Konjunktive Anfragen

2.1 Deskriptiver Ansatz: regelbasiert, graphisch und logikbasiert
2.2 Auswertungskomplexität
2.3 Algebraischer Ansatz: SPC-Algebra und SPJR-Algebra
2.4 Homomorphismus-Satz, Statische Analyse und Anfrageminimierung
2.5 Azyklische Anfragen
2.6 Mengen-Semantik vs. Multimengen-Semantik

Konjunktive Anfragen

Regelbasierte Konjunktive Anfragen — Informell

Beispiel-Anfrage:

(4) Welche Kinos (Name & Adresse) spielen einen Film mit “George Clooney”?

Andere Formulierung:

Wenn es in *Filme* ein Tupel $\langle x_{Titel}, x_{Regie}, \text{“George Clooney”}\rangle$ und
in *Programm* ein Tupel $\langle x_{Kino}, x_{Titel}, x_{Zeit}\rangle$ und
in *Orte* ein Tupel $\langle x_{Kino}, x_{Adr}, x_{Tel}\rangle$ gibt,
dann nimm das Tupel $\langle x_{Kino}, x_{Adr}\rangle$ in die Antwort auf

Als regelbasierte konjunktive Anfrage:

$$\begin{aligned} Ans(x_{Kino}, x_{Adr}) \leftarrow\ & Filme(x_{Titel}, x_{Regie}, \text{“George Clooney”}), \\ & Programm(x_{Kino}, x_{Titel}, x_{Zeit}), \\ & Orte(x_{Kino}, x_{Adr}, x_{Tel}) \end{aligned}$$

Regelbasierte Konjunktive Anfragen — Präzise

Definition 2.1

- **var** sei eine abzählbar unendliche Menge von Variablen(symbolen), die disjunkt zu den Mengen **att**, **dom**, **relname** ist.

  (Einzelne Variablen bezeichnen wir i.d.R. mit x, y, x_1, x_2, ...)
- Ein Term ist ein Element aus $\mathbf{var} \cup \mathbf{dom}$.
- Ein freies Tupel der Stelligkeit k ist ein Element aus $(\mathbf{var} \cup \mathbf{dom})^k$.

Definition 2.2

Sei **R** ein Datenbankschema.
Eine regelbasierte konjunktive Anfrage über **R** ist ein Ausdruck der Form

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

wobei $\ell \geqslant 0$, $R_1, \ldots, R_\ell \in \mathbf{R}$, $Ans \in \mathbf{relname} \backslash \mathbf{R}$ und $u, u_1, \ldots, u_\ell$ freie Tupel der Stelligkeiten $arity(Ans)$, $arity(R_1), \ldots, arity(R_\ell)$, so dass jede Variable, die in u vorkommt, auch in mindestens einem der Tupel $u_1, \ldots, u_\ell$ vorkommt.

Semantik regelbasierter konjunktiver Anfragen

Sei Q eine regelbasierte konjunktive Anfrage (über einem DB-Schema $\mathbf{R}$) der Form

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

- Mit $Var(Q)$ bezeichnen wir die Menge aller Variablen, die in Q vorkommen.
- Eine Belegung für Q ist eine Abbildung $\beta : Var(Q) \to \mathbf{dom}$.

  Wir setzen β auf natürliche Weise fort zu einer Abbildung von $Var(Q) \cup \mathbf{dom}$ nach $\mathbf{dom}$, so dass $\beta|_{\mathbf{dom}} = \mathrm{id}$. Für ein freies Tupel $u = \langle e_1, .., e_k \rangle$ setzen wir $\beta(u) := \langle \beta(e_1), .., \beta(e_k) \rangle$.
- Der Anfrage Q ordnen wir die folgende Anfragefunktion $[\![Q]\!]$ zu:

$$[\![Q]\!](\mathbf{I}) := \left\{ \beta(u) \; : \; \begin{array}{l} \beta \text{ ist eine Belegung für } Q, \text{ so dass} \\ \beta(u_i) \in \mathbf{I}(R_i), \text{ f.a. } i \in \{1, .., \ell\} \end{array} \right\}$$

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

Beispiele (1/2)

- Die Anfrage (6) Welche (je 2) Regisseure haben schon in Filmen gespielt, in denen der jeweils andere Regie geführt hat?

  lässt sich durch folgende regelbasierte konjunktive Anfrage ausdrücken:

$$\mathit{Antworten}(x, y) \leftarrow \mathit{Filme}(z_1, x, y), \mathit{Filme}(z_2, y, x)$$

- Die Anfrage (5) Läuft zur Zeit ein “Detlev Buck” Film?

  lässt sich durch folgende regelbasierte konjunktive Anfrage ausdrücken:

$$\mathit{Ans}() \leftarrow \mathit{Filme}(x, \text{“Detlev Buck”}, y),\ \mathit{Programm}(z, x, w)$$

Ans ist hier also ein Relations-Name der Stelligkeit 0.
Erinnern Sie sich an unsere Konvention, dass die Ausgabe “$\emptyset$” der Antwort “nein” entspricht, und die Ausgabe der Menge $\{\langle\rangle\}$, die aus dem “Tupel der Stelligkeit 0” besteht, der Antwort “ja” entspricht.

Beispiele (2/2)

Betrachte die Datenbank $\mathbf{I} := \{\mathbf{I}(R), \mathbf{I}(S)\}$ mit
$\mathbf{I}(R) := \{\langle a, a\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle c, b\rangle\}$ und $\mathbf{I}(S) := \{\langle d, a, b\rangle, \langle a, c, e\rangle, \langle b, a, c\rangle\}$.

- Die Anfrage $Q_1 :=$

$$Ans_1(x_1, x_2, x_3) \leftarrow R(x_1, y), S(y, x_2, x_3)$$

liefert auf $\mathbf{I}$ das Ergebnis $[\![Q_1]\!](\mathbf{I}) = \{\langle a, c, e\rangle, \langle a, a, c\rangle, \langle c, a, c\rangle\}$.

- Die Anfrage $Q_2 :=$

$$Ans_2(x, y) \leftarrow R(x, z_1),\ S(z_1, a, z_2),\ R(y, z_2)$$

liefert auf $\mathbf{I}$ das Ergebnis $[\![Q_2]\!](\mathbf{I}) = \{\langle a, b\rangle, \langle c, b\rangle\}$.

Bezeichnungen

Oft sagen wir kurz Regel, um eine regelbasierte konjunktive Anfrage zu bezeichnen.

Sei Q eine Regel der Form $Ans(u) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$

- $Ans(u)$ wird als Kopf der Regel bezeichnet.
- $R_1(u_1), \ldots, R_\ell(u_\ell)$ wird als Rumpf der Regel bezeichnet.
- Die Relations-Namen aus **R** werden als extensionale Datenbankprädikate (kurz: edb-Prädikate) bezeichnet.
  Wir schreiben $edb(Q)$, um die Menge aller edb-Prädikate zu bezeichnen, die in Q vorkommen.
- Der Relations-Name, der im Kopf von Q vorkommt, wird als intensionales Datenbankprädikat (kurz: idb-Prädikat) bezeichnet.
- Mit $adom(Q)$ bezeichnen wir die Menge aller Konstanten (also Elemente aus **dom**), die in Q vorkommen. ("*adom*" steht für "active domain")

Der “Active Domain” einer Datenbank

Definition 2.3

Sei **R** ein Datenbankschema und sei **I** eine Datenbank vom Schema **R**.

Der Active Domain von **I**, kurz: $adom(\mathbf{I})$, ist die Menge aller Elemente aus **dom**, die in **I** vorkommen. D.h.:

$$adom(\mathbf{I}) \;=\; \bigcup_{R \in \mathbf{R}} adom\big(\mathbf{I}(R)\big)$$

wobei für jedes R aus **R** gilt: $adom\big(\mathbf{I}(R)\big)$ ist die kleinste Teilmenge von **dom**, so dass jedes Tupel $t \in \mathbf{I}(R)$ eine Funktion von $sort(R)$ nach $adom\big(\mathbf{I}(R)\big)$ ist.

