

Konjunktive Anfragen (I)

- 2.1 Deskriptiver Ansatz: regelbasiert, graphisch und logikbasiert
- 2.2 Auswertungskomplexität
- 2.3 Algebraischer Ansatz: SPC-Algebra und SPJR-Algebra

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

Als **regelbasierte konjunktive Anfrage**:

$$\text{Ans}(x_{\text{Kino}}, x_{\text{Adr}}) \leftarrow \text{Filme}(x_{\text{Titel}}, x_{\text{Regie}}, \text{"George Clooney"}), \\ \text{Programm}(x_{\text{Kino}}, x_{\text{Titel}}, x_{\text{Zeit}}), \\ \text{Orte}(x_{\text{Kino}}, x_{\text{Adr}}, x_{\text{Tel}})$$

Konjunktive Anfragen (I)

- 2.1 Deskriptiver Ansatz: regelbasiert, graphisch und logikbasiert
- 2.2 Auswertungskomplexität
- 2.3 Algebraischer Ansatz: SPC-Algebra und SPJR-Algebra

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, \dots)
- ▶ Ein **Term** ist ein Element aus $\text{var} \cup \text{dom}$.
- ▶ Ein **freies Tupel** der Stelligkeit k ist ein Element aus $(\text{var} \cup \text{dom})^k$.

Definition 2.2

Sei \mathbf{R} ein Datenbankschema.

Eine **regelbasierte konjunktive Anfrage** über \mathbf{R} ist ein Ausdruck der Form

$$\text{Ans}(u) \leftarrow R_1(u_1), \dots, R_\ell(u_\ell)$$

wobei $\ell \geq 0$, $R_1, \dots, R_\ell \in \mathbf{R}$, $\text{Ans} \in \text{relname} \setminus \mathbf{R}$ und u, u_1, \dots, u_ℓ freie Tupel der Stelligkeiten $\text{arity}(\text{Ans}), \text{arity}(R_1), \dots, \text{arity}(R_\ell)$, so dass jede Variable, die in u vorkommt, auch in mindestens einem der Tupel u_1, \dots, u_ℓ vorkommt.

Semantik regelbasierter konjunktiver Anfragen

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

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

- ▶ Mit $Var(Q)$ bezeichnen wir die Menge aller Variablen, die in Q vorkommen.
- ▶ Eine **Belegung** für Q ist eine Abbildung $\beta : Var(Q) \rightarrow \mathbf{dom}$. Wir setzen β auf natürliche Weise fort zu einer Abbildung von $Var(Q) \cup \mathbf{dom}$ nach \mathbf{dom} , so dass $\beta|_{\mathbf{dom}} = \text{id}$. Für ein freies Tupel $u = \langle e_1, \dots, e_k \rangle$ setzen wir $\beta(u) := \langle \beta(e_1), \dots, \beta(e_k) \rangle$.
- ▶ Der Anfrage Q ordnen wir die folgende Anfragefunktion $\llbracket Q \rrbracket$ zu:

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

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

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 $\llbracket Q_1 \rrbracket(\mathbf{I}) = \{\langle a, c, e \rangle, \langle a, a, c \rangle, \langle c, a, c \rangle\}$.

- ▶ 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 $\llbracket Q_2 \rrbracket(\mathbf{I}) = \{\langle a, b \rangle, \langle c, b \rangle\}$.

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:

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

- ▶ Die Anfrage (5) Lläuft zur Zeit ein "Detlev Buck" Film?

lässt sich durch folgende regelbasierte konjunktive Anfrage ausdrücken:

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

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.

Bezeichnungen

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

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

- ▶ $Ans(u)$ wird als **Kopf** der Regel bezeichnet.
- ▶ $R_1(u_1), \dots, R_\ell(u_\ell)$ wird als **Rumpf** der Regel bezeichnet.
- ▶ Die Relations-Namen aus \mathbf{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 \mathbf{dom}), die in Q vorkommen. ("adom" steht für "active domain")

Der “Active Domain” einer Datenbank

Definition 2.3

Sei \mathbf{R} ein Datenbankschema und sei \mathbf{I} eine Datenbank vom Schema \mathbf{R} .

