Struktur Vorlesung

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

Kapitel 6

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)

Patient, Medikament, etc können in TBox definiert sein.

Ziel des Kapitels

Ziel des Kapitels:

- Einführen eines Formalismus für Instanzdaten (ABox)
- Auswahl Schlußfolgerungsprobleme, um mit Instanzdaten zu arbeiten (insb. verschiedene Varianten von Anfragebeantwortung)
- Algorithmen entwickeln / Komplexität analysieren.

Kapitel 6

ABoxen und Anfragebeantwortung

Grundlagen

ABox - Syntax

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

Diese entsprechen Konstanten im Sinne der Prädikatenlogik

Definition 6.1 (ABox Syntax)
Eine

- *Konzeptassertion* hat die Form $\boldsymbol{C(a)}$
- *Rollenassertion* hat die Form $\boldsymbol{r(a, b)}$

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

T6.1

ABox – Semantik

Definition 6.1 (ABox Semantik)
Interpretation $\mathcal{I}$

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

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

T6.1 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*
(und dasselbe gilt natürlich auch für TBoxen)

Wissensbasis

Definition 6.2 (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 Anwendungen haben $\mathcal{T}$ und $\mathcal{A}$ verschiedenen Status:

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

ABoxen und Anfragebeantwortung

Schlussfolgerungsprobleme

Grundlegende Schlussfolgerungsprobleme

Definition 6.3 (Konsistenz, Instanz)
Sei $\mathcal{K}$ Wissensbasis, $C(a)$ Konzeptassertion. Dann ist

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

T6.2

Grundlegende Schlussfolgerungsprobleme

Konsistenzproblem:

gegeben $\mathcal{K}$, entscheide ob $\mathcal{K}$ konsistent;

Instanzproblem:

gegeben $\mathcal{K}$ und $C(a)$, entscheide ob $\mathcal{K} \models C(a)$;

Reduktionen

Konsistenz- und (Nicht-)Instanzproblem wechselseitig polynomiell reduzierbar.

Lemma 6.4.

- $\mathcal{K}$ konsistent gdw. $\mathcal{K} \not\models \bot(a)$
- $(\mathcal{T}, \mathcal{A}) \models C(a)$ gdw. $(\mathcal{T}, \mathcal{A} \cup \{\neg C(a)\})$ inkonsistent.

T6.3

Erfüllbarkeit (von Konzepten) polynomiell reduzierbar auf Konsistenz.

Lemma 6.5.

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

Intuitiv:
Erfüllbarkeit entspricht genau KB Konsistenz mit ABoxen der Form $\{C(a)\}$

Anfragebeantwortung

Viele Anwendungen verwenden ABoxen wie (semantische) Datenbanken

Verschiedene Anfragesprachen möglich:

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

Berechnungsproblem, kein Entscheidungsproblem.

T6.2cont

ABoxen vs Relationale Datenbanken

Relationale Datenbanken, zur Erinnerung:

- *relationales Schema* ist Menge von Tabellennamen mit Stelligkeit
- konkrete *Datenbankinstanz* füllt die Tabellen mit Inhalten
- Anfragen sind in SQL formuliert

Einfache ABox: in jeder Konzeptassertion $C(a)$ ist C ein Konzept*name*.

Im Prinzip mögliche Betrachtungsweise:

- einfache ABox kann als Datenbankinstanz dargestellt werden
- Instanzanfrage kann (meist) als SQL-Anfrage dargestellt werden T6.4

Es gibt jedoch einen drastischen Unterschied in der Semantik!

ABoxen vs Relationale Datenbanken

Relationale Datenbanken machen die Closed World Assumption (CWA):

was nicht explizit als wahr angegeben ist, ist falsch.

Für ABoxen machen wir die Open World Assumption (OWA):

es sind nur solche Dinge falsch, die explizit als falsch angegeben sind

T6.4cont

Beachte:
die OWA ist notwendig, damit die Verwendung einer TBox Sinn macht!

T6.4cont

ABoxen vs Datenbanken

Der OWA/CWA Unterschied hat sehr 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

ABoxen und Anfragebeantwortung

Instanzanfragen

Übersicht

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

- berechenbar durch mehrfaches Entscheiden des Instanzproblems: überprüfe ob $\mathcal{K} \models C(a)$ für all Individuen a in $\mathcal{A}$
- Instanzproblem kann auf Konsistenzproblem reduziert werden

Wir konzentrieren uns also auf das Entscheiden von Konsistenz
(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. $\mathcal{T}$

Vervollständigung

Vervollständigung (Precompletion):

Turing-Reduktion der Konsistenz von Wissensbasen $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ auf die Erfüllbarkeit gewisser Konzepte bzgl. $\mathcal{T}$

Notation: $\mathsf{Ind}(\mathcal{A})$ ist Menge aller Individuen in $\mathcal{A}$