Ist Q eine Anfrage und **I** eine Datenbank, so setzen wir

$$adom(Q, \mathbf{I}) \;:=\; adom(Q) \cup adom(\mathbf{I})$$

Proposition 2.4

Für jede regelbasierte konjunktive Anfrage Q und jede Datenbank **I** *(vom passenden DB-Schema) gilt:* $adom\big([\![Q]\!](\mathbf{I})\big) \;\subseteq\; adom(Q, \mathbf{I})$.

Beweis: siehe Tafel.

Monotonie und Erfüllbarkeit

Sind **I** und **J** zwei Datenbanken vom gleichen Schema **R**, so sagen wir "**J** ist eine Erweiterung von **I**" und schreiben kurz "$\mathbf{J} \supseteq \mathbf{I}$" (bzw. "$\mathbf{I} \subseteq \mathbf{J}$"), falls für alle $R \in \mathbf{R}$ gilt: $\mathbf{I}(R) \subseteq \mathbf{J}(R)$ (d.h. jedes Tupel, das in einer Relation von **I** vorkommt, kommt auch in der entsprechenden Relation von **J** vor).

Definition 2.5

Sei **R** ein DB-Schema und sei q eine Anfragefunktion über **R**.

(a) q heißt monoton, falls für alle Datenbanken **I** und **J** über **R** gilt: Falls $\mathbf{I} \subseteq \mathbf{J}$, so ist $q(\mathbf{I}) \subseteq q(\mathbf{J})$.

(b) q heißt erfüllbar, falls es eine Datenbank **I** gibt mit $q(\mathbf{I}) \neq \emptyset$.

Satz 2.6

Jede regelbasierte konjunktive Anfrage ist monoton und erfüllbar.

Beweis: siehe Tafel.

Anwendung von Satz 2.6

Satz 2.6 liefert ein einfaches Kriterium, um zu zeigen, dass bestimmte Anfragefunktionen nicht durch eine regelbasierte konjunktive Anfrage beschrieben werden können:

> Wenn eine Anfragefunktion q nicht monoton ist, dann kann sie auch nicht durch eine regelbasierte konjunktive Aussage beschrieben werden.

Beispiel: Die Anfrage

(15) Welche Filme laufen nur zu 1 Uhrzeit?

ist nicht monoton, kann also nicht durch eine regelbasierte konjunktive Anfrage beschrieben werden.

Vorsicht: Dies heißt nicht, dass jede monotone Anfragefunktion durch eine Regel beschrieben werden kann!

“Graphische” Variante: Tableau-Anfragen

Beispiel-Anfrage:

(4) Welche Kinos (Name & Adresse) spielen einen Film mit “George Clooney”?

Darstellung als Tableau **T** (ähnlich zu QBE):

Filme

Titel	Regie	Schauspieler
x_{Titel}	x_{Regie}	“George Clooney”

Programm

Kino	Titel	Zeit
x_{Kino}	x_{Titel}	x_{Zeit}

Orte

Kino	Adresse	Telefon
x_{Kino}	x_{Adr}	x_{Tel}

Zugehörige Tableau-Anfrage: $\left(\mathbf{T},\ \langle x_{Kino},\ x_{Adr}\rangle\right)$

Tableaus — Präzise

Definition 2.7

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

  (D.h. ein Tableau über **R** ist eine "Datenbank vom Schema **R**, die als Einträge auch Variablen enthalten kann".)

- Eine Tableau-Anfrage über **R** (bzw. R) ist von der Form $(\mathbf{T}, u)$, wobei **T** ein Tableau über **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 $(\mathbf{T}, u)$.

Semantik von Tableau-Anfragen

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:

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

- 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 $\mathbf{I} \in inst(\mathbf{R})$.

Logikbasierte Variante: Konjunktiver Kalkül

Beispiel-Anfrage:

(4) Welche Kinos (Name & Adresse) spielen einen Film mit “George Clooney”?

Formulierung als Anfrage des konjunktiven Kalküls:

$$\left\{ \langle x_{Kino}, x_{Adr} \rangle \quad : \quad \exists x_{Titel}\, \exists x_{Regie}\, \exists x_{Zeit}\, \exists x_{Tel}\, \left(\begin{array}{l} \mathit{Filme}(x_{Titel}, x_{Regie}, \text{“George Clooney”}) \wedge \\ \mathit{Programm}(x_{Kino}, x_{Titel}, x_{Zeit}) \wedge \\ \mathit{Orte}(x_{Kino}, x_{Adr}, x_{Tel}) \end{array} \right) \right\}$$

Erinnerung an “Diskrete Modellierung”:

Ähnlichkeit zu Formeln der Logik erster Stufe (d.h. Prädikatenlogik).

Hier: eingeschränkte Variante, in der es nur $\exists$-Quantoren und Konjunktionen ($\wedge$) gibt.

Konjunktiver Kalkül (CQ) — Präzise

Definition 2.8

Sei **R** ein Datenbankschema.
Die Menge CQ[**R**] aller Formeln des konjunktiven Kalküls über **R** ist induktiv wie folgt definiert: (CQ steht für "conjunctive queries")

(A) $R(v_1, \ldots, v_r)$ gehört zu CQ[**R**],
für alle $R \in \mathbf{R}$, $r := \mathit{arity}(R)$ und $v_1, \ldots, v_r \in \mathbf{var} \cup \mathbf{dom}$.

(K) $(\varphi \wedge \psi)$ gehört zu CQ[**R**],
für alle $\varphi \in$ CQ[**R**] und $\psi \in$ CQ[**R**].

(E) $\exists x\, \varphi$ gehört zu CQ[**R**],
für alle $\varphi \in$ CQ[**R**] und $x \in \mathbf{var}$.

Insbesondere: Jede Formel in CQ[**R**] ist eine Formel der Logik erster Stufe über der Signatur $\mathbf{R} \cup \mathbf{dom}$ (wobei jedes Element aus **dom** als "Konstanten-Symbol" aufgefasst wird, das stets "mit sich selbst" interpretiert wird.

Semantik von CQ[**R**]

Sei φ eine CQ[**R**]-Formel.

- $adom(\varphi)$ bezeichnet die Menge aller Konstanten (also Elemente aus **dom**), die in φ vorkommen. $Var(\varphi)$ bezeichnet die Menge aller Variablen (also Elemente aus **var**), die in φ vorkommen.
- $frei(\varphi)$ bezeichnet die Menge aller Variablen, die frei in φ vorkommen.
  D.h.: $frei(R(v_1,..,v_r)) = \{v_1,..,v_r\} \cap \mathbf{var}$; $frei((\varphi \wedge \psi)) = frei(\varphi) \cup frei(\psi)$;
  $frei(\exists x \varphi) = frei(\varphi) \setminus \{x\}$.
- Eine Belegung für φ ist eine Abbildung $\beta : frei(\varphi) \to \mathbf{dom}$.
- Einer Datenbank **I** vom Schema **R** ordnen wir die logische Struktur

$$\mathcal{A}_{\mathbf{I}} := \left(\mathbf{dom},\ (\mathbf{I}(R))_{R \in \mathbf{R}},\ (c)_{c \in \mathbf{dom}}\right)$$

  zu. (Insbesondere ist $\mathcal{A}_{\mathbf{I}}$ eine σ-Struktur über der Signatur $\sigma := \mathbf{R} \cup \mathbf{dom}$.)
- Ist **I** eine Datenbank vom Schema **R** und β eine Belegung für φ, so sagen wir "**I** erfüllt φ unter β" und schreiben $\mathbf{I} \models \varphi[\beta]$ bzw. $(\mathbf{I}, \beta) \models \varphi$, um auszudrücken, dass $(\mathcal{A}_{\mathbf{I}}, \beta) \models \varphi$.