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

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

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

Ist Q eine Anfrage und \mathbf{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 \mathbf{I} (vom passenden DB-Schema) gilt: $adom(\llbracket Q \rrbracket(\mathbf{I})) \subseteq adom(Q, \mathbf{I})$.

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!

Monotonie und Erfüllbarkeit

Sind \mathbf{I} und \mathbf{J} zwei Datenbanken vom gleichen Schema \mathbf{R} , so sagen wir “ **\mathbf{J} ist eine Erweiterung von \mathbf{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 \mathbf{I} vorkommt, kommt auch in der entsprechenden Relation von \mathbf{J} vor).

Definition 2.5

Sei \mathbf{R} ein DB-Schema und sei q eine Anfragefunktion über \mathbf{R} .

- (a) q heißt **monoton**, falls für alle Datenbanken \mathbf{I} und \mathbf{J} über \mathbf{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 \mathbf{I} gibt mit $q(\mathbf{I}) \neq \emptyset$.

Satz 2.6

Jede **regelbasierte konjunktive Anfrage** ist **monoton** und **erfüllbar**.

Beweis: siehe Tafel.

“Graphische” Variante: Tableau-Anfragen

Beispiel-Anfrage:

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

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

<i>Filme</i>	Titel	Regie	Schauspieler
	x_{Titel}	x_{Regie}	“George Clooney”

<i>Programm</i>	Kino	Titel	Zeit
	x_{Kino}	x_{Titel}	x_{Zeit}

<i>Orte</i>	Kino	Adresse	Telefon
	x_{Kino}	x_{Adr}	x_{Tel}

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

Tableaus — 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 \mathbf{dom} , sondern auch Variablen aus \mathbf{var} haben kann".)
- ▶ Ein **Tableau \mathbf{T} über \mathbf{R}** ist eine Abbildung, die jedem $R \in \mathbf{R}$ ein Tableau über 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) .

Logikbasierte Variante: Konjunktiver Kalkül

Beispiel-Anfrage:

- (4) Welche Kinos (Name & Adresse) spielen einen Film mit "George Clooney"?

Formulierung als Anfrage des konjunktiven Kalküls:

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

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.

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 $\beta : Var(Q) \rightarrow \mathbf{dom}$.
- ▶ 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 $\llbracket Q \rrbracket$ zu:

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

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

Konjunktiver Kalkül (CQ) — Präzise

Definition 2.8

Sei \mathbf{R} ein Datenbankschema.

Die Menge $CQ[\mathbf{R}]$ aller Formeln des **konjunktiven Kalküls über \mathbf{R}** ist induktiv wie folgt definiert: (CQ steht für "conjunctive queries")

- (A) $R(v_1, \dots, v_r)$ gehört zu $CQ[\mathbf{R}]$,
für alle $R \in \mathbf{R}$, $r := arity(R)$ und $v_1, \dots, v_r \in \mathbf{var} \cup \mathbf{dom}$.
- (K) $(\varphi \wedge \psi)$ gehört zu $CQ[\mathbf{R}]$,
für alle $\varphi \in CQ[\mathbf{R}]$ und $\psi \in CQ[\mathbf{R}]$.
- (E) $\exists x \varphi$ gehört zu $CQ[\mathbf{R}]$,
für alle $\varphi \in CQ[\mathbf{R}]$ und $x \in \mathbf{var}$.

Insbesondere: Jede Formel in $CQ[\mathbf{R}]$ ist eine Formel der **Logik erster Stufe** über der Signatur $\mathbf{R} \cup \mathbf{dom}$ (wobei jedes Element aus \mathbf{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.: $frei(R(v_1, \dots, v_r)) = \{v_1, \dots, v_r\} \cap var$; $frei((\varphi \wedge \psi)) = frei(\varphi) \cup frei(\psi)$;
 $frei(\exists x\varphi) = frei(\varphi) \setminus \{x\}$.
- ▶ Eine Belegung für φ ist eine Abbildung $\beta : frei(\varphi) \rightarrow dom$.
- ▶ Einer Datenbank I vom Schema R ordnen wir die logische Struktur

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

zu. (Insbesondere ist \mathcal{A}_I eine σ -Struktur über der Signatur $\sigma := R \cup dom$.)

