Konjunktive Anfragen II

5.1 Homomorphismus-Satz, Statische Analyse und Anfrageminimierung
5.2 Azyklische Anfragen
5.3 Mengen-Semantik vs. Multimengen-Semantik

Zur Erinnerung (1/3): Tableau-Anfragen — Beispiel

Beispiel-Anfrage:

Filmtitel + Regisseur aller z.Zt. laufenden Filme, deren Regisseur schon mal mit “Liz Taylor” zusammengearbeitet hat.

Als regelbasierte konjunktive Anfrage:

$$Ans(x_T, x_R) \leftarrow Programm(x_K, x_T, x_Z),\ Filme(x_T, x_R, x_S),\ Filme(y_T, x_R, \text{“Liz Taylor”})$$

Als Tableau-Anfrage: $(\mathbf{T},\ \langle x_T,\ x_R\rangle)$ mit folgendem Tableau $\mathbf{T}$:

Programm

Kino	Titel	Zeit
x_K	x_T	x_Z

Filme

Titel	Regie	Schauspieler
x_T	x_R	x_S
y_T	x_R	“Liz Taylor”

Zur Erinnerung (2/3): Tableau-Anfragen — Präzise

Definition 2.7

Sei $\mathbf{R}$ ein Datenbankschema und R ein Relationsschema.

- Ein Tableau über R (auch: Einzel-Tableau) ist eine endliche Menge von freien Tupeln (also Tupeln über $\mathbf{dom} \cup \mathbf{var}$) der Stelligkeit $arity(R)$.

  (D.h. ein Tableau über R ist eine “Relation vom Schema R, die als Einträge nicht nur Elemente aus **dom**, sondern auch Variablen aus **var** haben kann”.)

- Ein Tableau $\mathbf{T}$ über $\mathbf{R}$ ist eine Abbildung, die jedem $R \in \mathbf{R}$ ein Tableau über $\mathbf{R}$ zuordnet.

  (D.h. ein Tableau über $\mathbf{R}$ ist eine “Datenbank vom Schema $\mathbf{R}$, die als Einträge auch Variablen enthalten kann”.)

- Eine Tableau-Anfrage über $\mathbf{R}$ (bzw. R) ist von der Form $(\mathbf{T}, u)$, wobei $\mathbf{T}$ ein Tableau über $\mathbf{R}$ (bzw. R) und u ein freies Tupel ist, so dass jede Variable, die in u vorkommt, auch in $adom(\mathbf{T})$ vorkommt.

  u heißt Zusammenfassung der Anfrage $(\mathbf{T}, u)$.

Zur Erinnerung (3/3): Tableau-Anfragen — Semantik

Sei $Q = (\mathbf{T}, u)$ eine Tableau-Anfrage.

- $Var(Q)$ bezeichnet die Menge aller Variablen, die in u oder $\mathbf{T}$ vorkommen.
  $adom(Q)$ bezeichnet die Menge aller Konstanten, die in u oder $\mathbf{T}$ vorkommen.

- Eine Belegung für Q ist eine Abbildung $\beta : Var(Q) \to \mathbf{dom}$.

- Sei $\mathbf{I}$ eine Datenbank vom Schema $\mathbf{R}$.

  Eine Belegung β für Q heißt Einbettung von $\mathbf{T}$ in $\mathbf{I}$, falls “$\beta(\mathbf{T}) \subseteq \mathbf{I}$”, d.h. f.a. $R \in \mathbf{R}$ gilt:

  $$\beta(\mathbf{T}(R)) := \{\beta(t) : t \in \mathbf{T}(R)\} \subseteq \mathbf{I}(R).$$

- Der Tableau-Anfrage Q ordnen wir die folgende Anfragefunktion $[\![Q]\!]$ zu:

  $$[\![Q]\!](\mathbf{I}) := \{\beta(u) : \beta \text{ ist eine Einbettung von } \mathbf{T} \text{ in } \mathbf{I}\}$$

  für alle Datenbanken $\mathbf{I} \in inst(\mathbf{R})$.

Konjunktive Anfragen II

Vorbemerkung

zum Thema "Statische Analyse" und "Optimierung"

- Globale Optimierung:
  Finde zur gegebenen Anfrage eine "minimale" äquivalente Anfrage
- Statische Analyse:
  Erfüllbarkeitsproblem, Äquivalenzproblem, Query Containment Problem
  - Für Rel.Algebra sind diese drei Probleme unentscheidbar (Kapitel 4).
  - Für regelbas. konj. Anfragen ist das Erfüllbarkeitsproblem trivial (Kapitel 2).
  - Für konj. Anfragen mit "$=$" (bzw. für SPC- bzw. SPJR-Anfragen) ist das Erfüllbarkeitsproblem in poly. Zeit lösbar (Übungsblatt 1).
  - Jetzt: Algorithmen für's Query Containment Problem und für's Äquivalenzproblem für konjunktive Anfragen.

Definition 5.1 (Äquivalenz und Query Containment)

Seien P und Q zwei Anfragen einer Anfragesprache über einem DB-Schema $\mathbf{R}$.

(a) Wir schreiben $Q \equiv P$ und sagen "Q ist äquivalent zu P", falls für alle Datenbanken $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ gilt: $[\![Q]\!](\mathbf{I}) = [\![P]\!](\mathbf{I})$.

