wie kann effiziente Anfragebeantwortung realisiert werden?
- Kann man relationale Datenbanksysteme (SQL-Datenbanken) einsetzen?

Kapitel 7: Effiziente Beschreibungslogiken

Relationale Datenbanken

Zur Erinnerung:

- **Relationales Schema** ist Menge von Tabellennamen mit Stelligkeit.
- Konkrete **Datenbankinstanz** füllt die Tabellen mit Inhalten.
- Anfragen sind in SQL formuliert.

Wir betrachten relationale Datenbanken, in denen alle Tabellen nur **eine oder zwei Spalten** haben.

- Einspaltige Tabellen entsprechen Konzeptnamen; zweispaltige Tabellen entsprechen Rollennamen.
- Datenbankinstanz entspricht dann **Interpretation** (nicht ABox): Datenbanken werden als *vollständig* behandelt (Semantik $\neg$).

T 7.4

ABoxen versus Datenbanken

(Un)vollständigkeit hat weitreichende Konsequenzen:

- **Anfragebeantwortung in relationalen Datenbanken**
  entspricht dem **Model-Checking-Problem:**
  - Datenbank = Modell
  - Anfrage = logische Formel
- **Anfragebeantwortung in Beschreibungslogik**
  entspricht logischer **Folgerbarkeit:**
  - KB = logische Theorie
  - Anfrage = logische Formel

Letzteres ist in der Regel ein **wesentlich schwierigeres Problem.**

Verschiedene wichtige Anfragesprachen

SQL = relationale Algebra = relationales Kalkül (Logik erster Stufe)

- Nützliche und weit verbreitete Anfragesprache.
- Führt im Zusammenhang mit unvollständigen Daten und TBoxen leider sofort zu **Unentscheidbarkeit.**

select-from-where-Anfragen = select-project-join-Anfragen = konjunktive Anfragen

- Wichtiges Fragment von SQL, keine Negation, keine Disjunktion
- In der Praxis sind $> 90\,\%$ aller SQL-Anfragen von dieser Art.

Datalog

- Regelbasierte Anfragesprache, Aussdrucksstärke orthogonal zu SQL
- Erlaubt Rekursion, aber keine Negation

Konjunktive Anfragen: Syntax

Von nun an sei V eine Menge von *Variablen*.

Definition 7.7 (Konjunktive Anfrage, CQ)

- Ein **Konzeptatom** hat die Form $A(x)$, A Konzeptname, $x \in \mathrm{V}$.
- Ein **Rollenatom** hat die Form $r(x, y)$, r Rollenname, $x, y \in \mathrm{V}$.

Eine **konjunktive Anfrage (CQ)** hat die Form $\exists \overline{y}\ \varphi(\overline{x}, \overline{y})$, wobei

- $\overline{y} = y_0 \cdots y_m$ die **quantifizierten Variablen** sind;
- $\overline{x} = x_0 \cdots x_n$ die **Antwortvariablen** sind;
- $\varphi(\overline{x}, \overline{y})$ Konjunktion von Konzept- und Rollenatomen über $\overline{x} \cup \overline{y}$ ist.

T 7.5

(„CQ“ steht für „conjunctive query“.)

CQs: Notation

Wir schreiben:

- x, y, z für Variablen
- $\overline{x}, \overline{y}, \overline{z}$ für Tupel von Variablen
- $\varphi(\overline{x}, \overline{y})$ für Konjunktionen von Atomen über Variablen $\overline{x} \cup \overline{y}$
- $q(\overline{x})$ für konjunktive Anfragen mit Antwortvariablen $\overline{x}$

CQs: Semantik

Definition 7.8 (Homomorphismus, Antwort bzgl. Interpretation)

Sei $\mathcal{I}$ Interpretation und $q(\overline{x}) = \exists \overline{y}\, \varphi(\overline{x}, \overline{y})$ mit $\overline{x} = x_1 \cdots x_n$.

Ein **Homomorphismus von $q(\overline{x})$ nach $\mathcal{I}$** ist eine Abbildung $h : \overline{x} \cup \overline{y} \to \Delta^{\mathcal{I}}$, so dass:

- $h(x) \in A^{\mathcal{I}}$ für alle Konzeptatome $A(x)$ in φ
- $(h(x), h(y)) \in r^{\mathcal{I}}$ für alle Rollenatome $r(x, y)$ in φ

Eine **Antwort auf $q(\overline{x})$ in $\mathcal{I}$** ist ein Tupel $\overline{a} = a_1 \cdots a_n$ von Elementen, so dass es einen Homomorphismus h von $q(\overline{x})$ nach $\mathcal{I}$ gibt mit $h(x_i) = a_i$ für $1 \leq i \leq n$.

ans$(q, \mathcal{I})$ bezeichnet die Menge aller Antworten auf $q(\overline{x})$ in $\mathcal{I}$.

T 7.6

Boolesche CQs

Boolesche konjunktive Anfrage: CQ q **ohne Antwortvariablen**

Wir schreiben $\boldsymbol{q \longrightarrow \mathcal{I}}$, wenn es Homomorphismus von q nach Interpretation $\mathcal{I}$ gibt.

Boolesche CQ q liefert für jede Interpretation $\mathcal{I}$ entweder

- **keine Antwort** (wenn $q \not\rightarrow \mathcal{I}$);
  wir schreiben dann $\boldsymbol{\mathcal{I} \not\models q}$, sagen „$\mathcal{I}$ macht q **falsch**“;
- oder **das leere Tupel ()** als einzige Antwort (wenn $q \longrightarrow \mathcal{I}$);
  wir schreiben dann $\boldsymbol{\mathcal{I} \models q}$, sagen „$\mathcal{I}$ macht q **wahr**“.

Bei Komplexitätsanalysen betrachten wir meist Boolesche Anfragen.

Die Ergebnisse übertragen sich im Prinzip auf Anfragen mit Antwortvariablen.

Datenkomplexität

In typischen Datenbank-Anwendungen sind die **Daten extrem groß**, **Anfragen** jedoch **verhältnismäßig klein.**

Datenkomplexität: Die Anfrage wird als fest angenommen, hat daher konstante Größe. Die Daten sind also die einzige Eingabe.

Im Gegensatz dazu **kombinierte Komplexität:**
Daten **und** Anfrage werden als Eingabe angesehen.

Einige beispielhafte Komplexitäten:

	Datenkomplexität	kombinierte Kompl.
CQ	in AC_0	NP-vollständig
SQL	in AC_0	PSPACE-vollständig
Datalog	P-vollständig	EXPTIME-vollständig

Datenkomplexität bildet praktische Beobachtungen **deutlich realistischer** ab!