

der Chase, jetzt formal ...

Datenbank-Instanzen

- 1 Wir fixieren drei unendliche, paarweise disjunkte Mengen Δ , Δ_{null} und V .
 - Δ ist der zugrunde gelegte Domain.
 - $\Delta_{null} = \{n_1, n_2, \dots\}$ ist die Menge der Null-Werte. Wir sprechen in diesem Zusammenhang von "labelled nulls".
 - V bezeichne die Menge der Variablen.
- 2 Eine Datenbank-Fakt ist ein \mathcal{R} -Atom mit Argumenten aus $\Delta \cup \Delta_{null}$.
- 3 Eine Datenbankinstanz \mathcal{I} ist eine (nicht notwendigerweise endliche) Menge von Datenbanken-Fakten.

Beispiel

$\mathcal{I} = \{fly(Fft, B, 600), fly(Fft, C, n_1), fly(n_2, C, n_1), \dots\}$
 n_1, n_2 sind Null-Werte (labelled nulls).

Umgekehrt kann eine Datenbank-Instanz in eine konjunktive Anfrage umgewandelt werden:

Beispiel

<i>ans</i>	<i>fly</i>	<i>hasAirport</i>
<i>n</i> ₁	<i>n</i> ₁ <i>n</i> ₂ <i>n</i> ₃	<i>n</i> ₁
	<i>n</i> ₂ <i>Bonn</i> <i>n</i> ₄	<i>n</i> ₂
		<i>Bonn</i>

$ans(X) \leftarrow fly(X, Y, D_1), fly(Y, Bonn, D_2), hasAirport(X), hasAirport(Y), hasAirport(Bonn)$

Konjunktive Anfragen als Datenbank-Instanz

Den Rumpf $body(\bar{x}, \bar{y})$ einer konjunktiven Anfrage

$$Q : ans(\bar{x}) \leftarrow body(\bar{x}, \bar{y})$$

fassen wir als eine Datenbank-Instanz $db(Q)$ auf, wobei wir die Variablen durch Null-Werte ersetzen (durch eine injektive Abbildung).

Das Ergebnis des Chase auf der Instanz $db(Q)$ bezeichnen wir weiterhin als Q^Σ .

Beispiel

$ans(X, D) \leftarrow fly(Bonn, X, D), fly(X, Bonn, D), hasAirport(X), hasAirport(Bonn)$

$db(Q) :$

<i>ans</i>	<i>fly</i>	<i>hasAirport</i>
<i>n</i> ₁ <i>n</i> ₂	<i>Bonn</i> <i>n</i> ₁ <i>n</i> ₂	<i>n</i> ₁
	<i>n</i> ₁ <i>Bonn</i> <i>n</i> ₂	<i>Bonn</i>

Der Chase-Algorithmus kann auf beliebigen endlichen (!) Datenbank-Instanzen angewendet werden, insbesondere auf $db(Q)$, wobei Q eine konjunktive Anfrage.

Anwendbarkeit eines Chase-Schritts

- Sei $\alpha \in \Sigma$ und \mathcal{I} eine endliche Datenbank-Instanz.
- Bezeichne $\tilde{\alpha}$ den Constraint der aus α hervorgeht indem man den \forall -Quantor und die vorangestellte Liste der allquantifizierten Variablen streicht.
- Sei $\bar{a} \in (\Delta \cup \Delta_{null})^*$.
- Das Tupel (α, \bar{a}) heißt anwendbar auf \mathcal{I} , falls $\mathcal{I} \not\models \tilde{\alpha}(\bar{a})$.
- Der Constraint α heißt anwendbar auf \mathcal{I} falls es ein $\bar{b} \in (\Delta \cup \Delta_{null})^*$ gibt, so dass (α, \bar{b}) anwendbar auf \mathcal{I} ist.

Definition: Homomorphismen

Seien $\mathcal{I}, \mathcal{I}'$ Datenbanken-Instanzen. Ein Homomorphismus von \mathcal{I} nach \mathcal{I}' ist eine Abbildung $h : \Delta \cup \Delta_{null} \cup V \rightarrow \Delta \cup \Delta_{null}$ mit

- für alle $c \in \Delta$ gilt $h(c) = c$ und
- für alle $R(t_1, \dots, t_n) \in I$ gilt $h(R(t_1, \dots, t_n)) := R(h(t_1), \dots, h(t_n)) \in I'$.