(b) Wir schreiben $Q \subseteq P$ und sagen "Q ist in P enthalten", falls für alle Datenbanken $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ gilt: $[\![Q]\!](\mathbf{I}) \subseteq [\![P]\!](\mathbf{I})$.

Zusammenhänge

- Äquivalenz vs. Containment:
  - $Q \equiv P \iff Q \subseteq P$ und $P \subseteq Q$
  - $Q \subseteq P \iff (Q \vee P) \equiv P$ ..... für Optimierung nutzen
- Containment vs. Erfüllbarkeit:
  - Q unerfüllbar $\implies Q \subseteq P$ (für alle Anfragen P)
  - $Q \subseteq P \iff (Q \wedge \neg P)$ ist unerfüllbar ..... für Optimierung nutzen
- Erfüllbarkeit vs. Äquivalenz:
  - Q unerfüllbar $\implies (Q \vee P) \equiv P$ (für alle Anfragen P) für Optimierung nutzen

Homomorphismen

Definition 5.2

Sei $\mathbf{R}$ ein Datenbankschema und
seien $Q' = (\mathbf{T}', u')$ und $Q = (\mathbf{T}, u)$ zwei Tableau-Anfragen über $\mathbf{R}$.

(a) Eine Substitiution für Q' ist eine Abbildung $\zeta : \mathit{Var}(Q') \to \mathbf{var} \cup \mathbf{dom}$.

Wie üblich setzen wir ζ auf natürliche Weise fort zu einer Abbildung von $\mathit{Var}(Q') \cup \mathbf{dom}$ nach $\mathbf{var} \cup \mathbf{dom}$, so dass $\zeta|_{\mathbf{dom}} = \mathrm{id}$.
Für ein freies Tupel $t = \langle e_1, .., e_k \rangle$ setzen wir $\zeta(t) := \langle \zeta(e_1), .., \zeta(e_k) \rangle$.
Für eine Menge M von freien Tupeln setzen wir $\zeta(M) := \{ \zeta(t) : t \in M \}$.

(b) Eine Substitution ζ für Q' heißt Homomorphismus von Q' auf Q, falls
- $\zeta(u') = u$ und
- $\zeta(\mathbf{T}') \subseteq \mathbf{T}$, d.h. für alle $R \in \mathbf{R}$ ist $\zeta(\mathbf{T}'(R)) \subseteq \mathbf{T}(R)$.

(Beachte: Dann gilt insbes., dass $\zeta(\mathit{Var}(Q')) \subseteq \mathit{Var}(Q)$)

Beispiel: $\mathbf{R} := \{R\}$, $Q' := (\mathbf{T}', \langle x, y \rangle)$, $Q := (\mathbf{T}, \langle x, y \rangle)$ mit

$\mathbf{T}'(R) :=$

A	B
x	y_1
x_1	y_1
x_1	y

$\mathbf{T}(R) :=$

A	B
x	y

Homomorphismus ζ von Q' auf Q: $\zeta : x, y, x_1, y_1 \mapsto x, y, x, y$.
Es gibt keinen Homomorphismus von Q auf Q'.

Die kanonische Datenbank $\mathbf{I}_Q^{Q'}$

("repräsentiere Var. in **T** durch Konstanten, die nicht in Q, Q' vorkommen")

Wir legen ein für alle Mal für jedes endliche $C \subseteq \mathbf{dom}$ eine injektive Abbildung $\alpha_C : \mathbf{var} \to \mathbf{dom} \setminus C$ fest. Wie üblich setzen wir α_C fort zu einer Abbildung von $\mathbf{var} \cup \mathbf{dom}$ nach $\mathbf{dom}$ mit $\alpha|_{\mathbf{dom}} = \mathrm{id}$.
Für die folgendermaßen definierte "Umkehrfunktion" $\alpha_C^{-1} : \mathbf{dom} \to \mathbf{var} \cup \mathbf{dom}$

$$\alpha_C^{-1}(a) := \begin{cases} a & \text{falls } a \notin \mathrm{Bild}(\alpha_C) \\ y & \text{falls } a = \alpha_C(y) \text{ für } y \in \mathbf{var} \end{cases}$$

gilt für alle $b \in \mathbf{var} \cup C$, dass $\alpha_C^{-1}(\alpha_C(b)) = b$.

Definition 5.3 *(repräsentiere $Q = (\mathbf{T}, u)$ durch eine Datenbank $\mathbf{I}_Q^{Q'}$ und ein Tupel $u_Q^{Q'}$)*

$Q' = (\mathbf{T}', u')$ und $Q = (\mathbf{T}, u)$ seien Tableau-Anfragen über einem DB-Schema **R**.
Die kanonische Datenbank $\mathbf{I}_Q^{Q'} \in \mathit{inst}(\mathbf{R})$ und das kanonische Tupel $u_Q^{Q'}$ sind folgendermaßen definiert: Für $C := \mathit{adom}(Q) \cup \mathit{adom}(Q')$ ist
$u_Q^{Q'} := \alpha_C(u)$ und $\mathbf{I}_Q^{Q'} := \alpha_C(\mathbf{T})$, d.h. $\mathbf{I}_Q^{Q'}(R) = \alpha_C(\mathbf{T}(R))$, für alle $R \in \mathbf{R}$.

