Anfragesprachen mit Rekursion — Datalog

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 ???

Am Ende von Kapitel 4 gesehen: Nicht ausdrückbar in Relationaler Algebra

Erreichbarkeits-Anfrage in SQL

Im SQL (ab SQL-99 Standard) kann die Anfrage “Gib alle Stationen aus, die von “Bockenheimer Warte” aus ohne Umsteigen zu erreichen sind” folgendermaßen ausgedrückt werden:

```
WITH RECURSIVE Erreichbar(Linie,Start,Ziel)
AS (
    SELECT Linie, Halt, NaechsterHalt
    FROM U-Bahn-Netz
 UNION ALL
    SELECT Erreichbar.Linie,
           Erreichbar.Start,
           U-Bahn-Netz.NaechsterHalt
    FROM Erreichbar, U-Bahn-Netz
    WHERE Erreichbar.Linie = U-Bahn-Netz.Linie AND
          Erreichbar.Ziel = U-Bahn-Netz.Halt
)
SELECT Ziel
FROM Erreichbar
WHERE Start=”Bockenheimer Warte”
```

Erreichbarkeits-Anfrage in Datalog

Die Anfrage

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

kann in Datalog folgendermaßen ausgedrückt werden:

$$
\begin{aligned}
\mathit{Erreichbar}(L, S, Z) &\leftarrow \mathit{U\text{-}Bahn\text{-}Netz}(L, S, Z) \\
\mathit{Erreichbar}(L, S, Z) &\leftarrow \mathit{Erreichbar}(L, S, Z'),\ \mathit{U\text{-}Bahn\text{-}Netz}(L, Z', Z) \\
\mathit{Ans}(Z) &\leftarrow \mathit{Erreichbar}(L, \text{"Bockenheimer Warte"}, Z)
\end{aligned}
$$

Anfragesprachen mit Rekursion — Datalog

Datalog: Syntax

Definition 7.1

(a) Eine Datalog-Regel ist ein Ausdruck der Form

$$R_0(u_0) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$$

wobei $\ell \geqslant 0$, $R_0, R_1, \ldots, R_\ell \in$ **relname** und $u_0, u_1, \ldots, u_\ell$ freie Tupel der Stelligkeiten $\mathit{arity}(R_0), \mathit{arity}(R_1), \ldots, \mathit{arity}(R_\ell)$ sind, so dass jede Variable, die in u_0 vorkommt, auch in mindestens einem der Tupel $u_1, \ldots, u_\ell$ vorkommt.

(b) Ein Datalog-Programm P ist eine endliche Menge von Datalog-Regeln.

(c) Eine Datalog-Anfrage (P, R) besteht aus einem Datalog-Programm P und einem Relations-Namen R, der in P vorkommt.

Notation

- Wie üblich bezeichnen wir mit $adom(P)$ bzw. $adom(Q)$ die Menge der Konstanten, die in einem Datalog-Programm P bzw. einer Datalog-Anfrage Q vorkommen.

  Für eine Datanbank $\mathbf{I}$ ist $adom(P, \mathbf{I}) := adom(P) \cup adom(\mathbf{I})$ und $adom(Q, \mathbf{I}) := adom(Q) \cup adom(\mathbf{I})$.

- $R_0(u_0)$ heißt Kopf der Regel $R_0(u_0) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$;
  $R_1(u_1), \ldots, R_\ell(u_\ell)$ heißt Rumpf der Regel.

- $edb(P)$ bezeichnet die Menge der Relations-Namen, die ausschließlich im Rumpf von Regeln in P vorkommen (die sog. “extensionalen Prädikate von P”).

- $idb(P)$ bezeichnet die Menge der Relations-Namen, die im Kopf mindestens einer Regel in P vorkommen (die sog. “intensionalen Prädikate von P”).

- $sch(P) := edb(P) \dot{\cup} idb(P)$ heißt Schema von P.

Beispiel

$$P := \left\{ \begin{array}{rcl} \mathit{Erreichbar}(L,S,Z) & \leftarrow & \mathit{U\text{-}Bahn\text{-}Netz}(L,S,Z)\ , \\ \mathit{Erreichbar}(L,S,Z) & \leftarrow & \mathit{Erreichbar}(L,S,Z'),\ \mathit{U\text{-}Bahn\text{-}Netz}(L,Z',Z)\ , \\ \mathit{Ans}(Z) & \leftarrow & \mathit{Erreichbar}(L, \text{“Bockenheimer Warte”}, Z) \end{array} \right\}$$

ist ein Datalog-Programm mit

- $\mathit{edb}(P) = \{\ \mathit{U\text{-}Bahn\text{-}Netz}\ \}$
- $\mathit{idb}(P) = \{\ \mathit{Erreichbar, Ans}\ \}$
- $\mathit{sch}(P) = \{\ \mathit{U\text{-}Bahn\text{-}Netz, Erreichbar, Ans}\ \}$

$Q_1 := (P, \mathit{Ans})$ ist eine Datalog-Anfrage;

$Q_2 := (P, \mathit{Erreichbar})$ ist noch eine Datalog-Anfrage.

Datalog: Semantik

Die Semantik von Datalog lässt sich auf verschiedene Weisen definieren:

- Fixpunkt-Semantik:
  "Regeln schrittweise anwenden, bis sich nichts mehr ändert"
- Modellbasierte Semantik:
  kleinste Datenbank über $sch(P)$, die alle "Regeln wahr macht"
- Beweisbasierte Semantik:
  "Fakten, die sich herleiten lassen, sind im Ergebnis"

Glücklicherweise kann man zeigen, dass alle drei Ansätze zum gleichen Resultat führen.
Wir betrachten im Folgenden hauptsächlich die Fixpunkt-Semantik, die folgendermaßen definiert ist:

Der "immediate consequence"-Operator T_P

Definition 7.2

Sei P ein Datalog-Programm.