Notation

- Mit CQ bezeichnen wir die Klasse aller CQ[**R**]-Formeln für alle Datenbankschemata **R**.
- Manchmal schreiben wir $\{x_1/a_1, \ldots, x_r/a_r\}$ um die Belegung $\beta : \{x_1, .., x_r\} \to \mathbf{dom}$ zu bezeichnen mit $\beta(x_i) = a_i$, f.a. $i \in \{1, .., r\}$.
- Für eine CQ-Formel φ schreiben wir oft $\varphi(x_1, .., x_r)$, um anzudeuten, dass $\mathit{frei}(\varphi) = \{x_1, .., x_r\}$.
- Ist $a_1, .., a_r \in \mathbf{dom}$, so schreiben wir vereinfachend $\mathbf{I} \models \varphi[a_1, .., a_r]$ an Stelle von $\mathbf{I} \models \varphi[\{x_1/a_1, \ldots, x_r/a_r\}]$.
- Ist $y \in \mathbf{dom} \cup \mathbf{var}$, so bezeichnet $\varphi(x_1/y, x_2, .., x_r)$ die Formel, die aus φ entsteht, indem jedes Vorkommen der Variablen x_1 durch y ersetzt wird.
- Beim Schreiben von Formeln lassen wir Klammern "(", ")" oft weg und schreiben $\exists x_1, .., x_n$ als Abkürzung für $\exists x_1 \, \exists x_2 \, \cdots \exists x_n$.
- Zwei CQ[**R**]-Formeln φ und ψ heißen äquivalent, falls $\mathit{frei}(\varphi) = \mathit{frei}(\psi)$ und für jede Datenbank $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ und jede Belegung β für φ (und ψ) gilt: $\mathbf{I} \models \varphi[\beta] \iff \mathbf{I} \models \psi[\beta]$.

Konjunktiver Kalkül: Syntax und Semantik

Definition 2.9

Sei **R** ein Datenbankschema.

Eine Anfrage des konjunktiven Kalküls ist von der Form

$$\{\langle e_1, \ldots, e_r \rangle \ : \ \varphi\}$$

wobei $\varphi \in \mathrm{CQ}[\mathbf{R}]$, $r \geqslant 0$ und $\langle e_1, \ldots, e_r \rangle$ ein freies Tupel ist, so dass $\mathit{frei}(\varphi) = \{e_1, .., e_r\} \cap \mathbf{var}$.

Semantik:

Einer Anfrage Q der Form $\{\langle e_1, .., e_r \rangle : \varphi\}$ ordnen wir die folgende Anfragefunktion $[\![Q]\!]$ zu:

$$[\![Q]\!](\mathbf{I}) \ := \ \left\{ \beta(\langle e_1, .., e_r \rangle) \ : \ \begin{array}{l} \beta \text{ ist eine Belegung für } \varphi \text{ mit} \\ \mathbf{I} \models \varphi[\beta] \end{array} \right\}$$

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

Wertebereich von Anfragen des konjunktiven Kalküls

Für eine Anfrage Q der Form $\{\langle e_1, \ldots, e_r \rangle : \varphi\}$ setzen wir

$$adom(Q) \;:=\; adom(\varphi) \cup \big(\{e_1, \ldots, e_r\} \cap \mathbf{dom}\big).$$

Ist Q eine Anfrage und $\mathbf{I}$ eine Datenbank, so setzen wir – wie üblich –

$$adom(Q, \mathbf{I}) \;:=\; adom(Q) \cup adom(\mathbf{I})$$

Analog zu Proposition 2.4 gilt auch für Anfragen des konjunktiven Kalküls:

Proposition 2.10

Für jede Anfrage Q des konjunktiven Kalküls und jede Datenbank $\mathbf{I}$ (vom passenden DB-Schema) gilt: $adom\big([\![Q]\!](\mathbf{I})\big) \;\subseteq\; adom(Q, \mathbf{I})$.

Beweis: Einfache Induktion über den Formelaufbau. Details: Übung.

Beispiel

Die Anfrage

> Gibt es einen Schauspieler, der sowohl in einem Film von “Detlev Buck” als auch in einem Film von “Stephen Spielberg” mitgespielt hat?

wird durch die folgende Anfrage des konjunktiven Kalküls beschrieben:

$$\left\{ \langle\rangle \;:\; \exists x_{Schauspieler} \left(\exists x_{Titel1}\, Filme(x_{Titel1}, \text{“Detlev Buck”}, x_{Schauspieler}) \wedge \exists x_{Titel2}\, Filme(x_{Titel2}, \text{“Stephen Spielberg”}, x_{Schauspieler}) \right) \right\}$$

und durch die dazu äquivalente Anfrage

$$\left\{ \langle\rangle \;:\; \exists x_{Schauspieler}\, \exists x_{Titel1}\, \exists x_{Titel2} \left(Filme(x_{Titel1}, \text{“Detlev Buck”}, x_{Schauspieler}) \wedge Filme(x_{Titel2}, \text{“Stephen Spielberg”}, x_{Schauspieler}) \right) \right\}$$

Eine Normalform für CQ

Definition 2.11

Eine CQ[**R**]-Formel $\varphi(x_1, \ldots, x_r)$ ist in Normalform, falls sie von der Form

$$\exists x_{r+1} \cdots \exists x_{r+s} \left(R_1(u_1) \wedge \cdots \wedge R_\ell(u_\ell) \right)$$

ist, mit $r, s, \ell \geqslant 0$, $R_1, \ldots, R_\ell \in \mathbf{R}$ und $u_1, \ldots, u_\ell$ freie Tupel über $\{x_1, \ldots, x_{r+s}\} \cup \mathbf{dom}$.

Analog ist eine Anfrage Q des konjunktiven Kalküls in Normalform, falls sie von der Form $\{\langle e_1, \ldots, e_r \rangle : \varphi\}$ ist, wobei φ eine CQ-Formel in Normalform ist.

Lemma 2.12

Jede CQ-Formel ist äquivalent zu einer CQ-Formel in Normalform.

Beweis: Übung.

Äquivalenz von Anfragesprachen

Definition 2.13

Seien $\mathcal{Q}_1$ und $\mathcal{Q}_2$ zwei Anfragesprachen. Wir schreiben

- $\mathcal{Q}_1 \leqslant \mathcal{Q}_2$ (bzw. $\mathcal{Q}_2 \geqslant \mathcal{Q}_1$; "$\mathcal{Q}_2$ ist mindestens so ausdrucksstark wie $\mathcal{Q}_1$"), falls jede Anfragefunktion, die durch eine Anfrage in $\mathcal{Q}_1$ ausgedrückt werden kann, auch durch eine Anfrage in $\mathcal{Q}_2$ ausgedrückt werden kann.

- $\mathcal{Q}_1 \equiv \mathcal{Q}_2$ ("$\mathcal{Q}_1$ und $\mathcal{Q}_2$ haben dieselbe Ausdrucksstärke") falls $\mathcal{Q}_1 \leqslant \mathcal{Q}_2$ und $\mathcal{Q}_2 \leqslant \mathcal{Q}_1$

- $\mathcal{Q}_1 < \mathcal{Q}_2$ (bzw. $\mathcal{Q}_2 > \mathcal{Q}_1$; "$\mathcal{Q}_2$ ist ausdrucksstärker als $\mathcal{Q}_1$") falls $\mathcal{Q}_1 \leqslant \mathcal{Q}_2$ und nicht $\mathcal{Q}_2 \leqslant \mathcal{Q}_1$.

Äquivalenz der bisher eingeführten Anfragesprachen

Lemma 2.14

Die Klassen der regelbasierten konjunktiven Anfragen, der Tableau-Anfragen und der Anfragen des konjunktiven Kalküls haben dieselbe Ausdrucksstärke.

Es gilt sogar: Jede Anfrage aus einer dieser drei Anfragesprachen kann in polynomieller Zeit in äquivalente Anfragen der beiden anderen Anfragesprachen übersetzt werden.

Beweis: siehe Tafel.

Einige Erweiterungen der Anfragesprachen

(1) Test auf "Gleichheit" von Variablen zulassen

(2) Hintereinanderausführung (Komposition) mehrerer Anfrage zulassen

Zu (1):

Beispiel-Anfrage (6): Welche (je 2) Schauspieler haben schon in Filmen gespielt, in denen der jeweils andere Regie geführt hat?

ausdrücken durch

$$Ans(y_1, y_2) \leftarrow Filme(x_1, y_1, z_1),\ Filme(x_2, y_2, z_2),\ y_1{=}z_2,\ y_2{=}z_1$$

aber auch, äquivalent, durch

$$Ans(y_1, y_2) \leftarrow Filme(x_1, y_1, y_2),\ Filme(x_2, y_2, y_1)$$

Regelbasierte konjunktive Anfragen mit "="

Prinzipiell:

Im Rumpf von Regeln auch Atome der Form "$x{=}y$" und "$x{=}c$" zulassen, für beliebige Variablen $x, y \in \mathbf{var}$ und Konstante $c \in \mathbf{dom}$.

