Relationenkalkül

Relationenkalkül

5.1 CALC$_{nat}$, CALC$_{adom}$ und CALC$_{di}$
5.2 Sicherer Relationenkalkül: CALC$_{sr}$
5.3 Statische Analyse und Auswertungskomplexität
5.4 Grenzen der Ausdrucksstärke

Motivation: Bisher kennengelernte Anfragesprachen

Algebraisch ...	äquivalent dazu ...	Logik-basiert
SPC-Algebra		konjunktiver Kalkül
relationale Algebra		Relationenkalkül

Der konjunktive Kalkül basiert auf einem Fragment der Logik erster Stufe.

Der Relationenkalkül basiert auf der vollen Logik erster Stufe (kurz: FO).

("FO" steht für "first-order logic")

Die Logik erster Stufe — FO[**R**]

Definition 5.1

Sei **R** ein Datenbankschema.
Die Menge FO[**R**] aller Formeln der Logik erster Stufe über **R** ist induktiv wie folgt definiert:

(A) "Relations-Atome": $R(v_1, .., v_r)$ gehört zu FO[**R**],
für alle $R \in \mathbf{R}$, $r := arity(R)$ und $v_1, .., v_r \in \mathbf{var} \cup \mathbf{dom}$.

(G) "Gleichheits-Atome": $v_1 = v_2$ gehört zu FO[**R**], für alle $v_1, v_2 \in \mathbf{var} \cup \mathbf{dom}$.

(BK) "Boolesche Kombinationen": Falls $\varphi_1 \in$ FO[**R**] und $\varphi_2 \in$ FO[**R**], so gehört auch jede der folgenden fünf Formeln zu FO[**R**]:
$\neg\varphi_1$, $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \rightarrow \varphi_2)$ und $(\varphi_1 \leftrightarrow \varphi_2)$.

(Q) "Quantoren": Ist $\varphi_1 \in$ FO[**R**] und ist $x \in \mathbf{var}$, so gehören auch die beiden folgenden Formeln zu FO[**R**]: $\exists x\, \varphi_1$ und $\forall x\, \varphi_1$.

Bemerkung: Benutzen wir die Notation aus der Vorlesung *Diskrete Modellierung*, so ist FO[**R**] genau die Menge aller Formeln aus FO[σ], für die Signatur $\sigma := \mathbf{R} \cup \mathbf{dom}$ (wobei jedes Element aus **dom** als "Konstanten-Symbol" aufgefasst wird, das stets "mit sich selbst" interpretiert wird).

Notationen und Anmerkungen

- Manchmal werden wir Formeln der Form
  - $(\varphi_1 \vee \varphi_2)$ als Abkürzung für $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$,
  - $(\varphi_1 \to \varphi_2)$ als Abkürzung für $(\neg\varphi_1 \vee \varphi_2)$
  - $(\varphi_1 \leftrightarrow \varphi_2)$ als Abkürzung für $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \neg\varphi_2)$
  - $\forall x\, \varphi_1$ als Abkürzung für $\neg\exists x \neg\varphi_1$

  auffassen.

- $adom(\varphi)$ 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.
  - $frei\big(R(v_1,..,v_r)\big) := \{v_1,..,v_r\} \cap \mathbf{var}$
  - $frei\big(v_1{=}v_2\big) := \{v_1, v_2\} \cap \mathbf{var}$
  - $frei\big(\neg\varphi_1\big) := frei\big(\varphi_1\big)$
  - $frei\big((\varphi_1 * \varphi_2)\big) := frei\big(\varphi_1\big) \cup frei\big(\varphi_2\big)$ (für alle $* \in \{\wedge, \vee, \to, \leftrightarrow\}$)
  - $frei\big(\exists x\, \varphi_1\big) := frei\big(\forall x\, \varphi_1\big) := frei\big(\varphi_1\big) \setminus \{x\}$

- Eine Belegung für φ ist eine Abbildung $\beta : frei(\varphi) \to \mathbf{dom}$.
- Sei $\mathbf{d} \subseteq \mathbf{dom}$. Eine Belegung für φ in $\mathbf{d}$ ist eine Abbildung $\beta : frei(\varphi) \to \mathbf{d}$.

Nat. Semantik von FO[**R**] und Rel.kalkül CALC

Wir werden verschiedene Varianten der Semantik von FO[**R**] betrachten.
Hier zunächst die natürliche Semantik:

Definition 5.2 (natürliche Semantik von FO[**R**]*)*

- Zur Erinnerung: Einer Datenbank **I** vom Schema **R** ordnen wir die logische Struktur $\mathcal{A}_{\mathbf{I}} := \big(\mathbf{dom},\ (\mathbf{I}(R))_{R\in\mathbf{R}},\ (c)_{c\in\mathbf{dom}}\big)$ zu.
- Ist **I** eine Datenbank vom Schema **R** und β eine Belegung für φ (also eine Abbildung $\beta : frei(\varphi) \to \mathbf{dom}$), so sagen wir "**I** erfüllt φ unter β" und schreiben $\mathbf{I} \models \varphi[\beta]$ bzw. $(\mathbf{I}, \beta) \models \varphi$, um auszudrücken, dass $(\mathcal{A}_{\mathbf{I}}, \beta) \models \varphi$.

Definition 5.3 (Relationenkalkül CALC*)*
Sei **R** *ein Datenbankschema. Eine Anfrage des Relationenkalküls* CALC[**R**] *ist von der Form* $\{\ \langle e_1,\ldots,e_r\rangle\ :\ \varphi\ \}$,
wobei $\varphi \in$ FO[**R**], $r \geqslant 0$ *und* $\langle e_1,..,e_r\rangle$ *ein freies Tupel ist (d.h.* $e_1,..,e_r \in \mathbf{var} \cup \mathbf{dom}$*), so dass* $frei(\varphi) = \{e_1,..,e_r\} \cap \mathbf{var}$.
Wir setzen $adom(Q) := adom(\varphi) \cup (\{e_1,..,e_r\} \cap \mathbf{dom})$.

Semantik: Einer CALC[**R**]*-Anfrage Q der obigen Form ordnen wir die folgende "Anfragefunktion"* $[\![Q]\!]_{nat}$ *zu (f.a.* $\mathbf{I} \in inst(\mathbf{R})$*):*

$$[\![Q]\!]_{nat}(\mathbf{I}) := \{\ \beta\big(\langle e_1,..,e_r\rangle\big)\ :\ \beta \text{ ist eine Belegung für } \varphi \text{ mit } \mathbf{I} \models \varphi[\beta]\ \}$$

Beispiele für CALC-Anfragen

- In welchen Filmen hat George Clooney mitgespielt, aber nicht selbst Regie geführt?

$$\Big\{\ \langle x_{Titel}\rangle\ :\ \exists x_{Regie}\ \Big(Filme(x_{Titel}, x_{Regie}, \text{“George Clooney”}) \wedge \neg Filme(x_{Titel}, \text{“George Clooney”}, \text{“George Clooney”})\ \Big)\ \Big\}$$

- Welche Filme laufen nur zu 1 Uhrzeit?