(a) Für jedes $R \in idb(P)$ seien $Q_{R,1}, \ldots, Q_{R,k_R}$ diejenigen Regeln aus P, in deren Kopf das Relationssymbol R steht, und seien $Q'_{R,1}, \ldots, Q'_{R,k_R}$ die Regeln, die aus $Q_{R,1}, \ldots, Q_{R,k_R}$ entstehen, indem im Kopf jeweils R durch das Relationssymbol Ans' ersetzt wird (mit $Ans' \notin sch(P)$ und $arity(Ans') = arity(R)$).

(b) Der "immediate consequence"-Operator $T_P : inst(sch(P)) \to inst(sch(P))$ ist folgendermaßen definiert:
Für jedes $\mathbf{I} \in inst(sch(P))$ und jedes $R \in sch(P)$ ist

$$T_P(\mathbf{I})(R) := \begin{cases} \mathbf{I}(R) & \text{falls } R \in edb(P) \\ [\![Q'_{R,1}]\!](\mathbf{I}) \cup \cdots \cup [\![Q'_{R,k_r}]\!](\mathbf{I}) & \text{falls } R \in idb(P) \end{cases}$$

Beispiel: Ist P das Datalog-Programm aus dem Beispiel der Erreichbarkeits-Anfrage, und besteht $\mathbf{I}(Erreichbar)$ aus allen Tupeln, die "mit max. i Zwischenhalten ohne Umsteigen zu erreichen sind", so besteht $T_P(I)(Erreichbar)$ aus allen Tupeln, die "mit max. $i+1$ Zwischenhalten ohne Umsteigen zu erreichen sind".

Monotonie von T_P

Bemerkung: Eine alternative (und äquivalente) Definition von T_P:

$$T_P(\mathbf{I})(R) := \{ \text{ unmittelbare } R\text{-Konsequenzen von } P \text{ über } \mathbf{I} \},$$

wobei ein Tupel t eine unmittelbare R-Konsequenz von P über $\mathbf{I}$ ist, falls

- $R \in edb(P)$ und $t \in \mathbf{I}(R)$ oder
- $R \in idb(P)$ und es eine Regel der Form $R(u) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$ in P und eine Belegung β gibt, so dass $\beta(u) = t$ und $\beta(u_i) \in \mathbf{I}(R_i)$, für alle $i \in \{1, .., \ell\}$.

Lemma 7.3

Für jedes Datalog-Programm P gilt:

Der Operator T_P ist monoton, d.h. für alle $\mathbf{I}, \mathbf{J} \in inst(sch(P))$ mit $\mathbf{I} \subseteq \mathbf{J}$ (d.h. $\mathbf{I}(R) \subseteq \mathbf{J}(R)$ f.a. $R \in inst(sch(P))$) gilt: $T_P(\mathbf{I}) \subseteq T_P(\mathbf{J})$.

Beweis: leicht (*Übung*).

Fixpunkte

Bemerkung 7.4

(a) Aus der Monotonie von T_P folgt für jedes $\mathbf{I} \in inst(sch(P))$ mit $\mathbf{I}(R) = \emptyset$ f.a. $R \in idb(P)$, dass:

$$\mathbf{I} \subseteq T_P(\mathbf{I}) \subseteq T_P\big(T_P(\mathbf{I})\big) \subseteq \cdots \subseteq T_P^i(\mathbf{I}) \subseteq T_P^{i+1}(\mathbf{I}) \subseteq \cdots$$

wobei $T_P^0(\mathbf{I}) := \mathbf{I}$ und $T_P^{i+1}(\mathbf{I}) := T_P\big(T_P^i(\mathbf{I})\big)$.

(b) Man sieht leicht, dass $adom(T_P(\mathbf{I})) \subseteq adom(P, \mathbf{I})$ und $adom(T_P^i(\mathbf{I})) \subseteq adom(P, \mathbf{I})$, f.a. $i \in \mathbb{N}$.

(c) Da $adom(P, \mathbf{I})$ nur endlich viele Elemente besitzt, muss es in der Inklusionskette aus (a) ein $i_0 \in \mathbb{N}$ mit $T_P^{i_0}(\mathbf{I}) = T_P^{i_0+1}(\mathbf{I})$ geben.
Aus der Monotonie von T_P folgt dann: $T_P^{i_0}(\mathbf{I}) = T_P^j(\mathbf{I})$ f.a. $j \geqslant i_0$.

Notation

- Ein $\mathbf{J} \in \mathit{inst}(\mathit{sch}(P))$ heißt Fixpunkt von T_P, falls $T_P(\mathbf{J}) = \mathbf{J}$.

- $\mathit{Ab\text{-}Stufe}(\mathbf{I}) := \min\{i_0 : T_P^{i_0}(\mathbf{I}) = T_P^{i_0+1}(\mathbf{I})\}$ heißt "Abschluss-Stufe" von T_P auf I.
  Die einzelnen Instanzen $T_P^0(\mathbf{I})$, $T_P^1(\mathbf{I})$, $T_P^2(\mathbf{I})$ etc. werden "Stufen des Fixpunktprozesses" genannt. $\mathit{Ab\text{-}Stufe}(P, \mathbf{I})$ gibt also diejenige Stufe an, ab der der Fixpunkt erreicht ist:

$$\mathbf{I} = T_P^0(\mathbf{I}) \subsetneq T_P^1(\mathbf{I}) \subsetneq \cdots \subsetneq T_P^{\mathit{Ab\text{-}Stufe}(P,\mathbf{I})}(\mathbf{I}) = T_P^{\mathit{Ab\text{-}Stufe}(P,\mathbf{I})+1}(\mathbf{I}) = T_P^j(\mathbf{I})$$

f.a. $j \geqslant \mathit{Ab\text{-}Stufe}(P, \mathbf{I})$

- $T_P^\omega(\mathbf{I}) := T_P^{\mathit{Ab\text{-}Stufe}(\mathbf{I})}(\mathbf{I})$. Klar: $T_P^\omega(\mathbf{I})$ ist ein Fixpunkt von T_P.

