

## Vorlesungsübersicht

## Beschreibungslogik

### Kapitel 7: ABoxen und Anfragebeantwortung

Sommersemester 2018      Thomas Schneider

AG Theorie der künstlichen Intelligenz (TdKI)

<http://tinyurl.com/ss18-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

TBoxen repräsentieren allgemeines, begriffliches Wissen.

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

Für medizinische Ontologien wie SNOMED z. B. Patientendaten:

Patient( $p_1$ )	Patient( $p_2$ )
Medikament( $m$ )	Krankheit( $k$ )
erhält( $p_1, m$ )	erhält( $p_2, m$ )
heilt( $m, k$ )	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

- 1 Grundlagen
- 2 Etwas Datenbanktheorie
- 3 Konjunktive Anfragen & Beschreibungslogik-TBoxen
- 4 Query Rewriting
- 5 Nachbemerkungen zur Vorlesung

## ABoxen: Syntax

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

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

# Kapitel 7: Effiziente Beschreibungslogiken

- 1 Grundlagen
- 2 Etwas Datenbanktheorie
- 3 Konjunktive Anfragen & Beschreibungslogik-TBoxen
- 4 Query Rewriting
- 5 Nachbemerkungen zur Vorlesung

## 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).

## 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.**  $(\mathcal{T} \cup \{\bar{A} \equiv \neg A\}, \mathcal{A} \cup \{\bar{A}(a)\})$  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  $\{A(a)\}$ .

## 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  $\mathcal{K} \models A(a)$ .

## 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*)

## Kapitel 7: Effiziente Beschreibungslogiken

1 Grundlagen

2 Etwas Datenbanktheorie

3 Konjunktive Anfragen & Beschreibungslogik-TBoxen

4 Query Rewriting

5 Nachbemerkungen zur Vorlesung

## 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?

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

## 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**:
  - Wissensbasis  $\mathcal{K}$  = logische Theorie
  - Anfrage = logische Formel

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

## 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,  $x \in V$ .
- Ein **Rollenatom** hat die Form  $r(x, y)$ ,  $r$  Rollenname,  $x, y \in V$ .

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

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

T 7.5: siehe nächste 3 Folien

(„CQ“ steht für „conjunctive query“.)

## 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, Ausdrucksstärke orthogonal zu SQL
- Erlaubt Rekursion, aber keine Negation

## CQs: Beispiel 1

„Alle Student\*innen, die eine Vorlesung hören“

$$q_1(x) = \exists y \text{StudentIn}(\underline{x}) \wedge \text{hört}(\underline{x}, y) \wedge \text{VL}(y)$$

**Konzeptatome:**  $\text{StudentIn}(\underline{x})$ ,  $\text{VL}(y)$

**Rollenatome:**  $\text{hört}(\underline{x}, y)$

**quantifizierte Variablen:**  $y$

**Antwortvariablen:**  $\underline{x}$

## CQs: Beispiel 2

„Alle Paare  $(a, b)$  mit  $a$  Student,  $b$  VL und  $a$  hört  $b$ “

$$q_2(x_1, x_2) = \text{StudentIn}(x_1) \wedge \text{hört}(x_1, x_2) \wedge \text{VL}(x_2)$$

**Konzeptatome:**  $\text{StudentIn}(x_1), \text{VL}(x_2)$

**Rollenatome:**  $\text{hört}(x_1, x_2)$

**quantifizierte Variablen:** keine

**Antwortvariablen:**  $x_1, x_2$

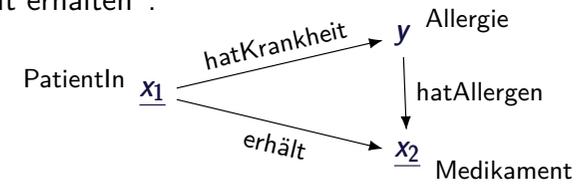
## CQs: Notation

**Wir schreiben:**

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

## CQs: Beispiel 3

„Alle Patient\*innen, die eine Allergie haben und ein passendes Medikament erhalten“:



$$q_3(x_1, x_2) = \exists y \text{ PatientIn}(x_1) \wedge \text{hatKrankheit}(x_1, y) \wedge \text{Allergie}(y) \wedge \text{erhält}(x_1, x_2) \wedge \text{hatAllergen}(y, x_2) \wedge \text{Medikament}(x_2)$$