Grundideen:

- Zerlege ABox in Teile: ein Teil pro Individuum $a \in \mathsf{Ind}(\mathcal{A})$.
- Für jedes $a \in \mathsf{Ind}(\mathcal{A})$, konstruiere Konzept C_a und teste ob C_a erfüllbar bzgl. $\mathcal{T}$
- Aus den gesammelten Modellen kann man Modell von $\mathcal{K}$ konstruieren.
- Grundidee: $C_a = \bigsqcap_{D(a) \in \mathcal{A}} D$ (funktioniert so aber noch nicht) T6.5

Vervollständigung

Bezüglich der Eingabe $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ nehmen wir an:

- Alle Konzepte in $\mathcal{A}$ sind in NNF
- $\mathcal{T}$ hat die Form $\{\top \sqsubseteq C_{\mathcal{T}}\}$ mit $C_{\mathcal{T}}$ in NNF

Definition 6.6. (Teilkonzepte von ABox und Wissensbasis)
Für alle ABoxen $\mathcal{A}$ und Wissensbasen $\mathcal{K} = (\mathcal{T}, \mathcal{A})$:

- $\mathsf{sub}(\mathcal{A}) = \bigcup_{C(a) \in \mathcal{A}} \mathsf{sub}(C)$
- $\mathsf{sub}(\mathcal{K}) = \mathsf{sub}(\mathcal{T}) \cup \mathsf{sub}(\mathcal{A})$

Vervollständigung

Definition 6.7. (Vervollständigung)
Vervollständigung von $\mathcal{ALC}$-Wissensbasis $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ ist Wissensbasis $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$ so dass:

1. $\mathcal{A}'$ erweitert $\mathcal{A}$ um Assertionen der Form $C(a)$ mit $C \in \text{sub}(\mathcal{K})$ und $a \in \text{Ind}(\mathcal{A})$
2. $C_{\mathcal{T}}(a) \in \mathcal{A}'$ für alle $a \in \text{Ind}(\mathcal{A})$
3. wenn $C \sqcap D(a) \in \mathcal{A}'$, dann $C(a) \in \mathcal{A}'$ und $D(a) \in \mathcal{A}'$
4. wenn $C \sqcup D(a) \in \mathcal{A}'$, dann $C(a) \in \mathcal{A}'$ oder $D(a) \in \mathcal{A}'$
5. wenn $\forall r.C(a) \in \mathcal{A}'$ und $r(a, b) \in \mathcal{A}'$, dann $C(b) \in \mathcal{A}'$

T6.5 cont

Vervollständigung

Lemma 6.8.
$\mathcal{K}$ konsistent gdw. es gibt Vervollständigung $\mathcal{K}' = (\mathcal{T}, \mathcal{A}')$ von $\mathcal{K}$ so dass
$C_a := \bigsqcap_{C(a) \in \mathcal{A}'} C$ erfüllbar bzgl. $\mathcal{T}$ für alle $a \in \mathsf{Ind}(\mathcal{A}')$. T6.7

Er gibt offensichtlich nur endlich viele Vervollständigungen

Theorem 6.9.
ABox Konsistenz in $\mathcal{ALC}$ ist entscheidbar. T6.8

Wir haben damit auch einen Algorithmus für Anfragebeantwortung.

Komplexität

Mit wenigen Ausnahmen:

Konsistenzproblem hat selbe Komplexität wie Erfüllbarkeit

Für $\mathcal{ALC}$ also zu erwarten:

- ExpTime-vollständig mit TBoxen;
- PSpace-vollständig ohne TBoxen.

Untere Schranken: Kapitel 5 + Lemma 6.5

Obere Schranken: im folgenden aus Kapitel 5 + Lemma 6.8

Resultat

Definition 6.10.
Größe $|\mathcal{A}|$ einer ABox $\mathcal{A}$ ist

$$\sum_{C(a)\in\mathcal{A}} |C| + 3 \quad + \quad 6 \cdot |\{r(a,b) \in \mathcal{A}\}|$$

Größe $|\mathcal{K}|$ einer WB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ ist $|\mathcal{T}| + |\mathcal{A}|$.

Lemma 6.11.
Für jede WB $\mathcal{K}$ gibt es höchstens 2^{n^2} Vervollständigungen, $n = |\mathcal{K}|$.