Fixpunkt-Semantik von Datalog

Definition 7.5

(a) Sei P ein Datalog-Programm, $\mathbf{R} := edb(P)$.
Jeder Datenbank $\mathbf{I} \in inst(\mathbf{R})$ ordnen wir die Datenbank $\hat{\mathbf{I}} \in inst(sch(P))$ mit

$$\hat{\mathbf{I}}(R) := \begin{cases} \mathbf{I}(R) & \text{falls } R \in edb(P) \\ \emptyset & \text{falls } R \in idb(P) \end{cases}$$

zu. Ausgewertet in $\mathbf{I}$ liefert P die Datenbank

$$[\![P]\!](\mathbf{I}) := T_P^\omega(\hat{\mathbf{I}}) \in inst(sch(P))$$

(b) Eine Datalog-Anfrage $Q = (P, R)$ liefert auf einer Datenbank $\mathbf{I} \in inst(edb(P))$ die Relation

$$[\![Q]\!](\mathbf{I}) := ([\![P]\!](\mathbf{I}))(R).$$

Naive Berechnung von $[\![P]\!](\mathbf{I})$

Bemerkung:

Ein einfacher Algorithmus, der bei Eingabe von P und $\mathbf{I}$ das Resultat $[\![P]\!](\mathbf{I})$ berechnet, kann folgendermaßen vorgehen:

(1) $\mathbf{J} := \hat{\mathbf{I}}$
(2) Berechne $\mathbf{J}' := T_P(\mathbf{J})$
(3) Falls $\mathbf{J}' \neq \mathbf{J}$, so $\big(\ \mathbf{J} := \mathbf{J}',$ GOTO (2) $\big)$
(4) Gib $\mathbf{J}$ aus

Datenkomplexität dieses Algorithmus: Polynomialzeit.

Fixpunkt-Semantik vs. Modellbasierte Semantik

Notation:

Für zwei Datenbanken $\mathbf{J}, \mathbf{J}' \in inst(\mathbf{R})$ bezeichnet $\mathbf{J} \cap \mathbf{J}'$ die Datenbank mit $(\mathbf{J} \cap \mathbf{J}')(R) := \mathbf{J}(R) \cap \mathbf{J}'(R)$, f.a. $R \in \mathbf{R}$.

Satz 7.6 (Knaster und Tarski)

Sei P ein Datalog-Programm und $\mathbf{I} \in inst(edb(P))$. Dann gilt:

$$[\![P]\!](\mathbf{I}) \;=\; \bigcap \{\, \mathbf{J} \in inst(sch(P)) \;:\; T_P(\mathbf{J}) = \mathbf{J} \;\; und \;\; \hat{\mathbf{I}} \subseteq \mathbf{J} \}$$

D.h.: $[\![P]\!](\mathbf{I})$ ist die kleinste Erweiterung von $\hat{\mathbf{I}}$, die ein Fixpunkt von T_P ist.

Beweis: Siehe Tafel.

Ausdrucksstärke von Datalog

Bemerkung 7.7

Die Ausdrucksstärken von Datalog-Anfragen und von Anfragen des Relationenkalküls sind unvergleichbar:

- Beispiel für eine Datalog-Anfrage, die nicht im Relationenkalkül formuliert werden kann:

  "Welche Stationen sind von "Bockenheimer Warte" aus ohne Umsteigen zu erreichen?"

- Beispiel für eine Relationenkalkül-Anfrage, die nicht in Datalog formuliert werden kann:

  "In welchen Kinos läuft kein Film um 17:00 Uhr?"

  Bzw. jede andere Relationenkalkül-Anfrage, die nicht monoton ist.

  Denn aus Lemma 7.3 folgt leicht, dass jede Datalog-Anfrage monoton ist.

Beweisbasierte Semantik von Datalog

Sichtweise:

- Ein Faktum ist ein Ausdruck der Form $R(t)$ mit $R \in \mathbf{relname}$ und $t \in \mathbf{dom}^{arity(R)}$.
- Eine Datenbank $\mathbf{I} \in inst(\mathbf{R})$ wird mit der folgenden Menge von Fakten identifiziert:

$$Fakten(\mathbf{I}) \;:=\; \{\; R(t) \;:\; R \in \mathbf{R} \text{ und } t \in \mathbf{I}(R) \;\}$$

- Für die beweisbasierte Semantik werden Datalog-Regeln als Schlussregel-Muster in Beweisen betrachtet.

Beweisbäume

Definition 7.8

Sei P ein Datalog-Programm und $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$. Ein Beweisbaum für ein Faktum $R(t)$ bzgl. $\mathbf{I}$ und P ist ein gerichteter Baum mit den folgenden Eigenschaften:

(1) Jeder Knoten ist mit einem Faktum markiert.

(2) Die Wurzel enthält das Faktum $R(t)$.

(3) Die Fakten an den Blättern sind Elemente aus $\mathit{Fakten}(\mathbf{I})$

(4) Für jeden inneren Knoten v mit Kindern $v_1, \ldots, v_\ell$ gibt es eine Regel $R_0(u_0) \leftarrow R_1(u_1), \ldots, R_\ell(u_\ell)$ in P und eine Belegung $\beta : \mathit{Var}(P) \to \mathbf{dom}$ so dass gilt:

- der Knoten v enthält das Faktum $R_0(\beta(u_0))$
- das Kind v_i von v enthält das Faktum $R_i(\beta(u_i))$ (für alle $i \in \{1, \ldots, \ell\}$).