- ▶ Ist I eine Datenbank vom Schema R und β eine Belegung für φ , so sagen wir " I erfüllt φ unter β " und schreiben $I \models \varphi[\beta]$ bzw. $(I, \beta) \models \varphi$, um auszudrücken, dass $(\mathcal{A}_I, \beta) \models \varphi$.

Konjunktiver Kalkül: Syntax und Semantik

Definition 2.9

Sei R ein Datenbankschema.

Eine Anfrage des konjunktiven Kalküls ist von der Form

$$\langle e_1, \dots, e_r \rangle : \varphi$$

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

Semantik:

Einer Anfrage Q der Form $\langle e_1, \dots, e_r \rangle : \varphi$ ordnen wir die folgende Anfragefunktion $\llbracket Q \rrbracket$ zu:

$$\llbracket Q \rrbracket(I) := \left\{ \beta(\langle e_1, \dots, e_r \rangle) : \beta \text{ ist eine Belegung für } \varphi \text{ mit } I \models \varphi[\beta] \right\}$$

für alle Datenbanken $I \in inst(R)$.

Notation

- ▶ Mit **CQ** bezeichnen wir die Klasse aller CQ[R]-Formeln für alle Datenbankschemata R .
- ▶ Manchmal schreiben wir $\{x_1/a_1, \dots, x_r/a_r\}$ um die Belegung $\beta : \{x_1, \dots, x_r\} \rightarrow dom$ zu bezeichnen mit $\beta(x_i) = a_i$, f.a. $i \in \{1, \dots, r\}$.
- ▶ Für eine CQ-Formel φ schreiben wir oft $\varphi(x_1, \dots, x_r)$, um anzudeuten, dass $frei(\varphi) = \{x_1, \dots, x_r\}$.
- ▶ Ist $a_1, \dots, a_r \in dom$, so schreiben wir vereinfachend $I \models \varphi[a_1, \dots, a_r]$ an Stelle von $I \models \varphi[\{x_1/a_1, \dots, x_r/a_r\}]$.
- ▶ Ist $y \in dom \cup var$, so bezeichnet $\varphi(x_1/y, x_2, \dots, x_r)$ 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 $\exists x_1, \dots, x_n$ als Abkürzung für $\exists x_1 \exists x_2 \dots \exists x_n$.
- ▶ Zwei CQ[R]-Formeln φ und ψ heißen äquivalent, falls $frei(\varphi) = frei(\psi)$ und für jede Datenbank $I \in inst(R)$ und jede Belegung β für φ (und ψ) gilt: $I \models \varphi[\beta] \iff I \models \psi[\beta]$.

Wertebereich von Anfragen des konjunktiven Kalküls

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

$$adom(Q) := adom(\varphi) \cup (\{e_1, \dots, e_r\} \cap dom).$$

Ist Q eine Anfrage und I eine Datenbank, so setzen wir – wie üblich –

$$adom(Q, I) := adom(Q) \cup adom(I)$$

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 I (vom passenden DB-Schema) gilt: $adom(\llbracket Q \rrbracket(I)) \subseteq adom(Q, I)$.

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:

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

und durch die dazu äquivalente Anfrage

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

Äquivalenz von Anfragesprachen

Definition 2.13

Seien Q_1 und Q_2 zwei Anfragesprachen. Wir schreiben

- ▶ $Q_1 \leq Q_2$ (bzw. $Q_2 \geq Q_1$; " Q_2 ist mindestens so ausdrucksstark wie Q_1 "), falls jede Anfragefunktion, die durch eine Anfrage in Q_1 ausgedrückt werden kann, auch durch eine Anfrage in Q_2 ausgedrückt werden kann.
- ▶ $Q_1 \equiv Q_2$ (" Q_1 und Q_2 haben dieselbe Ausdrucksstärke") falls $Q_1 \leq Q_2$ und $Q_2 \leq Q_1$
- ▶ $Q_1 < Q_2$ (bzw. $Q_2 > Q_1$; " Q_2 ist ausdrucksstärker als Q_1 ") falls $Q_1 \leq Q_2$ und nicht $Q_2 \leq Q_1$.