T6.9

Theorem 6.12.
Das Konsistenzproblem in $\mathcal{ALC}$ ist

1. ExpTime-vollständig mit generellen TBoxen;
2. PSpace-vollständig ohne TBoxen.

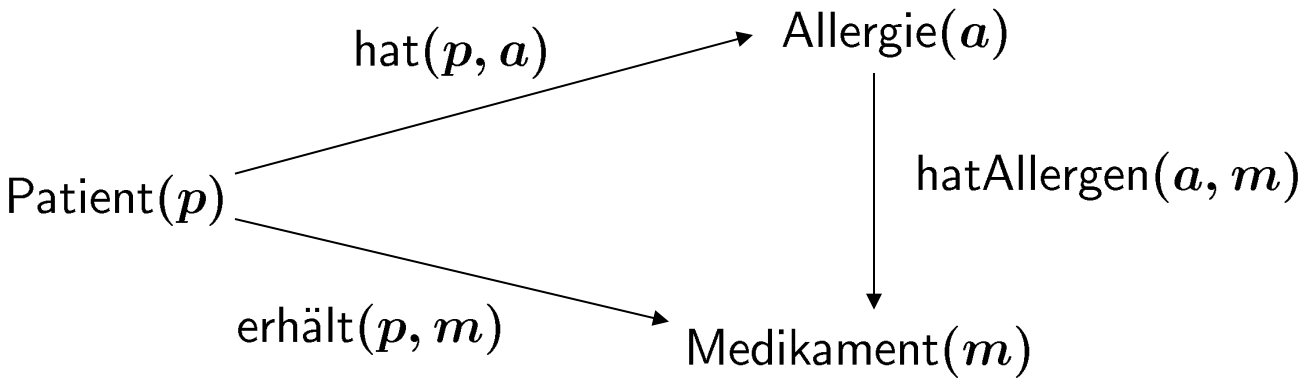
T6.10

ABoxen und Anfragebeantwortung

Konjunktive Anfragen

Instanzanfragen und ABoxen

ABoxen können komplexe relationale Strukturen beschreiben:



Konzept kann solche Struktur (mit p als Wurzel) nicht "abfragen"

(denn die Struktur ist nicht baumförmig, siehe Kapitel 3 + Unravelling)

Konjunktive Anfragen

Konjunktive Anfragen

- generalisieren Instanzanfragen in natürlicher Weise;
- erlauben Anfragen bzgl. der *relationalen Struktur* von ABoxen;
- entsprechen einem wichtigen Fragment von SQL;
- sind in der Datenbankwelt sehr weit verbreitet.

Konjunktive Anfrage - Syntax

Von nun an sei $\mathbf{V}$ eine Menge von *Variablen*.

Definition 6.13. (Konjunktive Anfrage)
Ein

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

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

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

T6.11

Konjunktive Anfrage - 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 $\overline{x}$

Konjunktive Anfrage - Semantik

Definition 6.14. (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 $\tau : \overline{x} \cup \overline{y} \to \Delta^{\mathcal{I}}$ ist *Homomorphismus* von $q(\overline{x})$ nach $\mathcal{I}$ wenn

- $\tau(x) \in A^{\mathcal{I}}$ für alle Konzeptatome $A(x)$ in φ;
- $(\tau(x), \tau(y)) \in r^{\mathcal{I}}$ 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 τ von $q(\overline{x})$ nach $\mathcal{I}$ gibt mit $\tau(x_i) = a_i$ für $1 \leq i \leq n$.

T6.12

Konjunktive Anfrage - Semantik

Definition 6.15. (Antwort bzgl. Wissensbasis)
Sei $q(\overline{x}) = \exists \overline{y}\, \varphi(\overline{x}, \overline{y})$ mit $\overline{x} = x_1 \cdots x_n$, $\mathcal{T}$ TBox, $\mathcal{A}$ ABox.

Antwort auf $q(\overline{x})$ *in* $\mathcal{A}$ *bzgl.* $\mathcal{T}$:
Tupel $\overline{a}$, das Antwort auf $q(\overline{x})$ ist in allen Modellen $\mathcal{I}$ von $\mathcal{T}$.

Menge aller Antworten auf $q(\overline{x})$ in $\mathcal{A}$ bzgl. $\mathcal{T}$ bezeichnen wir mit $\text{cert}_{\mathcal{T}}(q, \mathcal{A})$

Solche Antworten nennt man auch sichere Antworten (engl. certain answers)

Beantwortung konjunktiver Anfragen: gegeben $q(\overline{x})$, $\mathcal{A}$ und $\mathcal{T}$,
berechne $\text{cert}_{\mathcal{T}}(q, \mathcal{A})$

T6.13

Konjunktive Anfragen vs. Instanzanfragen

Konjunktive Anfrage $\exists \overline{y}\, \varphi(\overline{x}, \overline{y})$ kann als knoten- und kantenbeschrifteter gerichteter Graph $(V, E, \mathcal{L})$ gesehen werden:

- $V = \overline{x} \cup \overline{y}$;
- $(x, r, y) \in E$ wenn $r(x, y)$ Atom in $q(\overline{x})$ ist;
- $\mathcal{L}(x) = \{A \mid A(x)$ ist Atom in $q(\overline{x})\}$.