$$\Big\{\ \langle x_{Titel}\rangle\ :\ \exists x_{Kino}\,\exists x_{Zeit}\ \Big(\ Programm(x_{Kino}, x_{Titel}, x_{Zeit}) \wedge \forall y_{Kino}\,\forall y_{Zeit}\big(Programm(y_{Kino}, x_{Titel}, y_{Zeit}) \to y_{Zeit}{=}x_{Zeit}\ \big)\ \Big)\ \Big\}$$

- Welche Filme haben nur Schauspieler, die schon mal in einem Film von Stephen Spielberg mitgespielt haben?

$$\Big\{\ \langle x_T\rangle\ :\ \exists x_R \exists x_S\Big(Filme(x_T, x_R, x_S)\ \wedge \forall y_S\big(Filme(x_T, x_R, y_S) \to \exists y_T\, Filme(y_T, \text{“Stephen Spielberg”}, y_S)\big)\Big)\ \Big\}$$

Frage: Was ist mit . . . ?

$$\Big\{\langle x_T\rangle\ :\ \forall y_S\Big(\exists x_R\, Filme(x_T, x_R, y_S)\ \to\ \exists y_T\, Filme(y_T, \text{“Stephen Spielberg”}, y_S)\Big)\Big\}$$

Unsichere CALC-Anfragen

Beispiele:

(1) $Q_1 := \{\langle x_{Schausp}\rangle\ :\ \neg Filme(\text{“Knallhart”}, \text{“Detlev Buck”}, x_{Schausp})\ \}$

(2) $Q_2 := \{\langle x_S, y_T\rangle\ :\ Filme(\text{“Knallhart”}, \text{“Detlev Buck”}, x_S) \vee Filme(y_T, \text{“George Clooney”}, \text{“George Clooney”})\ \}$

(3) $Q_3 := \Big\{\langle x_T\rangle\ :\ \forall y_S\Big(\exists x_R\, Filme(x_T, x_R, y_S)\ \to\ \exists y_T\, Filme(y_T, \text{“Stephen Spielberg”}, y_S)\Big)\Big\}$

Auswertung mit $[\![\cdot]\!]_{nat}$*:* Q_1 liefert alle (unendlich vielen) $a \in \mathbf{dom}$, die *nicht* als Schauspieler im Film "Knallhart" mitgespielt haben.
Q_2 liefert alle (unendlich vielen) Tupel der Form $\langle a, b\rangle$, für die a ein Schaupieler in "Knallhart" und b ein beliebiges Element in **dom** ist oder b ein Film von und mit George Clooney und a ein beliebiges Element aus **dom**.
Q_3 liefert alle Filme, die nur solche Schauspieler haben, die schon mal mit Stephen Spielberg gearbeitet haben, aber auch **alle Elemente aus dom, die nicht Titel eines Films** sind.

Problem: Gemäß der bisherigen Definition der Semantik liefern die Anfragen Q_1, Q_2 und Q_3 stets "unendliche Ergebnisse" die gemäß unserer Definition keine Datenbank-Relationen sind (da die stets endlich sein müssen). Solche Anfrage heißen unsicher.

Eine weitere problematische Anfrage

$Q_4 := \{\langle x\rangle \ : \ \forall y\, R(x,y)\}$

Q_4 liefert als Ergebnis stets die leere Relation, d.h.
$[\![Q_4]\!]_{nat}(\mathbf{I}) = \emptyset$, für alle $\mathbf{I} \in inst(\mathbf{R})$.

Dies ist unschön.

Daher (und weil manche Anfragen unsicher sind in der nat. Semantik), betrachten wir im Folgenden andere Varianten der Semantik.

Relativierte Semantik von FO[**R**] und CALC[**R**]

Definition 5.4

Sei **R** ein Datenbankschema.

(a) Eine relativierte Datenbank über **R** ist ein Paar $(\mathbf{d}, \mathbf{I})$, wobei $\mathbf{I}$ eine Datenbank vom Schema **R** ist und **d** eine Menge, so dass $adom(\mathbf{I}) \subseteq \mathbf{d} \subseteq \mathbf{dom}$.

(b) Einer relativierten Datenbank $(\mathbf{d}, \mathbf{I})$ über **R** ordnen wir die logische Struktur $\mathcal{A}_{(\mathbf{d},\mathbf{I})} := \big(\mathbf{d},\ (\mathbf{I}(R))_{R\in\mathbf{R}},\ (c)_{c\in\mathbf{d}}\big)$ zu.

(c) Eine FO[**R**]-Formel φ heißt interpretierbar über $(\mathbf{d}, \mathbf{I})$, falls $adom(\varphi) \subseteq \mathbf{d}$.
Eine CALC[**R**]-Anfrage Q heißt interpretierbar über $(\mathbf{d}, \mathbf{I})$, falls $adom(Q) \subseteq \mathbf{d}$.

(d) Ist $(\mathbf{d}, \mathbf{I})$ eine relativierte Datenbank über **R**, ist φ eine FO[**R**]-Formel, die interpretierbar ist über $(\mathbf{d}, \mathbf{I})$, und ist β eine Belegung für φ in **d** (also eine Abbildung $\beta : frei(\varphi) \to \mathbf{d}$), so sagen wir "$\mathbf{I}$ erfüllt φ unter β relativ zu **d**" und schreiben $\mathbf{I} \models_{\mathbf{d}} \varphi[\beta]$ bzw. $(\mathbf{I}, \beta) \models_{\mathbf{d}} \varphi$, um auszudrücken, dass $(\mathcal{A}_{(\mathbf{d},\mathbf{I})}, \beta) \models \varphi$.

(e) Ist Q eine CALC[**R**]-Anfrage der Form $\{\langle e_1, \ldots, e_r\rangle \ : \ \varphi\}$ und $(\mathbf{d}, \mathbf{I})$ eine relativierte Datenbank über **R**, über der Q interpretierbar ist, so ist
$[\![Q]\!]_{\mathbf{d}}(\mathbf{I}) := \{\ \beta(\langle e_1, \ldots, e_r\rangle) \ : \ \beta \text{ ist eine Belegung für } \varphi \text{ über } \mathbf{d} \text{ mit } \mathbf{I} \models_{\mathbf{d}} \varphi[\beta]\ \}$

("Relativiert" soll andeuten, dass Quantifizierung relativiert wird auf Elemente aus **d**.)

Ein Beispiel

Beispiel 5.5

- **R** *enthalte ein 1-stelliges Relationssymbol* R
- $\mathbf{I}$ *sei die Datenbank mit*
  $\mathbf{I}(R) = \{\langle a\rangle, \langle b\rangle, \langle c\rangle\}$ *und* $\mathbf{I}(S) = \emptyset$ *für alle* $S \in \mathbf{R} \setminus \{R\}$.
- Q *sei die* CALC[**R**]*-Anfrage*

$$Q := \{\ \langle x\rangle \ : \ (\ R(x) \ \wedge\ \exists y\,(\neg R(y) \ \wedge\ \forall z\,(R(z) \vee z{=}y))\ )\ \}$$

- *Dann gilt:*
  - $[\![Q]\!]_{\{a,b,c\}}(\mathbf{I}) = \emptyset$
  - $[\![Q]\!]_{\{a,b,c,d\}}(\mathbf{I}) = \{\langle a\rangle, \langle b\rangle, \langle c\rangle\}$
  - $[\![Q]\!]_{\{a,b,c,d,e\}}(\mathbf{I}) = \emptyset$

Active Domain Semantik von FO[**R**] und CALC[**R**]

Definition 5.6

(a) Ist $\mathbf{I} \in inst(\mathbf{R})$ und $\varphi \in$ FO[**R**], so ist $adom(\varphi, \mathbf{I}) := adom(\varphi) \cup adom(\mathbf{I})$.
Ist $\mathbf{I} \in inst(\mathbf{R})$ und $Q \in$ CALC[**R**], so ist $adom(Q, \mathbf{I}) := adom(Q) \cup adom(\mathbf{I})$.