**Konzeptatome:**  $\text{PatientIn}(x_1), \text{Allergie}(y), \text{Medikament}(x_2)$

**Rollenatome:**  $\text{hatKrankheit}(x_1, y), \text{erhält}(x_1, x_2), \text{hatAllergen}(y, x_2)$

**quantifizierte Variablen:**  $y$

**Antwortvariablen:**  $x_1, x_2$

## CQs: Semantik

**Definition 7.8 (Homomorphismus, Antwort bzgl. Interpretation)**

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

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

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

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

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

## Boolesche CQs

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

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

Bei Komplexitätsanalysen betrachten wir meist Boolesche Anfragen.

Die Ergebnisse übertragen sich im Prinzip auf Anfragen mit Antwortvariablen.

## Kapitel 7: Effiziente Beschreibungslogiken

- 1 Grundlagen
- 2 Etwas Datenbanktheorie
- 3 Konjunktive Anfragen & Beschreibungslogik-TBoxen
- 4 Query Rewriting
- 5 Nachbemerkungen zur Vorlesung

## 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!**

## Ontology-Mediated Queries

Wir betrachten die Beantwortung konjunktiver Anfragen über **ABoxen** (statt Interpretationen) in Gegenwart von TBoxen.

Dazu möchten wir idealerweise Datenbanksysteme verwenden.

Es macht für diesen Zweck Sinn,

- **nicht** TBox und ABox zu einer Wissensbasis zusammenzufassen,
- **sondern** CQ und TBox zu einer erweiterten Anfrage.

Dies führt zum Begriff der **ontology-mediated query**.  
(auf deutsch also etwa: Ontologie-vermittelte Anfrage)

# Ontology-Mediated Queries

## Definition 7.9 (OMQ)

Eine **Ontology-mediated query (OMQ)** ist ein Paar  $Q = (q(\bar{x}), \mathcal{T})$  mit  $q(\bar{x})$  konjunktiver Anfrage und  $\mathcal{T}$  TBox.

**Antwort auf  $Q$  in ABox  $\mathcal{A}$ :** Tupel  $\bar{a} = a_1 \cdots a_n$  von Individuen, das eine Antwort auf  $q(\bar{x})$  in **allen** Modellen von  $\mathcal{A}$  und  $\mathcal{T}$  ist.

Menge aller Antworten auf  $Q$  in  $\mathcal{A}$  bezeichnen wir mit  $\text{cert}(Q, \mathcal{A})$ . Solche Antworten nennt man auch **sichere Antworten** (engl.: certain answers).

**Beachte:** 
$$\text{cert}(Q, \mathcal{A}) = \bigcap_{\substack{\mathcal{I} \text{ Modell} \\ \text{von } \mathcal{A} \text{ und } \mathcal{T}}} \text{ans}(q, \mathcal{I}) \quad \text{T 7.7}$$

# Datenkomplexität von OMQs

**Wollen zeigen:** Beantwortung konjunktiver Anfragen in  $\mathcal{ALC}$  ist **coNP-hart bzgl. Datenkomplexität**.

Per Reduktion vom Komplement von 3-Färbbarkeit

## Definition 7.10 (3-Färbbarkeit, zur Erinnerung)

Ein **ungerichteter** Graph  $G = (V, E)$  ist **3-färbbar**, wenn es eine Abbildung  $f : V \rightarrow \{R, G, B\}$  gibt, so dass  $f(v_1) \neq f(v_2)$  für alle  $\{v_1, v_2\} \in E$ .