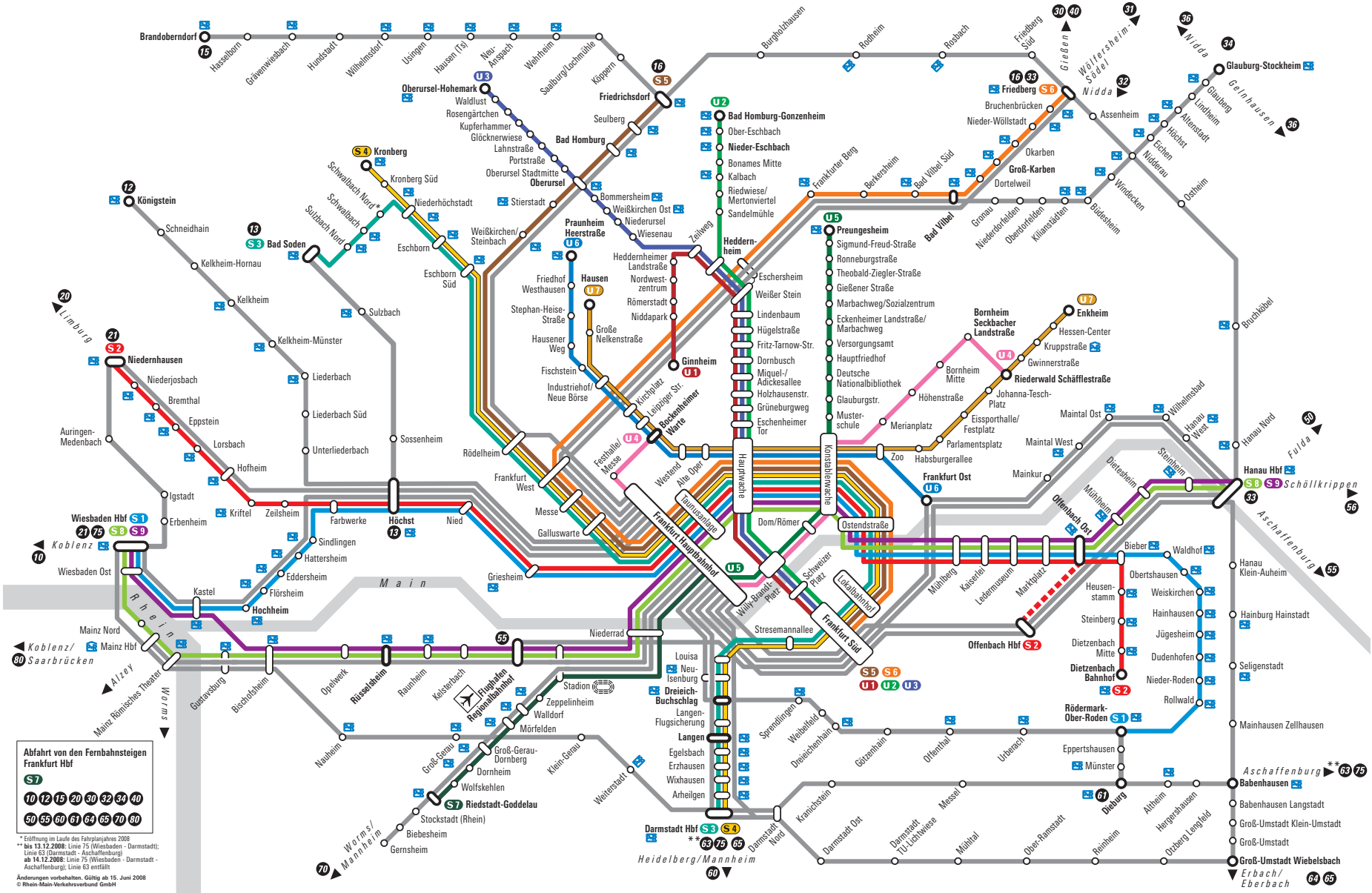
Beispiel: Beweisbäume für *Ans*(“Frankfurt Süd”) bzgl. dem Frankfurter U-Bahn-Netz und dem folgenden Datalog-Programm:

$$\begin{aligned} E(L,S,Z) &\leftarrow \mathit{U\text{-}Bahn\text{-}Netz}(L,S,Z) \\ E(L,S,Z) &\leftarrow E(L,S,Y),\ \mathit{U\text{-}Bahn\text{-}Netz}(L,Y,Z) \\ \mathit{Ans}(Z) &\leftarrow E(L, \text{“Hauptwache”}, Z) \end{aligned}$$

(siehe Tafel)

Rhein-Main-Verkehrsverbund

Schnellbahnplan



Fixpunkt-Semantik vs. Beweisbasierte Semantik

Satz 7.9

Für jedes Datalog-Programm P, alle $\mathbf{I} \in inst(edb(P))$ und alle Fakten $R(t)$ mit $R \in sch(P)$ und $t \in \mathbf{dom}^{arity(R)}$ gilt:

$$t \in \big([\![P]\!](\mathbf{I})\big)(R) \iff \text{es gibt einen Beweisbaum für } R(t) \text{ bzgl. } \mathbf{I} \text{ und } P.$$

Beweis: Übung.

Anfragesprachen mit Rekursion — Datalog

Erfüllbarkeit

Theorem 7.10

Das

ERFÜLLBARKEITSPROBLEM FÜR DATALOG-ANFRAGEN

Eingabe: Datalog-Anfrage $Q = (P, R)$

Frage: Gibt es ein $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$ so dass $[\![Q]\!](\mathbf{I}) \neq \emptyset$?

ist entscheidbar.

Beweis: Übung (siehe [AHV]-Buch, Kapitel 12.5)

Query Containment

Zwei Varianten:

(1) Uniformes Containment:

UNIFORMES CONTAINMENT PROBLEM FÜR DATALOG-PROGRAMME

Eingabe: Zwei Datalog-Programme P_1 und P_2 mit $edb(P_1) = edb(P_2)$ und $idb(P_1) = idb(P_2)$

Frage: Gilt $[\![P_1]\!](\mathbf{I}) \subseteq [\![P_2]\!](\mathbf{I})$ für alle $\mathbf{I} \in inst(edb(P_1))$?