Proposition 5.4

*$Q' = (\mathbf{T}', u')$ und $Q = (\mathbf{T}, u)$ seien Tableau-Anfragen über einem DB-Schema **R**.*
Dann gilt: Es gibt einen Homomorphismus von Q' auf Q $\iff$ $u_Q^{Q'} \in [\![Q']\!](\mathbf{I}_Q^{Q'})$

Beweis: Siehe Tafel.

Der Homomorphismus-Satz

Theorem 5.5 (Chandra, Merlin, 1977)

*Sei **R** ein Datenbankschema und*
*seien $Q' = (\mathbf{T}', u')$ und $Q = (\mathbf{T}, u)$ zwei Tableau-Anfragen über **R**. Dann gilt:*

$Q \subseteq Q'$ $\iff$ *es gibt einen Homomorphismus von Q' auf Q* $\iff$ $u_Q^{Q'} \in [\![Q']\!](\mathbf{I}_Q^{Q'})$.

Beweis: Siehe Tafel.

Korollar 5.6

Das

> Query Containment Problem für Tableau-Anfragen
> *Eingabe:* Tableau-Anfragen Q und Q' über einem DB-Schema **R**
> *Frage:* Ist $Q \subseteq Q'$ (d.h. gilt für alle $\mathbf{I} \in \mathit{inst}(\mathbf{R})$, dass $[\![Q]\!](\mathbf{I}) \subseteq [\![Q']\!](\mathbf{I})$) ?

ist NP*-vollständig.*

Beweis: Siehe Tafel.

Bemerkung: Wegen $(Q \equiv Q' \iff (Q \subseteq Q' \text{ und } Q' \subseteq Q))$ gibt es daher insbes. einen Algorithmus, der bei Eingabe zweier Tableau-Anfragen Q und Q' entscheidet, ob die beiden Anfragen äquivalent sind.

Tableau-Minimierung (1/2)

Definition 5.7

Sei **R** ein Datenbankschema.

(a) Eine Tableau-Anfrage $(\mathbf{T}, u)$ heißt minimal, falls es keine zu $(\mathbf{T}, u)$ äquivalente Tableau-Anfrage $(\mathbf{T}', u')$ gibt mit $|\mathbf{T}'| < |\mathbf{T}|$ (wobei $|\mathbf{T}|$ die Kardinalität, d.h. die Gesamtzahl von Tupeln in **T** bezeichnet).

(b) Zwei Tableau-Anfragen $Q' := (\mathbf{T}', u')$ und $Q := (\mathbf{T}, u)$ heißen isomorph, falls es eine Bijektion $\zeta : \mathit{Var}(Q') \to \mathit{Var}(Q)$ gibt, so dass $\zeta(\mathbf{T}') = \mathbf{T}$ und $\zeta(u') = u$.

Theorem 5.8 (Chandra, Merlin, 1977)

(a) *Zu jeder Tableau-Anfrage $(\mathbf{T}, u)$ gibt es ein $\mathbf{T}' \subseteq \mathbf{T}$ (d.h. für jedes $R \in \mathbf{R}$ ist $\mathbf{T}'(R) \subseteq \mathbf{T}(R)$), so dass die Anfrage $(\mathbf{T}', u)$ minimal und äquivalent zu $(\mathbf{T}, u)$ ist.*

(b) *Sind $(\mathbf{T}, u)$ und $(\mathbf{S}, v)$ zwei minimale äquivalente Tableau-Anfragen, so sind $(\mathbf{T}, u)$ und $(\mathbf{S}, v)$ isomorph.*

Beweis: (a): siehe Tafel; (b): Übung.

Tableau-Minimierung (2/2)

Korollar 5.9

(a) *Es gibt einen Algorithmus, der bei Eingabe einer Tableau-Anfrage $Q = (\mathbf{T}, u)$ eine minimale zu Q äquivalente Tableau-Anfrage $(\mathbf{T}', u)$ (mit $\mathbf{T}' \subseteq \mathbf{T}$) berechnet.*

(b) *Das Problem*

> *Eingabe:* Tableau-Anfrage $(\mathbf{T}, u)$ und Tableau $\mathbf{T}' \subseteq \mathbf{T}$
> *Frage:* Ist $(\mathbf{T}, u) \equiv (\mathbf{T}', u)$?

ist NP*-vollständig.*

Beweis: (a): siehe Tafel; (b): Übung.

Verbesserte Optimierung von SPJR-Anfragen

Vorgehensweise bei Eingabe einer SPJR-Anfrage Q:

(1) Übersetze Q in eine Tableau-Anfrage $(\mathbf{T}, u)$ (bzw. gib "$\emptyset$" aus, falls Q nicht erfüllbar ist)

(2) Finde minimales Tableau $\mathbf{T}' \subseteq \mathbf{T}$ so dass $(\mathbf{T}', u)$ äquivalent zu $(\mathbf{T}, u)$ ist.

(3) Übersetze $(\mathbf{T}', u)$ in eine äquivalente SPJR-Anfrage Q'

(4) Wende heuristische Optimierung (siehe Kapitel 3.2) auf Q' an und werte Q' aus.