Betrachte folgende OMQ  $Q = (q, \mathcal{T})$  (TBox in  $\mathcal{ALC}$ ):

$$\mathcal{T} = \{ \top \sqsubseteq R \sqcup G \sqcup B, \quad R \sqcap \exists r.R \sqsubseteq D, \\ G \sqcap \exists r.G \sqsubseteq D, \quad D \text{ steht für} \\ B \sqcap \exists r.B \sqsubseteq D \quad \text{„Defekt“} \}$$

$$q = \exists x D(x)$$

# Beantwortung von OMQs

## Anfragebeantwortung:

Gegeben  $Q = (q, \mathcal{T})$  und  $\mathcal{A}$ , berechne  $\text{cert}(Q, \mathcal{A})$ .

Bei **Datenkomplexität** betrachten wir wieder nur die Daten (also  $\mathcal{A}$ ) als Eingabe, aber  $Q$  (und damit sowohl  $q$  als auch  $\mathcal{T}$ ) als fest.

**Boolesche OMQ**  $Q = (q, \mathcal{T})$  liefert für jede ABox  $\mathcal{A}$  entweder

- **leeres Tupel ( $\epsilon$ )** als einzige Antwort, notiert  $\mathcal{A} \models Q$  ( $q \rightarrow \mathcal{I}$  für alle Modelle  $\mathcal{I}$  von  $\mathcal{A}$  und  $\mathcal{T}$ ) oder
- **keine Antwort**, notiert  $\mathcal{A} \not\models Q$  ( $q \not\rightarrow \mathcal{I}$  für mindestens ein Modell  $\mathcal{I}$  von  $\mathcal{A}$  und  $\mathcal{T}$ ).

Für Boolesche OMQs ist Anfragebeantwortung also ein **Entscheidungsproblem**.

# Datenkomplexität von OMQs

$$\mathcal{T} = \{ \top \sqsubseteq R \sqcup G \sqcup B, \quad R \sqcap \exists r.R \sqsubseteq D, \quad D \text{ steht für} \\ G \sqcap \exists r.G \sqsubseteq D, \quad \text{„Defekt“} \\ B \sqcap \exists r.B \sqsubseteq D \}$$

$$q = \exists x D(x)$$

$$Q = (q, \mathcal{T})$$

## Graph-ABox:

- verwendet keine Konzeptnamen und nur den Rollennamen  $r$
- wenn  $r(a, b) \in \mathcal{A}$ , dann  $r(b, a) \in \mathcal{A}$

Offensichtlich sind Graph-ABoxen **dasselbe** wie ungerichtete Graphen.

## Lemma 7.11

Für alle Graph-ABoxen  $\mathcal{A}$ :  $\mathcal{A}$  nicht 3-färbbar gdw.  $\mathcal{A} \models Q$  **T 7.8**

Dies liefert wie gewünscht eine Reduktion von **Nicht-3-Färbbarkeit** auf Beantwortung der Anfrage  $Q$ .

# Kapitel 7: Effiziente Beschreibungslogiken

- 1 Grundlagen
- 2 Etwas Datenbanktheorie
- 3 Konjunktive Anfragen & Beschreibungslogik-TBoxen
- 4 Query Rewriting**
- 5 Nachbemerkungen zur Vorlesung

# Query rewriting: Ziel

Wollen nun Datenbanksysteme verwenden, um OMQs zu beantworten: TBoxen ins relationale Datenbanksystem „schummeln“.

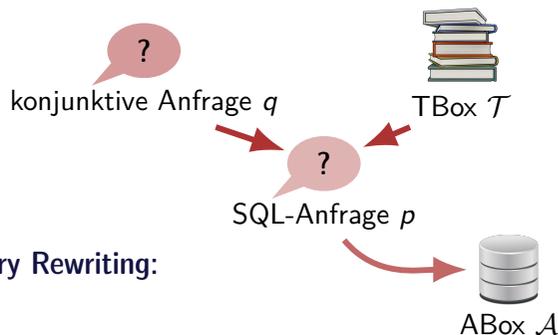
**Wir nehmen an:**

Die ABox  $\mathcal{A}$  ist als Interpretation  $\mathcal{I}$  in der Datenbank gespeichert:

$$\begin{aligned} \Delta^{\mathcal{I}} &= \text{Ind}(\mathcal{A}) \\ A^{\mathcal{I}} &= \{a \mid A(a) \in \mathcal{A}\} \\ r^{\mathcal{I}} &= \{(a, b) \mid r(a, b) \in \mathcal{A}\} \end{aligned}$$