Ein Homomorphismus wird in natürlicher Weise fortgesetzt auf $(\Delta \cup \Delta_{null} \cup V)^*$.

Unterschied zu Enthaltensein-Abbildungen?

Beispiel

$$\mathcal{I} = \{R(a, b)\}$$

$$\Sigma := \{\forall x_1, x_2 (R(x_1, x_2) \rightarrow \exists y S(x_2, y))\}$$

Chase-Schritt: TGD

- Sei (α, \bar{a}) anwendbar auf \mathcal{I} , wobei α eine TGD ist. Die Liste der allquantifizierten Variablen in α sei \bar{x} .
- Der Chase-Schritt $\mathcal{I} \xrightarrow{\alpha, \bar{a}} \mathcal{J}$ ist definiert wie folgt.
 - Sei h ein Homomorphismus mit $h(\bar{x}) = \bar{a}$.
 - Für jede existentiell quantifizierte Variable y in α wähle einen "neuen" Null-Wert $n_y \in \Delta_{null}$ und definiere $h(y) := n_y$.
 - Setze $\mathcal{J} := \mathcal{I} \cup h(\text{head}(\alpha))$.

Was gilt nun in \mathcal{J} ?

Chase-Schritt: EGD

- Sei (α, \bar{a}) anwendbar auf \mathcal{I} , wobei α eine EGD ist.
- Seien x_1, x_2 die beiden Variablen, welche in $\text{head}(\alpha)$ vorkommen.
- Sei h ein Homomorphismus mit $h(\bar{x}) = \bar{a}$.
- Wenn $h(x_1), h(x_2) \in \Delta$, dann ist der Chase-Schritt undefiniert, d.h. der Chase schlägt fehl.
- Ansonsten ist der Chase-Schritt $\mathcal{I} \xrightarrow{\alpha, \bar{a}} \mathcal{J}$ definiert wie folgt.
 - Definiere einen neuen Homomorphismus h' , der sich von h höchstens an den Stellen x_1, x_2 unterscheidet. Falls $h(x_1) \in \Delta$ dann $h'(x_1) := h'(x_2) := h(x_1)$. Falls $h(x_1) \notin \Delta$ dann $h'(x_2) := h'(x_1) := h(x_2)$.
 - Setze $\mathcal{J} := h'(\mathcal{I})$.

Was gilt nun in \mathcal{J} ?

Beispiel

$$\mathcal{I} = \{R(n_1, n_2), R(n_1, n_3)\}$$

$$\Sigma := \{\forall x_1, x_2, x_3 (R(x_1, x_2) \wedge R(x_1, x_3) \rightarrow x_2 = x_3)\}$$

Das Ergebnis eines Chase-Schritts hängt davon ab ob n_1, n_2, n_3 Null-Werte sind, oder nicht.

Chase-Sequenzen

Eine Chase-Sequenz ist eine (nicht-notwendigerweise endliche) Folge von Chase-Schritten $\mathcal{I}_0 \xrightarrow{\alpha_0, \vec{a}_0} \mathcal{I}_1 \xrightarrow{\alpha_1, \vec{a}_1} \mathcal{I}_2 \dots$, wobei wir keinerlei Annahme darüber treffen in welcher Reihenfolge Constraints angewendet werden. Falls eine solche Sequenz endlich ist und kein weiterer Chase-Schritt mehr anwendbar ist, dann sagen wir dass die Sequenz terminiert.

Achtung: unterschiedliche Reihenfolgen der Constraint-Anwendung führen zu unterschiedlichen Chase-Resultaten.

Satz

Sei \mathcal{J} die letzte Datenbank-Instanz in einer terminierenden Chase-Sequenz ausgehend von der Instanz \mathcal{I} .

- Es gilt $\mathcal{J} \models \Sigma$.
- Es gilt $\mathcal{J} \models \mathcal{I}$.

Sei n ein Null-Wert.