Minimierung des Tableaus $\widehat{=}$ Minimierung der Anzahl der Joins,
denn: Anzahl Zeilen im Tableau = 1 + Anzahl Join-Operationen bei der Auswertung

Beispiel 5.10

- $\mathbf{R} = \{R\}$, wobei R die Attribute A, B, C hat.
- $Q := \pi_{A,B}\big(\sigma_{B=\text{"5"}}(R)\big) \bowtie \pi_{B,C}\Big(\pi_{A,B}(R) \bowtie \pi_{A,C}(\sigma_{B=\text{"5"}}(R))\Big)$
- zugehörige Tableau-Anfrage: $(\mathbf{T}, u)$ mit $u = \langle x_A, \text{"5"}, z_C\rangle$ und $\mathbf{T}(R) = \big\{\ \langle x_A, \text{"5"}, x_C\rangle,\ \langle y_A, \text{"5"}, y_C\rangle,\ \langle y_A, \text{"5"}, z_C\rangle\ \big\}$
- minimales Tableau $\mathbf{T}'$: $\mathbf{T}'(R) = \big\{\ \langle x_A, \text{"5"}, x_C\rangle,\ \langle y_A, \text{"5"}, z_C\rangle\ \big\}$
- zugehörige SPJR-Anfrage: $Q' := \pi_{A,B}\big(\sigma_{B=\text{"5"}}(R)\big) \bowtie \pi_{B,C}\big(\sigma_{B=\text{"5"}}(R)\big)$

Konjunktive Anfragen II

Motivation

- Ziel jetzt: Teilklasse der Klasse der konjunktiven Anfragen, für die das Auswertungsproblem in Polynomialzeit lösbar ist (kombinierte Komplexität)
- $\rightsquigarrow$ azyklische konjunktive Anfragen
- Wir werden in diesem Kapitel oft nur Boolesche Anfragen betrachten (. . . wenn wir die effizient auswerten können, dann können wir die Konstruktion aus dem Beweis von Theorem 2.20 benutzen, um auch Anfragen auszuwerten, deren Ergebnis die Stelligkeit $\geqslant 1$ hat)
- In der Literatur: Verschiedene äquivalente Definitionen (bzw. Charakterisierungen) der azyklischen Booleschen konjunktiven Anfragen, etwa:
  - regelbasierte konjunktive Anfragen mit azyklischem Hypergraph
  - regelbasierte konjunktive Anfragen der Hyperbaum-Weite 1
  - regelbasierte konjunktive Anfragen, die einen Join-Baum besitzen
  - Boolesche Semijoin-Anfragen
  - konjunktive Sätze des Guarded Fragment

Beispiel

Beispiel-Datenbank mit Relationen

- T mit Attributen *Student*, *Kurs*, *Semester* ..... T steht für "Teilnehmer"
- D mit Attributen *Prof*, *Kurs*, *Semester* ..... D steht für "Dozent"
- E mit Attributen *Person1*, *Person2* ..... steht für "Person1 ist Elternteil von Person2"

Anfrage 1: Gibt es einen Dozenten, dessen Tochter/Sohn an irgendeinem Kurs teilnimmt?

Als regelbasierte konjunktive Anfrage $Q_1 :=$

$$Ans() \leftarrow E(x_P, x_S),\ D(x_P, x_K, x_Z),\ T(x_S, y_K, y_Z)$$

Auswertung als "Semijoin-Anfrage": $\pi_{\langle\rangle}\Big(\big(E(x_P, x_S) \ltimes D(x_P, x_K, x_Z)\big) \ltimes T(x_S, y_K, y_Z)\Big)$

Anfrage 2: Gibt es einen Studenten, der an einem Kurs teilnimmt, der von seinem/r Vater/Mutter veranstaltet wird?

Als regelbasierte konjunktive Anfrage $Q_2 :=$

$$Ans() \leftarrow E(x_P, x_S),\ D(x_P, x_K, x_Z),\ T(x_S, x_K, x_Z)$$

Auswertung: $\pi_{\langle\rangle}\Big(E(x_P, x_S) \bowtie D(x_P, x_K, x_Z) \bowtie T(x_S, x_K, x_Z)\Big)$

Auswertung durch eine "Semijoin-Anfrage" ist nicht möglich.

Join-Bäume & Azyklische regelbasierte konj. Anfragen

Definition 5.11

(a) Sei $Q := Ans(u) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$ eine regelbasierte konjunktive Anfrage. Ein Join-Baum von Q ist ein Baum mit Knotenmenge $\{R_1(u_1), \ldots, R_\ell(u_\ell)\}$, so dass für alle Knoten $R_i(u_i)$ und $R_j(u_j)$ die folgende "Weg-Eigenschaft" gilt:

> Jede Variable x, die sowohl in u_i als auch in u_j vorkommt, kommt in jedem Knoten vor, der auf dem (eindeutig bestimmten) Weg zwischen $R_i(u_i)$ und $R_j(u_j)$ liegt.

(b) Eine regelbasierte konjunktive Anfrage Q (beliebiger Stelligkeit) heißt azyklisch, falls es einen Join-Baum für Q gibt.