$\rightsquigarrow$ regelbasierte konjunktive Anfragen mit "="

$\rightsquigarrow$ Konjunktiver Kalkül mit "="

Aber Vorsicht: Die Regel

$$Q := \quad Ans(x, y) \leftarrow R(x),\ y{=}z$$

ausgewertet in einer Datenbank $\mathbf{I}$ mit $\mathbf{I}(R) = \{a\}$, liefert als Ergebnis

$$[\![Q]\!](\mathbf{I}) \ = \ \{\langle a, d\rangle : d \in \mathbf{dom}\} \ = \ \{a\} \times \mathbf{dom}$$

Da **dom** unendlich viele Elemente hat, ist $[\![Q]\!](\mathbf{I})$ also eine "unendliche Relation"; per Definition (siehe Folie 21), sind als Relationen aber nur endliche Mengen erlaubt.

Daher syntaktische Einschränkung auf bereichsbeschränkte Anfragen, um zu garantieren, dass das Ergebnis einer Anfrage stets eine endliche Relation ist . . .

Bereichsbeschränkte konjunktive Anfragen mit "="

Präzise:

Jede Variable x, die im Rumpf der Regel vorkommt, muss auch in einem im Rumpf der Regel stehenden Atom der Form $R(u)$ oder $x{=}c$, für ein $c \in \mathbf{dom}$, vorkommen — oder es muss eine Kette von Gleichheits-Atomen der Form $x{=}y_1, y_1{=}y_2, \ldots, y_j{=}z$ und ein Atom der Form $z{=}c$ für eine Konstante $c \in \mathbf{dom}$ geben oder ein Atom der Form $R(u)$ im Rumpf der Regel geben, so dass die Variable z im freien Tupel u vorkommt.

$\rightsquigarrow$ bereichsbeschränkte regelbasierte konjunktive Anfragen mit "="

Analog wird die Klasse $\mathrm{CQ}^{=}$ aller bereichsbeschränkten Formeln des konjunktiven Kalküls mit "=" definiert.

Beobachtung 2.15

Jede $\mathrm{CQ}^{=}$-Formel ist entweder unerfüllbar oder äquivalent zu einer CQ-Formel. (Details siehe Übung.)

Beispiel für eine $\mathrm{CQ}^{=}$-Anfrage, die nicht erfüllbar ist:

$$\{ \langle x \rangle : \ (R(x) \ \wedge \ x{=}a \ \wedge \ x{=}b \ ) \ \}$$

wobei a und b zwei verschiedene Elemente aus **dom** sind.

Einige Erweiterungen der Anfragesprachen

(1) Test auf “Gleichheit” von Variablen zulassen

(2) Hintereinanderausführung (Komposition) mehrerer Anfrage zulassen

Zu (2):

Wende Anfrage auf das Resultat einer (oder mehrerer) Anfragen an . . .

Regelbasierte konjunktive Programme

Definition 2.16

Sei **R** ein Datenbankschema.
Ein regelbasiertes konjunktives Programm über **R** (mit oder ohne "=") hat die Form

$$\begin{array}{lcl} S_1(u_1) & \leftarrow & \text{Rumpf}_1 \\ S_2(u_2) & \leftarrow & \text{Rumpf}_2 \\ \vdots & \vdots & \vdots \\ S_m(u_m) & \leftarrow & \text{Rumpf}_m \end{array}$$

wobei $m \geqslant 1$, $S_1, \ldots, S_m$ sind paarweise verschiedene Relations-Namen aus **relname** $\setminus$ **R** und für jedes $i \in \{1, \ldots, m\}$ gilt:

$$Q_i \;:=\; S_i(u_i) \leftarrow \text{Rumpf}_i$$

ist eine regelbasierte konjunktive Anfrage über $\big(\mathbf{R} \cup \{S_j : 1 \leqslant j < i\}\big)$ (mit oder ohne "=").

Die Relations-Namen aus **R**, die im Programm vorkommen, heißen extensionale Prädikate (edb-Prädikate). Die Relations-Namen $S_1, \ldots, S_m$ heißen intensionale Prädikate (idb-Prädikate).

Semantik regelbasierter konjunktiver Programme

Ausgewertet über einer Datenbank $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ beschreibt obiges Programm P der vorigen Folie die Relationen

$$[\![P(S_1)]\!](\mathbf{I}), \quad \ldots, \quad [\![P(S_m)]\!](\mathbf{I}),$$

die induktiv für alle $i \in \{1, \ldots, m\}$ folgendermaßen definiert sind:

$$[\![P(S_i)]\!](\mathbf{I}) \;:=\; [\![Q_i]\!](\mathbf{J}_{i-1})$$

wobei $\mathbf{J}_{i-1}$ die Erweiterung der Datenbank $\mathbf{I}$ um die Relationen $\mathbf{J}(S_j) := [\![P(S_j)]\!](\mathbf{I})$, für alle j mit $1 \leqslant j < i$ ist.

Äquivalenz von Regeln und Programmen

Beobachtung 2.17

Für jedes regelbasierte konjunktive Programm P über einem Relationsschema $\mathbf{R}$ (mit oder ohne "=") und jedes idb-Prädikat S von P gibt es eine regelbasierte konjunktive Anfrage Q (mit "=") über $\mathbf{R}$, so dass

$$[\![Q]\!](\mathbf{I}) \;=\; [\![P(S)]\!](\mathbf{I})$$

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

Beispiel: Sei $\mathbf{R} = \{Q, R\}$. Betrachte folgendes regelbasierte konjunktive Programm P:

$$\begin{aligned} S_1(x,z) &\leftarrow Q(x,y),\ R(y,z,w) \\ S_2(x,y,z) &\leftarrow S_1(x,w),\ R(w,y,v),\ S_1(v,z) \end{aligned}$$

Die von P durch S_2 definierte Anfrage ist äquivalent zu

$$\begin{aligned} S_2(x,y,z) \leftarrow\ & x{=}x',\ w{=}z',\ Q(x',y'),\ R(y',z',w'), \\ & R(w,y,v), \\ & v{=}x'',\ z{=}z'',\ Q(x'',y''),\ R(y'',z'',w'') \end{aligned}$$

Konjunktive Anfragen

Auswertungskomplexität konjunktiver Anfragen

AUSWERTUNGSPROBLEM FÜR CQ: (kombinierte Komplexität)

Eingabe: Anfrage Q des konjunktiven Kalküls,
Datenbank $\mathbf{I}$ (von einem zu Q passenden Schema $\mathbf{R}$)

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

Schön wäre: Algorithmus, der zur Lösung dieses Problems mit Zeit polynomiell in "Größe der Eingabe + Größe der Ausgabe" auskommt.

Frage: Gibt es einen solchen Algorithmus?

Größe der Eingabe: $k + n$, wobei

- $k := ||Q||$ die Länge der Anfrage
  (betrachtet als Wort über dem Alphabet $\mathbf{dom} \cup \mathbf{var} \cup \mathbf{R} \cup \{\exists, \wedge, (,), \langle, \rangle, \{, :, \}\}$)
- $n := ||\mathbf{I}||$ die Größe der Datenbank, also

$$n := ||\mathbf{I}|| := \sum_{R \in \mathbf{R}} ||\mathbf{I}(R)|| \quad \text{wobei} \quad ||\mathbf{I}(R)|| := \underbrace{\mathit{arity}(R) \cdot |\mathbf{I}(R)|}_{\text{Anzahl Tupel in } \mathbf{I}(R)}$$

Größe der Ausgabe: $||\ [\![Q]\!](\mathbf{I})\ ||$:= "Stelligkeit" · "Anzahl Tupel im Ergebnis"

Auswertung konjunktiver Anfragen

Proposition 2.18

Das Auswertungsproblem für CQ *lässt sich in Zeit* $\mathcal{O}\big((k+n)^k\big)$ *lösen.*

Beweis: siehe Tafel.

Bemerkung: Das ist exponentiell in der Länge der Anfrage.

Frage: Geht das effizienter?

Boolesche Anfragen

- Zur Erinnerung:

  Boolesche Anfragen sind “ja / nein”-Anfragen,
  d.h. Anfragen, deren Ergebnis die Stelligkeit 0 hat.