Beispiel

$$\mathcal{I} = \{R(a, b), R(a, n)\}$$

$$\Sigma := \{ \forall x_1, x_2 (R(x_1, x_2) \rightarrow \exists y S(x_2, y)), \\ \forall x_1, x_2, x_3 (R(x_1, x_2) \wedge R(x_1, x_3) \rightarrow x_2 = x_3) \}$$

Welche Chase-Sequenzen sind möglich?

Welche Eigenschaften haben verschiedene Chase-Ergebnisse gemeinsam?

Universelle Eigenschaft

Satz

Sei \mathcal{J} die letzte Datenbank-Instanz in einer terminierenden Chase-Sequenz ausgehend von der Instanz \mathcal{I} .

Falls \mathcal{K} eine Datenbankinstanz ist mit $\mathcal{K} \models \mathcal{I} \cup \Sigma$ dann gibt es einen Homomorphismus h von \mathcal{J} nach \mathcal{K} .

Deshalb bezeichnen wir \mathcal{J} als universelle Lösung.

Folgerung

Für je zwei Chase-Ergebnisse $\mathcal{J}, \mathcal{J}'$ gibt es einen Homomorphismus von \mathcal{J} nach \mathcal{J}' und einen Homomorphismus von \mathcal{J}' nach \mathcal{J} .

Dieser Punkt erlaubt es uns von **dem** Chase-Ergebnis \mathcal{I}^Σ zu sprechen, d.h. ein Chase-Ergebnis ist eindeutig bis auf Homomorphie, was für alle bisher bekannten Anwendungen ausreicht. Mit anderen Worten: jede Anwendungsaufgabe kann mit jedem beliebigen Chase-Resultat gelöst werden.



Eine andere Formulierung ...

Satz

Seien Q und Q' konjunktive Anfragen und existiere Q^Σ . Dann gilt:

- $Q \sqsubseteq_\Sigma Q'$ genau dann wenn $Q^\Sigma \sqsubseteq Q'$.
- $Q \sqsubseteq_\Sigma Q'$ genau dann wenn $Q^\Sigma \sqsubseteq Q'^\Sigma$.

Beweis der zweiten Aussage mit folgendem Lemma (Anfragen werden als Datenbank-Instanz betrachtet).

Lemma

Sei $\mathcal{I} \models \Sigma$ und h ein Homomorphismus von \mathcal{J} nach \mathcal{I} . Falls \mathcal{I}^Σ endlich ist, dann kann h zu einem Homomorphismus von \mathcal{J}^Σ nach \mathcal{I} erweitert werden.

**Beweis des Lemmas:**

Induktion über die Anzahl der Chase-Schritte:

- Falls keine Chase-Schritte gemacht werden ist die Behauptung klar.
- Wir skizzieren den Fall eines Chase-Schritts $\mathcal{J}_i \xrightarrow{\alpha, \bar{a}} \mathcal{J}_{i+1}$ für den Fall, dass eine TGD angewendet wird.
Gemäß Induktionsvoraussetzung gibt es einen Homomorphismus h_i von \mathcal{J}_i nach \mathcal{I} mit $h_i \supseteq h$. Sei $\alpha = \forall \bar{x}(\varphi(\bar{x}) \rightarrow \exists \bar{y}\psi(\bar{x}, \bar{y}))$. Seien \bar{n} die Null-Werte, welche in diesem Schritt neu eingeführt werden, d.h. $\mathcal{J}_{i+1} \models \psi(\bar{a}, \bar{n})$. Es gilt $\mathcal{J}_i \models \varphi(\bar{a})$, also $\mathcal{I}^\Sigma \models \varphi(h(\bar{a}))$. Wir wissen, dass $\mathcal{I} \models \Sigma$ gilt, demnach gibt es \bar{b} , so dass $\mathcal{I} \models \psi(h(\bar{a}), \bar{b})$. Wir setzen h_{i+1} gleich h_i auf allen Werten aus \mathcal{J}_i und setzen außerdem $h_{i+1}(\bar{n}) := \bar{b}$. Man rechnet nach, dass h_{i+1} ein Homomorphismus von \mathcal{J}_{i+1} nach \mathcal{I} ist.
- Für den Fall eines Chase-Schritts mit Hilfe einer EGD, setzen wir $h_{i+1} = h_i$.

