Vorlesung
Logik und Datenbanken

Nicole Schweikardt

Johann Wolfgang Goethe-Universität Frankfurt am Main

Sommersemester 2008

Konjunktive Anfragen (I)

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

Konjunktive Anfragen (I)

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 **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 **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-Namen, 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 "$\mathbf{I} \subseteq \mathbf{J}$", 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.: $frei(R(v_1, \ldots, v_r)) = \{v_1, \ldots, v_r\} \cap \mathbf{var}$; $frei((\varphi \wedge \psi)) = frei(\varphi) \cup frei(\psi)$;
  $frei(\exists x \varphi) = frei(\varphi) \setminus \{x\}$.
- 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 aus 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:

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

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 (I)

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

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

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

- 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 $B \in \text{P} \implies A \in \text{P}$
  - $A \notin \text{P}$ und $A \leqslant_p B \implies B \notin \text{P}$
  - 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 Anfragen des konjunktiven Kalküls 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: 3-SAT *ist* NP*-vollständig.*

Das Problem 3-SAT ist dabei folgendermaßen definiert:

3-SAT

Eingabe: Aussagenlogische Formel ψ in 3-KNF

Frage: Ist ψ erfüllbar?
(D.h. gibt es eine Belegung der aussagenlogischen Variablen von ψ mit Werten aus $\{0, 1\}$, die ψ erfüllt?)

3-KNF bedeutet:
Formeln sind Konjunktionen von Klauseln; Klauseln sind Disjunktionen aus genau 3 Literalen; Literale sind aussagenlogische Variablen oder negierte aussagenlogische Variablen.

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 (I)