Eine Normalform für CQ

Definition 2.11

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

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

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

Analog ist eine Anfrage Q des konjunktiven Kalküls in Normalform, falls sie von der Form $\{\langle e_1, \dots, e_r \rangle : \varphi\}$ 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 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

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

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 $x=y_1, y_1=y_2, \dots, y_j=z$ 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.

↪ **bereichsbeschränkte regelbasierte konjunktive Anfragen mit "="**

Analog wird die Klasse $CQ^=$ aller **bereichsbeschränkten Formeln des konjunktiven Kalküls mit "="** definiert.

Beobachtung 2.15

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

Beispiel für eine $CQ^=$ -Anfrage, die nicht erfüllbar ist:

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

wobei a und b zwei verschiedene Elemente aus \mathbf{dom} sind.

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 $c \in \mathbf{dom}$.

↪ **regelbasierte konjunktive Anfragen mit "="**

↪ **Konjunktiver Kalkül mit "="**

Aber Vorsicht: Die Regel

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

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

$$\llbracket Q \rrbracket(I) = \{ \langle a, d \rangle : d \in \mathbf{dom} \} = \{a\} \times \mathbf{dom}$$

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

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

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 \mathbf{R} ein Datenbankschema.

Ein **regelbasiertes konjunktives Programm** über \mathbf{R} (mit oder ohne “=”) hat die Form

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

wobei $m \geq 1$, S_1, \dots, S_m sind paarweise verschiedene Relations-Namen aus $\text{relname} \setminus \mathbf{R}$ und für jedes $i \in \{1, \dots, m\}$ gilt:

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

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

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

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

$$\llbracket Q \rrbracket(\mathbf{I}) = \llbracket P(S) \rrbracket(\mathbf{I})$$

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

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

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

Die von P durch S_2 definierte Anfrage ist äquivalent zu

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

Semantik regelbasierter konjunktiver Programme

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

$$\llbracket P(S_1) \rrbracket(\mathbf{I}), \dots, \llbracket P(S_m) \rrbracket(\mathbf{I}),$$

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

$$\llbracket P(S_i) \rrbracket(\mathbf{I}) := \llbracket Q_i \rrbracket(\mathbf{J}_{i-1})$$

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

Konjunktive Anfragen (I)

2.1 Deskriptiver Ansatz: regelbasiert, graphisch und logikbasiert

2.2 Auswertungskomplexität

2.3 Algebraischer Ansatz: SPC-Algebra und SPJR-Algebra

Auswertungskomplexität konjunktiver Anfragen

AUSWERTUNGSPROBLEM FÜR CQ: (kombinierte Komplexität)

Eingabe: Anfrage Q des konjunktiven Kalküls,
Datenbank I (von einem zu Q passenden Schema R)

Aufgabe: Berechne $\llbracket Q \rrbracket(I)$

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

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

$$n := \llbracket I \rrbracket := \sum_{R \in R} \llbracket I(R) \rrbracket \quad \text{wobei} \quad \llbracket I(R) \rrbracket := \underbrace{\text{arity}(R)} \cdot \llbracket I(R) \rrbracket$$

Anzahl Tupel in $I(R)$

Größe der Ausgabe: $\llbracket \llbracket Q \rrbracket(I) \rrbracket :=$ "Stelligkeit" \cdot "Anzahl Tupel im Ergebnis"

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.

Auswertung konjunktiver Anfragen

Proposition 2.18

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

Beweis: siehe Tafel.

Bemerkung: Das ist exponentiell in der Länge der Anfrage.

Frage: Geht das effizienter?

Algorithmen mit Verzögerung $f(k, n)$

Definition 2.19

Sei $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, 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 I nach und nach genau die Tupel aus $\llbracket Q \rrbracket(I)$ ausgibt und

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

je höchstens $f(\llbracket Q \rrbracket, \llbracket I \rrbracket)$ 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.

