
Datenbanken

Datalog

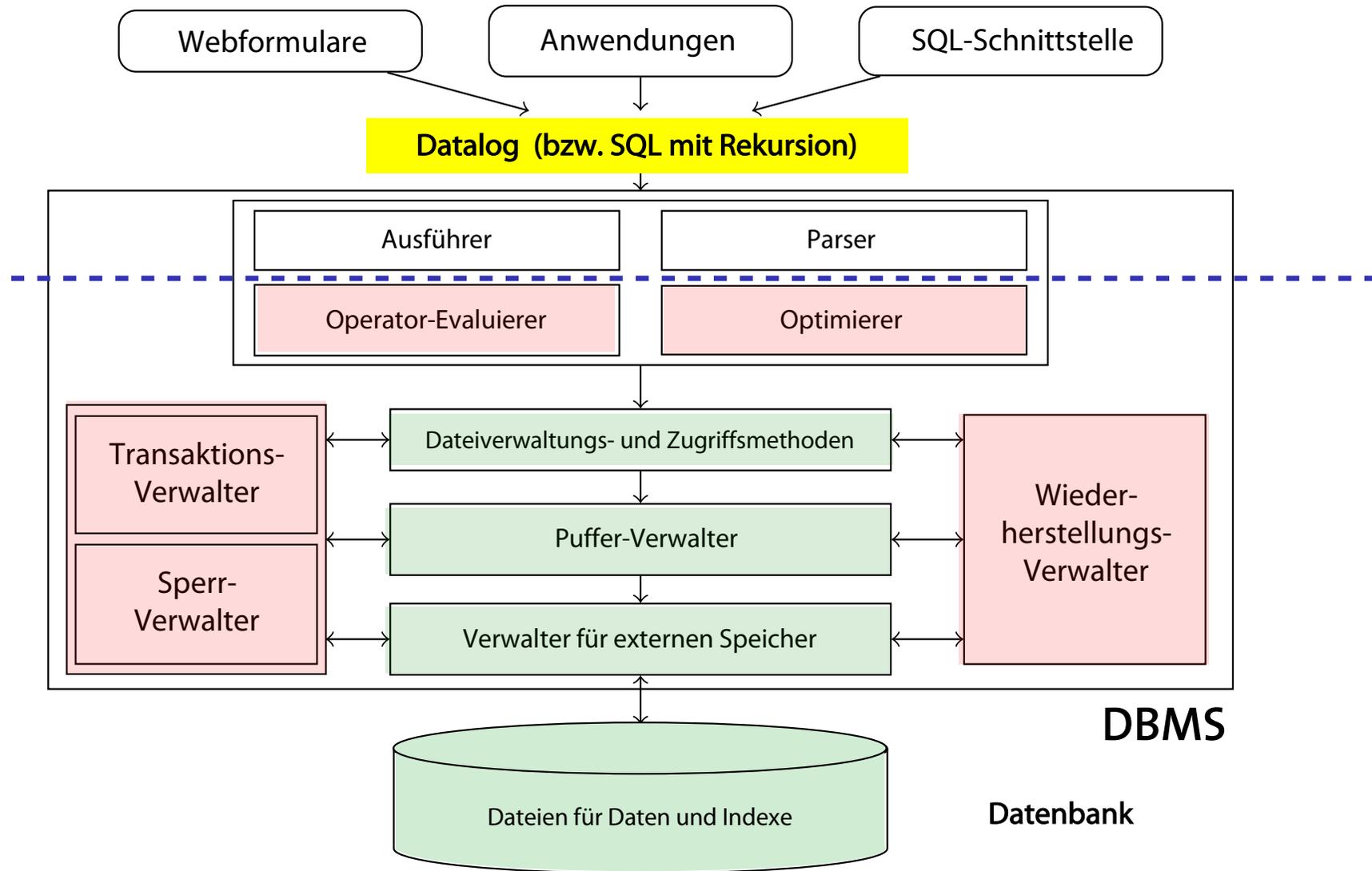
Dr. Özgür Özçep
Universität zu Lübeck
Institut für Informationssysteme



Einordnung



Architektur eines DBMS



Motivierendes Beispiel: transitive Hülle



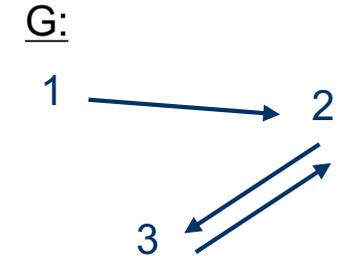
Datalog: Rekursive Anfragen

Einige Darstellungen wurden mit Änderungen übernommen aus einer Präsentation von Stephanie Scherzinger und von Ullman CS345