Intuitiv:

Instanzanfragen = baumförmige konjunktive Anfragen.

mit Wurzel einziger Antwortvariable T6.14

Konjunktive Anfragen und SQL

Datenbankwelt:

Konjunktive Anfragen entsprechen *select-project-join*-Anfragen, eine eingeschränkte aber sehr wichtige Art von SQL-Anfrage

In Datenbanksystemen

- sind >90% aller gestellten Anfragen select-project-join;
- sind die query engines speziell auf solche Anfragen hin optimiert.

Man kann zeigen, dass die Beantwortung beliebiger SQL-Anfragen bzgl. $\mathcal{ALC}$-Wissensbasen unentscheidbar ist!

Beantwortung konjunktiver Anfragen

- Reduktion auf mehrere Konsistenztests,
  wie auch bei der Beantwortung von Instanzanfragen

- Die Reduktion ist technisch aber viel aufwendiger und
  es sind exponentiell viele Konsistenztests notwendig.

- Im Fall von $\mathcal{ALC}$ erhält man trotzdem ein ExpTime Verfahren

- Leichte Erweiterungen der BL können das aber ändern,
  z.B. im Fall von $\mathcal{ALCI}$:
  - Beantworten von Instanzanfragen immernoch ExpTime-vollständig
  - Beantworten von Konjunktiven Anfragen 2ExpTime-vollständig

Konjunktive Anfragen mit Ungleichheit

Schon leichte Erweiterungen resultieren in Unentscheidbarkeit.

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

Wir betrachten der Einfachheit halber *Boolsche (erweiterte) Anfragen*
d.h. Anfragen ohne Antwortvariablen

Boolsche Anfrage q liefert für jede ABox $\mathcal{A}$ entweder

- keine Antwort
  (es gibt Modell $\mathcal{I}$ von $\mathcal{A}$ und $\mathcal{T}$, so dass kein Hom. von q nach $\mathcal{I}$ existiert)
- das leere Tupel () als einzige Antwort; wir schreiben dann $\mathcal{T}, \mathcal{A} \models q$
  (für alle Modelle $\mathcal{I}$ von $\mathcal{A}$ und $\mathcal{T}$ gilt: es gibt Hom. von q nach $\mathcal{I}$)