(b) Ist $\varphi \in$ FO[**R**], $\mathbf{I}$ eine Datenbank vom Schema **R** und β eine Belegung für φ in $adom(\varphi, \mathbf{I})$ (also eine Abbildung $\beta : frei(\varphi) \to adom(\varphi, \mathbf{I})$), so sagen wir "$\mathbf{I}$ erfüllt φ unter β in der Active Domain Semantik" und schreiben $\mathbf{I} \models_{adom} \varphi[\beta]$ bzw. $(\mathbf{I}, \beta) \models_{adom} \varphi$, um auszudrücken, dass $\mathbf{I} \models_{\mathbf{d}} \varphi[\beta]$ für $\mathbf{d} := adom(\varphi, \mathbf{I})$ gilt.

(c) Ist Q eine CALC[**R**]-Anfrage der Form $\{\langle e_1, \ldots, e_r\rangle \ : \ \varphi\}$, so definiert Q in der Active Domain Semantik die folgende Anfragefunktion

$$[\![Q]\!]_{adom}(\mathbf{I}) := [\![Q]\!]_{adom(Q,\mathbf{I})}(\mathbf{I})$$

Bemerkungen zur natürlichen Semantik und zur Active Domain Semantik

- Die Semantik $[\![\cdot]\!]_{nat}$ ist vom Standpunkt der Logik aus "natürlich"; vom Standpunkt des Datenbanknutzers aber eher unnatürlich, da manche Anfragen unendliche Ergebnisse liefern (d.h. unsicher sind).
- Die Active Domain Semantik $[\![\cdot]\!]_{adom}$ liefert immer ein endliches Ergebnis; aber das Ergebnis kann sich (vom Standpunkt des Datenbanknutzers) auf unnatürliche Weise ändern, wenn ein neuer Wert in die Datenbank eingetragen wird (selbst wenn dieser scheinbar nichts mit der Anfrage zu tun hat).

  Beispiel: $Q := \{ \langle x \rangle \ : \ (\ R(x) \ \wedge \ \exists y (\neg R(y) \ \wedge \ \forall z (R(z) \vee z{=}y)) \ ) \ \}$
- Radikaler Schritt: Betrachte nur Anfragen, die bereichsunabhängig sind, d.h. bei denen es egal ist, ob sie in $[\![\cdot]\!]_{nat}$, $[\![\cdot]\!]_{adom}$ oder in $[\![\cdot]\!]_{\mathbf{d}}$ für irgendeine Menge $\mathbf{d}$ mit $adom(Q, \mathbf{I}) \subseteq \mathbf{d} \subseteq \mathbf{dom}$ auswertet werden.

*Definition 5.7 (bereichunabhängige CALC[**R**]-Anfragen)*

Eine Anfrage $Q \in \text{CALC}[\mathbf{R}]$ heißt bereichsunabhängig (domain independent), falls für jede Datenbank $\mathbf{I} \in inst(\mathbf{R})$ und alle $\mathbf{d}, \mathbf{d}' \subseteq \mathbf{dom}$ mit $adom(Q, \mathbf{I}) \subseteq \mathbf{d}, \mathbf{d}'$ gilt: $[\![Q]\!]_{\mathbf{d}}(\mathbf{I}) = [\![Q]\!]_{\mathbf{d}'}(\mathbf{I})$.

Beispiele für bereichsunabhängige CALC-Anfragen

- Welche Filme laufen in mindestens 2 Kinos?

  $\Big\{ \langle x_T \rangle \ : \ \exists x_K \, \exists x_Z \, \exists y_K \, \exists y_Z \big(Programm(x_K, x_T, x_Z) \wedge Programm(y_K, x_T, y_Z) \wedge \neg x_K{=}y_K \big) \Big\}$
- In welchen Filmen hat George Clooney mitgespielt, aber nicht selbst Regie geführt?

  $\Big\{ \langle x_{Titel} \rangle \ : \ \exists x_{Regie} \ \Big(Filme(x_{Titel}, x_{Regie}, \text{"George Clooney"}) \wedge \neg Filme(x_{Titel}, \text{"George Clooney"}, \text{"George Clooney"}) \ \Big) \Big\}$
- Welche Filme laufen nur zu 1 Uhrzeit?

  $\Big\{ \langle x_{Titel} \rangle \ : \ \exists x_{Kino} \, \exists x_{Zeit} \ \Big(\ Programm(x_{Kino}, x_{Titel}, x_{Zeit}) \wedge \forall y_{Kino} \, \forall y_{Zeit} \big(Programm(y_{Kino}, x_{Titel}, y_{Zeit}) \rightarrow y_{Zeit}{=}x_{Zeit} \ \big) \ \Big) \Big\}$
- Welche Filme haben nur Schauspieler, die schon mal in einem Film von Stephen Spielberg mitgespielt haben?

  $\Big\{ \langle x_T \rangle \ : \ \exists x_R \exists x_S \Big(Filme(x_T, x_R, x_S) \ \wedge \ \forall y_S \big(Filme(x_T, x_R, y_S) \rightarrow \exists y_T \ Filme(y_T, \text{"Stephen Spielberg"}, y_S) \big) \Big) \Big\}$

  Nicht bereichunabhängig ist ...

  $\Big\{ \langle x_{Titel} \rangle \ : \ \forall y_S \Big(\exists x_R \ Filme(x_{Titel}, x_R, y_S) \ \rightarrow \ \exists y_T \ Filme(y_T, \text{"Stephen Spielberg"}, y_S) \Big) \Big\}$

CALC_{nat}, CALC_{adom}, CALC_{di}

Definition 5.8

$\text{CALC}_{nat}[\mathbf{R}]$ bezeichnet die Klasse aller in der natürlichen Semantik $[\![\cdot]\!]_{nat}$ interpretierten CALC[**R**]-Anfragen.

$\text{CALC}_{adom}[\mathbf{R}]$ bezeichnet die Klasse aller in der Active Domain Semantik $[\![\cdot]\!]_{adom}$ interpretierten CALC[**R**]-Anfragen.