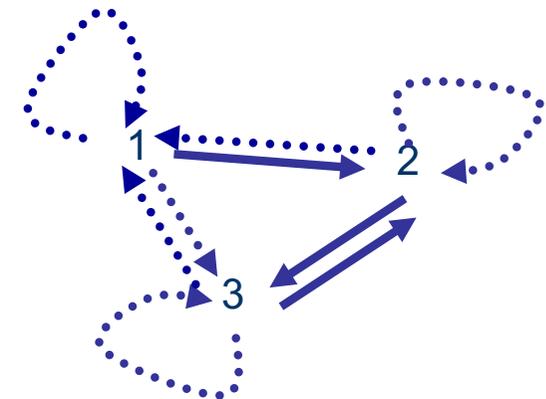
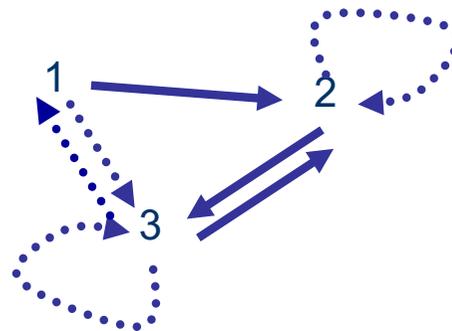
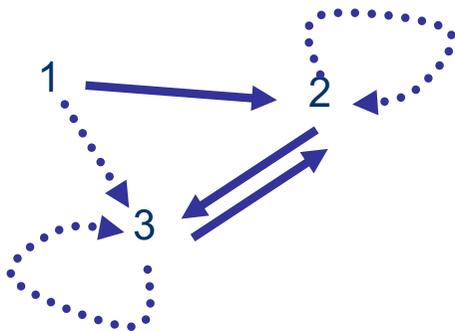
$$\forall x,y (T(x,y) \leftarrow G(x,y))$$

$$\forall x,y,z (T(x,y) \leftarrow (G(x,z) \wedge T(z,y)))$$

$$G(1, 2), G(2, 3), G(3, 2)$$



Mögliche Lösungen:



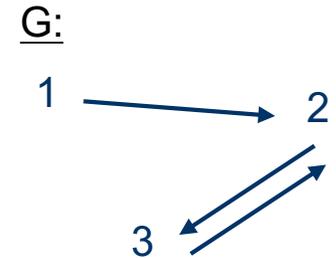
Herve Gallaire und Jack Minker *Logic and Data Bases*.
Symposium on Logic and Data Bases, Centre d'études et de
recherches de Toulouse in „Advances in Data Base Theory“.
Plenum Press, New York **1978**

Minimale-Modell-Semantik

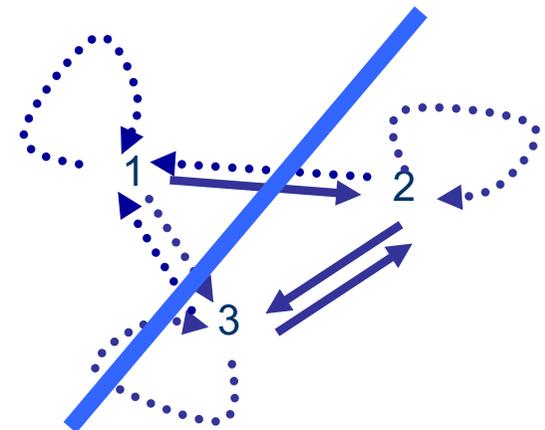
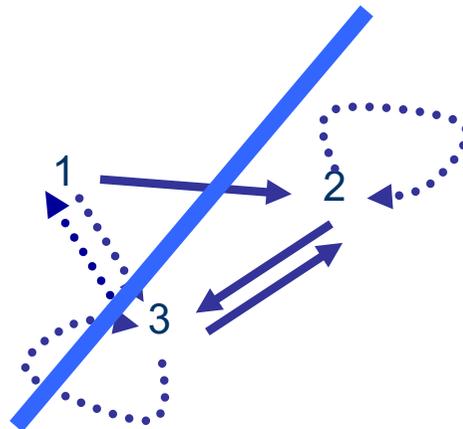
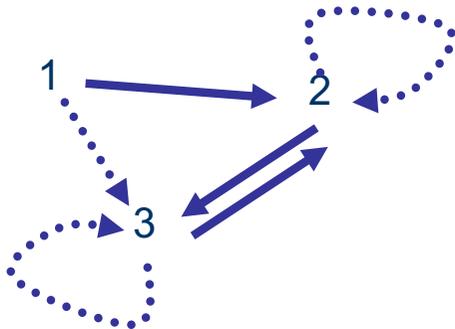
$$\forall x,y (T(x,y) \leftarrow G(x,y))$$