(2) Herkömmliches Query Containment:

QUERY CONTAINMENT PROBLEM FÜR DATALOG-ANFRAGEN

Eingabe: Zwei Datalog-Anfragen (P_1, R) und (P_2, R) mit $edb(P_1) = edb(P_2)$ und $R \in idb(P_1) \cap idb(P_2)$

Frage: Gilt $[\![(P_1, R)]\!](\mathbf{I}) \subseteq [\![(P_2, R)]\!](\mathbf{I})$ für alle $\mathbf{I} \in inst(edb(P_1))$?

Theorem 7.11 *(hier ohne Beweis)*

(a) *Das uniforme Containment Problem für Datalog-Programme ist entscheidbar.*

(b) *Das Query Containment Problem für Datalog-Anfragen ist unentscheidbar.*

Beschränktheit (Boundedness)

- Wir haben gesehen: $[\![P]\!](\mathbf{I})$ kann berechnet werden, indem man nach und nach

$$\hat{\mathbf{I}},\ T_P(\hat{\mathbf{I}}),\ T_P(T_P(\hat{\mathbf{I}})),\ \ldots,\ T_P^{\textit{Ab-Stufe}(P,\mathbf{I})}(\hat{\mathbf{I}}),\ T_P^{\textit{Ab-Stufe}(P,\mathbf{I})+1}(\hat{\mathbf{I}})$$

  berechnet.
- Anzahl der Iterationen, die dafür nötig sind: $\textit{Ab-Stufe}(P,\mathbf{I}) + 1$
- Frage: Wie groß kann $\textit{Ab-Stufe}(P,\mathbf{I})$ werden?
- Klar: – in jeder Iteration kommt mindestens ein Faktum dazu;
  – $\textit{adom}([\![P]\!](\mathbf{I})) \subseteq \textit{adom}(P,\mathbf{I})$
- Daher: $\textit{Ab-Stufe}(P,\mathbf{I}) \leqslant \sum_{R \in \textit{idb}(P)} |\textit{adom}(P,\mathbf{I})|^{\textit{arity}(R)}$
- Besonders "schön" sind solche Datalog-Programme P, bei denen $\textit{Ab-Stufe}(P,\mathbf{I})$ gar nicht von $\mathbf{I}$ abhängt. Solche Programme heißen beschränkt (engl.: bounded).
- Präzise: Ein Datalog-Programm P heißt beschränkt, falls eine Zahl $d \in \mathbb{N}$ gibt, so dass für alle $\mathbf{I} \in \textit{inst}(\textit{edb}(P))$ gilt: $\textit{Ab-Stufe}(P,\mathbf{I}) \leqslant d$.

  Den kleinsten solchen Wert d nennen maximale Rekursionstiefe von P.
- Wenn man weiß, dass ein Programm P bschränkt ist und maximale Rekursionstiefe d hat, so kann man leicht eine zu P äquivalente SPCU-Anfrage konstruieren.

Beispiel

Das Datalog-Programm

$$\begin{aligned} E(L,S,Z) &\leftarrow U\text{-}Bahn\text{-}Netz(L,S,Z) \\ E(L,S,Z) &\leftarrow E(L,S,Y),\ U\text{-}Bahn\text{-}Netz(L,Y,Z) \\ Ans(Z) &\leftarrow E(L,\text{“Bockenheimer Warte”},Z) \end{aligned}$$

ist nicht beschränkt.

Das Datalog-Programm

$$\begin{aligned} Kauft(x,y) &\leftarrow Mag(x,y) \\ Kauft(x,y) &\leftarrow Person(x),\ Trendsetter(z),\ Kauft(z,y) \end{aligned}$$

ist beschränkt mit maximaler Rekursionstiefe 2 und äquivalent zum nicht-rekursiven Programm

$$\begin{aligned} Kauft(x,y) &\leftarrow Mag(x,y) \\ Kauft(x,y) &\leftarrow Person(x),\ Trendsetter(z),\ Mag(z,y) \end{aligned}$$

bzw. zur SPCU-Anfrage $Mag \cup \Big(Person \times \pi_3\big(\sigma_{1=1}(Trendsetter \times Mag)\big)\Big)$

Beschränktheit (Boundedness)

Theorem 7.12 *(hier ohne Beweis)*

Das

BOUNDEDNESS PROBLEM FÜR DATALOG-PROGRAMME

Eingabe: Datalog-Programm P

Frage: Ist P beschränkt, d.h. gibt es ein $d \in \mathbb{N}$ so dass $\mathit{Ab\text{-}Stufe}(P, \mathbf{I}) \leqslant d$ für alle $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$?

ist unentscheidbar.

Anfragesprachen mit Rekursion — Datalog

Nicht-Rekursives Datalog — Beispiel

Beispiel 7.13

(a) Das Datalog-Programm

$$\begin{aligned} \mathit{Kauft}(x,y) &\leftarrow \mathit{Mag}(x,y) \\ \mathit{Kauft}(x,y) &\leftarrow \mathit{Person}(x),\ \mathit{Trendsetter}(z),\ \mathit{Mag}(z,y) \end{aligned}$$

ist nicht-rekursiv.

(b) Äquivalent dazu, aber NICHT nicht-rekursiv ist

$$\begin{aligned} \mathit{Kauft}(x,y) &\leftarrow \mathit{Mag}(x,y) \\ \mathit{Kauft}(x,y) &\leftarrow \mathit{Person}(x),\ \mathit{Trendsetter}(z),\ \mathit{Kauft}(z,y) \end{aligned}$$

Nicht-Rekursives Datalog — Präzise

Definition 7.14

Sei P ein Datalog-Programm.

(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 aller nicht-rekursiven Datalog-Programme besteht aus allen Datalog-Programmen P, deren Abhängigkeitsgraph G_P azyklisch ist.

Beispiele für Abhängigkeitsgraphen: *siehe Tafel*

“Nicht-rekursiv” vs. “beschränkt” (“bounded”)

Proposition 7.15

(a) *Jedes nr-Datalog-Programm ist beschränkt.*

(b) *Jedes beschränkte Datalog-Programm ist äquivalent zu einem nr-Datalog-Programm.*

Beweis: Einfache Übung.