**Unentscheidbarkeit der Chase-Terminierung****Satz**

Sei Σ eine Menge von Tuple-generating und Equality-generating Dependencies. Es ist im Allgemeinen unentscheidbar ob der Chase mit Σ auf einer fixen Instanz \mathcal{I} terminiert.

⇒ Problematisch in der Praxis



Hinreichende Terminierungsbedingung für den Chase

Lösungsansatz

Hinreichende, statisch überprüfbare Bedingungen über der Constraintmenge, die Chase-Terminierung auf allen Instanzen garantieren.

Hinreichende Terminierungsbedingungen für den Chase

Es wurden verschiedene solcher Bedingungen entwickelt, z.B.

- Acyclicity (wird in der Vorlesung behandelt)
- Weak Acyclicity (wird auf Übungsblatt 3 behandelt)
- Stratification
- ...

⇒ Diese Bedingungen erlauben es, den Chase-Algorithmus in vielen praxisrelevanten Szenarien anzuwenden

Acyclicity

Definition

Sei Σ eine Menge von Tuple-generating und Equality-generating Dependencies. Σ heißt *azyklisch* genau dann wenn der Relationsgraph $rel(\Sigma)$ keinen Zyklus enthält.

Satz

Sei Σ eine azyklische Menge von Tupel-generating und Equality-generating Dependencies. Dann terminiert der Chase-Algorithmus mit Σ für jede Datenbankinstanz.

Intuition: Wenn der Relationsgraph keine Zyklen besitzt können Constraints nie zyklisch feuern.

Acyclicity

Definition

Sei Σ eine Menge von Tuple-generating und Equality-generating Dependencies. Konstruiere den *Relationsgraphen* $rel(\Sigma) = (V, E)$ wie folgt. Für jede Tuple-generating Dependency $\varphi_{tgd} \in \Sigma$

$$\varphi_{tgd} := \forall \bar{x}(R_1(\bar{x}) \wedge \dots \wedge R_m(\bar{x}) \rightarrow \exists \bar{y}S_1(\bar{x}, \bar{y}) \wedge \dots \wedge S_n(\bar{x}, \bar{y}))$$

- erweitere V um die Menge aller Relationsbezeichner in φ_{tgd} , d.h. setze $V := V \cup \{R_1, \dots, R_n, S_1, \dots, S_m\}$ und
- erweitere E um alle Kanten $R_i \rightarrow S_j$ für alle $1 \leq i \leq m, 1 \leq j \leq n$, d.h. setze $E := E \cup \{(R_i, S_j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$.

Anmerkung: Equality-generating Dependencies spielen bei der Konstruktion des Relationsgraphen keine Rolle.

Beispiel Acyclicity

Beispiel

Betrachte die Constraintmenge $\Sigma_1 := \{\alpha_1\}$ mit

$$\alpha_1 := \forall x_1, x_2, y \text{ fly}(x_1, x_2, y) \rightarrow \text{hasAirport}(x_1) \wedge \text{hasAirport}(x_2).$$

Der Relationsgraph $rel(\Sigma_1) = (V_1, E_1)$ ist definiert als

$$V_1 := \{ \text{fly}, \text{hasAirport} \}, \\ E_1 := \{ (\text{fly}, \text{hasAirport}) \}.$$

Offensichtlich enthält $rel(\Sigma_1)$ keinen Zyklus. Gemäß vorigem Satz terminiert der Chase mit Σ_1 für jede Datenbankinstanz.

Beispiel Acyclicity

Beispiel

Betrachte die Constraintmenge $\Sigma_2 := \{\alpha_2\}$ mit

$$\alpha_2 := \forall x_1, x_2, y \text{ rail}(x_1, x_2, y) \rightarrow \text{rail}(x_2, x_1, y).$$

Der Relationsgraph $rel(\Sigma_2) = (V_2, E_2)$ ist definiert als

$$V_2 := \{ \text{rail} \}, \\ E_2 := \{ (\text{rail}, \text{rail}) \}.$$

- $rel(\Sigma_2)$ enthält einen Zyklus \implies keine Terminierungsgarantien ableitbar.
- Man kann sich leicht klarmachen, dass der Chase mit Σ_2 dennoch immer terminiert. Dies zeigt, dass Acyclicity nur eine hinreichende Bedingung ist.