Beispiele:

Join-Baum für $Q_1 := Ans() \leftarrow E(x_P, x_S),\ D(x_P, x_K, x_Z),\ T(x_S, y_K, y_Z)$

Es gibt keinen Join-Baum für $Q_2 := Ans() \leftarrow E(x_P, x_S),\ D(x_P, x_K, x_Z),\ T(x_S, x_K, x_Z)$

Join-Baum für
$Q_3 := Ans() \leftarrow R(y,z),\ P(x,y),\ S(y,z,u),\ S(z,u,w),\ T(y,z),\ T(z,u),\ R(z',y')$

Effiziente Auswertung von azyklischen Booleschen konjunktiven Anfragen

Vorgehensweise:

Eingabe: Boolesche regelbasierte konjunktive Anfrage Q, Datenbank $\mathbf{I}$

Ziel: Berechne $[\![Q]\!](\mathbf{I})$

(1) Teste, ob Q azyklisch ist und konstruiere ggf. einen Join-Baum T für Q.
(Details dazu: später)

(2) Nutze T zur Konstruktion einer Booleschen Semijoin-Anfrage Q', die äquivalent zu Q ist.
(Details dazu: gleich)

(3) Werte Q' in $\mathbf{I}$ aus
(das geht gemäß Proposition 5.13 in Zeit $\mathcal{O}(||Q'||^2 \cdot ||\mathbf{I}|| \cdot \log(||Q'|| \cdot ||\mathbf{I}||))$)

Semijoin-Anfragen

Definition 5.12

Sei $\mathbf{R}$ ein Datenbankschema. Die Klasse der Semijoin-Anfragen über $\mathbf{R}$ ist induktiv wie folgt definiert:

(A) Jedes Relations-Atom $R(v_1, \ldots, v_r)$, für $R \in \mathbf{R}$, $r := arity(R)$ und $v_1, \ldots, v_r \in \mathbf{var} \cup \mathbf{dom}$ ist eine Semijoin-Anfrage der Sorte $(v_1, \ldots, v_r)$.

Semantik: Für jede Datenbank $\mathbf{I} \in inst(\mathbf{R})$ ist $[\![R(v_1, \ldots, v_r)]\!](\mathbf{I}) :=$

$$\left\{ \langle \beta(v_1), \ldots, \beta(v_r) \rangle \ : \ \begin{array}{l} \beta : (\{v_1, \ldots, v_r\} \cap \mathbf{var}) \to \mathbf{dom} \text{ ist eine Belegung} \\ \text{so dass } \langle \beta(v_1), \ldots, \beta(v_r) \rangle \in \mathbf{I}(R) \end{array} \right\}$$

(S) Sind Q_1 und Q_2 Semijoin-Anfragen der Sorten $(v_1, \ldots, v_r)$ und $(v'_1, \ldots, v'_s)$, so ist $Q := (Q_1 \ltimes Q_2)$ eine Semijoin-Anfrage der Sorte $(v_1, \ldots, v_r)$.

Semantik: Für jede Datenbank $\mathbf{I} \in inst(\mathbf{R})$ ist $[\![Q]\!](\mathbf{I}) :=$

$$\left\{ \langle a_1, \ldots, a_r \rangle \in [\![Q_1]\!](\mathbf{I}) \ : \ \begin{array}{l} \text{es gibt ein } \langle b_1, \ldots, b_s \rangle \in [\![Q_2]\!](\mathbf{I}), \text{ so dass} \\ \text{für alle } i,j \text{ mit } v_i = v'_j \in \mathbf{var} \text{ gilt: } a_i = b_j \end{array} \right\}$$

Eine Boolesche Semijoin-Anfrage über $\mathbf{R}$ ist von der Form $\pi_{\langle\rangle}(Q)$, wobei Q eine Semijoin-Anfrage über $\mathbf{R}$ ist.

Auswertung von Semijoin-Anfragen

Proposition 5.13

Das Auswertungsproblem für Semijoin-Anfragen bzw. Boolesche Semijoin-Anfragen ist in Zeit $\mathcal{O}(k^2 \cdot n \cdot \log(k \cdot n))$ lösbar

(für k = Größe der eingegebenen Semijoin-Anfrage und n = Größe der Datenbank).

Beweis: siehe Tafel

Semijoin-Anfragen vs. Join-Bäume

Lemma 5.14

(a) *Es gibt einen Algorithmus, der bei Eingabe einer Semijoin-Anfrage Q in Zeit $\mathcal{O}(||Q||)$ eine zu Q äquivalente azyklische regelbasierte konjunktive Anfrage Q' und einen Join-Baum für Q' berechnet.*

(b) *Es gibt einen Algorithmus, der bei Eingabe einer azyklischen Booleschen regelbasierten konjunktiven Anfrage Q und eines Join-Baums T für Q in Zeit $\mathcal{O}(||Q||)$ eine zu Q äquivalente Boolesche Semijoin-Anfrage Q' berechnet.*

Beweis: (a): Übung. (b): siehe Tafel.

Folgerung: Mit azyklischen Booleschen regelbasierten konjunktiven Anfragen kann man genau dieselben Anfragefunktionen ausdrücken wie mit Booleschen Semijoin-Anfragen.

Vorsicht: Dies gilt nicht, wenn man an Stelle von Booleschen Anfragen Anfragen beliebiger Stelligkeit betrachtet.

Konstruktion eines Join-Baums

Lemma 5.15

Es gibt einen Polynomialzeit-Algorithmus, der bei Eingabe einer regelbasierten konjunktiven Anfrage Q entscheidet, ob Q azyklisch ist und ggf. einen Join-Baum für Q konstruiert.