$\text{CALC}_{di}[\mathbf{R}]$ bezeichnet die Klasse aller bereichsunabhängigen CALC[**R**]-Anfragen (interpretiert in $[\![\cdot]\!]_{adom}$ oder, äquivalent dazu, in $[\![\cdot]\!]_{nat}$).
("di" steht für "domain independent")

Satz 5.9 (CALC_{adom}, CALC_{di} und relationaler Algebra)
Die folgenden Anfragesprachen können genau dieselben Anfragefunktionen ausdrücken:

(a) CALC_{di}
(b) CALC_{adom}
(c) *relationale Algebra (unbenannte oder benannte Perspektive).*

*Und für jedes feste Datenbankschema **R** gilt: Jede Anfrage aus einer dieser Sprachen kann in polynomieller Zeit in äquivalente Anfragen der anderen Sprachen übersetzt werden.*

Beweis: Siehe Tafel.

Erinnerung: nr-Datalog$^\neg$

Sei P ein Datalog-Programm (mit Negation).

(a) Der Abhängigkeitsgraph G_P von P ist der gerichtete Graph mit Knotenmenge $V_P := sch(P)$ und Kantenmenge

$$E_P := \left\{ (S, R) \ : \ \begin{array}{l} \text{es gibt eine Regel in } P, \text{ in deren Kopf } R \text{ und in} \\ \text{deren Rumpf } S \text{ vorkommt} \end{array} \right\}$$

(b) Die Klasse nr-Datalog$^\neg$ aller nicht-rekursiven Datalog-Programme mit Negation besteht aus allen Datalog-Programmen P mit Negation, deren Abhängigkeitsgraph G_P azyklisch ist.

Ausdrucksstärke von nr-Datalog$^\neg$

Jetzt können wir Satz 3.25 beweisen:

Satz 5.10

Die folgenden Anfragesprachen können genau dieselben Anfragefunktionen beschreiben:

(a) *nr-Datalog$^\neg$-Anfragen*

(b) *Relationale Algebra*

(c) *CALC_{di}*

Bemerkung: Die Übersetzung von nr-Datalog$^\neg$ in eine der anderen Sprachen kann mehr als polynomiell viel Zeit beanspruchen, da nr-Datalog-Programme u.U. viel kürzer sind als äquivalente Anfragen der anderen Sprachen.

Beweis: (b) $\Leftrightarrow$ (c): schon gezeigt (Satz 5.9).
(b) $\Rightarrow$ (a): Einfache Induktion nach dem Aufbau der relationalen Algebra: erweitere den Beweis von Satz 3.17, (b) $\Rightarrow$ (a) (Übung)
(a) $\Rightarrow$ (c): Erweitere den Beweis von Satz 3.17, (a) $\Rightarrow$ (c) (siehe Tafel).

Bereichsunabhängigkeit — Pro & Contra CALC_{di}

Vorteile:

- logik-basierte Anfragesprache, die die "richtige" Ausdrucksstärke hat (äquivalent zur relationalen Algebra; insbesondere sind alle CALC_{di}-Anfragen "sicher")
- keine unerwünschten Effekte, wie sie bei CALC_{adom} auftreten können (vgl. Beispiel 5.5).
- Ergebnis der Anfragen ist unabhängig davon, in welcher Semantik ($[\![\cdot]\!]_{adom}$ oder $[\![\cdot]\!]_{nat}$ oder $[\![\cdot]\!]_{\mathbf{d}}$) sie interpretiert werden

Frage:

- Wie kann man bei einer gegebenen CALC-Anfrage Q feststellen, ob Q zu CALC_{di} gehört, d.h. ob Q bereichsunabhängig ist?

Antwort — großer Nachteil von CALC_{di}:

- Das ist unentscheidbar, d.h. es gibt keinen Algorithmus, der bei Eingabe einer beliebigen CALC[**R**]-Anfrage Q nach endlich vielen Schritten anhält und entscheidet, ob Q bereichsabhängig ist oder nicht.
- Beweis: auf den nächsten Folien ...

Der Satz von Trakhtenbrot

Boris A. Trakhtenbrot, russisch-israelischer Mathematiker, geb. 1921;
alternative Schreibweisen: Trachtenbrot bzw. Trahtenbrot

Theorem 5.11 (Trakhtenbrot, 1950) *(hier ohne Beweis)*

*Sei **R** ein Datenbankschema, das mindestens ein Relationssymbol enthält, dessen Stelligkeit $\geqslant 2$ ist. Dann ist das folgende Problem unentscheidbar:*

ERFÜLLBARKEITSPROBLEM FÜR FO[**R**] ÜBER ENDLICHEN DATENBANKEN
Eingabe: Ein FO[**R**]-Satz ψ (*Zur Erinnerung:* "Satz" heißt: $\mathit{frei}(\psi) = \emptyset$)
Frage: Gibt es eine (endliche) Datenbank $\mathbf{I} \in \mathit{inst}(\mathbf{R})$, für die gilt $\mathbf{I} \models_{adom} \psi$?

Beweis: Siehe Vorlesung Logik in der Informatik.

Bereichsunabhängigkeit ist unentscheidbar

Satz 5.12

*Sei **R** ein Datenbankschema, das mindestens ein Relationssymbol R enthält, dessen Stelligkeit $\geqslant 2$ ist. Dann ist das folgende Problem unentscheidbar:*

BEREICHSUNABHÄNGIGKEIT FÜR CALC[**R**]
Eingabe: Eine CALC[**R**]-Anfrage Q
Frage: Ist Q bereichsunabhängig (d.h., gehört Q zu CALC_{di}[**R**]) ?

Beweis: Einfache Folgerung aus dem Satz von Trakhtenbrot; siehe Tafel.

Relationenkalkül

Motivation

- CALC$_{di}$ ist eine logik-basierte Anfragesprache, die die "richtige Ausdrucksstärke" hat (nämlich dieselbe wie die relationale Algebra).
- Wegen der Unentscheidbarkeit (Satz 5.12) ist CALC$_{di}$ als Anfragesprache für praktische Zwecke aber ungeeignet, da man bei Eingabe einer "Anfrage" aus CALC[**R**] nicht feststellen kann, ob dies eine Anfrage in CALC$_{di}$ ist.
- Ziel jetzt: Finde ein syntaktisches Kriterium für CALC[**R**]-Anfragen, das
  (a) algorithmisch leicht nachprüfbar ist,
  (b) Bereichsunabhängigkeit garantiert und
  (c) zu einer Anfragesprache führt, die immer noch dieselbe Ausdrucksstärke wie die relationale Algebra hat.

$\rightsquigarrow$ Sicherer Relationenkalkül CALC$_{sr}$: entscheidbare Teilklasse von CALC$_{di}$, die dieselbe Ausdrucksstärke wie CALC$_{di}$ hat.