T6.15

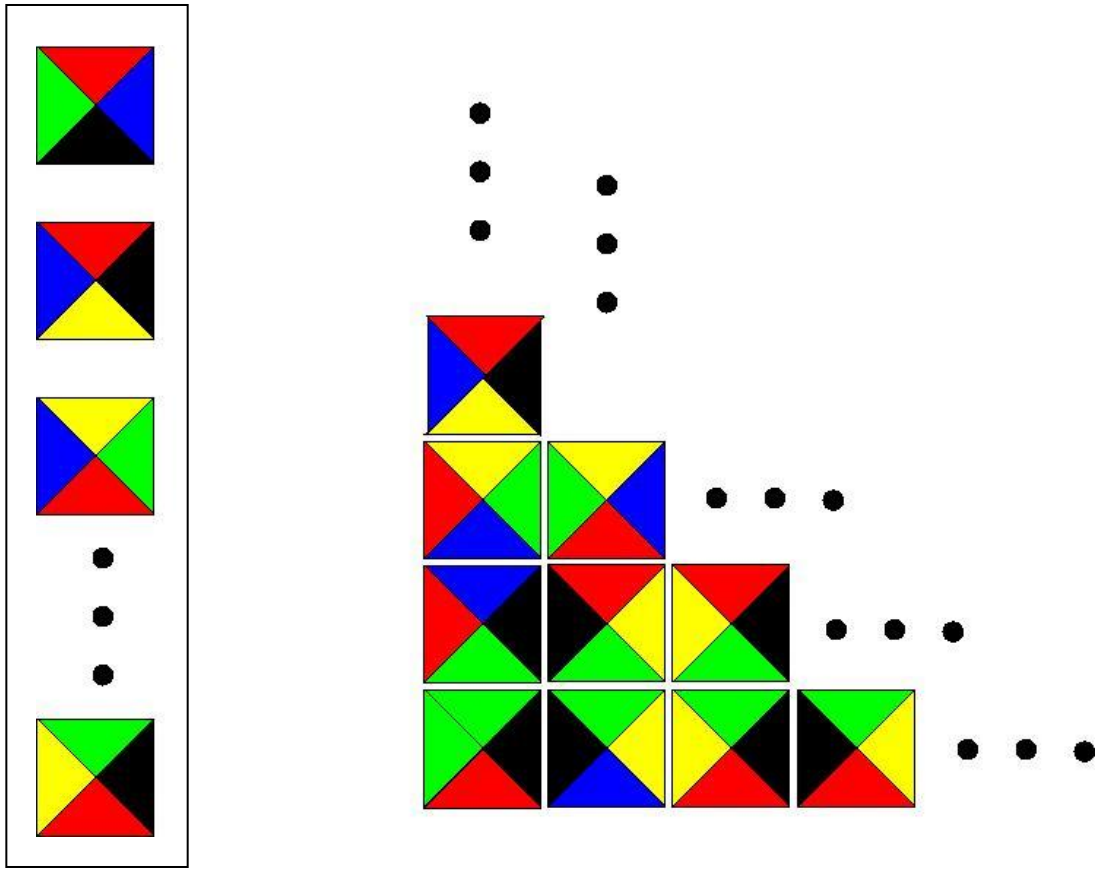
Konjunktive Anfragen mit Ungleichheit

Theorem 6.19

Für gegebene ABox $\mathcal{A}$, $\mathcal{ALC}$-TBox $\mathcal{T}$ und erweiterte konjunktive Anfrage q ist unentscheidbar, ob $\mathcal{T}, \mathcal{A} \models q$.

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 6.20 [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 $\pi : \mathbb{N} \times \mathbb{N} \longrightarrow T$ 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 6.21
Es ist unentscheidbar ob ein gegebenes Dominosystem eine Lösung hat.

Die Reduktion

Ziel:

Gegeben Domino System $\mathcal{D}$, konstruiere $\mathcal{ALC}$-TBox $\mathcal{T}_\mathcal{D}$ und erweiterte konjunktive Anfrage q, so dass $\mathcal{D}$ Lösung hat gdw $\mathcal{T}_\mathcal{D}, \emptyset \not\models q$.

leere ABox

Intuition:

Modelle $\mathcal{I}$ von $\mathcal{T}_\mathcal{D}$ so dass *kein* Hom. von q nach $\mathcal{I}$ $\approx$ Lösungen für $\mathcal{D}$.

"q-Gegenmodelle"

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

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

T6.16

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:

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

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

T6.17

Die Reduktion

Was noch fehlt:

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

Wir erreichen das mittels q:

$$\exists y_0 \exists y_1 \exists y_2 \exists y_3 \exists y_2' \, h(y_0, y_1) \wedge v(y_1, y_2) \wedge$$
$$v(y_0, y_3) \wedge h(y_3, y_2') \wedge y_2 \neq y_2' \qquad \text{T6.17 cont}$$

Lemma 6.22
$\mathcal{D}$ hat Lösung gdw. $\mathcal{T}_\mathcal{D}, \emptyset \not\models q$. T6.18

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

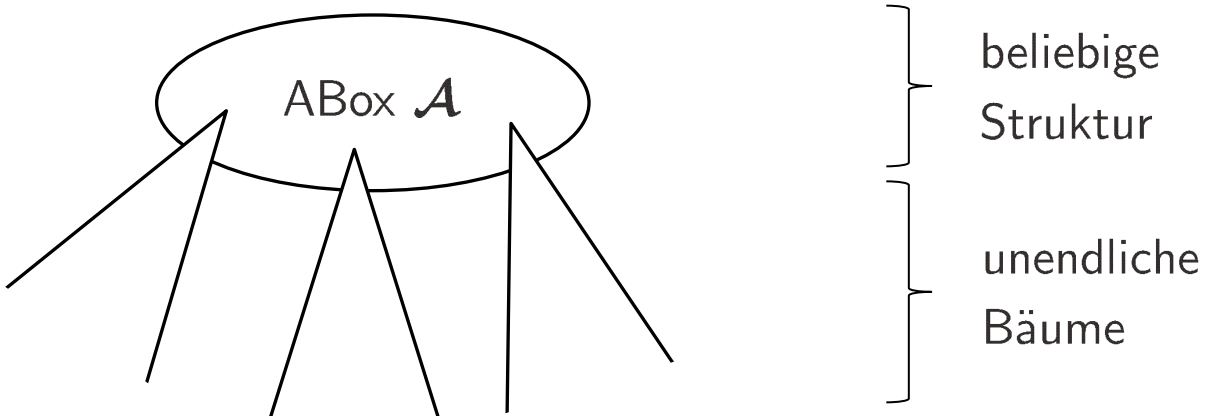
Theorem 6.19 folgt unmittelbar.

Nachbemerkung

Wie konnte so plötzlich aus entscheidbarem Problem unentscheidbares werden?