Beweis:

Algorithmus: **Eingabe:** Anfrage Q der Form $Ans(u) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$

(1) $V := \{R_1(u_1), \ldots, R_\ell(u_\ell)\}$ ..... *Knotenmenge*

(2) $E := \emptyset$ ..... *Kantenmenge*

(3) alle Elemente von V sind unmarkiert

(4) alle Variablen sind unmarkiert

(5) Wiederhole so lange, bis sich nichts mehr ändert:

  (5.1) Falls es unmarkierte Knoten $R_i(u_i)$ und $R_j(u_j)$ (mit $i \neq j$) gibt, so dass alle unmarkierten Variablen aus u_j in u_i vorkommen, so markiere den Knoten $R_j(u_j)$ und füge in E eine Kante zwischen $R_i(u_i)$ und $R_j(u_j)$ ein.

  (5.2) Markiere sämtliche Variablen x, für die gilt:
  "Es gibt genau einen unmarkierten Knoten, in dem x vorkommt."

(6) Falls es nur noch einen unmarkierten Knoten gibt, so gib (V, E) aus;
sonst gib aus: "Q ist nicht azyklisch".

Zum Beweis von Lemma 5.15 (1/2)

Beispiel: Probelauf des Algorithmus für die Anfragen

- $Q_1 := Ans() \leftarrow E(x_P, x_S),\ D(x_P, x_K, x_Z),\ T(x_S, y_K, y_Z)$
- $Q_2 := Ans() \leftarrow E(x_P, x_S),\ D(x_P, x_K, x_Z),\ T(x_S, x_K, x_Z)$
- $Q_3 := Ans() \leftarrow R(y, z),\ P(x, y),\ S(y, z, u),\ S(z, u, w),\ T(y, z),\ T(z, u),\ R(z', y')$

Notation:

- Zeitpunkt t = Beginn des t-ten Durchlaufs durch Zeile (5)
- $w_1^t, \ldots, w_{r_t}^t$: die zu Zeitpunkt t noch unmarkierten Knoten
- MV^t : Menge der zum Zeitpunkt t bereits markierten Variablen
- E^t : die Kantenmenge zum Zeitpunkt t

Zum Beweis von Lemma 5.15 (2/2)

Die Korrektheit des Algorithmus folgt direkt aus den folgenden Behauptungen 1 & 2:

Behauptung 1: Zu jedem Zeitpunkt t gilt:

$(1)_t$: E^t ist ein Wald aus Bäumen $T_1^t, \ldots, T_{r_t}^t$, deren Wurzeln die Knoten $w_1^t, \ldots, w_{r_t}^t$ sind.

$(2)_t$: Jeder dieser Bäume erfüllt die Weg-Eigenschaft, d.h. für alle $i \in \{1, \ldots, r_t\}$, alle Knoten $v, v' \in T_i^t$ und jede Variable x, die sowohl in v als auch in v' vorkommt, gilt: x kommt in jedem Knoten auf dem Weg zwischen v und v' vor.

$(3)_t$: Jede unmarkierte Variable (d.h. jede Variable, die nicht zu MV^t gehört), die in einem Baum T_k^t vorkommt, kommt auch in dessen Wurzel w_k^t vor.

$(4)_t$: Es gibt keine markierte Variable (d.h. aus MV^t), die in 2 verschiedenen Bäumen T_i^t und T_j^t vorkommt.

Beweis: Induktion nach t. $t = 1$: klar. $t \mapsto t+1$: Nachrechnen (Übung).

Behauptung 2:
Wenn Q azyklisch ist, so endet der Algorithmus mit nur *einem* unmarkierten Knoten.

Beweis: Siehe Tafel.

Auswertungskomplexität azyklischer Boolescher konjunktiver Anfragen

Theorem 5.16 (Yannakakis, 1981)

Das

AUSWERTUNGSPROBLEM FÜR AZYKLISCHE REGELBASIERTE KONJUNKTIVE ANFRAGEN

Eingabe: Regelbasierte konjunktive Anfrage Q und Datenbank $\mathbf{I}$

Aufgabe: Falls Q ayzklisch ist, so berechne $[\![Q]\!](\mathbf{I})$;
ansonsten gib "Q ist nicht azyklisch" aus.

kann in Zeit polynomiell in $||Q|| + ||\mathbf{I}|| + ||[\![Q]\!](\mathbf{I})||$ *gelöst werden.*

Beweis:

- *Algo für Boolesche Anfragen:* Nutze Lemma 5.15, Lemma 5.14 (b) und Proposition 5.13.
- *Algo für Anfragen beliebiger Stelligkeit:* Nutze Algo für Boolesche Anfragen und die Konstruktion aus dem Beweis von Theorem 2.20. Beachte dabei, dass sämtliche Booleschen Anfragen, die zur Auswertung einer azyklischen Anfrage Q gestellt werden, denselben Join-Baum besitzen wie Q und daher insbesondere azyklisch sind.

Bemerkung: Es ist bekannt, dass das Auswertungsproblem für Boolesche azyklische regelbasierte konjunktive Anfragen vollständig ist für die Komplexitätsklasse LOGCFL (Gottlob, Leone, Scarcello, 1998).

Konjunktives Guarded Fragment GF(CQ)

Definition 5.17

Sei $\mathbf{R}$ ein Datenbankschema.

Mit GF(CQ)[$\mathbf{R}$] bezeichnen wir die Menge aller Formeln des konjunktiven Kalküls CQ[$\mathbf{R}$] (vgl. Definition 2.8), die zum Guarded Fragment GF[$\mathbf{R}$] gehören, d.h. ... (Details: siehe Tafel)

Konjunktive Sätze des Guarded Fragment sind Formeln aus GF(CQ)[$\mathbf{R}$], die keine freien Variablen besitzen.