Safe-Range Normalform (SRNF)

Für jede FO[**R**]-Formel φ sei SRNF(φ) die FO[**R**]-Formel, die aus φ entsteht, indem man die folgenden Regeln so lange anwendet, bis keine Regel mehr anwendbar ist:

- Alle quantifizierten Variablen werden so umbenannt, dass sie paarweise verschieden sind und verschieden von den freien Variablen der Formel.
- Teilformeln der Form $\forall x \psi$ werden ersetzt durch $\neg\exists x \neg\psi$
- Implikationspfeile $\rightarrow$ und "genau-dann-wenn"-Pfeile $\leftrightarrow$ werden durch Kombinationen von $\wedge$, $\vee$, $\neg$ ersetzt:
  - Teilformeln der Form $(\psi_1 \rightarrow \psi_2)$ werden ersetzt durch $(\neg\psi_1 \ \vee \ \psi_2)$
  - Teilformeln der Form $(\psi_1 \leftrightarrow \psi_2)$ werden ersetzt durch $\big((\psi_1 \wedge \psi_2) \vee (\neg\psi_1 \ \wedge \ \neg\psi_2)\big)$
- Negationszeichen werden "nach innen geschoben":
  - $\neg(\psi_1 \wedge \psi_2)$ wird ersetzt durch $(\neg\psi_1 \vee \neg\psi_2)$
  - $\neg(\psi_1 \vee \psi_2)$ wird ersetzt durch $(\neg\psi_1 \wedge \neg\psi_2)$
- "doppelte Negationszeichen" werden gelöscht: Teilformeln der Form $\neg\neg\psi$ werden ersetzt durch ψ

Offensichtlich gilt: Die Formel SRNF(φ) ist äquivalent zur Formel φ.
Falls $\varphi = \text{SRNF}(\varphi)$, so sagen wir: φ ist in Safe-Range Normalform (kurz: SRNF).

Beispiel für Transformation von φ zu SRNF(φ)

Beispiel 5.13

$$\varphi := \exists x_R \exists x_S \Big(\mathit{Filme}(x_T, x_R, x_S) \ \wedge \ \forall y_S \big(\mathit{Filme}(x_T, x_R, y_S) \rightarrow \exists y_T \, \mathit{Filme}(y_T, \text{“Stephen Spielberg”}, y_S) \big) \Big)$$

$$\rightsquigarrow \exists x_R \exists x_S \Big(\mathit{Filme}(x_T, x_R, x_S) \ \wedge \ \neg\exists y_S \neg \big(\mathit{Filme}(x_T, x_R, y_S) \rightarrow \exists y_T \, \mathit{Filme}(y_T, \text{“Stephen Spielberg”}, y_S) \big) \Big)$$

$$\rightsquigarrow \exists x_R \exists x_S \Big(\mathit{Filme}(x_T, x_R, x_S) \ \wedge \ \neg\exists y_S \neg \big(\neg\mathit{Filme}(x_T, x_R, y_S) \vee \exists y_T \, \mathit{Filme}(y_T, \text{“Stephen Spielberg”}, y_S) \big) \Big)$$

$$\rightsquigarrow \exists x_R \exists x_S \Big(\mathit{Filme}(x_T, x_R, x_S) \ \wedge \ \neg\exists y_S \big(\neg\neg\mathit{Filme}(x_T, x_R, y_S) \wedge \neg\exists y_T \, \mathit{Filme}(y_T, \text{“Stephen Spielberg”}, y_S) \big) \Big)$$

$$\rightsquigarrow \exists x_R \exists x_S \Big(\mathit{Filme}(x_T, x_R, x_S) \ \wedge \ \neg\exists y_S \big(\mathit{Filme}(x_T, x_R, y_S) \wedge \neg\exists y_T \, \mathit{Filme}(y_T, \text{“Stephen Spielberg”}, y_S) \big) \Big)$$

$$=: \text{SRNF}(\varphi)$$

Bereichsbeschränkte Variablen einer Formel in SRNf

Definition 5.14

Per Induktion nach dem Aufbau von SRNf-Formeln φ definieren wir $rr(\varphi)$ folgendermaßen:
($rr(\varphi)$ steht für "range restricted variables of φ")

- $rr\big(R(v_1,\ldots,v_{arity(R)})\big) := \{v_1,\ldots,v_{arity(R)}\} \cap \mathbf{var}$
- $rr\big(x{=}a\big) := rr\big(a{=}x\big) := \{x\}$, falls $x \in \mathbf{var}$ und $a \in \mathbf{dom}$
- $rr\big(v_1{=}v_2\big) := \emptyset$, falls $\{v_1, v_2\} \subseteq \mathbf{dom}$ oder $\{v_1, v_2\} \subseteq \mathbf{var}$
- $rr\big(\varphi_1 \vee \varphi_2\big) := rr(\varphi_1) \cap rr(\varphi_2)$
- Für $x, y \in \mathbf{var}$ gilt:
  $rr\big(\psi \wedge x{=}y\big) := rr\big(x{=}y \wedge \psi\big) := \begin{cases} rr(\psi) \cup \{x,y\} & \text{falls } \{x,y\} \cap rr(\psi) \neq \emptyset \\ rr(\psi) & \text{sonst} \end{cases}$
- Ist weder φ_1 noch φ_2 von der Form $x{=}y$ für $x, y \in \mathbf{var}$, so gilt:
  $rr\big(\varphi_1 \wedge \varphi_2\big) := rr(\varphi_1) \cup rr(\varphi_2)$
- $rr\big(\exists x \psi\big) := \begin{cases} \bot & \text{falls } x \notin rr(\psi) \\ rr(\psi) \setminus \{x\} & \text{sonst} \end{cases}$

Dabei gilt: "Einmal $\bot \rightsquigarrow$ immer $\bot$", d.h. für alle Mengen X gilt:
$\bot = \bot \setminus X = \bot \cap X = X \cap \bot = \bot \cap \bot = \bot \cup X = X \cup \bot = \bot \cup \bot$

- $rr(\neg\psi) := \begin{cases} \bot & \text{falls } rr(\psi) = \bot \\ \emptyset & \text{sonst} \end{cases}$

Bemerkung: Insbesondere gilt $rr(\varphi) = \bot$ oder $rr(\varphi) \subseteq frei(\varphi)$

Beispiele für $rr(\cdot)$

Beispiel 5.15

(a) Für die SRNf-Formel

$$\varphi := \exists x_R \exists x_S \Big(Filme(x_T, x_R, x_S) \wedge \neg\exists y_S \big(Filme(x_T, x_R, y_S) \wedge \neg\exists y_T\, Filme(y_T, \text{“Stephen Spielberg”}, y_S) \big) \Big)$$

gilt $rr(\varphi) = \{x_T\} = frei(\varphi)$.

(b) Für die SRNf-Formel

$$\psi := \neg\exists y_S \Big(\exists x_R\, Filme(x_T, x_R, y_S) \wedge \neg\big(\exists y_T\, Filme(y_T, \text{“Stephen Spielberg”}, y_S)\big) \Big)$$

gilt $rr(\psi) = \emptyset \subsetneq frei(\psi) = \{x_T\}$.
($\psi = \text{SRNF}(\psi')$, wobei ψ' die letzte Formel von Folie "Kapitel 5, Seite 14" ist.
In der Active Domain Semantik ist ψ äquivalent zu φ)

(c) Für die SRNf-Formel

$$\chi := R(x) \wedge \exists y \Big(\neg R(y) \wedge \neg\exists z \big(\neg R(z) \wedge \neg z{=}y \big) \Big)$$

gilt $rr(\chi) = \bot$. (χ ist die Formel SRNF("Formel aus Bsp. 5.5"))

Der Sichere Relationenkalkül CALC$_{sr}$

Definition 5.16

Sei **R** ein Datenbankschema.