NP-Vollständigkeit

Zur Erinnerung:

- Ein Entscheidungsproblem B heißt **NP-vollständig** (bzgl. Polynomialzeit-Reduktionen), falls gilt:
 - $B \in \text{NP}$ und
 - B ist **NP-hart**, d.h. für jedes Problem $A \in \text{NP}$ gibt es eine **Polynomialzeit-Reduktion** f von A auf B (kurz: $f : A \leq_p B$)
- 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 Eingabe $f(w)$ abbildet, so dass gilt

$$w \text{ ist eine "ja"-Instanz für } A \iff f(w) \text{ ist eine "ja"-Instanz für } B.$$
 Anschaulich bedeutet $A \leq_p B$, 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)
 - $A \leq_p B$ und $B \in \text{P}$ \implies $A \in \text{P}$
 - $A \notin \text{P}$ und $A \leq_p B$ \implies $B \notin \text{P}$
 - A NP-hart und $A \leq_p B$ \implies B NP-hart.

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:
 - Jeder Lauf von T bei Eingabe w hat die Länge $\leq N^c$ und endet mit der Ausgabe "ja" oder "nein".
 - Ist w eine "ja"-Instanz für B , so gibt es mindestens einen Lauf von T auf w , der mit der Ausgabe "ja" endet.
 - Ist w eine "nein"-Instanz für B , so endet jeder Lauf von T auf w mit der Ausgabe "nein".

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 $(k+n)^{O(1)}$ 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.

Algebraische Anfragen — Informell

Beispiel-Anfrage:

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

Als Anfrage in der SPJR-Algebra:

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

Als Anfrage in der SPC-Algebra:

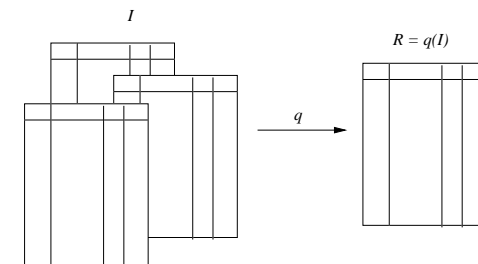
$$\pi_{7,8} \left(\sigma_{4=7} \left(\sigma_{1=5} \left(\sigma_{3 = \text{"George Clooney"}} (\text{Filme} \times \text{Programm} \times \text{Orte}) \right) \right) \right)$$

Konjunktive Anfragen (I)

- 2.1 Deskriptiver Ansatz: regelbasiert, graphisch und logikbasiert
- 2.2 Auswertungskomplexität
- 2.3 Algebraischer Ansatz: SPC-Algebra und SPJR-Algebra

SPC-Algebra bzw. SPJR-Algebra

Zur Erinnerung:



▶ Mathematik:

Algebraische Struktur $\hat{=}$ 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 $a \in \mathbf{dom}$ und eine natürliche Zahl $j \geq 1$.
Dieser Operator kann angewendet werden auf Relationen I der Stelligkeit $\geq j$ 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 $j, k \geq 1$.
Dieser Operator kann angewendet werden auf Relationen I der Stelligkeit $\geq \max(j, k)$ 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 (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 $m+n$:

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

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 $\langle t(1), \dots, t(m), s(1), \dots, s(n) \rangle$ zu bezeichnen.

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

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

Projektion:

Operator π_{j_1, \dots, j_k} , für (nicht notwendigerweise paarweise verschiedene) natürliche Zahlen $j_1, \dots, j_k \geq 1$ (und $k \geq 0$).

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

$$\pi_{j_1, \dots, j_k}(I) := \{ \langle t(j_1), \dots, t(j_k) \rangle : t \in I \}$$

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

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

Definition 2.23

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

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

Die **Semantik** $\llbracket Q \rrbracket$ von SPC[\mathbf{R}]-Anfragen Q ist induktiv auf die offensichtliche Art definiert.

Verallgemeinerung des Selektions-Operators

Eine **positive konjunktive Selektionsbedingung** ist eine Formel F der Form

$$\gamma_1 \wedge \dots \wedge \gamma_n$$

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

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

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 $A \in \mathit{sort}(R)$ 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 $A, B \in \mathit{sort}(R)$ 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.