- Klar:

  Falls wir zeigen können, dass das Auswertungsproblem für Boolesche Anfragen des konjunktiven Kalküls schwierig ist, so ist es auch für allgemeine Anfragen des konjunktiven Kalküls schwierig.

- Wir werden sehen, dass umgekehrt aber auch gilt:

  Falls wir einen Algorithmus haben, der das Auswertungsproblem für *Boolesche* Anfragen des konjunktiven Kalküls löst, so können wir diesen Algorithmus verwenden, um das Auswertungsproblem für *beliebige* Anfragen des konjunktiven Kalküls zu lösen.

Algorithmen mit Verzögerung $f(k, n)$

Definition 2.19

Sei $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, sei $\mathcal{A}$ eine Datenbankanfragesprache und sei $\mathbb{A}$ irgendein Algorithmus, der bei allen Eingaben terminiert.

Wir sagen:

> das Auswertungsproblem für $\mathcal{A}$ kann unter Rückgriff auf $\mathbb{A}$ mit Verzögerung $f(k, n)$ gelöst werden,

falls es einen Algorithmus $\mathbb{B}$ gibt, der bei Eingabe einer Anfrage Q aus $\mathcal{A}$ und einer Datenbank $\mathbf{I}$ nach und nach genau die Tupel aus $[\![Q]\!](\mathbf{I})$ ausgibt und

- vor der Ausgabe des ersten Tupels,
- zwischen der Ausgabe von je zwei aufeinanderfolgenden Tupeln,
- nach der Ausgabe des letzten Tupels

je höchstens $f(||Q||, ||\mathbf{I}||)$ viele Elementarschritte oder Schritte, in denen der Algorithmus $\mathbb{A}$ aufgerufen wird, macht.

Boolesche Anfragen $\rightsquigarrow$ beliebige Anfragen

Theorem 2.20

Sei $\mathbb{A}$ ein Algorithmus, der das Auswertungsproblem für Boolesche Anfragen des konjunktiven Kalküls löst.

Dann gibt es einen Algorithmus $\mathbb{B}$, der das Auswertungsproblem für (beliebige) Anfragen des konjunktiven Kalküls unter Rückgriff auf $\mathbb{A}$ mit Verzögerung $\mathcal{O}(k^3 \cdot n)$ löst.

Beweis: siehe Tafel.

Folgerung: Falls wir das Auswertungsproblem für *Boolesche* Anfragen des konjunktiven Kalküls effizient lösen können, dann können wir es auch für *beliebige* Anfragen des konjunktiven Kalküls effizient lösen.

Die Komplexitätsklasse NP

Zur Erinnerung:

- Komplexitätsklassen bestehen aus Entscheidungsproblemen, d.h. Problemen mit "ja/nein"-Ausgabe
  - $\rightsquigarrow$ das Auswertungsproblem für Boolesche Anfragen passt gut zum Konzept der klassischen Komplexitätstheorie;
  - $\rightsquigarrow$ das Auswertungsproblem für beliebige Anfragen nicht

- Ein Entscheidungsproblem B gehört zur Klasse NP, falls es eine nichtdeterministische Turingmaschine T und eine Konstante c gibt, so dass für jede Zahl N und jede zum Problem B passende Eingabe w der Größe N gilt:

  (1) Jeder Lauf von T bei Eingabe w hat die Länge $\leqslant N^c$ und endet mit der Ausgabe "ja" oder "nein".

  (2) Ist w eine "ja"-Instanz für B, so gibt es mindestens einen Lauf von T auf w, der mit der Ausgabe "ja" endet.

  (3) Ist w eine "nein"-Instanz für B, so endet jeder Lauf von T auf w mit der Ausgabe "nein".

NP-Vollständigkeit

Zur Erinnerung:

- Ein Entscheidungsproblem B heißt NP-vollständig (bzgl. Polynomialzeit-Reduktionen), falls gilt:

  (1) $B \in \mathrm{NP}$ und
  (2) B ist NP-hart, d.h. für jedes Problem $A \in \mathrm{NP}$ gibt es eine Polynomialzeit-Reduktion f von A auf B (kurz: $f : A \leqslant_p B$)

- Eine Polynomialzeit-Reduktion f von A auf B ist eine (deterministisch) in polynomieller Zeit berechenbare Funktion, die jede zum Problem A passende Eingabe w auf eine zum Problem B passende Eingaben $f(w)$ abbildet, so dass gilt

  $$w \text{ ist eine “ja”-Instanz für } A \iff f(w) \text{ ist eine “ja”-Instanz für } B.$$

  Anschaulich bedeutet $A \leqslant_p B$, dass A “höchstens so schwer” wie B ist. NP-vollständige Probleme sind also “die schwierigsten Probleme” in NP.

- Der Begriff Polynomialzeit-Reduktion ist so definiert, dass folgendes gilt (wobei P die Klasse aller Entscheidungsprobleme bezeichnet, die deterministisch in Polynomialzeit lösbar sind)
  - $A \leqslant_p B$ und $B \in \mathrm{P} \implies A \in \mathrm{P}$
  - $A \notin \mathrm{P}$ und $A \leqslant_p B \implies B \notin \mathrm{P}$
  - A NP-hart und $A \leqslant_p B \implies B$ NP-hart.

Auswertungskomplexität konjunktiver Anfragen

Theorem 2.21 (Chandra, Merlin, 1977)
Das Auswertungsproblem für Boolesche regelbasierte konjunktive Anfragen ist NP*-vollständig.* *(kombinierte Komplexität)*

Beweis: siehe Tafel . . .

Zum Nachweis der NP-Härte benutzen wir das folgende Resultat, das Sie bereits aus der Veranstaltung Algorithmentheorie kennen:

Theorem: CLIQUE *ist* NP*-vollständig.*

Das Problem CLIQUE ist dabei folgendermaßen definiert:

CLIQUE

Eingabe: Ein endlicher ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$

Frage: Enthält G eine k-Clique?
(D.h. gibt es k verschiedene Knoten in G, von denen jeder mit jedem durch eine Kante verbunden ist?)

endlicher ungerichteter Graph:
jede Kante in E ist eine 2-elementige Teilmenge von V.

Folgerung aus der NP-Vollständigkeit

Folgerung:

Falls P $\neq$ NP, so gibt es keinen Algorithmus, der das Auswertungsproblem für Boolesche Anfragen des konjunktiven Kalküls deterministisch in Zeit $(k+n)^{\mathcal{O}(1)}$ löst.

Bemerkung 2.22 (hier ohne Beweis)

Unter Verwendung einer stärkeren Annahme aus der Parametrischen Komplexitätstheorie lässt sich folgendes zeigen:

> *Theorem (Papadimitriou, Yannakakis, 1997)*
>
> *Falls* FPT $\neq$ W[1]*, so gibt es keinen Algorithmus, der das Auswertungsproblem für Boolesche Anfragen des konjunktiven Kalküls deterministisch in Zeit* $f(k)\cdot n^c$ *löst*
>
> *(wobei* f *irgendeine berechenbare Funktion und* c *irgendeine Konstante ist).*

FPT und W[1] sind Komplexitätsklassen, die in der parametrischen Komplexitätstheorie Rollen spielen, die in etwa mit den Rollen von P und NP in der klassischen Komplexitätstheorie vergleichbar sind.

Konjunktive Anfragen

Algebraische Anfragen — Informell

Beispiel-Anfrage:

(4) Welche Kinos (Name & Adresse) spielen einen Film mit “George Clooney”?