CALC$_{sr}$[**R**] bezeichnet die Klasse aller CALC[**R**]-Anfragen der Form $\{\langle e_1,\ldots,e_r\rangle : \varphi\}$, für die gilt: $rr\big(\text{SRNF}(\varphi)\big) = frei(\varphi)$.

Bemerkung:

Aus den Definitionen von SRNF$(\cdot)$ und $rr(\cdot)$ erhält man direkt einen Algorithmus, der bei Eingabe einer FO[**R**]-Formel φ überprüft, ob $rr\big(\text{SRNF}(\varphi)\big) = frei(\varphi)$.

Somit gibt es also auch einen Algorithmus, der bei Eingabe einer CALC[**R**]-Anfrage Q entscheidet, ob Q zu CALC$_{sr}$[**R**] gehört oder nicht.

Ziel jetzt:

Zeige, dass alle CALC$_{sr}$-Anfragen bereichsunabhängig sind und dass CALC$_{sr}$ dieselben Anfragefunktionen ausdrücken kann wie die relationale Algebra.

Beispiele für CALC$_{sr}$-Anfragen

Anfragen in CALC$_{sr}$:

- Welche Filme laufen in mindestens 2 Kinos?
  $$\Big\{ \langle x_T \rangle \;:\; \exists x_K\, \exists x_Z\, \exists y_K\, \exists y_Z \big(Programm(x_K, x_T, x_Z) \wedge Programm(y_K, x_T, y_Z) \wedge \neg x_K{=}y_K \big) \Big\}$$
- In welchen Filmen hat George Clooney mitgespielt, aber nicht selbst Regie geführt?
  $$\Big\{ \langle x_{Titel} \rangle \;:\; \exists x_{Regie} \Big(Filme(x_{Titel}, x_{Regie}, \text{“George Clooney”}) \wedge \neg Filme(x_{Titel}, \text{“George Clooney”}, \text{“George Clooney”}) \Big) \Big\}$$
- Welche Filme haben nur Schauspieler, die schon mal in einem Film von Stephen Spielberg mitgespielt haben?
  $$\Big\{ \langle x_T \rangle \;:\; \exists x_R \exists x_S \Big(Filme(x_T, x_R, x_S) \wedge \forall y_S \big(Filme(x_T, x_R, y_S) \rightarrow \exists y_T\, Filme(y_T, \text{“Stephen Spielberg”}, y_S) \big) \Big) \Big\}$$

Nicht in CALC$_{sr}$ sind ...

- $\Big\{ \langle x_T \rangle \;:\; \forall y_S \Big(\exists x_R\, Filme(x_T, x_R, y_S) \rightarrow \exists y_T\, Filme(y_T, \text{“Stephen Spielberg”}, y_S) \Big) \Big\}$
- $\Big\{ \langle x \rangle \;:\; R(x) \wedge \exists y \Big(\neg R(y) \wedge \forall z \, \big(R(z) \vee z{=}y \big) \Big) \Big\}$

Ein technisches Lemma

Lemma 5.17

Sei **R** *ein Datenbankschema.*

Für jede FO[**R**]*-Formel* φ *in SRNf mit* $rr(\varphi) \neq \bot$,
für jede Datenbank $\mathbf{I} \in inst(\mathbf{R})$,
für jedes $\mathbf{d} \subseteq \mathbf{dom}$ *mit* $adom(\varphi, \mathbf{I}) \subseteq \mathbf{d}$ *und*
für jede Belegung β *für* φ *in* $\mathbf{d}$ *gilt:*

(1) *Falls* $\mathbf{I} \models_{\mathbf{d}} \varphi[\beta]$, *so gilt für alle* $x \in rr(\varphi)$: $\beta(x) \in adom(\varphi, \mathbf{I})$.

und

(2) *Für alle* $\mathbf{d}' \subseteq \mathbf{dom}$ *mit* $adom(\varphi, \mathbf{I}) \subseteq \mathbf{d}'$ *und* $\beta(frei(\varphi)) \subseteq \mathbf{d}'$ *gilt:*

$$\mathbf{I} \models_{\mathbf{d}} \varphi[\beta] \quad \Longleftrightarrow \quad \mathbf{I} \models_{\mathbf{d}'} \varphi[\beta]$$

Beweis: Siehe Tafel.

Relationenkalkül vs. Relationale Algebra

Korollar 5.18
Die folgenden Anfragesprachen können genau dieselben Anfragefunktionen ausdrücken:

(a) $CALC_{sr}$

(b) $CALC_{di}$

(c) $CALC_{adom}$

(d) *relationale Algebra (unbenannte oder benannte Perspektive)*

Und für jedes feste Datenbankschema **R** *gilt: Jede Anfrage aus einer dieser Sprachen kann in polynomieller Zeit in äquivalente Anfragen der anderen Sprachen übersetzt werden.*

Beweis: Siehe Tafel.

Im weiteren Verlauf der Vorlesung schreiben wir manchmal "Relationenkalkül" oder einfach CALC, um irgendeine der Varianten $CALC_{sr}$, $CALC_{di}$ bzw. $CALC_{adom}$ des Relationenkalküls zu bezeichnen.

In der Literatur wird oft der Begriff "Relational vollständige Sprache" benutzt, um Anfragesprachen zu bezeichnen, die mindestens die Ausdrucksstärke der relationalen Algebra (bzw., äquivalent dazu, des Relationenkalküls) besitzen.

Relationenkalkül

Zur Erinnerung (aus Kapitel 4)

Wunschliste für bessere Optimierung:

- zum "Löschen leerer Zwischenergebnisse":
  Test, ob eine gegebene (Teil-)Anfrage Q nicht erfüllbar ist ("$Q \equiv \emptyset$")
- zum "Löschen von Redundanzen":
  Test, ob zwei (Teil-)Anfragen Q und P äquivalent sind ("$Q \equiv P$")

- Algorithmen zum Lösen dieser Probleme für konjunktive Anfragen (s. Kap. 2)
- Jetzt: Nicht-Entscheidbarkeit dieser Probleme für allgemeinere Anfragen (relationale Algebra)

Statische Analyse für Relational vollständige Sprachen

Satz 5.19

Sei **R** *ein Datenbankschema, das mindestens ein Relationssymbol R enthält, dessen Stelligkeit* $\geqslant 2$ *ist. Dann sind die folgenden Probleme unentscheidbar:*

ERFÜLLBARKEITSPROBLEM FÜR REL. ALGEBRA ÜBER **R**
Eingabe: Anfrage Q der relationalen Algebra über dem Datenbankschema **R**
Frage: Ist Q erfüllbar (d.h. gibt es mind. ein $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ mit $[\![Q]\!](\mathbf{I}) \neq \emptyset$) ?

ÄQUIVALENZPROBLEM FÜR REL. ALGEBRA ÜBER **R**
Eingabe: Anfragen Q und P der relationalen Algebra über dem DB-Schema **R**
Frage: Ist $Q \equiv P$ (d.h. gilt für alle $\mathbf{I} \in \mathit{inst}(\mathbf{R})$, dass $[\![Q]\!](\mathbf{I}) = [\![P]\!](\mathbf{I})$) ?