Azyklische Boolesche Konjunktive Anfragen

Satz 5.18

Die folgenden Anfragesprachen können genau dieselben Booleschen Anfragefunktionen ausdrücken:

(a) *azyklische Boolesche regelbasierte konjunktive Anfragen,*

(b) *Boolesche Semijoin-Anfragen,*

(c) *konjunktive Sätze des Guarded Fragment.*

Und jede Anfrage aus einer dieser Anfragesprachen kann in polynomieller Zeit in äquivalente Anfragen der anderen Sprachen übersetzt werden.

Gemäß Theorem 5.16 ist das Auswertungsproblem also für jede dieser Anfragesprachen in Polynomialzeit lösbar (kombinierte Komplexität).

Beweis: (a) $\Longleftrightarrow$ (b): Lemma 5.14. (a) $\Longleftrightarrow$ (c): Übung.

Konjunktive Anfragen II

Motivation

bisher: Mengen-Semantik (engl.: set semantics):

- DB-Relation = eine Menge von Tupeln
- Duplikate eines Tupels werden eliminiert

$$\{t\} = \{t,t\} = \{t,t,t\} = \cdots$$

in SQL:

- keine Duplikat-Elimination bei Anfragen der Form
  `SELECT * FROM ... WHERE ...`
- falls Duplikat-Elimination explizit gewünscht:
  `SELECT DISTINCT * FROM ... WHERE ...`

betrachte jetzt: Multimengen-Semantik (engl: bag semantics):

$$\{t\} \neq \{t,t\} \neq \{t,t,t\} \neq \cdots$$

Beispiel

Datenbankschema:

- 2-stellige Relation *Hersteller* mit Attributen *Name* und *Ort*
- 2-stellige Relation *Bauteil* mit Attributen *Teil* und *Lager*

"Datenbank" $\mathbf{I}_F$ mit

$\mathbf{I}_{Flugzeug}(Hersteller)$

Name	*Ort*
Boeing	Seattle
Boeing	New York
Airbus	Hamburg

Notation:

- $|\langle \text{Boeing,Seattle} \rangle|_{\mathbf{I}_F(Hersteller)} = 1$
- $|\langle \text{Boeing,New York} \rangle|_{\mathbf{I}_F(Hersteller)} = 1$
- $|\langle \text{Airbus,Hamburg} \rangle|_{\mathbf{I}_F(Hersteller)} = 1$

$\mathbf{I}_F(Bauteil)$

Teil	*Lager*
Motor	Seattle
Motor	Seattle
Flügel	Portland
Cockpit	Seattle
Cockpit	Seattle
Cockpit	Seattle

Notation:

- $|\langle \text{Motor,Seattle} \rangle|_{\mathbf{I}_F(Bauteil)} = 2$
- $|\langle \text{Flügel,Portland} \rangle|_{\mathbf{I}_F(Bauteil)} = 1$
- $|\langle \text{Cockpit,Seattle} \rangle|_{\mathbf{I}_F(Bauteil)} = 3$

Multimengen (Bags) und Multimengen-Datenbanken

Sei M eine Menge.

- Eine Multimenge B über M ist eine Abbildung $B : M \to \mathbb{N}_{\geqslant 0}$
- Notation: Für $a \in M$ schreibe $|a|_B$ an Stelle von $B(a)$
  $|a|_B = i$ bedeutet: das Element a kommt i-mal in der Multimenge B vor.
- B heißt endlich, falls die Menge $\{a \in M : |a|_B \neq 0\}$ endlich ist.
- Für Multimengen B und B' über M gilt:
  - $B =_b B' \quad :\Longleftrightarrow \quad |a|_B = |a|_{B'}$, für alle $a \in M$
  - $B \subseteq_b B' \quad :\Longleftrightarrow \quad |a|_B \leqslant |a|_{B'}$, für alle $a \in M$
  - Insbesondere gilt: $B =_b B' \iff \Big(B \subseteq_b B' \text{ und } B' \subseteq_b B\Big)$
  - $B \cup_b B' \quad :=$
    die Multimenge B'' über M mit $|a|_{B''} := |a|_B + |a|_{B'}$, für alle $a \in M$

Definition 5.19

Sei $\mathbf{R}$ ein Datenbankschema.
Eine Multimengen-Datenbank $\mathbf{I} \in inst_b(\mathbf{R})$ ordnet jedem Relationssymbol $R \in \mathbf{R}$ eine endliche Multimenge $\mathbf{I}(R)$ über $\mathbf{dom}^{arity(R)}$ zu.