Als Anfrage in der SPJR-Algebra:

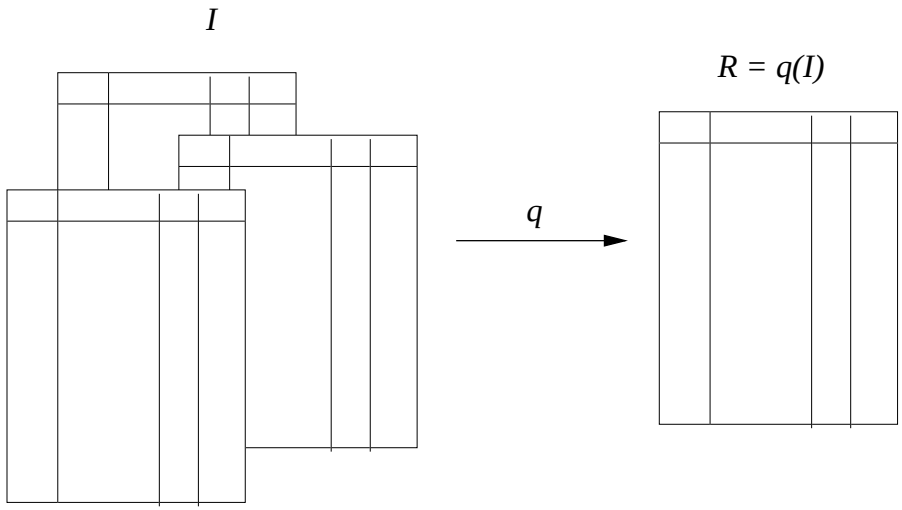
$$\pi_{Kino,Adresse}\left(\sigma_{\substack{Schauspieler = \\ \text{“George Clooney”}}}\left(Filme \bowtie Programm \bowtie Orte\right)\right)$$

Als Anfrage in der SPC-Algebra:

$$\pi_{7,8}\left(\sigma_{4=7}\left(\sigma_{1=5}\left(\sigma_{3 = \text{“George Clooney”}}\left(Filme \times Programm \times Orte\right)\right)\right)\right)$$

SPC-Algebra bzw. SPJR-Algebra

Zur Erinnerung:



- Mathematik:

  Algebraische Struktur $\widehat{=}$ Grundmenge + Operationen

- Hier:
  - Operationen auf (endlichen) Relationen
  - Speziell: Projektion, Selektion,
    Cartesisches Produkt bzw. Join und Umbenennung

Unbenannte Perspektive: Die SPC-Algebra (1/4)

Operationen: Selektion σ, Projektion π, Cartesisches Produkt $\times$

Selektion: Zwei Varianten:

- Operator $\sigma_{j=a}$, für eine Konstante $a \in$ **dom** und eine natürliche Zahl $j \geqslant 1$.

  Dieser Operator kann angewendet werden auf Relationen I der Stelligkeit $\geqslant j$ und liefert als Ausgabe die folgende Teilmenge der Relation I:

$$\sigma_{j=a}(I) := \{t \in I : \text{in der } j\text{-ten Komponente von } t \text{ steht ein } a\}$$

- Operator $\sigma_{j=k}$, für zwei natürliche Zahlen $j, k \geqslant 1$.

  Dieser Operator kann angewendet werden auf Relationen I der Stelligkeit $\geqslant \max(j,k)$ und liefert als Ausgabe die folgende Teilmenge der Relation I:

$$\sigma_{j=k}(I) := \left\{ t \in I \ : \begin{array}{l} \text{in der } j\text{-ten Komponente von } t \text{ steht derselbe Eintrag} \\ \text{wie in der } k\text{-ten Komponente von } t \end{array} \right\}$$

Selektion ist eine “horizontale” Operation: sie wählt einzelne Tabellen-Zeilen aus.

“$j=a$” und “$j=k$” werden Selektionsbedingungen genannt.

Unbenannte Perspektive: Die SPC-Algebra (2/4)

Operationen: Selektion σ, Projektion π, Cartesisches Produkt $\times$

Projektion:

Operator $\pi_{j_1,\ldots,j_k}$, für (nicht notwendigerweise paarweise verschiedene) natürliche Zahlen $j_1,\ldots,j_k \geqslant 1$ (und $k \geqslant 0$).

Dieser Operator kann angewendet werden auf Relationen I der Stelligkeit $\geqslant \max\{j_1,\ldots,j_k\}$ und liefert als Ausgabe die folgende Relation der Stelligkeit k:

$$\pi_{j_1,\ldots,j_k}(I) \;:=\; \left\{ \left\langle t(j_1),\ldots,t(j_k)\right\rangle \;:\; t \in I \right\}$$

Projektion ist eine “vertikale” Operation: sie wählt einzelne Tabellen-Spalten aus und arrangiert sie in möglicherweise anderer Reihenfolge

Unbenannte Perspektive: Die SPC-Algebra (3/4)

Operationen: Selektion σ, Projektion π, Cartesisches Produkt $\times$

Cartesisches Produkt:

Operator $\times$

Dieser Operator kann angewendet werden auf Relationen I und J beliebiger Stelligkeiten m und n und liefert als Ausgabe die folgende Relation der Stelligkeit $m+n$:

$$I \times J := \left\{ \left\langle t(1), \ldots, t(m), s(1), \ldots, s(n) \right\rangle : t \in I \text{ und } s \in J \right\}$$

Wir benutzen den Operator manchmal auch für einzelne Tupel:
Sind t und s Tupel der Stelligkeiten m und n, so schreiben wir $t \times s$, um das Tupel $\langle t(1), \ldots, t(m), s(1), \ldots, s(n) \rangle$ zu bezeichnen.

Unbenannte Perspektive: Die SPC-Algebra (4/4)

Definition 2.23

Sei **R** ein Datenbankschema. Die Klasse der Anfragen der SPC-Algebra über **R** (kurz: SPC[**R**]) ist induktiv wie folgt definiert:

- Für alle Relations-Namen $R \in \mathbf{R}$ ist R eine SPC[**R**]-Anfrage der Stelligkeit $\mathit{arity}(R)$.
- Für alle Konstanten $c \in \mathbf{dom}$ ist $\{\langle c \rangle\}$ eine SPC[**R**]-Anfrage der Stelligkeit 1.
- Ist Q eine SPC[**R**]-Anfrage der Stelligkeit m, sind j, k natürliche Zahlen aus $\{1, \ldots, m\}$ und ist $a \in \mathbf{dom}$ eine Konstante, so sind auch $\sigma_{j=a}(Q)$ und $\sigma_{j=k}(Q)$ SPC[**R**]-Anfragen der Stelligkeit m.
- Ist Q eine SPC[**R**]-Anfrage der Stelligkeit m, ist $k \geqslant 0$ und sind $j_1, \ldots, j_k$ natürliche Zahlen aus $\{1, \ldots, m\}$, so ist $\pi_{j_1, \ldots, j_k}(Q)$ eine SPC[**R**]-Anfrage der Stelligkeit k.
- Sind Q und P zwei SPC[**R**]-Anfragen der Stelligkeiten m und n, so ist $(Q \times P)$ eine SPC[**R**]-Anfrage der Stelligkeit $m+n$.

Die Semantik $[\![Q]\!]$ von SPC[**R**]-Anfragen Q ist induktiv auf die offensichtliche Art definiert.

Verallgemeinerung des Selektions-Operators

Eine positive konjunktive Selektionsbedingung ist eine Formel F der Form

$$\gamma_1 \wedge \cdots \wedge \gamma_n$$

wobei $n \geqslant 1$ und jedes γ_i eine Selektionsbedingung der Form $j_i{=}a_i$ oder $j_i{=}k_i$ für natürliche Zahlen $j_i, k_i \geqslant 1$ und Konstanten $a_i \in \mathbf{dom}$.

Der Selektionsoperator σ_F hat dieselbe Wirkung wie die Hintereinanderausführung der Selektionsoperatoren σ_{γ_i} für alle $i \in \{1, \ldots, n\}$.

Eine Normalform für SPC-Anfragen

Sei **R** ein Relationsschema.

Definition 2.24

Eine SPC[**R**]-Anfrage ist in Normalform, falls sie von der Form

$$\pi_{j_1,\ldots,j_k}\Big(\{\langle c_1\rangle\}\times\cdots\times\{\langle c_m\rangle\}\times\sigma_F\big(R_1\times\cdots\times R_\ell\big)\Big)$$

ist, für $k, m, \ell \geqslant 0$, paarweise verschiedene Elemente $j_1,\ldots,j_k$, so dass $\{1,\ldots,m\}\subseteq\{j_1,\ldots,j_k\}$, Konstanten $c_1,\ldots,c_m \in \mathbf{dom}$, $R_1,\ldots,R_\ell \in \mathbf{R}$ und F eine positive konjunktive Selektionsbedingung.

Proposition 2.25

Für jede SPC[**R**]*-Anfrage Q gibt es eine* SPC[**R**]*-Anfrage Q' in Normalform, die dieselbe Anfragefunktion definiert.*

Beweis: Übung.

