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

Kapitel 7

ABoxen und Anfragebeantwortung

Ziel des Kapitels

TBoxen repräsentieren allgemeines, begriffliches Wissen.

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

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)
- Schlussfolgerungsprobleme betrachten, um mit Instanzdaten zu arbeiten (insb. verschiedene Varianten von Anfragebeantwortung)
- Anfragebeantwortung mit Datenbanksystemen und Ontologien (Query Rewriting)

ABoxen und Anfragebeantwortung

Grundlagen

ABoxen – Syntax

Wir reservieren eine unendliche Menge von *Individuen* $\boldsymbol{a, b, \ldots}$
Diese entsprechen Konstanten im Sinne der Prädikatenlogik.

Definition 7.1 (ABoxen, Syntax)

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

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

T7.1

Mit $\mathrm{Ind}(\mathcal{A})$ bezeichnen wir 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.

T7.1 cont

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

T7.2

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 wechselseitig polynomiell reduzierbar:

Lemma 7.5

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

T7.3

Erfüllbarkeit (von Konzepten) 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!

T7.2cont

Instanzanfragen in ALC

Instanzanfrage A and Wissensbasis $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

- 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, wenn $\mathcal{T} = \emptyset$)
- Erweiterung von Tableau-Algorithmen
- Reduktion auf Erfüllbarkeit von Konzepten bzgl. $\mathcal{T}$ (Precompletion)

Komplexität

Mit wenigen Ausnahmen:

Konsistenzproblem hat dieselbe Komplexität wie Erfüllbarkeit.

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

Konjunktive Anfragen führen zu noch höheren Komplexitäten:

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

Wie kann effiziente Anfragebeantwortung realisiert werden?

Kann man relationale Datenbanksysteme (SQL-Datenbanken) einsetzen?

Kapitel 7

Etwas Datenbanktheorie

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: Rollennamen

Datenbankinstanz entspricht dann *Interpretation*, nicht etwa ABox:
Datenbanken werden als *vollständig* behandelt (Semantik Negation).

T7.4

ABoxen vs. 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 und 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 $\mathbf{V}$ eine Menge von *Variablen*.

Definition 7.7 (Konjunktive Anfrage, CQ)

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

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

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

T7.5

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 $\overline{x} = x_1 \cdots x_n$.

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

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

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

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

T7.6

Boolesche CQs

Boolesche konjunktive Anfrage: CQ q ohne Antwortvariablen

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

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

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

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-vollst.
SQL	in AC_0	PSPACE-vollst.
Datalog	P-vollständig	EXPTIME-vollst.

Datenkomplexität bildet praktische Beobachtungen deutlich realistischer ab!

Kapitel 7

Konjunktive Anfragen und Beschreibungslogik-TBoxen

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 einer $Ontology\text{-}Mediated\ Query.$

(auf deutsch also etwa: Ontologie-vermittelte Anfrage)

Ontology-Mediated Queries

Definition 7.9 (OMQ)

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

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

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

Beachte:
$$\mathsf{cert}(\boldsymbol{Q}, \boldsymbol{\mathcal{A}}) = \bigcap_{\boldsymbol{\mathcal{I}} \text{ Modell von } \boldsymbol{\mathcal{A}} \text{ und } \boldsymbol{\mathcal{T}}} \mathsf{ans}(\boldsymbol{q}, \boldsymbol{\mathcal{I}})$$

T7.7

Beantwortung von OMQs

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

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

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

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

Für Boolesche OMQs ist Anfragebeantwortung also ein *Entscheidungsproblem.*

Datenkomplexität von OMQs

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

Per Reduktion vom Komplement von 3-Färbbarkeit für ungerichtete Graphen.