$$\forall x,y,z (T(x,y) \leftarrow (G(x,z) \wedge T(z,y)))$$

$$G(1,2), G(2,3), G(3,2)$$



Mögliche Lösungen:



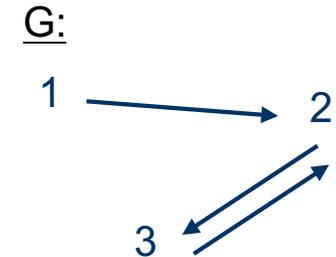
→ *Wähle minimales Modell* ←
T soll kleinste Menge von Fakten enthalten, so dass Regeln wahr sind

Minimale-Modell-Semantik

$$\forall x,y (T(x,y) \leftarrow G(x,y))$$

$$\forall x,y,z (T(x,y) \leftarrow (G(x,z) \wedge T(z,y)))$$

$$G(1,2), G(2,3), G(3,2)$$

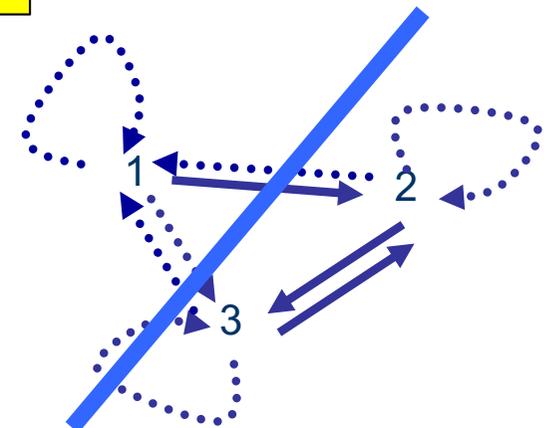
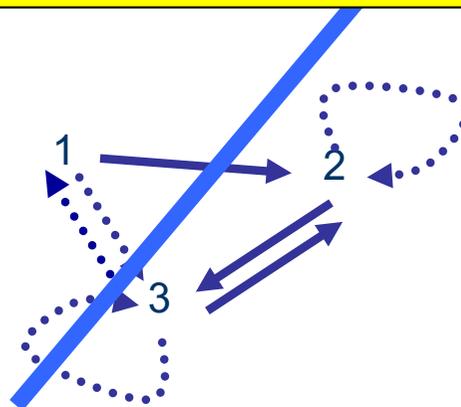
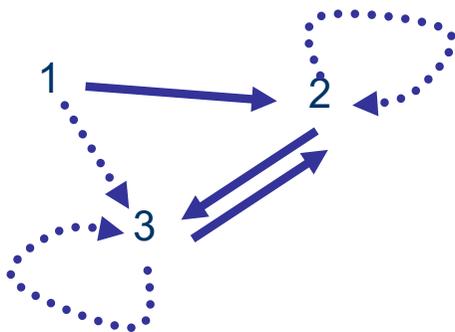


Terminologie:

G is ein EDB-Prädikat (extensionale DB)

T ist ein IDB-Prädikat (intensionale DB)

Mögliche Lösungen:



→ *Wähle minimales Modell* ←
T soll kleinste Menge von Fakten enthalten, so dass Regeln wahr sind

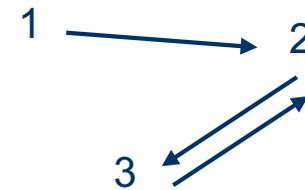
Erreichbarkeit in Graphen (T = transitive Hülle von G):

$$\mathbf{T(x, y) :- G(x, y)}$$

$$\mathbf{T(x, y) :- G(x, z), T(z, y)}$$

Intuition

Transitive Hülle:

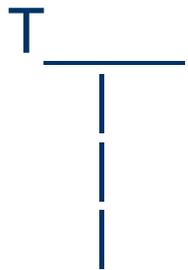


$$T(x, y) :- G(x, y)$$

$$T(x, y) :- G(x, z), T(z, y)$$

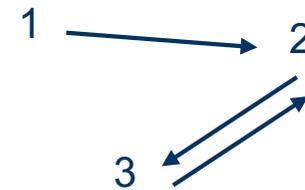
G

1		2
2		3
3		2



Intuition

Transitive Hülle:



$$T(x, y) :- G(1, 2)$$

$$T(x, y) :- G(x, z), T(z, y)$$

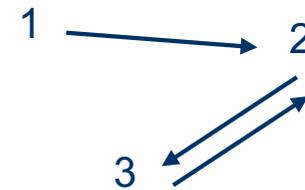
G

1	2
2	3
3	2

(1) Bilde ab von den DB-Relationen in den Regelrumpf

Intuition

Transitive Hülle:



$T(1, 2) \text{ :- } G(1, 2)$
 $T(x, y) \text{ :- } G(x, z), T(z, y)$

G

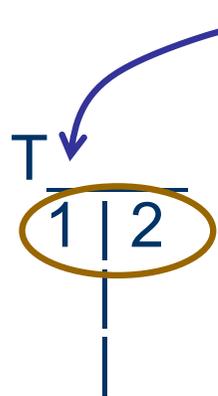
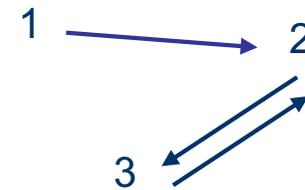
1	2
2	3
3	2

T

(2) Propagiere Bindungen auf Regelkopf

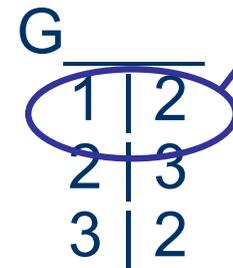
Intuition

Transitive Hülle:



$$T(1, 2) \text{ :- } G(1, 2)$$

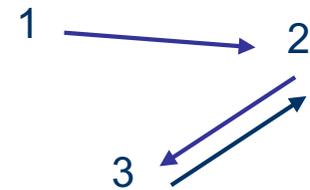
$$T(x, y) \text{ :- } G(x, z), T(z, y)$$



(2) Propagiere Bindungen auf Regelkopf

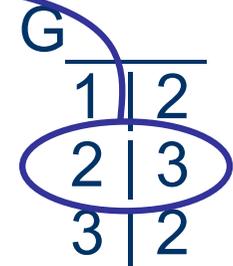
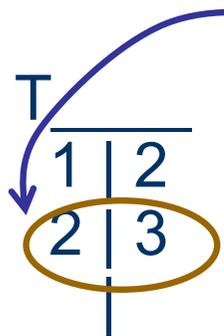
Intuition

Transitive Hülle:



$$T(x, y) :- G(x, y)$$

$$T(x, y) :- G(x, z), T(z, y)$$



Intuition

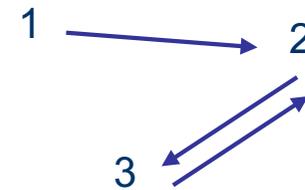
Transitive Hülle:

T

1	2
2	3
3	2

$$T(x, y) :- G(x, y)$$

$$T(x, y) :- G(x, z), T(z, y)$$

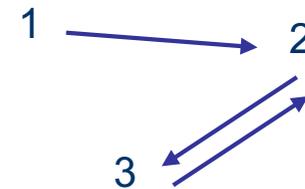


G

1	2
2	3
3	2

Intuition

Transitive Hülle:



$T(x, y) :- G(x, y)$

$T(x, y) :- G(1, 2), T(2, 3)$

T

1	2
2	3
3	2

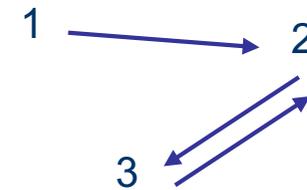
G

1	2
2	3
3	2

(1) Setze Tupel aus den Relationen in den Regelrumpf ein

Intuition

Transitive Hülle:



T	
1	2
2	3
3	2

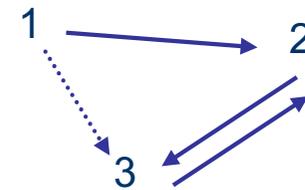
$T(x, y) \text{ :- } G(x, y)$
 $T(1, 3) \text{ :- } G(1, 2), T(2, 3)$