Benannte Perspektive: Die SPJR-Algebra (1/7)

Operationen: Selektion σ, Projektion π, Join $\bowtie$, Umbenennung (Renaming) δ

Attribut-Namen an Stelle von Spaltennummern !

Selektion: Zwei Varianten:

- Operator $\sigma_{A=a}$, für eine Konstante $a \in \mathbf{dom}$ und einen Attribut-Namen A.

  Dieser Operator kann angewendet werden auf R-Relationen I mit $A \in sort(R)$ und liefert als Ausgabe die folgende Teilmenge der Relation I:

$$\sigma_{A=a}(I) := \{ t \in I : t(A) = a \}$$

- Operator $\sigma_{A=B}$, für zwei Attribut-Namen A und B.

  Dieser Operator kann angewendet werden auf R-Relationen I mit $A, B \in sort(R)$ und liefert als Ausgabe die folgende Teilmenge der Relation I:

$$\sigma_{A=B}(I) := \{ t \in I : t(A) = t(B) \}$$

"$A{=}a$" und "$A{=}B$" werden Selektionsbedingungen genannt.

Benannte Perspektive: Die SPJR-Algebra (2/7)

Operationen: Selektion σ, Projektion π, Join $\bowtie$, Umbenennung (Renaming) δ

Projektion:

Operator $\pi_{A_1,\ldots,A_k}$, für paarweise verschiedene Attribut-Namen $A_1, \ldots, A_k$ (und $k \geqslant 0$).

Dieser Operator kann angewendet werden auf R-Relationen I mit $sort(R) \supseteq \{A_1, \ldots, A_k\}$ und liefert als Ausgabe die folgende Relation der Sorte $\{A_1, \ldots, A_k\}$:

$$\pi_{A_1,\ldots,A_k}(I) := \left\{ \left\langle A_1 : t(A_1),\ \ldots,\ A_k : t(A_k) \right\rangle \ :\ t \in I \right\}$$

Benannte Perspektive: Die SPJR-Algebra (3/7)

Operationen: Selektion σ, Projektion π, Join $\bowtie$, Umbenennung (Renaming) δ

An Stelle des Cartesischen Produkts: Natürlicher Join

Beispiel: Wenn I und J zwei Relationen mit disjunkten Attributmengen (d.h. Spaltenbezeichnungen) sind, so bilde einfach das herkömmliche Cartesische Produkt.

I

A	B
1	a
2	b
3	c

$\bowtie$

J

C	D	E
e	f	g
h	i	j

$=$

$I \bowtie J$

A	B	C	D	E
1	a	e	f	g
1	a	h	i	j
2	b	e	f	g
2	b	h	i	j
3	c	e	f	g
3	c	h	i	j

Frage: Was soll passieren, wenn I und J gemeinsame Attribute haben?

Antwort: Zueinander passende Tupel werden verschmolzen.

Beispiel:

I

A	B
1	a
2	b
3	c

$\bowtie$

J

B	C	D
a	f	g
a	i	j
c	l	m

$=$

$I \bowtie J$

A	B	C	D
1	a	f	g
1	a	i	j
3	c	l	m

Benannte Perspektive: Die SPJR-Algebra (4/7)

Operationen: Selektion σ, Projektion π, Join $\bowtie$, Umbenennung (Renaming) δ

Natürlicher Join:

Operator $\bowtie$

Dieser Operator kann angewendet werden auf eine R-Relation I und eine S-Relation J beliebiger Sorten und liefert als Ausgabe die folgende Relation der Sorte $\Sigma := sort(R) \cup sort(S)$:

$$I \bowtie J := \Big\{ \text{Tupel } t \text{ der Sorte } \Sigma : \text{es gibt Tupel } t' \in I \text{ und } t'' \in J \text{ so dass } t|_{sort(R)} = t' \text{ und } t|_{sort(S)} = t'' \Big\}$$

Wir benutzen den Operator manchmal auch für einzelne Tupel:
Sind t' und t'' Tupel der Sorten $sort(R)$ und $sort(S)$, so schreiben wir $t' \bowtie t''$, um das Tupel t der Sorte $sort(R) \cup sort(S)$ mit $t|_{sort(R)} = t'$ und $t|_{sort(S)} = t''$ zu bezeichnen (bzw. den Wert *"undefiniert"*, falls es kein solches Tupel gibt, d.h. falls $t'|_{sort(R) \cap sort(S)} \neq t''|_{sort(R) \cap sort(S)}$).

Benannte Perspektive: Die SPJR-Algebra (5/7)

Operationen: Selektion σ, Projektion π, Join $\bowtie$, Umbenennung (Renaming) δ

Umbenennung (Renaming):

Operator δ_f, für eine Umbenennungsfunktion f, d.h. eine injektive Funktion $f : U \to \mathbf{att}$, für eine beliebige endliche Menge U von Attribut-Namen.

Dieser Operator kann angewendet werden auf R-Relationen I, für die gilt: $U \subseteq sort(R)$ und $\big(sort(R) \setminus U\big) \cap f(U) = \emptyset$ und liefert die folgende Relation der Sorte $f(U) \cup \big(sort(R) \setminus U\big)$:

$$\delta_f(I) := \left\{ \text{Tupel } t \ : \begin{array}{l} \text{ex } t' \in I \text{ so dass f.a. } A \in U \text{ gilt } t'(A) = t(f(A)) \\ \text{und f.a. } A \in sort(R) \setminus U \text{ gilt } t'(A) = t(A) \end{array} \right\}$$

Oft schreiben wir $A_1 \cdots A_k \mapsto B_1 \cdots B_k$ um die Umbenennungsfunktion f mit Definitionsbereich $U = \{A_1, \ldots, A_k\}$ und Werten $f(A_i) = B_i$, für alle $i \in \{1, \ldots, k\}$, zu bezeichnen.

Benannte Perspektive: Die SPJR-Algebra (6/7)

Definition 2.26

Sei **R** ein Datenbankschema. Die Klasse der Anfragen der SPJR-Algebra über **R** (kurz: SPJR[**R**]) ist induktiv wie folgt definiert:

- Für alle Relations-Namen $R \in \mathbf{R}$ ist R eine SPJR[**R**]-Anfrage der Sorte $sort(R)$.
- Für alle Konstanten $c \in \mathbf{dom}$ und alle Attribut-Namen $A \in \mathbf{att}$ ist $\{\langle A : c \rangle\}$ eine SPJR[**R**]-Anfrage der Sorte $\{A\}$.
- Ist Q eine SPJR[**R**]-Anfrage der Sorte Σ, sind $A, B \in \Sigma$ und ist $a \in \mathbf{dom}$ eine Konstante, so sind auch $\sigma_{A=a}(Q)$ und $\sigma_{A=B}(Q)$ SPJR[**R**]-Anfragen der Sorte Σ.
- Ist Q eine SPJR[**R**]-Anfrage der Sorte Σ, ist $k \geqslant 0$ und sind $A_1, \ldots, A_k$ paarweise verschiedene Elemente aus Σ, so ist $\pi_{A_1,\ldots,A_k}(Q)$ eine SPJR[**R**]-Anfrage der Sorte $\{A_1, \ldots, A_k\}$.
- Sind Q und P zwei SPJR[**R**]-Anfragen der Sorten Σ und Π, so ist $(Q \bowtie P)$ eine SPJR[**R**]-Anfrage der Sorte $\Sigma \cup \Pi$.
- Ist Q eine SPJR[**R**]-Anfrage der Sorte Σ und ist $f : \Sigma \to \mathbf{att}$ eine Umbenennungsfunktion, so ist $\delta_f(Q)$ eine SPJR[**R**]-Anfrage der Sorte $f(\Sigma)$.

Benannte Perspektive: Die SPJR-Algebra (7/7)

Die Semantik $[\![Q]\!]$ von SPJR[**R**]-Anfragen Q ist induktiv auf die offensichtliche Art definiert.