Ausdrucksstärke von nr-Datalog

Satz 7.16

Die folgenden Anfragesprachen können genau dieselben Anfragefunktionen beschreiben:

(a) *SPCU bzw. SPJRU*

(b) *nr-Datalog-Anfragen*

(c) *bereichsunabhängiger positiver existentieller Kalkül PE-CALC$_{di}$*

Bemerkung: Die Übersetzung von nr-Datalog 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: (c) $\Rightarrow$ (a): Bringe die gegebene PE-CALC$_{di}$-Anfrage zunächst in Normalform, also als Diskunktion von bereichsunabhängigen Anfragen des konjunktiven Kalküls.
Aus Kapitel 2 wissen wir, dass jede Anfrage des konj. Kalküls äquivalent zu einer SPC-Anfrage ist. Disjunktionen von Anfragen des konj. Kalküls sind dann äquivalent zu einer SPCU-Anfrage.

(a) $\Rightarrow$ (b): Einfache Induktion nach dem Aufbau von SPCU-Anfragen.

(b) $\Rightarrow$ (c): siehe Tafel.

Datalog mit Negation

Ziel: Auch Negationszeichen "¬" in Datalog-Regeln zulassen.

Definition 7.17

(a) Ein Literal ist ein Relationsatom $R(u)$ oder ein negiertes Relationsatom $\neg R(u)$. Ein Literal der Form $R(u)$ heißt positiv; ein Literal der Form $\neg R(u)$ heißt negativ bzw. negiert.

(b) Eine Datalog$^\neg$-Regel ist ein Ausdruck der Form

$$R_0(u_0) \leftarrow L_1(u_1), \ldots, L_\ell(u_\ell)$$

wobei $\ell \geqslant 0$, $R_0 \in$ **relname**, u_0 ein freies Tupel der Stelligkeit $arity(R_0)$ und $L_1(u_1), \ldots, L_\ell(u_\ell)$ Literale, so dass jede Variable, die in u_0 vorkommt, auch in mindestens einem positiven Literal $L_i(u_i)$ vorkommt.

(c) Ein Datalog$^\neg$-Programm P ist eine endliche Menge von Datalog$^\neg$-Regeln.

(d) Eine Datalog$^\neg$-Anfrage (P, R) besteht aus einem Datalog$^\neg$-Programm P und einem Relations-Namen R, der in P vorkommt.

Frage: Was soll die Semantik von Datalog¬ sein?

Beispiel 7.18

Anfrage:

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

Als Datalog¬-Anfrage:

$$
\begin{aligned}
E(L,S,Z) &\leftarrow \textit{U-Bahn-Netz}(L,S,Z)\\
E(L,S,Z) &\leftarrow E(L,S,Y),\ \textit{U-Bahn-Netz}(L,Y,Z)\\
\textit{Erreichbar\_BW}(Z) &\leftarrow E(L,\text{"Bockenheimer Warte"},Z)\\
\textit{Station}(S) &\leftarrow \textit{U-Bahn-Netz}(L,S,Z)\\
\textit{Station}(Z) &\leftarrow \textit{U-Bahn-Netz}(L,S,Z)\\
\textit{Ans}(Z) &\leftarrow \textit{Station}(Z),\ \neg\textit{Erreichbar\_BW}(Z)
\end{aligned}
$$

Hier: Semantik intuitiv klar.

Probleme mit der Semantik von Datalog$^\neg$

Beispiel 7.19

(a) $R(x) \leftarrow A(x),\ \neg R(x)$

"Ausgewertet" über einer DB $\mathbf{I} \in \mathit{inst}(\{A\})$ gilt:

$$T_P^i(\hat{\mathbf{I}})(R) = \begin{cases} \emptyset & \text{falls } i \text{ gerade} \\ \mathit{adom}(\mathbf{I}) & \text{falls } i \text{ ungerade} \end{cases}$$

Somit: Die Folge $\left(T_P^i(\hat{\mathbf{I}})\right)_{i \geqslant 0}$ hat keinen Fixpunkt.

Außerdem: $T_P(\cdot)$ hat überhaupt keinen Fixpunkt.

(b) $R(x) \leftarrow A(x),\ \neg S(x)$
$S(x) \leftarrow A(x),\ \neg R(x)$

Hier hat $T_P(\cdot)$ zwei verschiedene minimale Fixpunkte (minimal bzgl. $\subseteq$):

- FP_1 mit $\mathrm{FP}_1(R) = \emptyset$ und $\mathrm{FP}_1(S) = \mathit{adom}(\mathbf{I})$
- FP_2 mit $\mathrm{FP}_2(R) = \mathit{adom}(\mathbf{I})$ und $\mathrm{FP}_2(S) = \emptyset$

Inflationäre Fixpunkte

Um eine Semantik für Datalog$^\neg$ festzulegen, könnte man einfach per Definition erzwingen, dass die einzelnen Stufen des Fixpunktprozesses ineinander enthalten sind:

An Stelle von $T_P(\cdot)$ betrachte den Operator $\Gamma_P(\cdot)$ mit

$$\Gamma_P(\mathbf{J}) \;:=\; \mathbf{J} \cup T_P(\mathbf{J})$$

Dann gilt:

$$\hat{\mathbf{I}} \subseteq \Gamma_P(\hat{\mathbf{I}}) \subseteq \Gamma_P(\Gamma_P(\hat{\mathbf{I}})) \subseteq \cdots \subseteq \Gamma_P^i(\hat{\mathbf{I}}) \subseteq \Gamma_P^{i+1}(\hat{\mathbf{I}}) \subseteq \cdots$$

Da $adom(P, \mathbf{I})$ endlich ist, wird irgendwann ein (eindeutig definierter) Fixpunkt von $\Gamma_P(\cdot)$ erreicht. Dieser heißt inflationärer Fixpunkt von P auf $\mathbf{I}$.

Problem mit der inflationären Fixpunkt-Semantik von Datalog¬

Diese Semantik ist für viele Programme unnatürlich.