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

Wenn $\mathcal{K} \not\models q$, dann gibt es immer ein q-Gegenmodell der Form



Die Konstruktion des Gitters zeigt:

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

Zusammenfassung

- In vielen Anwendungen reichen Erfüllbarkeit und Subsumption nicht aus
- ABoxen machen aus DLs "semantische Datenbanken"
- Instanzanfragen sind genauso schwierig wie Erfüllbarkeit
- Konjunktive Anfragen sind ausdrucksstärker, aber in vielen Fällen auch schwerer zu beantworten
- Noch mehr Ausdrucksstärke führt meist zu Unentscheidbarkeit

Query Rewriting

Ziel des Abschnitts

Anfragebeantwortung in $\mathcal{ALC}$ ist recht komplex:

- ExpTime bzw. sogar 2-ExpTime sind sehr hohe Komplexitäten
- Insbesondere bei großen Datenmengen ist das problematisch
- Bei Anwendungen, in denen die *Daten im Vordergrund stehen*, möchte man echte Datenbankfunktionalität *plus Ontologie / TBox*

Natürliche Frage:

Kann man SQL-Datenbanksysteme auch mit Ontologien verwenden?

Wir werden sehen: das hängt ab von Ontologiesprache/Beschreibungslogik

Generelle Annahmen

In diesem Kapitel nehmen wir folgendes an:

- alle ABoxen sind $einfach$, also:
  in Konzeptassertionen $\boldsymbol{C(a)}$ muss $\boldsymbol{C}$ ein Konzept$name$ sein.
- Die ABox ist in der relationalen Datenbank gespeichert

Bereits gesehen: dabei wird

- jeder Konzeptname zu einer gleichnamigen Tabelle mit einer Spalte
- jeder Rollenname zu einer gleichnamigen Tabelle mit zwei Spalten

Wie können wir die TBox ins Datenbanksystem "schummeln"?

Query Rewriting

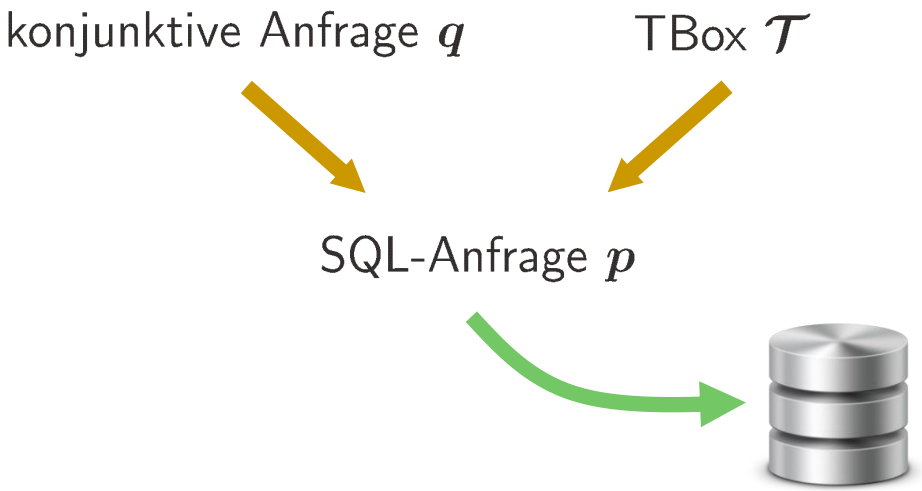
Definition 6.21 [Rewriting]

Sei $q(\overline{x})$ konjunktive Anfrage und $\mathcal{T}$ TBox. SQL-Anfrage $p(\overline{x})$ ist *SQL-Rewriting von* $\langle q, \mathcal{T} \rangle$ wenn für alle ABoxen $\mathcal{A}$ gilt:

$$\text{cert}_{\mathcal{T}}(q, \mathcal{A}) = \text{ans}(p, \mathcal{A}).$$

Antwort einer SQL-Datenbank auf Anfrage p bei Datenbank $\mathcal{A}$

Die Idee von Query Rewriting ist also:



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.

Einige beispielhafte Komplexitäten:

	Datenkompl.	Komb. Kompl.
CQ	in AC0	NP-c.
SQL	in AC0	PSpace-c.
Datalog	P-c.	ExpTime-c.

Datenkomplexität ist deutlich realistischer!

Datenkomplexität

Betrachte folgende TBox und Anfrage:

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

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

$\boldsymbol{D}$ steht für "Defekt"

Graph-ABox: verwendet keine Konzeptnamen und nur den Rollennamen $\boldsymbol{r}$ so dass 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 6.22

Für alle Graph-ABoxen $\mathcal{A}$ gilt: $\mathcal{A}$ ist nicht 3-färbbar gdw. $\mathcal{T}, \mathcal{A} \models \boldsymbol{q}$.

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

T6.19

Rewritability: zu schön, um wahr zu sein?

Man kann zeigen, dass Nicht-3-Färbbarkeit nicht in SQL ausdrückbar ist.

Es gibt aber viel einfachere Anfragen, für die das der Fall ist:

$$\mathcal{T} = \{ \;\; S \sqsubseteq M, \;\; M \sqsubseteq \forall r.M \;\; \}$$

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

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

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

Lemma 6.23
Für alle Reach-ABoxen $\mathcal{A}$ gilt:
T-Knoten erreichbar von S-Knoten gdw. $\mathcal{T}, \mathcal{A} \models q$.