Wie bei der SPC-Algebra lassen wir wieder eine Verallgemeinerung des Selektions-Operators zu:

- Eine positive konjunktive Selektionsbedingung ist eine Formel F der Form

$$\gamma_1 \wedge \cdots \wedge \gamma_n$$

  wobei $n \geqslant 1$ und jedes γ_i eine Selektionsbedingung der Form $A_i{=}a_i$ oder $A_i{=}B_i$ für Attribut-Namen A_i, B_i und Konstanten $a_i \in \mathbf{dom}$.

- Der Selektionsoperator σ_F hat dieselbe Wirkung wie die Hintereinanderausführung der Selektionsoperatoren σ_{γ_i} für alle $i \in \{1, \ldots, n\}$.

Eine Normalform für SPJR-Anfragen

Sei **R** ein Relationsschema.

Definition 2.27

Eine SPJR[**R**]-Anfrage ist in Normalform, falls sie von der Form

$$\pi_{B_1,\ldots,B_k}\Big(\{\langle A_1 : c_1\rangle\} \bowtie \cdots \bowtie \{\langle A_m : c_m\rangle\} \bowtie \sigma_F\big(\delta_{f_1}(R_1) \bowtie \cdots \bowtie \delta_{f_\ell}(R_\ell)\big)\Big)$$

ist, für $k, m, \ell \geqslant 0$, $B_1, \ldots, B_k, A_1, \ldots, A_m \in \mathbf{att}$ so dass $\{A_1, \ldots, A_m\} \subseteq \{B_1, \ldots, B_k\}$ und die $A_1, \ldots, A_m$ paarweise verschieden, $c_1, \ldots, c_m \in \mathbf{dom}$, $R_1, \ldots, R_\ell \in \mathbf{R}$, eine positive konjunktive Selektionsbedingung F und Umbenennungsfunktionen $f_1, \ldots, f_\ell$, so dass die Sorten von $\delta_{f_1}(R_1), \ldots, \delta_{f_\ell}(R_\ell)$ paarweise disjunkt sind und keins der $A_1, \ldots, A_m$ als Attribut von einem der $\delta_{f_j}(R_j)$ vorkommt.

Proposition 2.28

Für jede SPJR[**R**]*-Anfrage Q gibt es eine* SPJR[**R**]*-Anfrage Q' in Normalform, die dieselbe Anfragefunktion definiert.*

Beweis: Übung.

Nicht-Erfüllbare Anfragen

Bemerkung:

Sowohl in der SPC-Algebra als auch in der SPJR-Algebra lassen sich unerfüllbare Anfragen ausdrücken.

Beispiel: Die Anfrage $Q :=$

$$\sigma_{3=\text{“George Clooney”}}\Big(\sigma_{3=\text{“Wolfgang Völz”}}\big(\mathit{Filme}\big)\Big)$$

wählt in einer Datenbank vom Schema **KINO** genau diejenigen Tupel $t = (a, b, c)$ aus der *Filme*-Relation aus, für deren dritte Komponente c gilt: $c =$ “George Clooney” und $c =$ “Wolfgang Völz”.

Solche Tupel kann es aber nicht geben!

Für jede Datenbank **I** vom Schema **KINO** gilt also: $[\![Q]\!](\mathbf{I}) = \emptyset$.

Somit ist die Anfrage Q nicht erfüllbar.

Äquivalenz der Ausdrucksstärke der SPC-Algebra und der SPJR-Algebra

Lemma 2.29

Die SPC-Algebra und die SPJR-Algebra können genau dieselben Anfragen ausdrücken.

Es gilt sogar: Jede Anfrage aus einer dieser Anfragesprachen kann in polynomieller Zeit in eine äquivalente Anfrage der anderen Anfragesprache übersetzt werden.

Beweis: siehe Tafel.

Äquivalenz des deskriptiven und des algebraischen Ansatzes

Theorem 2.30 (Äquivalenz konjunktiver Anfragesprachen)

Die folgenden Anfragesprachen können genau dieselben erfüllbaren Anfragefunktionen ausdrücken:

(a) *die Klasse der regelbasierten konjunktiven Anfragen*

(b) *die Klasse der Tableau-Anfragen*

(c) *die Klasse der Anfragen des konjunktiven Kalküls*

(d) *die Anfragen der SPC-Algebra*

(e) *die Anfragen der SPJR-Algebra.*

Es gilt sogar: Jede Anfrage aus einer dieser Anfragesprachen kann in polynomieller Zeit in äquivalente Anfragen der anderen Anfragesprachen übersetzt werden.

Beweis: siehe Tafel.

Konjunktive Anfragen

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:

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

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

  (D.h. ein Tableau über **R** ist eine "Datenbank vom Schema **R**, die als Einträge auch Variablen enthalten kann".)

- Eine Tableau-Anfrage über **R** (bzw. R) ist von der Form $(\mathbf{T}, u)$, wobei **T** ein Tableau über **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:

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

- 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 $\mathbf{I} \in inst(\mathbf{R})$.

Vorbemerkung zum Thema “Statische Analyse, Optimierung”

Optimierung: Finde zur gegebenen Anfrage eine “minimale” äquivalente Anfrage

Definition 2.31 (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 $\mathbf{I} \in inst(\mathbf{R})$ gilt: $[\![Q]\!](\mathbf{I}) = [\![P]\!](\mathbf{I})$.

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

Statische Analyse:

- Äquivalenzproblem: Teste, bei Eingabe zweier Anfragen Q, P, ob $Q \equiv P$.
- Query Containment Problem: Teste, bei Eingabe zweier Anfragen Q, P, ob $Q \subseteq P$.
- Erfüllbarkeitsproblem: Teste, bei Eingabe einer Anfrage Q, ob Q erfüllbar ist (d.h. ob es eine DB $\mathbf{I}$ gibt, so dass $[\![Q]\!](\mathbf{I}) \neq \emptyset$ ist).

Bekannt:

- Für regelbas. konj. Anfragen ist das Erfüllbarkeitsproblem trivial (Satz 2.6).
- 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

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 2.32

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

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

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 $\alpha_C^{-1}\big(\alpha_C(b)\big) = b$.

Definition 2.33 *(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. $\mathbf{I}_Q^{Q'}(R) = \alpha_C\big(\mathbf{T}(R)\big)$, für alle $R \in \mathbf{R}$.

Proposition 2.34

*$Q' = (\mathbf{T}', u')$ 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 2.35 (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 2.36

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 $\big(Q \equiv Q' \iff (Q \subseteq Q' \text{ und } Q' \subseteq Q)\big)$ 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 2.37

Sei $\mathbf{R}$ 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 $\mathbf{T}$ 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 2.38 (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 2.39

(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.

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 2.40

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

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 2.41

(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 2.43 in Zeit $\mathcal{O}\big(||Q'||^2 \cdot ||\mathbf{I}|| \cdot \log(||Q'|| \cdot ||\mathbf{I}||)\big)$)

Semijoin-Anfragen

Definition 2.42

Sei **R** ein Datenbankschema. Die Klasse der Semijoin-Anfragen über **R** ist induktiv wie folgt definiert:

(A) Jedes Relations-Atom $R(v_1, \ldots, v_r)$, für $R \in \mathbf{R}$, $r := \mathit{arity}(R)$ 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 $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ 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 $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ 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 **R** ist von der Form $\pi_{\langle\rangle}(Q)$, wobei Q eine Semijoin-Anfrage über **R** ist.

Auswertung von Semijoin-Anfragen

Proposition 2.43

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 2.44

(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 2.45

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 2.45 (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 2.45 (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 2.46 (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 2.45, Lemma 2.44 (b) und Proposition 2.43.
- *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 2.47

Sei **R** ein Datenbankschema.

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

Konjunktive Sätze des Guarded Fragment sind Formeln aus GF(CQ)[**R**], die keine freien Variablen besitzen.

Azyklische Boolesche Konjunktive Anfragen

Satz 2.48

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

Beweis: (a) $\Longleftrightarrow$ (b): Lemma 2.44. (a) $\Longleftrightarrow$ (c): Übung.

Konjunktive Anfragen

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 2.49

Sei **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 $Q := \quad Ans(u) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$ 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:

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

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: $\quad Q \equiv_b Q' \iff \Big(Q \subseteq_b Q' \text{ und } Q' \subseteq_b Q \Big)$

  Insbes: Äquivalenz ist höchstens so schwer wie Query Containment.

Ergebnisse

Theorem 2.50 (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 2.37).*

(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 2.51 (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