G	
1	2
2	3
3	2

(2) Übertrage Bindungen in den Regelkopf

Intuition

Transitive Hülle:



$$T(x, y) \text{ :- } G(x, y)$$

$$T(1, 3) \text{ :- } G(1, 2), T(2, 3)$$

T	
1	2
2	3
3	2
1	3

G	
1	2
2	3
3	2

(2) Übertrage Bindungen in den Regelkopf

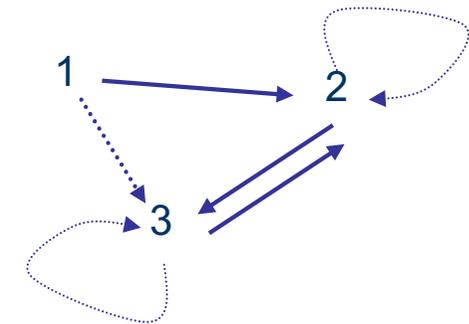
Intuition

Transitive Hülle:

T	
1	2
2	3
3	2
1	3
2	2
3	3

$$T(x, y) :- G(x, y)$$

$$T(x, y) :- G(x, z), T(z, y)$$



G	
1	2
2	3
3	2

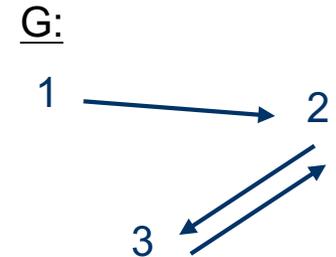
... Wiederhole bis Fixpunkt erreicht

Korrektheit

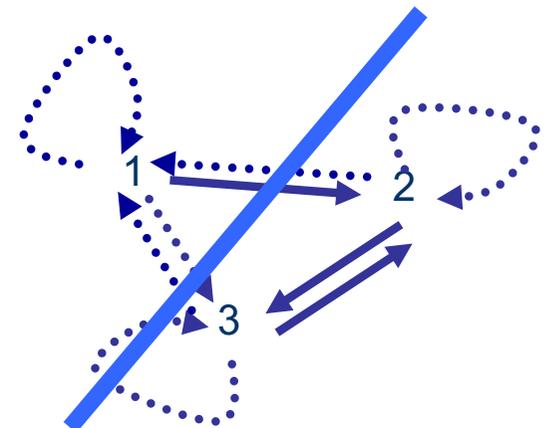
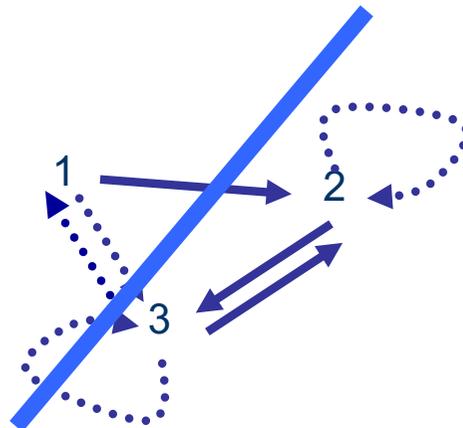
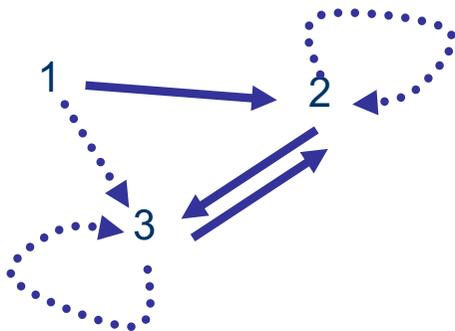
$$\forall x,y \ (T(x,y) \leftarrow G(x,y))$$

$$\forall x,y,z \ (T(x,y) \leftarrow (G(x,z) \wedge T(z,y)))$$

$$G(1,2), G(2,3), G(3, 2)$$



Mögliche Lösungen:



→ *Wähle minimales Modell* ←
T soll kleinste Menge von Fakten enthalten, so dass Regeln wahr sind

Weiteres Beispiel



Rekursive Anfragen in Datalog über einer DB

<i>Kurz</i>	<i>Name</i>	<i>Oberabt</i>
MFSW	Mainframe SW	LTSW
UXSW	Unix SW	LTSW
PCSW	PC SW	LTSW
LTSW	Leitung SW	NULL
PERS	Personal	NULL

Abteilungen

$\forall x,y (\text{Unterabteilung}(x,y) \leftarrow \exists w \text{ Abteilung}(x,w,y))$