Zur Erinnerung:
Ein ungerichteter Graph $G = (V, E)$ ist *3-färbbar*, wenn es eine Abbildung $f : V \to \{R, G, B\}$ 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 $\mathcal{ALC}$):

$$\mathcal{T} = \{ \quad \top \sqsubseteq R \sqcup G \sqcup B, \quad R \sqcap \exists r.R \sqsubseteq D, \quad G \sqcap \exists r.G \sqsubseteq D, \quad B \sqcap \exists r.B \sqsubseteq D \quad \}$$

D steht für „Defekt“

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

Datenkomplexität von OMQs

$$\mathcal{T} = \{ \quad \top \sqsubseteq \boldsymbol{R} \sqcup \boldsymbol{G} \sqcup \boldsymbol{B}, \quad \boldsymbol{R} \sqcap \exists \boldsymbol{r}.\boldsymbol{R} \sqsubseteq \boldsymbol{D}, \quad \boldsymbol{G} \sqcap \exists \boldsymbol{r}.\boldsymbol{G} \sqsubseteq \boldsymbol{D}, \quad \boldsymbol{B} \sqcap \exists \boldsymbol{r}.\boldsymbol{B} \sqsubseteq \boldsymbol{D} \quad \}$$

$\boldsymbol{D}$ steht für „Defekt“

$$\boldsymbol{q} = \exists \boldsymbol{x}\, \boldsymbol{D}(\boldsymbol{x})$$

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

Graph-ABox:

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

Offensichtlich sind Graph-ABoxen dasselbe wie ungerichtete Graphen.

Lemma 7.10

Für alle Graph-ABoxen $\mathcal{A}$ gilt: $\mathcal{A}$ ist nicht 3-färbbar gdw. $\mathcal{A} \models \boldsymbol{Q}$ T7.8

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

Query Rewriting

Ziel

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

Wir nehmen an:

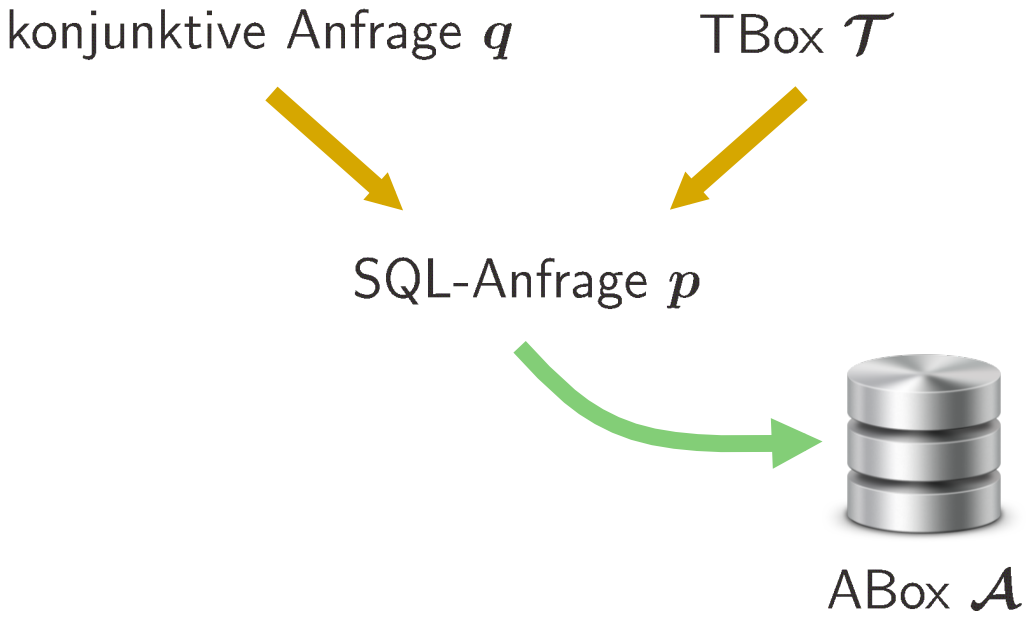
Die ABox $\mathcal{A}$ ist als Interpretation $\mathcal{I}$ in der Datenbank gespeichert:

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

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

Query Rewriting

Die Idee von Query Rewriting:



Definition 7.11 [Rewriting]

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

SQL-Anfrage $p(\overline{x})$ ist *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, dass Nicht-3-Färbbarkeit *nicht* in SQL ausdrückbar ist.