T6.20

Man kann zeigen, dass Erreichbarkeit ebenfalls nicht in SQL ausdrückbar ist.

Query Rewriting

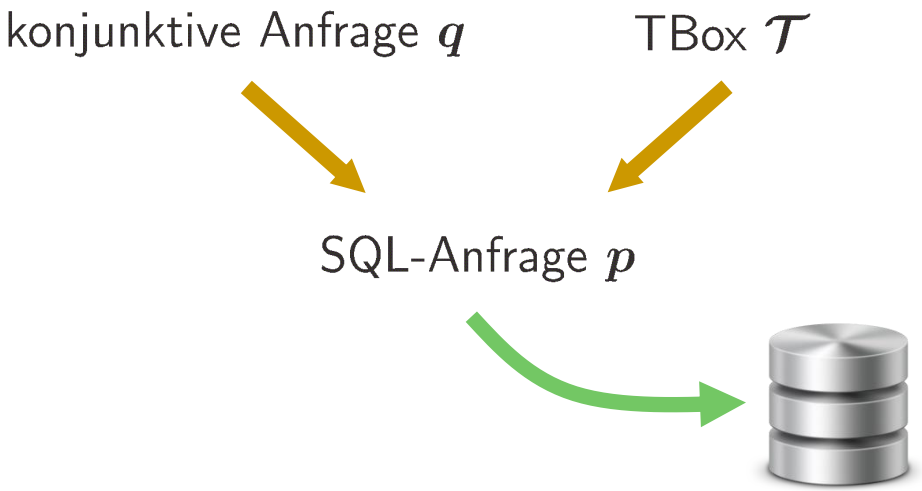
Definition 6.21 [Rewriting]

Sei $q(\overline{x})$ konjunktive Anfrage und $\mathcal{T}$ TBox. SQL-Anfrage $p(\overline{x})$ ist *SQL-Rewriting von* $\langle q, \mathcal{T} \rangle$ wenn für alle ABoxen $\mathcal{A}$ gilt:

$$\mathsf{cert}_{\mathcal{T}}(q, \mathcal{A}) = \mathsf{ans}(p, \mathcal{A}).$$

Antwort einer SQL-Datenbank auf Anfrage p bei Datenbank $\mathcal{A}$

Die Idee von Query Rewriting ist also:



DL-Lite

Definition 6.24 [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 $\boldsymbol{C_1} \sqsubseteq \boldsymbol{C_2}$
- Rolleninklusionen $\boldsymbol{r_1} \sqsubseteq \boldsymbol{r_2}$ ($\boldsymbol{r_1}, \boldsymbol{r_2}$ Rollennamen)

Wir verzichten hier der Einfachheit haber auf negative Inklusionen

$$\boldsymbol{C_1} \sqsubseteq \neg \boldsymbol{C_2} \text{ und } \boldsymbol{r_1} \sqsubseteq \neg \boldsymbol{r_2}$$

die in der ursprünglichen Definition ebenfalls zugelassen sind. T6.21

DL-Lite

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

T6.21 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 $d \in C^{\mathcal{U}_i}$, $C \sqsubseteq A \in \mathcal{T}$, $d \notin A^{\mathcal{U}_{i+1}}$, 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 $s^{\mathcal{U}_{i+1}} = s^{\mathcal{U}_i} \cup \{(d,e)\}$

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

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 6.25

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.

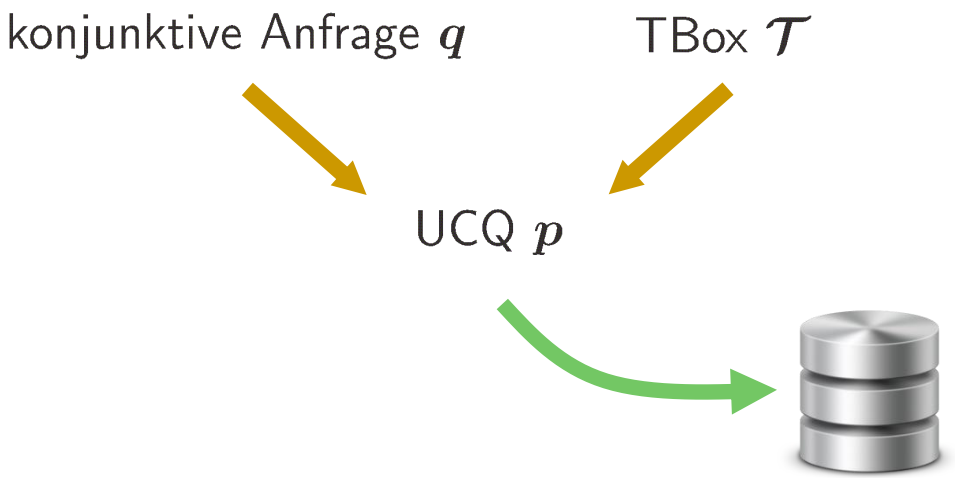
Der Einfachheit halber beschränken wir uns auf Boolsche Anfragen.