QUERY CONTAINMENT PROBLEM FÜR REL. ALGEBRA ÜBER **R**
Eingabe: Anfragen Q und P der relationalen Algebra über dem DB-Schema **R**
Frage: Ist $Q \subseteq P$ (d.h. gilt für alle $\mathbf{I} \in \mathit{inst}(\mathbf{R})$, dass $[\![Q]\!](\mathbf{I}) \subseteq [\![P]\!](\mathbf{I})$) ?

Beweis: Einfache Folgerung aus dem Satz von Trakhtenbrot (Theorem 5.11) und der Äquivalenz zwischen relationaler Algebra und dem Relationenkalkül. Details: *Übung*.

Auswertungsproblem: Boolesche Anfragen $\rightsquigarrow$ beliebige Anfragen

Theorem 5.20

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

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

Beweis: Identisch zum Beweis von Theorem 2.20.

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

Zur Erinnerung: In Kapitel 2 haben wir gesehen, dass das Auswertungsproblem bereits für Boolesche Anfragen des konjunktiven Kalküls NP-vollständig ist.

Die Komplexitätsklasse PSPACE ("polynomieller Platz")

Zur Erinnerung:

- Ein Entscheidungsproblem B gehört zur Klasse PSPACE, falls es eine (deterministische) 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:
  Die Berechnung von T bei Eingabe w benutzt $\leqslant N^c$ Bandzellen und endet mit der Ausgabe "ja", falls w eine "ja"-Instanz für B ist, bzw. mit der Ausgabe "nein", falls w eine "nein"-Instanz für B ist.
- Es gilt: $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE} \subseteq \mathrm{EXPTIME}$.
- Ein Entscheidungsproblem B heißt PSPACE-vollständig (bzgl. Polynomialzeit-Reduktionen), falls gilt:
  (1) $B \in \mathrm{PSPACE}$ und
  (2) B ist PSPACE-hart, d.h. für jedes Problem $A \in \mathrm{PSPACE}$ 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
  $$w \text{ ist eine "ja"-Instanz für } A \iff f(w) \text{ ist eine "ja"-Instanz für } B.$$
- Es gilt: (A PSPACE-hart und $A \leqslant_p B$) $\Longrightarrow$ B PSPACE-hart.

Auswertungskomplexität des Relationenkalküls

Theorem 5.21 (Chandra, Merlin, 1977)
Das Auswertungsproblem für Boolesche Anfragen des Relationenkalküls ist PSPACE*-vollständig.* *(kombinierte Komplexität)*

Beweis: Siehe Tafel ...

Zum Nachweis der PSPACE-Härte benutzen wir das folgende Resultat (hier ohne Beweis):

Theorem (Stockmeyer, Meyer, 1973) QBF *ist* PSPACE*-vollständig.*

Das Problem QBF ist dabei folgendermaßen definiert: (QBF steht für "Quantified Boolean Formulas")

QBF
Eingabe: Quantifizierte Aussagenlogische Formel ψ
Frage: Hat ψ den Wert 1 ?

"Quantifizierte Aussagenlogische Formel" bedeutet:
ψ ist von der Form $\exists X_1 \, \forall X_2 \, \exists X_3 \cdots Q_m X_m \, \chi$ wobei $m \geqslant 1$, $Q_m = \exists$, falls m ungerade und $Q_m = \forall$, falls m gerade, χ eine Aussagenlogische Formel über den Variablen $X_1, \ldots, X_m$.
Die Formel ψ hat den Wert 1, falls gilt: Es gibt eine Belegung von X_1 (mit 0 oder 1), so dass es für jede Belegung von X_2 eine Belegung von X_3 gibt, so dass ... χ gilt.
(Andernfalls hat ψ den Wert 0.)

Datenkomplexität des Relationenkalküls (1/3)

Datenkomplexität:

- Blickwinkel: Anfrage ist fest; die Eingabe besteht "nur" aus der Datenbank
- Rechtfertigung: i.d.R. ist die Anfrage kurz, die Datenbank aber sehr groß. Bei DB-Anwendungen in der Praxis außerdem oft: einige (wenige) Standard-Anfragen, die immer wieder gestellt werden (d.h.: Anfagen fest, die Einträge in der Datenbank können sich mit der Zeit aber ändern).
- Notation: Für jede feste Anfrage Q des Relationenkalküls (bzw. der relationalen Algebra) bezeichnet EVAL_Q das folgende Auswertungsproblem:

> EVAL_Q
> *Eingabe:* Datenbank $\mathbf{I}$ (vom zu Q "passenden" Datenbankschema)
> *Aufgabe:* Berechne $[\![Q]\!](\mathbf{I})$.

- Für jede feste Anfrage Q der relationalen Algebra kann EVAL_Q in polynomieller Zeit gelöst werden (denn gemäß Proposition 3.4 in Zeit $(k+n)^{\mathcal{O}(k)}$, wobei $k := ||Q||$)
- Dies lässt sich noch deutlich verbessern zur Aussage: EVAL_Q kann in konstanter Zeit gelöst werden auf Parallelrechnern mit polynomiell vielen Prozessoren. Genauer: EVAL_Q gehört zur Komplexitätsklasse AC^0 ...

Datenkomplexität des Relationenkalküls (2/3)

Schaltkreise:

- Wir betrachten Schaltkreise mit $\wedge$, $\vee$, $\neg$-Gattern; wobei $\wedge$ und $\vee$-Gatter beliebig viele Eingänge haben dürfen
- Zum "Zugriff" auf die Eingabe-Datenbank $\mathbf{I}$ außerdem:
  Für jedes $R \in \mathbf{R}$, $r := \mathit{arity}(R)$ und alle $i_1, \ldots, i_r \in \{1, \ldots, |\mathit{adom}(\mathbf{I})|\}$ ein mögliches "Eingabe-Gatter" der Form "$R(i_1, \ldots, i_r)$" zum Testen, ob das aus den entsprechenden Werten gebildete Tupel zur Relation $\mathbf{I}(R)$ gehört oder nicht.
  Analog auch für jedes $c \in \mathbf{dom}$ Eingabe-Gatter der Form "$i{=}c$" zum Testen, ob das i-te Element in $\mathit{adom}(\mathbf{I})$ die Konstante c ist.