Es gibt aber viel einfachere OMQs, für die das der Fall ist (TBox in $\mathcal{EL}$):

$$\mathcal{T} = \{\ T \sqsubseteq M,\ \ \exists r.M \sqsubseteq M\ \}$$

$$q = \exists x\,(M(x) \wedge S(x))$$

Reach-ABox: verwendet nur den Rollennamen r
und enthält Assertionen $S(a)$ und $T(b)$, sonst keine Konzeptassertionen.

Reach-ABoxen sind gerichtete Graphen mit markiertem Start und Ziel.

T7.9

Lemma 7.12

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

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

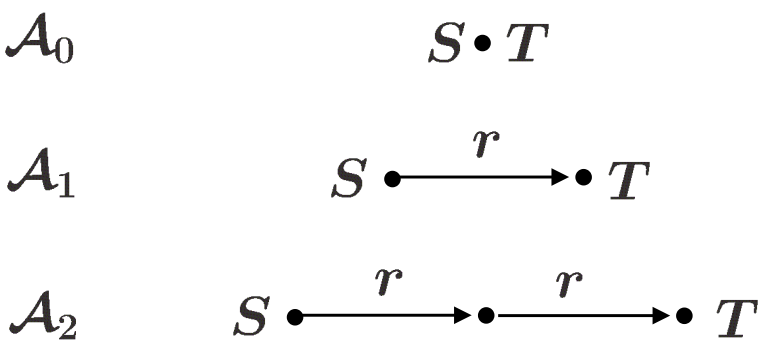
(Beweis ist recht elementar – probiert es aus.)

Rewritability: zu schön, um wahr zu sein?

Erreichbarkeit ist ebenfalls nicht in SQL ausdrückbar.

Grund ist die *Nicht-Lokalität* dieser Anfrage:
man findet ABoxen $\mathcal{A}_0, \mathcal{A}_1, \ldots,$ die immer größer werden und so dass gilt:

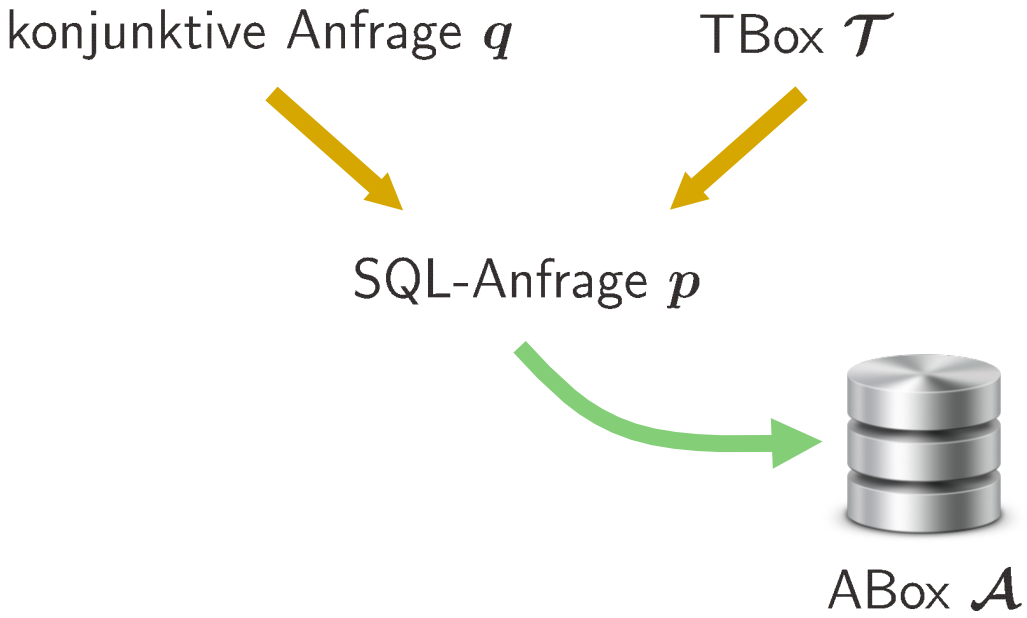
$$\mathcal{A}_i \models Q, \text{ aber } \mathcal{A}' \not\models Q \text{ für alle } \mathcal{A}' \subseteq \mathcal{A}_i$$



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

Query Rewriting

Die Idee von Query Rewriting:



Definition 7.11 [Rewriting]

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

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

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

DL-Lite

Definition 7.13 [DL-Lite]

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

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

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

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

Wir verzichten hier der Einfachheit haber auf negative Inklusionen

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

die in der ursprünglichen Definition ebenfalls zugelassen sind.