Lemma 6.26

Für jede konjunktive Anfrage q gilt: $\mathcal{T}, \mathcal{A} \models q$ gdw. $\mathcal{U} \models q$. T6.24

Wir verwenden universelle Modelle nun, um SQL-Rewritings zu konstruieren.

Diese Rewritings werden sogar Disjunktionen von CQs (UCQs) sein, also:



Rewritings

Wesentliche Einsicht: Anfragebeantwortung bzgl. DL-Lite TBoxen ist *lokal* in folgendem Sinne (vergl. Erreichbarkeits-Beispiel in $\mathcal{ALC}$):

Theorem 6.27
Wenn $\mathcal{A}, \mathcal{T} \models q$, dann gibt es ein $\mathcal{A}' \subseteq \mathcal{A}$ mit $|\mathsf{Ind}(\mathcal{A}')| \leq |q| \cdot (|\mathcal{T}| + 1)$ so dass $\mathcal{A}', \mathcal{T} \models q$.

T6.25

Es reicht also, nur kleine Ausschnitte der ABox zu betrachten

Diese Ausschnitte lassen sich mit konjunktiven Anfragen beschreiben.

Rewritings

Offensichtlich kann man jede ABox $\mathcal{B}$ als konjunktive Anfrage $q_{\mathcal{B}}$ auffassen.

Zum Beispiel:

ABox $\quad A(a),\ r(a,b),\ s(b,a),\ B(b)$

wird Anfrage $\quad \exists x\, \exists y\, A(x) \wedge r(x,y) \wedge s(y,x) \wedge B(y)$

Für konjunktive Anfrage q und DL-Lite TBox $\mathcal{T}$ sei

$$p = q_{\mathcal{B}_1} \vee \cdots \vee q_{\mathcal{B}_n}$$

wobei $\mathcal{B}_1, \ldots, \mathcal{B}_n$ alle ABoxen sind mit $\mathcal{B}_i, \mathcal{T} \models q$ und $|\mathsf{Ind}(\mathcal{B}_i)| \leq |q| \cdot (|\mathcal{T}| + 1)$

Wir betrachten hier nur ABoxen, die ausschließlich Symbole aus q und $\mathcal{T}$ enthalten

Außerdem wählen wir eine ABox pro Isomorphieklasse aus

Damit müssen wir nur endlich viele ABoxen betrachten.

Rewritings

Theorem 6.28

$p = q_{\mathcal{B}_1} \vee \cdots \vee q_{\mathcal{B}_n}$ ist ein SQL-Rewriting von $\langle q, \mathcal{T} \rangle$.

Lemma 6.28 [Vollständigkeit von p]

$\mathcal{T}, \mathcal{A} \models q$ impliziert $\mathcal{A} \models p$

T6.26

Für die Korrektheit brauchen wir eine weitere Beobachtung:

ABox Homomorphismus von $\mathcal{A}_1$ nach $\mathcal{A}_2$:

Funktion $h : \mathsf{Ind}(\mathcal{A}_1) \to \mathsf{Ind}(\mathcal{A}_2)$, die Konzept- und Rollenatome erhält.

Wenn $\mathcal{A} \models p$, dann gibt es Disjunkt $q_{\mathcal{B}_i}$ in p mit $\mathcal{B}_i \to \mathcal{A}$

Es muss aber nicht gelten $\mathcal{B}_i \subseteq \mathcal{A}$ (woraus sofort $\mathcal{T}, \mathcal{A} \models q$ folgen würde)

Rewritings

Lemma 6.29
$\mathcal{T}, \mathcal{A}_1 \models q$ und $\mathcal{A}_1 \to \mathcal{A}_2$ impliziert $\mathcal{T}, \mathcal{A}_2 \models q$.

T6.27

Damit ist die Korrektheit nun leicht zu zeigen.

Lemma 6.30 [Korrektheit von p]
$\mathcal{A} \models p$ impliziert $\mathcal{T}, \mathcal{A} \models q$.

T6.28

Final Words

- SQL-Rewritings existieren in DL-Lite immer, können aber exp groß werden
- Hauptgrund für Rewritability in DL-Lite ist *Lokalität* (Theorem 6.27)
- In anderen BLen gibt es Fälle, wo SQL-Rewritings nicht existieren

  In der Praxis gibt es sie in vielen Fällen aber doch.
- Eine andere wichtige Zielsprache für Rewritings ist Datalog