Wir unterscheiden nicht explizit zwischen diesen 2 Darstellungen; verwenden ABox  $\mathcal{A}$  auch als Interpretation.

# Query rewriting: Idee und Definition



Die Idee von Query Rewriting:

### Definition 7.12 (Rewriting)

Sei  $Q = (q(\bar{x}), \mathcal{T})$  eine OMQ.

Eine SQL-Anfrage  $p(\bar{x})$  ist ein **SQL-Rewriting** von  $Q$ , wenn **für alle ABoxen  $\mathcal{A}$**  gilt:

$$\text{cert}(Q, \mathcal{A}) = \text{ans}(p, \mathcal{A})$$

↑  
ABox als Interpretation

# Rewritability: zu schön, um wahr zu sein?

Man kann zeigen: Nicht-3-Färbbarkeit ist **nicht in SQL ausdrückbar**. Dies ist aber schon für **viel einfachere OMQs** der Fall (TBox in  $\mathcal{EL}$ ):

$$\begin{aligned} \mathcal{T} &= \{ \text{Ziel} \sqsubseteq \text{Markiert}, \exists r. \text{Markiert} \sqsubseteq \text{Markiert} \} \\ q &= \exists x (\text{Markiert}(x) \wedge \text{Start}(x)) \end{aligned}$$

**Reach-ABox:**

- verwendet nur den Rollennamen  $r$ , beliebige Rollenassertionen
- enthält als Konzeptassertionen nur  $\text{Start}(a)$  und  $\text{Ziel}(b)$
- $\hat{=}$  gerichteter Graph mit markiertem Start & Ziel **T 7.9**

### Lemma 7.13

Für alle Reach-ABoxen  $\mathcal{A}$  gilt:

$$b \text{ erreichbar von } a \text{ gdw. } \mathcal{A} \models Q$$

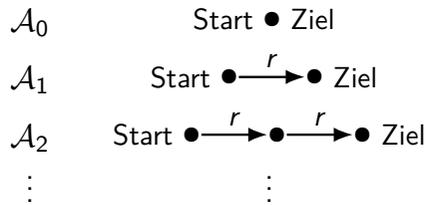
(Beweis ist recht elementar – probiert es aus.)

## Rewritability: zu schön, um wahr zu sein?

**Erreichbarkeit ist ebenfalls nicht in SQL ausdrückbar.**

Grund ist die **Nicht-Lokalität** dieser Anfrage:  
es gibt ABoxen  $\mathcal{A}_0, \mathcal{A}_1, \dots$  aufsteigender Größe, so dass gilt:

$$\mathcal{A}_i \models Q, \text{ aber } \mathcal{A}' \not\models Q \text{ für alle } \mathcal{A}' \subseteq \mathcal{A}_i$$



(Das bezieht sich auf SQL 1&2. Ab SQL 3 sind einige nicht-lokale Anfragen mittels linearer Rekursion ausdrückbar, z. B. Erreichbarkeit; dennoch ist auch in aktuellen SQL-Versionen nicht jede  $\mathcal{EL}$ -OMQ ausdrückbar.)

## DL-Lite

**Wir werden sehen:**

Für konjunktive Anfragen und DL-Lite-TBoxen gibt es **immer** SQL-Rewritings. **T 7.10 Forts.**

Zentrale Technik im Umgang mit DL-Lite: **universelle Modelle**

Hierbei handelt es sich um ein **einzelnes** Modell, das genau die „certain answers“ (definiert über **alle** Modelle) liefert.