$\forall x,y (\text{Unterabteilung}(x,y) \leftarrow \exists w \exists z(\text{Abteilung}(z,w,y) \wedge \text{Unterabteilung}(x,z)))$

Rekursive Anfragen in SQL

<i>Kurz</i>	<i>Name</i>	<i>Oberabt</i>
MFSW	Mainframe SW	LTSW
UXSW	Unix SW	LTSW
PCSW	PC SW	LTSW
LTSW	Leitung SW	NULL
PERS	Personal	NULL

Abteilungen

```
create recursive view Unterabteilungen
select r.kurz, r.oberabt
from Abteilungen r
union
select u.kurz, o.oberabt
from Abteilungen o,
     Unterabteilungen u
where o.kurz = u.oberabt
```

PostgreSQL-
Syntax

- Beim Start der Rekursion enthält „Unterabteilungen“ nur die Tupel der ersten Teilanfrage.
- Tupel, die sich durch den Join in der zweiten Teilanfrage ergeben, werden der Extension von „Unterabteilungen“ für die nächste Iteration hinzugefügt.
- Abbruch der Rekursion, sobald die zweite Teilanfrage bei Verwendung der Ergebnisse aus der vorigen Iteration keine zusätzlichen Ergebnistupel mehr liefert → Fixpunkt.

Basisterminologie zu Datalog



Pures Datalog: Terminologie

- Annahme: Extensionale Datenbank-Prädikate (EDB) gegeben
- Syntax: **Kopf :-Rumpf**
 - Gelesen von rechts nach links: Wenn **Rumpf**, dann **Kopf**
- Atom = Prädikat angewandt auf Argumente (= Variable oder Konstante)
(Bsp: $T(x,y)$ oder $G(a,x)$)
- **Kopf** (Head) ist ein Atom
- **Rumpf** (Body) ist Konjunktion von Atomen
 - Konjunktion als Komma notiert
- Variablen im Kopf = ausgezeichnete (distinguished)

Variablen



Aufgabe:

Wie sind Regeln mit leerem Rumpf zu lesen?

Aufgabe:

Wie sind Regeln mit leerem Rumpf zu lesen?

Lösung: Ein leerer Rumpf ist eine Konjunktion von 0 Atomen, das entspricht einer Tautologie (immer wahr). Also entspricht eine Regel mit leerem Rumpf einem Fakt.

$T(a,b) : -$

entspricht

$T(a,b)$

Pures Datalog: Terminologie (Forts.)

- Atome des Rumpfs auch Teilziele (subgoals) genannt
- Prädikate, die im Kopf erscheinen, heißen intensionale Datenbank Prädikate (IDB)
- Teilziele können EDB- oder IDB-Prädikate enthalten
- Datalog Programm = endliche Menge von Regeln.
 - Ein spezielles IDB-Prädikat (meist Q) wird als Anfrage-/Antwortprädikat verstanden, dessen Extension die Antwortmenge präsentiert.

$$T(x, y) \text{ :- } G(x, y)$$

$$T(x, y) \text{ :- } G(x, z), T(z, y)$$

$$Q(y) \text{ :- } T(1, y)$$

Sichere (safe) Anfragen

- Unendliche Antwortmengen sollen verhindert werden
- Kann durch syntaktische Einschränkungen an Variablen realisiert werden

- Unsichere Anfrage:

Verboten: `Numbered_edges (x, y, num) :- G(x, y)`

- Antwortmenge besteht aus Tripeln, bei denen das dritte Argument `num` nicht durch die Datenbank gebunden werden kann
- Ist der Typ von `num` unendlich (natürliche Zahlen), bekäme man unendliche Antwortmenge

→ Variablen im Kopf müssen im Rumpf vorkommen
(Wertebereichsbeschränktheit)

Sichere Anfragen

- Sicherheit wird durch syntaktische Einschränkungen garantiert, diese sind jedoch manchmal zu stark
- Semantisches Kriterium: Domänenunabhängigkeit (domain independence)
 - Antworten sollen nur von Datenbank und Anfrage (und nicht Wertebereichen (Typen)) abhängen
 - Problem: I.A. ist nicht entscheidbar, ob eine gegebene Anfrage domänenunabhängig ist, daher Kompromiss mit Syntaxeinschränkung für Variable

Aufgabe:

Für die Sicherheit einer Anfrage ist nicht gefordert, dass Variablen im Rumpf auch im Kopf vorkommen müssen. Warum nicht?
Welchen SQL-Konstrukten entsprechen solche Variablen?