T7.10

DL-Lite

Wir werden sehen:
für konjunktive Anfragen und DL-Lite-TBoxen gibt es stets SQL-Rewritings.

T7.10 cont

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.

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 $\mathcal{U}_0$:

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

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 und 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** $S^{\mathcal{U}_{i+1}} = S^{\mathcal{U}_i} \cup \{(d,e)\}$

(R, S: Rollennamen r oder Inverse r^-)

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

T7.11

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 $\mathcal{T}$.

Homomorphismus von Interpretation $\mathcal{I}$ nach Interpretation $\mathcal{J}$
ist Abbildung $h : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$, 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 $\mathcal{I} \to \mathcal{J}$, wenn es Homomorphismus von $\mathcal{I}$ nach $\mathcal{J}$ gibt.

Lemma 7.14

Für jedes Modell $\mathcal{I}$ von $\mathcal{A}$ und $\mathcal{T}$ gilt: $\mathcal{U} \to \mathcal{I}$.

Universelle Modelle

Wir können nun die Haupteigenschaft von universellen Modellen beweisen.

Der Einfachheit halber im Folgenden nur Boolesche Anfragen.

Lemma 7.15
Für jede OMQ $\boldsymbol{Q} = (\boldsymbol{q}, \boldsymbol{\mathcal{T}})$ gilt: $\boldsymbol{\mathcal{A}} \models \boldsymbol{Q}$ gdw. $\boldsymbol{\mathcal{U}} \models \boldsymbol{q}$

Übersetzt auf nicht-Boolesche Anfragen heißt das: $\text{cert}(\boldsymbol{Q}, \boldsymbol{\mathcal{A}}) = \text{ans}(\boldsymbol{q}, \boldsymbol{\mathcal{U}})$

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

Universelle Modelle

Lemma 7.15

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

Beweis: „⇒“.

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

„⇐“.

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

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

Nach Lemma 7.14 gibt es weiterhin Homomorphismus $\boldsymbol{g}$ von $\boldsymbol{\mathcal{U}}$ in jedes beliebige Modell $\boldsymbol{\mathcal{I}}$ von $\boldsymbol{\mathcal{A}}$ und $\boldsymbol{\mathcal{T}}$.

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

Also $\boldsymbol{\mathcal{A}} \models \boldsymbol{Q}$.

Lokalität

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

Theorem 7.16

Sei $Q = (q, \mathcal{T})$ OMQ mit DL-Lite-TBox $\mathcal{T}$. Wenn $\mathcal{A} \models Q$, dann gibt es ein $\mathcal{A}' \subseteq \mathcal{A}$ mit $|\mathsf{Ind}(\mathcal{A}')| \leq |Q|^2 + |Q|$, so dass $\mathcal{A}' \models Q$. T7.13

Beachte: die Größe von $\mathcal{A}'$ hängt nur von q und $\mathcal{T}$ ab! T7.12

Definition 7.17

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

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

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

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

Rewritings

Korollar 7.18

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

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

(bis auf Individuenumbenennung)

Lemma 7.19

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

$$\mathcal{B} \subseteq \mathcal{A} \text{ (bis auf Individuenumbenennung)} \quad \text{gdw.} \quad \mathcal{A} \models q_{\mathcal{B}} \qquad \text{T7.14}$$

Damit ist die Konstruktion eines SQL-Rewritings offensichtlich:

$$q_{\mathcal{B}_1} \text{ UNION } \cdots \text{ UNION } q_{\mathcal{B}_n}$$

wobei $\mathcal{B}_1, \ldots, \mathcal{B}_n$ alle (endlich vielen) kleinen $\mathcal{Q}$-Zeugen sind.

Zusammenfassung

- Nicht-Lokalität ist der Grund, warum für $\mathcal{EL}$-OMQs und $\mathcal{ALC}$-OMQs im Allgemeinen keine SQL-Rewritings existieren.