## DL-Lite

**Definition 7.14 (DL-Lite)**

Ein **DL-Lite-Konzept** hat eine der folgenden Formen:

- $A$  (Konzeptname)
- $\top$  (Top-Konzept)
- $\exists r$  (**unqualifizierte** Existenzrestriktion) kurz für  $\exists r.\top$
- $\exists r^-$  (**unqualifizierte** inverse Existenzrestriktion) kurz für  $\exists r^-\top$

Eine **DL-Lite-TBox** ist eine endliche Menge von

- Konzeptinklusionen  $C_1 \sqsubseteq C_2$
- Rolleninklusionen  $R_1 \sqsubseteq R_2$  ( $R_1, R_2$ : Rollennamen  $r$  oder Inverse  $r^-$ )

Wir verzichten hier der Einfachheit halber auf negative Inklusionen

$$C_1 \sqsubseteq \neg C_2 \quad \text{und} \quad R_1 \sqsubseteq \neg R_2,$$

die in der ursprünglichen Definition ebenfalls zugelassen sind.

**T 7.10**

## Universelle Modelle

Sei  $\mathcal{A}$  ABox und  $\mathcal{T}$  DL-Lite-TBox. **Universelles Modell  $\mathcal{U}$**  für  $\mathcal{A}$  und  $\mathcal{T}$ :

**Schritt 1:** Starte mit  $\mathcal{U}_0$ :

$$\begin{aligned} \Delta^{\mathcal{U}_0} &= \text{Ind}(\mathcal{A}) \\ A^{\mathcal{U}_0} &= \{a \mid A(a) \in \mathcal{A}\} \\ r^{\mathcal{U}_0} &= \{(a, b) \mid r(a, b) \in \mathcal{A}\} \end{aligned}$$

**Schritt 2:** Wende erschöpfend folgende Regeln an:

- R1** Wenn  $d \in C^{\mathcal{U}_i}$ ,  $C \sqsubseteq A \in \mathcal{T}$ ,  $d \notin A^{\mathcal{U}_i}$ , dann  $A^{\mathcal{U}_{i+1}} = A^{\mathcal{U}_i} \cup \{d\}$
- R2** Wenn  $d \in C^{\mathcal{U}_i}$ ,  $C \sqsubseteq \exists R \in \mathcal{T}$ ,  $d \notin (\exists R)^{\mathcal{U}_i}$ , dann erweitere  $\Delta^{\mathcal{U}_i}$  um neues Element  $e$ ; setze  $R^{\mathcal{U}_{i+1}} = R^{\mathcal{U}_i} \cup \{(d, e)\}$
- R3** Wenn  $(d, e) \in R^{\mathcal{U}_i}$ ,  $R \sqsubseteq S \in \mathcal{T}$ ,  $(d, e) \notin S^{\mathcal{U}_i}$ , dann setze  $S^{\mathcal{U}_{i+1}} = S^{\mathcal{U}_i} \cup \{(d, e)\}$   
( $R, S$ : Rollennamen  $r$  oder Inverse  $r^-$ )

$\mathcal{U}$  ergibt sich im **Limit** bei fairer Regelanwendung.

**T 7.11**

## Universelle Modelle

Offensichtlich ist  $\mathcal{U}$  Modell für  $\mathcal{A}$  und  $\mathcal{T}$ :

- Bereits  $\mathcal{U}_0$  ist per Definition Modell von  $\mathcal{A}$ .
- Da keine Regel mehr anwendbar ist, ist  $\mathcal{U}$  Modell von  $\mathcal{T}$ .

**Kanonizität:** mittels Homomorphismen

### Definition 7.15

Ein **Homomorphismus** von Interpretation  $\mathcal{I}$  nach Interpretation  $\mathcal{J}$  ist eine Abbildung  $h : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{J}}$ , so dass:

- $d \in A^{\mathcal{I}}$  **impliziert**  $h(d) \in A^{\mathcal{J}}$
- $(d, e) \in r^{\mathcal{I}}$  **impliziert**  $(h(d), h(e)) \in r^{\mathcal{J}}$

Wir schreiben  $\mathcal{I} \rightarrow \mathcal{J}$ , wenn es Homomph. von  $\mathcal{I}$  nach  $\mathcal{J}$  gibt.