Definition 5.22 ($\text{AC}^0 \mathrel{\widehat{=}}$ *"Probleme, die mit Schaltkreisen konst. Tiefe und polyn. Größe lösbar sind"*)

Sei Q eine Boolesche Anfrage an Datenbanken vom Schema $\mathbf{R}$.
Das Problem EVAL_Q gehört genau dann zur Komplexitätsklasse AC^0, wenn es eine Familie $(C_m)_{m \geqslant 1}$ von Schaltkreisen (der obigen Art) gibt, so dass gilt:

- Es gibt eine Zahl $d \in \mathbb{N}$, so dass für jedes $m \geqslant 1$ gilt: C_m hat die Tiefe $\leqslant d$.
- Es gibt eine Zahl $d' \in \mathbb{N}$, so dass für jedes $m \geqslant 1$ gilt: C_m besteht aus maximal $m^{d'}$ vielen Gattern.
- Für jedes $m \geqslant 1$ und jede Datenbank $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ mit $|\mathit{adom}(\mathbf{I})| = m$ gilt:
  C_m akzeptiert Eingabe $\mathbf{I} \iff [\![Q]\!](\mathbf{I}) =$ "ja"

Datenkomplexität des Relationenkalküls (3/3)

> *Theorem 5.23 (Immerman, 1987)*
> *Für jede Boolesche Anfrage Q des Relationenkalküls gehört EVAL_Q zu AC^0*

Beweis: Hier nur die Beweisidee (Details: *Übung*) anhand der Booleschen Anfrage

$$Q := \Big\{ \langle\rangle : \underbrace{\exists x\, \forall y\, \exists z\, \big(R(x,y) \wedge \neg z{=}c\big)}_{=:\varphi} \Big\}$$

Ansatz zur Konstruktion von C_m (auszuwerten über einer Datenbank $\mathbf{I}$ mit $|\mathit{adom}(\mathbf{I})| = m$) :

$$\varphi \quad \rightsquigarrow \quad \bigvee_{i=1}^{m} \bigwedge_{j=1}^{m} \bigvee_{\ell=1}^{m} \Big(R(i,j) \wedge \neg \ell{=}c \Big) \quad \rightsquigarrow \quad \text{Schaltkreis } C_m \text{ (siehe Tafel) der Tiefe } \leqslant ||\varphi||$$

Relationenkalkül

Beispiel: Frankfurter U-Bahn-Netz

Hier vereinfacht: Eine Relation *U-Bahn-Netz* mit Attributen *Linie*, *Halt*, *nächsterHalt*

U-Bahn-Netz

Linie	*Halt*	*nächsterHalt*
U4	Bockenheimer Warte	Festhalle/Messe
U4	Festhalle/Messe	Hauptbahnhof
U4	Hauptbahnhof	Willy-Brandt-Platz
U4	Willy-Brandt-Platz	Dom/Römer
...	...	...
U7	...	...
U7	Kirchplatz	Leipziger Str.
U7	Leipziger Str.	Bockenheimer Warte
U7	Bockenheimer Warte	Westend
...	...	...

Anfrage:

Gib alle Stationen aus, die von "Bockenheimer Warte" aus ohne Umsteigen zu erreichen sind.

(1) mit max. 1 Zwischenhalt:

$$\Big\{\langle x_S\rangle : \exists x_L\Big(\textit{U-Bahn-Netz}(x_L, \text{“Bockenheimer Warte”}, x_S) \vee \exists x_Z\big(\textit{U-Bahn-Netz}(x_L, \text{“Bockenheimer Warte”}, x_Z) \wedge \textit{U-Bahn-Netz}(x_L, x_Z, x_S)\big)\Big)\Big\}$$

(2) mit max. 2 Zwischenhalten: *analog*

(3) mit beliebig vielen Zwischenhalten ???

Gaifman-Lokalität einer Anfrage

Definition 5.24

Sei **R** ein Datenbankschema, sei $r \geqslant 1$.
Eine Anfrage Q der Stelligkeit r heißt Gaifman-lokal, falls es ein $d \geqslant 0$ gibt, so dass für alle Datenbanken $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ und alle Tupel $a \in \mathbf{dom}^r$ und $b \in \mathbf{dom}^r$ gilt:
Falls $\mathcal{N}_d^{\mathbf{I},Q}(a) \cong \mathcal{N}_d^{\mathbf{I},Q}(b)$, so $\big(a \in [\![Q]\!](\mathbf{I}) \iff b \in [\![Q]\!](\mathbf{I})\big)$.

Notation hierbei:

- $\mathcal{N}_d^{\mathbf{I},Q}(a)$: die d-Nachbarschaft von $a = \langle a_1, \ldots, a_r\rangle$, d.h.: die Datenbank $\mathbf{I}$, eingeschränkt auf die Elemente $N_d^{\mathbf{I},Q}(a)$, wobei $N_0^{\mathbf{I},Q}(a) := \{a_1, \ldots, a_r\} \cup \mathit{adom}(Q)$

$$N_{i+1}^{\mathbf{I},Q}(a) := N_i^{\mathbf{I},Q}(a) \cup \left\{ c \in \mathbf{dom} : \begin{array}{l}\text{es gibt ein Tupel } t \text{ in } \mathbf{I}, \text{ in dem sowohl } c \text{ als auch}\\ \text{mind. ein Element aus } N_i^{\mathbf{I},Q}(a) \text{ vorkommt}\end{array} \right\}$$

- $\mathcal{N}_d^{\mathbf{I},Q}(a) \cong \mathcal{N}_d^{\mathbf{I},Q}(b)$: $\mathcal{N}_d^{\mathbf{I},Q}(a)$ ist isomorph zu $\mathcal{N}_d^{\mathbf{I},Q}(b)$, d.h. es gibt eine bijektive Abbildung f von $X := N_d^{\mathbf{I},Q}(a)$ nach $Y := N_d^{\mathbf{I},Q}(b)$, so dass gilt:
  - $f(a) = b$,
  - $f(c) = c$, für alle $c \in \mathit{adom}(Q)$,
  - für jedes $R \in \mathbf{R}$ und jedes Tupel $t \in X^{\mathit{arity}(R)}$ gilt: $t \in \mathbf{I}(R) \iff f(t) \in \mathbf{I}(R)$.

Gaifman-Lokalität des Relationenkalküls

Zur Erinnerung:

Eine Anfrage Q der Stelligkeit r heißt Gaifman-lokal, falls es ein $d \geqslant 0$ gibt, so dass für alle Datenbanken $\mathbf{I} \in \mathit{inst}(\mathbf{R})$ und alle Tupel $a \in \mathbf{dom}^r$ und $b \in \mathbf{dom}^r$ gilt:
Falls $\mathcal{N}_d^{\mathbf{I},Q}(a) \cong \mathcal{N}_d^{\mathbf{I},Q}(b)$, so $\big(a \in [\![Q]\!](\mathbf{I}) \iff b \in [\![Q]\!](\mathbf{I})\big)$.

Theorem 5.25 (Gaifman-Lokalität) *(hier ohne Beweis)*

Jede Anfrage Q des Relationenkalküls $\mathrm{CALC}_{\mathit{adom}}$ *ist Gaifman-lokal.*

Beweis: Siehe Vorlesung Logik in der Informatik.

Anwendung der Gaifman-Lokalität

zum Nachweis, dass eine Anfragefunktion q nicht im Relationenkalkül beschreibbar ist

Beispiel 5.26

Die Anfrage "Gib alle Stationen aus, die von "Bockenheimer Warte" aus ohne Umsteigen zu erreichen sind" kann nicht im Relationenkalkül (also auch nicht in der relationalen Algebra) beschrieben werden.

Beweis: Siehe Tafel.

Bemerkung:

Mit etwas anderen Methoden kann man auch zeigen, dass der Relationenkalkül "nicht zählen kann". Zum Beispiel kann die Anfrage "Hat Stephen Spielberg mehr Filme gedreht als Alfred Hitchcock?" nicht im Relationenkalkül ausgedrückt werden.

Beweismethode:
Ehrenfeucht-Fraïssé-Spiele; siehe Vorlesung Logik in der Informatik.