- In DL-Lite sind Existenzrestriktionen extrem eingeschränkt: *unquantifiziert*
- Dies resultiert in Lokalität (kleine Q-Zeugen).

- Lokale Anfragen sind in SQL ausdrückbar.
- Hintergrund ist die Äquivalenz von SQL und Prädikatenlogik erster Stufe; für diese ist *Lokalität* eines der wichtigsten Werkzeuge.

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.

Kapitel 7

Konjunktive Anfragen mit Ungleichheit

(nur informativ; nicht mehr Bestandteil der VL)

Konjunktive Anfragen mit Ungleichheit

Schon leicht erweiterte Anfragen resultieren in Unentscheidbarkeit.

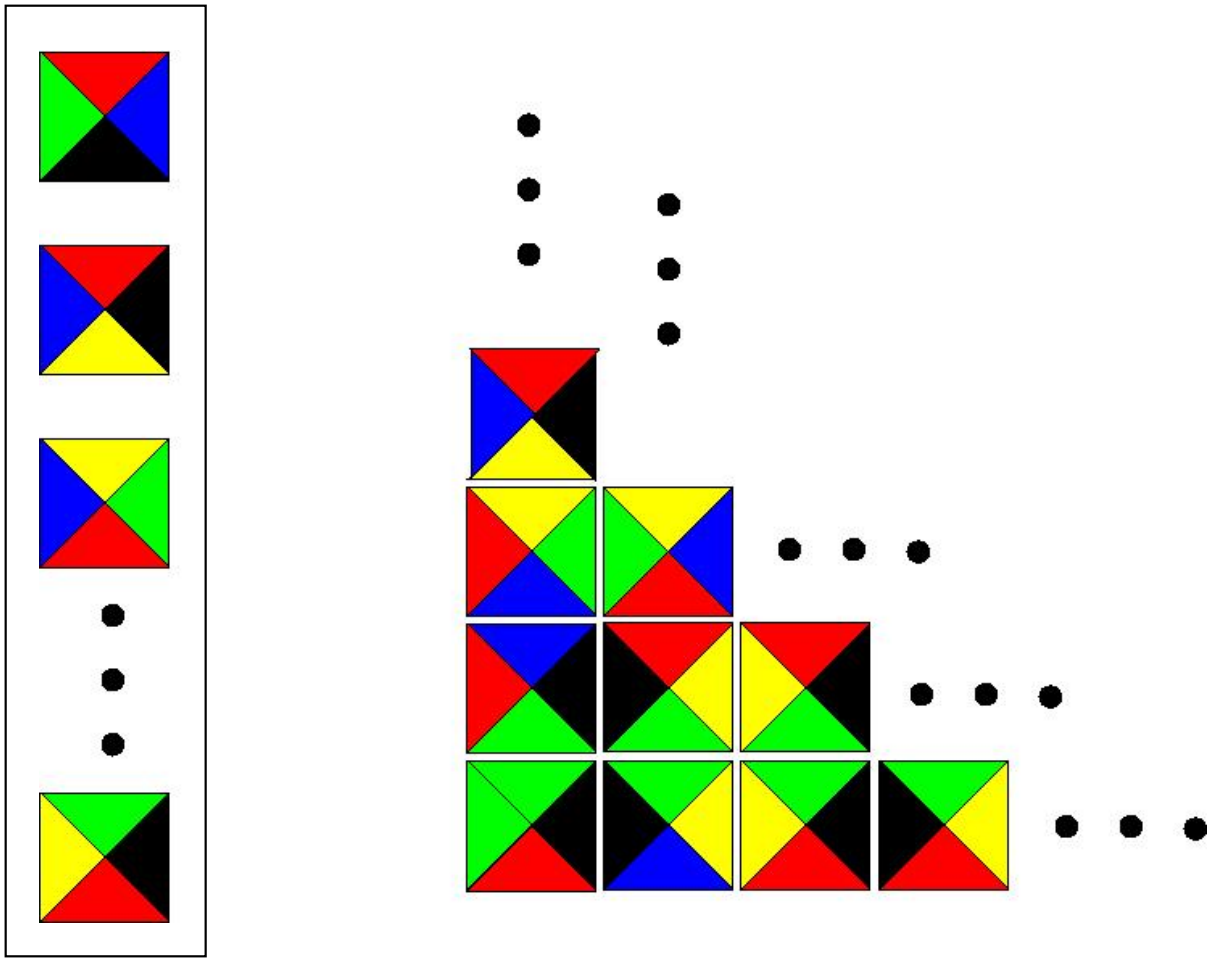
Erweiterte konjunktive Anfrage: erlaubt zusätzlich Atome $x \neq y$