Komplexere hinreichende Bedingungen notwendig um hier Chase Terminierung abzuleiten, siehe z.B. *Weak Acyclicity* auf Übungsblatt 3.



Anfrageminimierung mittels Chase

Algorithmus zur Berechnung minimaler Σ -äquivalenter Anfragen

Eingabe: Q, Σ

- 1 Berechne Q^Σ (falls existent).
- 2 Betrachte alle Teilanfragen $Q' \subseteq Q^\Sigma$ bottom-up, d.h. erst Anfragen mit einem Atom, dann Anfragen mit zwei Atomen . . .
- 3 Überprüfe jeweils, ob $Q'^\Sigma \subseteq Q^\Sigma$. Falls ja, so ist $Q' (\equiv_\Sigma Q^\Sigma \equiv_\Sigma Q)$ eine minimale äquivalente Anfrage zu Q unter Σ .
- 4 Betrachte ggf. noch ausstehende Teilanfragen gleicher Größe, um weitere minimale Σ -äquivalente Anfragen zu finden.

Beachte: Es können mehrere minimale Anfragen existieren, i.A. sogar exponentiell viele.



Anfrageminimierung mittels Chase

Satz

Sei Σ eine Menge von Tuple-generating und Equality-generating Dependencies und Q eine Konjunktive Anfrage. Wenn Q^Σ existiert, dann

- ist jede minimale Anfrage $Q' \equiv_\Sigma Q$ eine Teilanfrage von Q^Σ .
- gilt für jede Teilanfrage $Q' \subseteq Q^\Sigma$: $Q' \equiv_\Sigma Q^\Sigma \Leftrightarrow Q'^\Sigma \subseteq Q^\Sigma$.

Beweis der zweiten Behauptung

Umformen der rechten Seite gemäß dem Satz auf Folie 43 ergibt die Äquivalenz $Q' \equiv_\Sigma Q^\Sigma \Leftrightarrow Q' \sqsubseteq_\Sigma Q^\Sigma$.

Diese Äquivalenz gilt per Definition von \equiv_Σ genau dann wenn $Q^\Sigma \sqsubseteq_\Sigma Q'$ gilt. Aus der Annahme $Q' \subseteq Q^\Sigma$ folgt aber direkt $Q^\Sigma \sqsubseteq_\Sigma Q'$, was $Q^\Sigma \sqsubseteq_\Sigma Q'$ impliziert.



Anfrageminimierung mittels Chase

Beispiel

Betrachten Sie die Constraintmenge $\Sigma = \{\alpha_1, \alpha_2\}$ mit

$$\alpha_1: \forall x_1, x_2, d \text{ fly}(x_1, x_2, d) \rightarrow \text{hasAirport}(x_1) \wedge \text{hasAirport}(x_2) \\ \alpha_2: \forall x_1, x_2, d \text{ fly}(x_1, x_2, d) \rightarrow \text{fly}(x_2, x_1, d)$$

und die Anfrage $Q: \text{ans}(X) \leftarrow \text{fly}(\text{Bonn}, X, Y), \text{hasAirport}(X)$.

Nach Chase mit α_1 und α_2 erhalten wir Q^Σ :

$$\text{ans}(X) \leftarrow \text{fly}(\text{Bonn}, X, D), \text{fly}(X, \text{Bonn}, D), \text{hasAirport}(X), \text{hasAirport}(\text{Bonn})$$

Wir betrachten nun die Teilanfragen von Q^Σ bottom-up und stellen fest, dass für

$$Q': \text{ans}(X) \leftarrow \text{fly}(\text{Bonn}, X, D) \\ Q'': \text{ans}(X) \leftarrow \text{fly}(X, \text{Bonn}, D)$$

die Beziehungen $Q'^\Sigma \subseteq Q^\Sigma$ und $Q''^\Sigma \subseteq Q^\Sigma$ gelten. Folglich sind Q' und Q'' minimale Σ -äquivalente Anfragen zu Q .