### Lemma 7.16

Für jedes Modell  $\mathcal{I}$  von  $\mathcal{A}$  und  $\mathcal{T}$  gilt:  $\mathcal{U} \rightarrow \mathcal{I}$  **T 7.12**

## Universelle Modelle

### Lemma 7.17

Für jede OMQ  $Q = (q, \mathcal{T})$  gilt:  $\mathcal{A} \models Q$  gdw.  $\mathcal{U} \models q$

**Beweis:** „ $\Rightarrow$ “.

Wenn  $\mathcal{U} \not\models q$ , dann  $\mathcal{A} \not\models Q$ , da  $\mathcal{U}$  Modell von  $\mathcal{A}$  und  $\mathcal{T}$ .

„ $\Leftarrow$ “.

Angenommen  $\mathcal{U} \models q$ .

Dann gibt es Homomorphismus  $h$  von  $q$  nach  $\mathcal{U}$ .

Nach Lemma 7.16 gibt es weiterhin Homomorphismus  $g$  von  $\mathcal{U}$  in jedes beliebige Modell  $\mathcal{I}$  von  $\mathcal{A}$  und  $\mathcal{T}$ .

Komposition  $h(g(\cdot))$  liefert Homomorphismus von  $q$  nach  $\mathcal{I}$ ; also  $\mathcal{I} \models q$  für alle Modelle  $\mathcal{I}$  von  $\mathcal{A}$  und  $\mathcal{T}$ .

Also  $\mathcal{A} \models Q$ . □

## Universelle Modelle

Können nun **Haupteigenschaft von universellen Modellen** beweisen.

Der Einfachheit halber im Folgenden **nur Boolesche Anfragen**.

### Lemma 7.17

Für jede OMQ  $Q = (q, \mathcal{T})$  gilt:  $\mathcal{A} \models Q$  gdw.  $\mathcal{U} \models q$

Entsprechend für nicht-Boolesche OMQs:  $\text{cert}(Q, \mathcal{A}) = \text{ans}(q, \mathcal{U})$

$\rightsquigarrow$  Das universelle Modell repräsentiert also genau die „certain answers“!

## Lokalität

**Anfragebeantwortung bzgl. DL-Lite-TBoxen ist lokal:**

### Theorem 7.18

Sei  $Q = (q, \mathcal{T})$  OMQ mit DL-Lite-TBox  $\mathcal{T}$ . Wenn  $\mathcal{A} \models Q$ , dann gibt es  $\mathcal{A}' \subseteq \mathcal{A}$  mit  $|\text{Ind}(\mathcal{A}')| \leq |Q|^2 + |Q|$  und  $\mathcal{A}' \models Q$ . **T 7.14**

**Beachte:** die Größe von  $\mathcal{A}'$  hängt nur von  $q$  und  $\mathcal{T}$  ab! **T 7.13**

### Definition 7.19 (kleine Zeugen)

Sei  $Q = (q, \mathcal{T})$  OMQ mit DL-Lite-TBox  $\mathcal{T}$ ,  $n = |Q|^2 + |Q|$  und  $\text{Ind} = \{a_1, \dots, a_n\}$  feste Menge von  $n$  Individuennamen.

ABox  $\mathcal{A}$  ist **kleiner Q-Zeuge**, wenn

- $\mathcal{A}$  nur Symbole aus  $Q$  verwendet;
- $\text{Ind}(\mathcal{A}) \subseteq \text{Ind}$  (also  $|\text{Ind}(\mathcal{A})| \leq n$ ) und
- $\mathcal{A} \models Q$ .