Erweiterte OMQ: entsprechend

Theorem 7.20
Für gegebene ABox $\mathcal{A}$ und erweiterte $\mathcal{ALC}$-OMQ Q ist unentscheidbar, ob $\mathcal{A} \models Q$.

Beweis: Reduktion des unentscheidbaren Dominoproblems.

Domino Problem



Gegeben: endliche Menge von Domino-$Typen$.

Frage: kann man damit den ersten Quadranten der Ebene parkettieren so dass alle benachbarten Kanten dieselbe Farbe haben?

Domino Problem

Definition 7.21 [Domino System]
Domino System ist Paar $\mathcal{D} = (F, T)$ mit

- F endliche Menge von *Farben*;
- T endliche Menge von 4-Tupeln $t = (t_o, t_u, t_\ell, t_r) \in F^4$, den *Domino Typen*

Abbildung $\pi : \mathbb{N} \times \mathbb{N} \to T$ ist *Lösung* für $\mathcal{D}$ wenn

- $\pi(i,j)_r = \pi(i+1,j)_\ell$ für alle $i, j \geq 0$;
- $\pi(i,j)_o = \pi(i,j+1)_u$ für alle $i, j \geq 0$;.

Theorem 7.22
Es ist unentscheidbar ob ein gegebenes Dominosystem eine Lösung hat.

Die Reduktion

Ziel:

Gegeben Domino System $\mathcal{D}$, konstruiere erweiterte $\mathcal{ALC}$-OMQ $Q = (q, \mathcal{T})$ so dass $\mathcal{D}$ Lösung hat gdw $\emptyset \not\models Q$.

|

leere ABox

Intuition:

Modelle $\mathcal{I}$ von $\mathcal{T}$ so dass $q \not\to \mathcal{I}$ $\approx$ Lösungen für $\mathcal{D}$.

"q-Gegenmodelle"

Signatur von $\mathcal{T}$ und q:

- Rollennamen v, h für vertikale/horizontale Nachbarschaft
- Konzeptname A_t für jedes $t \in T$.

Die Reduktion

1. Jeder Punkt hat horizontalen und vertikalen Nachfolger:

$$\top \sqsubseteq \exists \boldsymbol{h}.\top \sqcap \exists \boldsymbol{v}.\top$$

2. Jeder Punkt hat genau einen Typ:

$$\top \sqsubseteq \bigsqcup_{t \in \boldsymbol{T}} \left(\boldsymbol{A_t} \sqcap \bigsqcap_{t' \in \boldsymbol{T}, t \neq t'} \neg \boldsymbol{A_{t'}} \right)$$

3. Benachbarte Kanten haben dieselbe Farbe:

$$\top \sqsubseteq \bigsqcap_{t \in \boldsymbol{T}} \left(\boldsymbol{A_t} \rightarrow \forall \boldsymbol{h}. \bigsqcup_{t' \in \boldsymbol{T} \text{ und } t_r = t'_\ell} \boldsymbol{A_{t'}} \right)$$

$$\top \sqsubseteq \bigsqcap_{t \in \boldsymbol{T}} \left(\boldsymbol{A_t} \rightarrow \forall \boldsymbol{v}. \bigsqcup_{t' \in \boldsymbol{T} \text{ und } t_o = t'_u} \boldsymbol{A_{t'}} \right)$$

Die Reduktion

Was noch fehlt:

$(*)$ jeder hv-Nachfolger ist auch vh-Nachfolger.