Eine Normalform für SPC-Anfragen

Sei \mathbf{R} ein Relationsschema.

Definition 2.24

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

$$\pi_{j_1, \dots, j_k} \left(\{ \langle c_1 \rangle \} \times \dots \times \{ \langle c_m \rangle \} \times \sigma_F(R_1 \times \dots \times R_\ell) \right)$$

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

Proposition 2.25

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

Beweis: Übung.

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

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

Projektion:

Operator π_{A_1, \dots, A_k} , für paarweise verschiedene Attribut-Namen A_1, \dots, A_k (und $k \geq 0$).

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

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

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.

$$I \bowtie J = I \bowtie J$$

A	B
1	a
2	b
3	c

C	D	E
e	f	g
h	i	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:

$$I \bowtie J = I \bowtie J$$

A	B
1	a
2	b
3	c

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

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

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 bijektive Funktion $f : U \rightarrow \mathbf{att}$, für eine beliebige endliche Menge U von Attribut-Namen.

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

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

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

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 $\Sigma := \text{sort}(R) \cup \text{sort}(S)$:

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

Wir benutzen den Operator manchmal auch für einzelne Tupel:

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

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

Definition 2.26

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

- ▶ Für alle Relations-Namen $R \in \mathbf{R}$ ist R eine SPJR[\mathbf{R}]-Anfrage der Sorte $\text{sort}(R)$.
- ▶ Für alle Konstanten $c \in \mathbf{dom}$ und alle Attribut-Namen $A \in \mathbf{att}$ ist $\{(A : c)\}$ eine SPJR[\mathbf{R}]-Anfrage der Sorte $\{C\}$.
- ▶ Ist Q eine SPJR[\mathbf{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[\mathbf{R}]-Anfragen der Sorte Σ .
- ▶ Ist Q eine SPJR[\mathbf{R}]-Anfrage der Sorte Σ , ist $k \geq 0$ und sind A_1, \dots, A_k paarweise verschiedene Elemente aus Σ , so ist $\pi_{A_1, \dots, A_k}(Q)$ eine SPJR[\mathbf{R}]-Anfrage der Sorte $\{A_1, \dots, A_k\}$.
- ▶ Sind Q und P zwei SPJR[\mathbf{R}]-Anfragen der Sorten Σ und Π , so ist $(Q \bowtie P)$ eine SPJR[\mathbf{R}]-Anfrage der Sorte $\Sigma \cup \Pi$.
- ▶ Ist Q eine SPJR[\mathbf{R}]-Anfrage der Sorte Σ und ist $f : \Sigma \rightarrow \mathbf{att}$ eine Umbenennungsfunktion, so ist $\delta_f(Q)$ eine SPJR[\mathbf{R}]-Anfrage der Sorte $f(\Sigma)$.

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

Die **Semantik** $\llbracket Q \rrbracket$ 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

$$\gamma_1 \wedge \dots \wedge \gamma_n$$

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

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

Nicht-Erfüllbare Anfragen

Bemerkung:

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

Beispiel: Die Anfrage $Q :=$

$$\sigma_{3=\text{George Clooney}} \left(\sigma_{3=\text{Wolfgang Völz}} (\text{Filme}) \right)$$

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: $c = \text{George Clooney}$ und $c = \text{Wolfgang Völz}$.

Solche Tupel kann es aber nicht geben!

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

Somit ist die Anfrage Q nicht erfüllbar.

Eine Normalform für SPJR-Anfragen

Sei **R** ein Relationenschema.

Definition 2.27

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

$$\pi_{B_1, \dots, B_k} \left(\left(\{ \langle A_1 : c_1 \rangle \} \bowtie \dots \bowtie \{ \langle A_m : c_m \rangle \} \bowtie \sigma_F (\delta_{f_1}(R_1) \bowtie \dots \bowtie \delta_{f_\ell}(R_\ell)) \right) \right)$$

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

Äquivalenz der Ausdruckskraft 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.