**Offensichtlich:** es gibt nur **endlich viele** kleine Q-Zeugen.

## Nun: Rewritings für DL-Lite-OMQs

### Korollar 7.20

Sei  $Q = (q, T)$  OMQ mit DL-Lite-TBox  $\mathcal{T}$ . Dann gilt:

$\mathcal{A} \models Q$  gdw. es gibt kleinen  $Q$ -Zeugen  $\mathcal{A}'$  mit  $\mathcal{A}' \subseteq^* \mathcal{A}$

### Lemma 7.21

Für jede ABox  $\mathcal{B}$  gibt es Boolesche SQL-Anfrage  $q_{\mathcal{B}}$ , so dass für alle ABoxen  $\mathcal{A}$ :

$\mathcal{B} \subseteq^* \mathcal{A}$  gdw.  $\mathcal{A} \models q_{\mathcal{B}}$  T 7.15

Damit ist die Konstruktion eines SQL-Rewritings offensichtlich:

$q_{\mathcal{B}_1} \text{ UNION } \dots \text{ UNION } q_{\mathcal{B}_k}$

wobei  $\mathcal{B}_1, \dots, \mathcal{B}_k$  alle (endlich vielen) kleinen  $Q$ -Zeugen sind

\*(bis auf Individuenumbenennung)

## Anfragebeantwortung jenseits von DL-Lite

- Wir hatten gesehen: auch für  $\mathcal{EL}$ -OMQs müssen SQL-Rewritings nicht existieren (Erreichbarkeit).
- Man kann für  $\mathcal{EL}$ -OMQs aber stets **Datalog**-Rewritings finden; auch dafür gibt es sehr effiziente Systeme.
- Anstatt die TBox in die **Anfrage** zu integrieren, kann man sie in die **ABox** integrieren (**Materialisierung**).
- Materialisierung funktioniert stets für  $\mathcal{EL}$ , erlaubt auch hier die Verwendung von SQL-Datenbanken.
- Materialisierung funktioniert im Allgemeinen **nicht** in  $\mathcal{ALC}$ ; in **praktisch relevanten Fällen** aber oftmals doch.

## Zusammenfassung

- Nicht-Lokalität ist der Grund, warum für  $\mathcal{EL}$ - und  $\mathcal{ALC}$ -OMQs im Allgemeinen **keine** SQL-Rewritings existieren.
- In DL-Lite sind  $\exists$ -Restriktionen extrem eingeschränkt: **unquantifiziert**
- Dies resultiert in **Lokalität** (kleine  $Q$ -Zeugen).
- Lokale Anfragen sind in SQL ausdrückbar.
- Hintergrund ist die Äquivalenz von SQL und Prädikatenlogik; für diese ist Lokalität eines der wichtigsten Werkzeuge.

## Und nun ...

- 1 Grundlagen
- 2 Etwas Datenbanktheorie
- 3 Konjunktive Anfragen & Beschreibungslogik-TBoxen
- 4 Query Rewriting
- 5 Nachbemerkungen zur Vorlesung

# Auslassungen

## Viele relevante Themen ausgelassen:

- „eingebaute“ Datentypen (Zahlen, Strings, etc.) zur Repräsentation von z. B. Alter, Gewicht, Distanz, etc.
- Zusammenhang zur Automatentheorie
- ★ Modularität von Ontologien, Modulextraktion
- ★ Erklärungen von Inferenzen (Subsumtionen, Unerfüllbarkeit)
- ★ Erweiterungen von DLs: z. B. temporale DLs, probabilistische DLs
- Entwurfsmethodologien für Ontologien
- Anwendungen wie das Semantische Web

★ Aktuelle Forschung, auch in unserer AG

★ ebenso; **mehr dazu morgen!**

# OWL

## OWL 1.0: $\mathcal{ALC}$ plus

- Zahlenrestriktionen
- Inverse Rollen
- Transitive Rollen
- Nominale (Konzepte mit genau einer Instanz in jedem Modell)
- etc.

Erfüllbarkeit ist entscheidbar und NExpTime-vollständig.