Wir erreichen das mittels q:

$$\exists y_0 \exists y_1 \exists y_2 \exists y_3 \exists y_2' \, h(y_0, y_1) \wedge v(y_1, y_2) \wedge v(y_0, y_3) \wedge h(y_3, y_2') \wedge y_2 \neq y_2'$$

Lemma 7.23
$\mathcal{D}$ hat Lösung gdw. $\emptyset \not\models Q$.

Beachte: für Korrektheit müssen h und v nicht unbedingt Funktionen sein

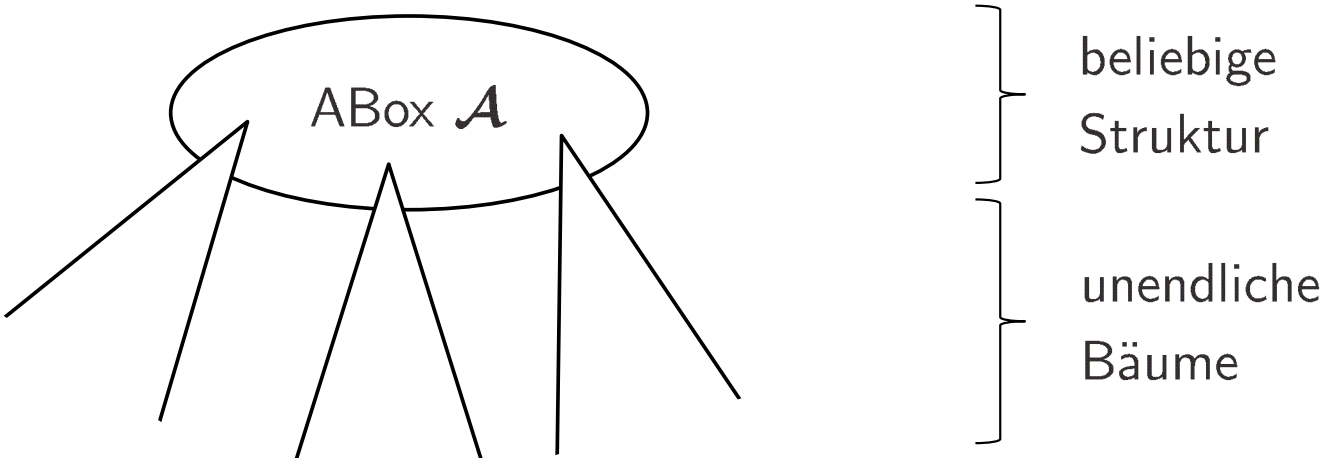
Theorem 7.20 folgt unmittelbar.

Nachbemerkung

Wie konnte so plötzlich aus entscheidbarem Problem unentscheidbares werden?

Auch für das Beantworten (nicht-erweiterter) konjunktiver Anfragen gibt es eine Art Baummodelleigenschaft.

Wenn $\mathcal{A} \not\models Q$, dann gibt es immer ein Gegenmodell der Form



Die Konstruktion des Gitters zeigt:

Mit Ungleichheit in der Anfrage gibt es keine baumförmigen Gegenmodelle mehr!

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 TBoxen, Modulextraktion
- Erklärungen von Inferenzen (Subsumtionen, Unerfüllbarkeit)
- Erweiterungen von DLs: z. B. temporale DLs, probabilistische DLs
- Entwurfsmethodologien für TBoxen
- Anwendungen wie das Semantische Web

OWL

Vom W3C (World Wide Web Committee) standardisierte Ontologiesprache:

- soll im semantischen Web Verwendung finden
- basiert auf Beschreibungslogik
- hat XML-Syntax, kann in RDFS (Resource Description Framework) übersetzt werden:

Zwei Versionen von OWL:

- OWL 1.0 (W3C-Standard 2004)
- OWL 2.0 (W3C-Standard 2009)

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

OWL 2.0: OWL 1.0 plus

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

Erfüllbarkeit ist 2NExpTime-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)

Danke für's Teilnehmen!

Es folgt:

- Mini-Übung zu ABoxen und Anfragebeantwortung
- kurze Auswertung der Evaluationsergebnisse