Anfragen mit Mulitmengen-Semantik

Beispiel:

SQL-Anfrage:

```
SELECT B1.Teil, B2.Teil
FROM Bauteil B1, Bauteil B2
WHERE B1.Lager = B2.Lager
```

regelbasiert:

$Ans(x,y) \leftarrow Bauteil(x,z),\ Bauteil(y,z)$

Auswertung der SQL-Anfrage in Datenbank mit

$\mathbf{I}_F(Bauteil)$

Teil	*Lager*
Motor	Seattle
Motor	Seattle
Flügel	Portland
Cockpit	Seattle
Cockpit	Seattle
Cockpit	Seattle

- bilde Kreuzprodukt
  $\mathbf{I}_F(Bauteil) \times \mathbf{I}_F(Bauteil)$
  (ohne Duplikatelimination)
- wähle die Tupel, in denen die Lager-Komponenten gleich sind
- streiche die Spalten mit den Lager-Komponenten

Liefert als Ergebnis die Multimenge M mit

- $|\langle \text{Motor,Motor} \rangle|_M = 4 \quad |\langle \text{Flügel,Flügel} \rangle|_M = 1 \quad |\langle \text{Cockpit,Cockpit} \rangle|_M = 9$
- $|\langle \text{Motor,Cockpit} \rangle|_M = 6 = |\langle \text{Cockpit,Motor} \rangle|_M$

Konjunktive Anfragen mit Multimengen-Semantik

Multimengen-Semantik $[\![Q]\!]_b$:

Sei $Q := Ans(u) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$ eine regelbasierte konjunktive Anfrage der Stelligkeit r über einem Datenbankschema $\mathbf{R}$.

- Eine Belegung β für Q ist, wie bisher, eine Abbildung $\beta : Var(Q) \to \mathbf{dom}$.
- Die Auswertung von Q in einer Multimengen-Datenbank $\mathbf{I} \in inst_b(\mathbf{R})$ liefert als Ergebnis die Multimenge $[\![Q]\!]_b(\mathbf{I})$, so dass für alle Tupel $t \in \mathbf{dom}^r$ gilt:

$$|\, t \,|_{[\![Q]\!]_b(\mathbf{I})} \quad := \sum_{\substack{\beta \,:\, \beta \text{ Belegung} \\ \text{für } Q \text{ mit } \beta(u)=t}} \Big(\; |\beta(u_1)|_{\mathbf{I}(R_1)} \cdot |\beta(u_2)|_{\mathbf{I}(R_2)} \cdot \cdots \cdot |\beta(u_\ell)|_{\mathbf{I}(R_\ell)} \; \Big)$$

Äquivalenz von Anfragen & Query Containment:

Seien Q und Q' zwei regelbasierte konjunktive Anfragen derselben Stelligkeit r über einem Datenbankschema $\mathbf{R}$.

- $Q \equiv_b Q' \quad :\Longleftrightarrow \quad [\![Q]\!]_b(\mathbf{I}) =_b [\![Q']\!]_b(\mathbf{I})$, für alle $\mathbf{I} \in inst_b(\mathbf{R})$
- $Q \subseteq_b Q' \quad :\Longleftrightarrow \quad [\![Q]\!]_b(\mathbf{I}) \subseteq_b [\![Q']\!]_b(\mathbf{I})$, für alle $\mathbf{I} \in inst_b(\mathbf{R})$
- Klar: $Q \equiv_b Q' \iff \Big(Q \subseteq_b Q' \text{ und } Q' \subseteq_b Q \Big)$

  Insbes: Äquivalenz ist höchstens so schwer wie Query Containment.

Ergebnisse

Theorem 5.20 (Chaudhuri, Vardi, 1993) *(hier ohne Beweis)*

(a) *Seien Q und Q' regelbasierte konjunktive Anfragen derselben Stelligkeit über demselben Datenbankschema. Dann ist $Q \equiv_b Q'$ genau dann, wenn die zu Q und Q' gehörenden Tableau-Anfragen isomorph sind (im Sinne von Definition 5.7).*

(b) *Das Problem*

Äquivalenz konj. Anfragen bzgl. Multimengen-Semantik
Eingabe: regelbasierte konjunktive Anfragen Q und Q'
Frage: Ist $Q \equiv_b Q'$?

ist genauso schwer wie das Graph-Isomorphie-Problem.

Folgerungen und eine offene Frage

- Das Äquivalenzproblem bzgl. Multimengen-Semantik liegt in NP und ist "vermutlich nicht NP-vollständig" (da vermutet wird, dass das Graph-Isomorphie-Problem nicht NP-hart ist).

  Somit: "Äquivalenz bzgl. Multimengen-Semantik" ist vermutlich einfacher als "Äquivalenz bzgl. Mengen-Semantik".

- Das "Query Containment Problem bzgl. Multimengen-Semantik" ist vermutlich schwerer als das "Query Containment Problem bzgl. Mengen-Semantik".

Offene Forschungsfrage:

Bisher ist nicht bekannt, ob das Query Containment Problem für konjunktive Anfragen bzgl. Multimengen-Semantik überhaupt entscheidbar ist.

Konj. Anfragen mit $\neq$ in Multimengen-Semantik

Konjunktive regelbasierte Anfragen mit $\neq$ sind von der Form

$$Ans(u) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell), U_1, \ldots, U_m$$

wobei $R_1(u_1), \ldots, R_\ell(u_\ell)$ Relations-Atome und $U_1, \ldots, U_m$ Ungleichungen der Form $x \neq y$ mit $x, y \in \mathbf{var} \cup \mathbf{dom}$ sind.

Theorem 5.21 (Jayram, Kolaitis, Vee, 2006) *(hier ohne Beweis)*

Das Problem

Query Containment für konj. Anfragen mit $\neq$ bzgl. Multimengen-Semantik
Eingabe: Q und Q' : regelbasierte konjunktive Anfragen mit $\neq$
Frage: Ist $Q \subseteq_b Q'$?

ist nicht entscheidbar.

Ab jetzt wieder

— und bis zum Ende des Semesters —

Mengen-Semantik