Beispiel 7.20

$$
\begin{aligned}
E(L,S,Z) &\leftarrow \textit{U-Bahn-Netz}(L,S,Z)\\
E(L,S,Z) &\leftarrow E(L,S,Y),\ \textit{U-Bahn-Netz}(L,Y,Z)\\
\textit{Erreichbar\_BW}(Z) &\leftarrow E(L,\text{“Bockenheimer Warte”},Z)\\
\textit{Station}(S) &\leftarrow \textit{U-Bahn-Netz}(L,S,Z)\\
\textit{Station}(Z) &\leftarrow \textit{U-Bahn-Netz}(L,S,Z)\\
\textit{Ans}(Z) &\leftarrow \textit{Station}(Z),\ \neg\textit{Erreichbar\_BW}(Z)
\end{aligned}
$$

“Intuitive Semantik”:

Alle Stationen, die von “Bockenheimer Warte” aus nicht ohne Umsteigen zu erreichen sind.

Inflationäre Fixpunkt-Semantik: Alle Stationen.

Frage: Was soll die Semantik von Datalog$^\neg$ sein?

Probleme:

- Inflationäre Fixpunkt-Semantik: unnatürlich (siehe Beispiel 7.20)
- Fixpunkt-Semantik via $T_P^\omega(\hat{\mathbf{I}})$:
  für manche Datalog$^\neg$-Programme undefiniert (siehe Beispiel 7.19)
- Semantik via "kleinster Fixpunkt" von $T_P(\cdot)$:
  ist für manche Datalog$^\neg$-Programme nicht eindeutig (siehe Beispiel 7.19)
- Beweisbasierte Semantik für Datalog$^\neg$: ???

Aber für Programme wie in Beispiel 7.20 ist die Semantik "intuitiv klar".

$\rightsquigarrow$ Betrachte im Folgenden nur Datalog$^\neg$-Programme von eingeschränkter Form:

- Semipositives Datalog$^\neg$
- nr-Datalog$^\neg$
- Stratifiziertes Datalog$^\neg$

Semipositives Datalog$^\neg$

Negation ist nur bei edb-Prädikaten erlaubt.

Man kann leicht zeigen, dass für jedes semipositive Datalog$^\neg$-Programm P und alle $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$ gilt:

- $T_P(\cdot)$ hat einen eindeutig bestimmten kleinsten Fixpunkt $\mathbf{J}$ mit $\mathbf{J}|_{\mathit{edb}(P)} = \mathbf{I}$.
- Dieser wird von der Sequenz

$$\hat{\mathbf{I}}, \quad T_P(\hat{\mathbf{I}}), \quad T_P^2(\hat{\mathbf{I}}), \quad T_P^3(\hat{\mathbf{I}}), \quad \ldots$$

  erreicht. Notation für diesen Fixpunkt: $T_P^\omega(\hat{\mathbf{I}})$.

Definition der Semantik von semipositiven Datalog$^\neg$-Programmen P:
Für alle $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$ setze

$$[\![P]\!](\mathbf{I}) \;:=\; T_P^\omega(\hat{\mathbf{I}})$$

Stratifiziertes Datalog$^\neg$ — Beispiel

$$
\begin{aligned}
E(L,S,Z) &\leftarrow U\text{-}Bahn\text{-}Netz(L,S,Z)\\
E(L,S,Z) &\leftarrow E(L,S,Y),\ U\text{-}Bahn\text{-}Netz(L,Y,Z)\\
Erreichbar\_BW(Z) &\leftarrow E(L,\text{“Bockenheimer Warte”},Z)\\
Station(S) &\leftarrow U\text{-}Bahn\text{-}Netz(L,S,Z)\\
Station(Z) &\leftarrow U\text{-}Bahn\text{-}Netz(L,S,Z)\\
Ans(Z) &\leftarrow Station(Z),\ \neg Erreichbar\_BW(Z)
\end{aligned}
$$

- Die Negation ist hier nicht mit der Rekursion verschränkt.
- Sie kann angewendet werden, nachdem $Erreichbar\_BW(\cdot)$ vollständig berechnet ist.
- $\rightsquigarrow$ Grundidee für stratifiziertes Datalog$^\neg$.

Stratifiziertes Datalog$^\neg$ — Präzise

Definition 7.21

Sei P ein Datalog$^\neg$-Programm.
Eine Stratifizierung von P ist eine Folge $P^1, \ldots, P^m$ von Datalog$^\neg$-Programmen, so dass $m \geqslant 1$ ist und es eine Abbildung $\sigma : idb(P) \to \{1, \ldots, m\}$ gibt, so dass gilt:

(1) $P^1, \ldots, P^m$ ist eine Partition von P (d.h. $P = P^1 \dot\cup \cdots \dot\cup P^m$),

(2) für jedes idb-Prädikat R von P gilt: alle Regeln, in deren Kopf R vorkommt, gehören zu $P^{\sigma(R)}$,

(3) kommt ein $S \in idb(P)$ im Rumpf einer Regel mit Kopf R vor, so ist $\sigma(S) \leqslant \sigma(R)$, und

(4) kommt ein $S \in idb(P)$ negiert im Rumpf einer Regel mit Kopf R vor, so ist $\sigma(S) < \sigma(R)$.

Notation:

- P^i heißt i-tes Stratum bzw. i-te Schicht der Stratifizierung $P^1, \ldots, P^m$
  ("Stratum": lat. für "Schicht"; Plural: Strata)
- σ heißt Stratifizierungs-Abbildung
- Ein Datalog$^\neg$-Programm P heißt stratifizierbar, falls es eine Stratifizierung von P gibt. Stratifiziertes Datalog$^\neg$ bezeichnet die Menge aller stratifizierbaren Datalog$^\neg$-Programme.