# OWL

Vom W3C (World Wide Web Committee, [w3.org](http://w3.org)) standardisierte Ontologiesprache:

- soll im semantischen Web Verwendung finden
- basiert auf Beschreibungslogik
- hat verschiedene menschen- und maschinenverständliche Syntaxen, u. a. XML, RDFS (Resource Description Framework)

## Zwei Versionen von OWL:

- OWL 1.0 (W3C-Standard 2004)
- OWL 2.0 (W3C-Standard 2009)  
siehe auch: <https://www.w3.org/TR/owl2-overview/>

# OWL

## OWL 2.0: OWL 1.0 plus

- reflexive und symmetrische Rollen
- Rolleninklusionen  $r_1 \circ r_2 \sqsubseteq s$
- etc.

Erfüllbarkeit ist N2ExpTime-vollständig.

## Wegen hoher Komplexität:

drei offizielle **Profile** mit besseren Berechnungseigenschaften, z. B.

- Schlussfolgern in Polynomialzeit (z. B. Profile EL, QL: basierend auf  $\mathcal{EL}$  bzw. DL-Lite)
- effiziente Beantwortung konjunktiver Anfragen mit Standard-DB-Systemen (Profil QL)

## Das war's (fast)

# Danke für's Teilnehmen! 😊

Übermorgen:

★ **Aktuelle Forschung:**

- Überblick zum Thema Modularität von Ontologien
- kurze Auswertung der Evaluationsergebnisse
- Werbung für Wahlveranstaltung im Wintersemester

## Literatur für dieses Kapitel



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

[Basis]

An Introduction to Description Logic.

Cambridge University Press, 2017.

Kapitel 7: Query Answering

In SUUB verfügbar: <https://tinyurl.com/suub-intro-dl-ebook>

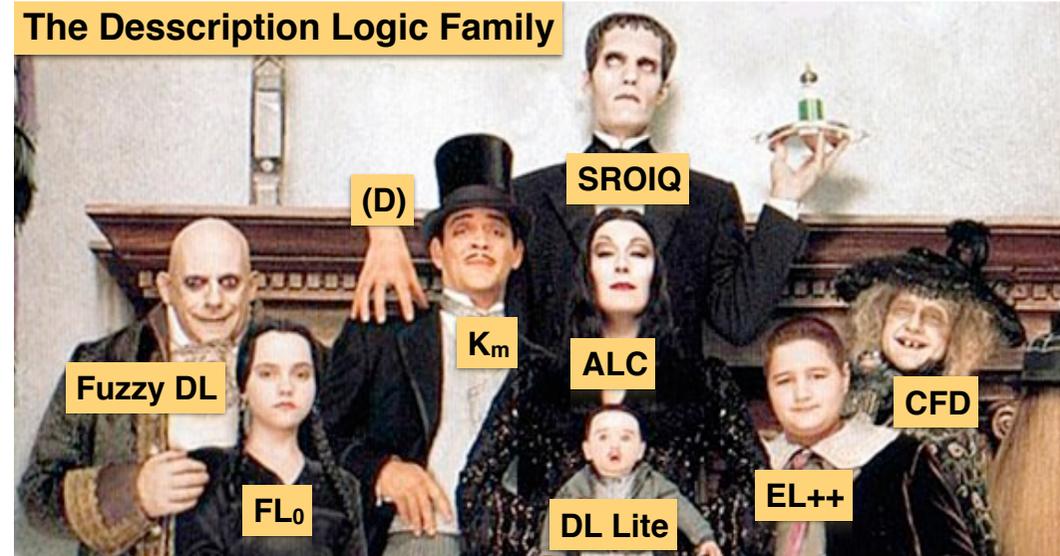
<https://tinyurl.com/suub-intro-dl>

[Weiterführend]

Carsten Lutz' aktuelle Arbeiten zum Thema Query Answering/Rewriting:

<http://www.informatik.uni-bremen.de/tdki/research/papers.html>

## The Description Logic Family



Idee und Collage: Uli Sattler und ich (ESSLLI-Kurs 2016 über DLs)