Stratifiziertes Datalog$^\neg$ — Beispiele

$P^1 :=$

$$
\begin{array}{rcl}
E(L,S,Z) & \leftarrow & BVG(L,S,Z) \\
E(L,S,Z) & \leftarrow & E(L,S,Y),\ U\text{-}Bahn\text{-}Netz(L,Y,Z) \\
Erreichbar\_BW(Z) & \leftarrow & E(L,\text{“Bockenheimer Warte”},Z) \\
Station(S) & \leftarrow & U\text{-}Bahn\text{-}Netz(L,S,Z) \\
Station(Z) & \leftarrow & U\text{-}Bahn\text{-}Netz(L,S,Z)
\end{array}
$$

$P^2 :=$

$$
Ans(Z) \leftarrow Station(Z),\ \neg Erreichbar\_BW(Z)
$$

ist eine Stratifizierung des Datalog$^\neg$-Programms aus Beispiel 7.20.

Das Datalog$^\neg$-Programm $R(x) \leftarrow A(x),\ \neg R(x)$ ist nicht stratifizierbar.

Das Datalog$^\neg$-Programm

$$
\begin{array}{rcl}
R(x) & \leftarrow & A(x),\ \neg S(x) \\
S(x) & \leftarrow & A(x),\ \neg R(x)
\end{array}
$$

auch nicht.

Test auf Stratifizierbarkeit

Abhängigkeitsgraph für Datalog$^\neg$-Programme

Definition 7.22

Sei P ein Datalog$^\neg$-Programm. Der Abhängigkeitsgraph $G_P = (V_P, E_P^+, E_P^-)$ ist der gerichtete Graph mit

- Knotenmenge $V_P := sch(P)$
- Kantenmengen

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

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

Beispiel: *Siehe Tafel:* Abhängigkeitsgraph für

$$\begin{array}{rcl} E(L,S,Z) & \leftarrow & \textit{U-Bahn-Netz}(L,S,Z) \\ E(L,S,Z) & \leftarrow & E(L,S,Y),\ \textit{U-Bahn-Netz}(L,Y,Z) \\ \textit{Station}(S) & \leftarrow & \textit{U-Bahn-Netz}(L,S,Z) \\ \textit{Station}(Z) & \leftarrow & \textit{U-Bahn-Netz}(L,S,Z) \\ \textit{Erreichbar\_BW}(Z) & \leftarrow & E(L, \text{“Bockenheimer Warte”}, Z) \\ \textit{Ans}(Z) & \leftarrow & \textit{Station}(Z),\ \neg \textit{Erreichbar\_BW}(Z) \end{array}$$

Test auf Stratifizierbarkeit

Proposition 7.23

Für jedes Datalog$^\neg$-Programm P gilt:

$$P \text{ ist stratifizierbar} \iff \begin{array}{l}\text{Im Abhängigkeitsgraph } G_P \text{ gibt es keinen}\\ \text{Kreis, in dem eine Kante aus } E_P^- \text{ vorkommt.}\end{array}$$

Beweis:

"$\Longrightarrow$": Sei σ eine Stratifizierungs-Abbildung von P.
Angenommen, es gibt einen Kreis von R nach R, auf dem mindestens eine Kante aus E_P^- vorkommt.
Dann gilt: $\sigma(R) > \sigma(R)$. Widerspruch!

"$\Longleftarrow$": Idee: Nutze eine topologische Sortierung der starken Zusammenhangskomponenten von $(E_P^+ \cup E_P^-)$, um die einzelnen Schichten zu definieren.

Details: *Übung*.

Stratifiziertes Datalog$^\neg$ — Semantik

- Sei $P^1, \ldots, P^m$ eine Stratifizierung eines Datalog$^\neg$-Programms P.
- Betrachte jede Schicht P^i als ein semipositives Datalog$^\neg$-Programm mit $edb(P^i) \subseteq edb(P) \cup idb(P^1) \cup \cdots \cup idb(P^{i-1})$
- Sei $\mathbf{I} \in edb(P)$.

  Die Semantik $[\![P]\!](\mathbf{I})$ von P auf $\mathbf{I}$ ist folgendermaßen definiert: $[\![P]\!](\mathbf{I}) := \mathbf{I}^m$, wobei

$$\begin{array}{lcl} \mathbf{I}^1 & := & [\![P^1]\!](\mathbf{I}) \\ \mathbf{I}^2 & := & [\![P^2]\!](\mathbf{I}^1) \\ \vdots & \vdots & \vdots \\ \mathbf{I}^m & := & [\![P^m]\!](\mathbf{I}^{m-1}) \end{array}$$

Man kann leicht zeigen, dass folgendes gilt:

(a) Obige Definition hängt nicht von der konkreten Wahl der Stratifizierung von P ab.
D.h. für je zwei verschiedene Stratifizierungen $P^1, \ldots, P^m$ und $Q^1, \ldots, Q^n$ von P gilt:
$[\![P^m]\!](\mathbf{I}^{m-1}) = [\![Q^n]\!](\mathbf{I}^{n-1})$.

(b) $[\![P]\!](\mathbf{I})$ ist der (eindeutig definierte) kleinste Fixpunkt $\mathbf{J}$ von $T_P(\cdot)$ mit $\mathbf{J}|_{edb(P)} = \mathbf{I}$.

Spezialfall: nr-Datalog$^\neg$

$$\begin{aligned} \text{nr-Datalog}^\neg &= \text{nr-Datalog mit Negation} \\ &= \text{stratifiziertes Datalog}^\neg\text{, eingeschränkt auf Programme } P\text{,} \\ &\quad \text{in deren Abhängigkeitsgraph es keinen gerichteten Kreis} \\ &\quad \text{(über } (E_P^+ \cup E_P^-)\text{) gibt.} \end{aligned}$$

Satz 7.24

Die folgenden Anfragesprachen können genau dieselben Anfragefunktionen beschreiben:

(a) *Relationale Algebra*

(b) *Relationenkalkül (bereichsunabhängig bzw. safe-range)*

(c) *nr-Datalog$^\neg$-Anfragen.*

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

Beweis: Einfache Induktion nach dem Aufbau von nr-Datalog$^\neg$ bzw. nach dem Aufbau der relationalen Algebra.