Aufgabe:

Für die Sicherheit einer Anfrage ist nicht gefordert, dass Variablen im Rumpf auch im Kopf vorkommen müssen. Warum nicht? Welchen SQL-Konstrukten entsprechen solche Variablen?

Lösung:

- Variablen im Rumpf, die nicht auch im Kopf vorkommen, sind von einem Existenzquantor gebunden. (Das Reinziehen vom Allquantor solcher Variablen macht sie zu einem Existenzquantor: Siehe das w in Unterabteilung auf S. 21).
- Die dahinter steckende "harmlose" Operation ist die der Projektion: Diese Variablen entsprechen gerade den wegprojizierten Attributen.
- Wenn solche Variablen mehrfach im Rumpf auftreten, dann haben sie die Rolle von Verbundattributen, die wegprojiziert werden. ³¹ Ein Beispiel ist das z in der zweiten Regel für Unterabteilung auf S. 21)

Semantik von purem Datalog



Semantik von purem Datalog

- 3 verschiedene Ansätze
 - Modelltheoretisch
 - Fixpunkttheoretisch
 - Beweistheoretisch

Modelltheoretisch

- Regeln werden als logische Sätze verstanden, die die Menge möglicher Modelle einschränken
- Anfangsbeispiel zur transitiven Hülle
 - $Q = T = \text{Antwortprädikat}$
 - $\forall x,y (T(x,y) \leftarrow G(x,y))$
 - $\forall x,y,z (T(x,y) \leftarrow (G(x,z) \wedge T(z,y)))$
- Allerdings gibt es meistens mehrere Modelle
 - In Datalog betrachtet man das minimale Modell
 - (Minimalität bezieht sich auf Teilmengenbeziehung bzgl. der Extensionen der IDB-Prädikate)

Warum minimales Modell?

- Hängt auch mit „**Closed-World-Assumption**“ zusammen
- Betrachte diejenigen Fakten, die nicht zwingend als „wahr“ beweisbar sind, als falsch
- Warnung: Die Wahl des minimalen Modells ist nicht selbstverständlich.
 - Siehe z.B. die Debatte in der Semantik des (rekursiven definierten) Wahrheitsprädikats in der Philosophie (der Logik), insbesondere Fixpunktansatz von Saul Kripke

Der Fixpunktansatz

- Modelltheoretischer Ansatz ist nicht-konstruktiv, führt nicht zwingend zur Konstruktion von Lösungen
- Betrachte Semantik durch Auswahl eines Fixpunktes
 - (Siehe Graphbeispiel)
- Anfrage wird iteriert, bis Fixpunkt erreicht: $F(P) = P$ für IDB-Prädikat P bzw. Tupel von IDB-Prädikaten P
→ Bottom-up Evaluation.

Bottom-up Evaluation

- Starte mit den Fakten der Datenbankinstanz (EDB-Fakten)
- Verwende Regeln von rechts (Rumpf) nach links (Kopf), um neue Fakten abzuleiten
- Wiederhole, bis keine Fakten mehr abgeleitet werden können

→ Produziert alle (!) Fakten, die beweisbar sind.

Bottom-up: Naive Evaluation

Gegeben sind EDB-Fakten:

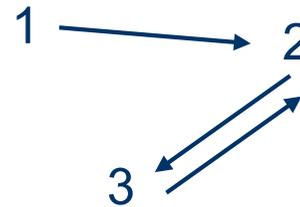
1. Initialisiere alle IDB-Prädikate jeweils mit leerer Menge
2. Instanziiere mit Konstanten alle Variablen in allen Regeln auf alle (konsistenten) Weisen. Falls alle Teilziele (des Rumpfes einer Regel) wahr werden, dann leite auch (instanziierten) Kopf der Regel ab.
3. Wiederhole (2) in Runden solange wie neue IDB-Fakten abgeleitet werden können

Bottom-up: Naive Evaluation

Gegeben sind EDB-Fakten:

1. Initialisiere alle IDB-Prädikate jeweils mit leerer Menge
2. Instanziiere mit Konstanten alle Variablen in allen Regeln auf alle (konsistenten) Weisen. Falls alle Teilziele (des Rumpfes einer Regel) wahr werden, dann leite auch (instanziierten) Kopf der Regel ab.
3. Wiederhole (2) in Runden solange wie neue IDB-Fakten abgeleitet werden können

G:



T:



$T(x, y) :- G(x, y)$

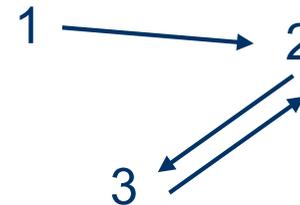
$T(x, y) :- G(x, z), T(z, y)$

Bottom-up: Naive Evaluation

Gegeben sind EDB-Fakten:

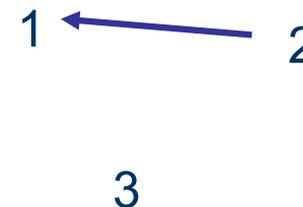
1. Initialisiere alle IDB-Prädikate jeweils mit leerer Menge
2. Instanziiere mit Konstanten alle Variablen in allen Regeln auf alle (konsistenten) Weisen. Falls alle Teilziele (des Rumpfes einer Regel) wahr werden, dann leite auch (instanziierten) Kopf der Regel ab.
3. Wiederhole (2) in Runden solange wie neue IDB-Fakten abgeleitet werden können

G:



T:

$G(2,1)$ ist falsch!



$T(x, y) :- G(2, 1)$

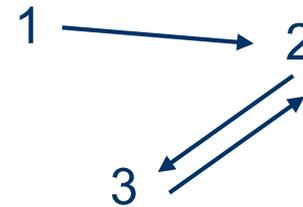
$T(x, y) :- G(x, z), T(z, y)$

Bottom-up: Naive Evaluation

Gegeben sind EDB-Fakten:

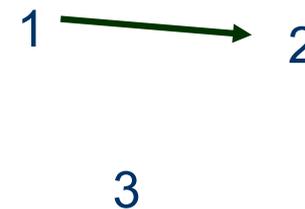
1. Initialisiere alle IDB-Prädikate jeweils mit leerer Menge
2. Instanziiere mit Konstanten alle Variablen in allen Regeln auf alle (konsistenten) Weisen. Falls alle Teilziele (des Rumpfes einer Regel) wahr werden, dann leite auch (instanziierten) Kopf der Regel ab.
3. Wiederhole (2) in Runden solange wie neue IDB-Fakten abgeleitet werden können

G:



T:

G(1,2) ist wahr!



$T(1, 2) :- G(1, 2)$

$T(x, y) :- G(x, z), T(z, y)$

Bottom-up: Naive Evaluation

Gegeben sind EDB-Fakten:

1. Initialisiere alle IDB-Prädikate jeweils mit leerer Menge
2. Instanziiere mit Konstanten alle Variablen in allen Regeln auf alle (konsistenten) Weisen. Falls alle Teilziele (des Rumpfes einer Regel) wahr werden, dann leite auch (instanziierten) Kopf der Regel ab.
3. Wiederhole (2) in Runden solange wie neue IDB-Fakten abgeleitet werden können

(2) Ist sinnvoll (gibt endlich viele Instanziiierungen), sofern Regeln sicher

Bottom-up: Naive Evaluation

Gegeben sind EDB-Fakten:

1. Initialisiere alle IDB-Prädikate jeweils mit leerer Menge
2. Instanziiere mit Konstanten alle Variablen in allen Regeln auf alle (konsistenten) Weisen. Falls alle Teilziele (des Rumpfes einer Regel) wahr werden, dann leite auch (instanziierten) Kopf der Regel ab.
3. Wiederhole (2) in Runden solange wie neue IDB-Fakten abgeleitet werden können

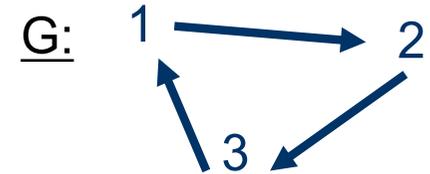
Algorithmus ergibt kleinsten (!) Fixpunkt der Regeln + EDB-Fakten.

Existenz garantiert wegen Monotonie: Extension der rekursiven IDB-Prädikate wächst! (Satz von Tarski-Knaster)

Bottom-up: Seminaive (inkrementell)

- Effizienterer Ansatz zur Evaluierung der Regeln
 - Naive Evaluation reproduziert bereits abgeleitete Fakten
- Idee: Falls in Runde i ein **neuer** Fakt abgeleitet wurde, dann muss eine Regel genutzt worden sein, in der mindestens ein Teilziel auf solche Fakten angewandt wurde, die in Runde $i-1$ abgeleitet wurden.
- Für jedes IDB Prädikat P führe Buch über
 - ΔP : die in der letzten Runde abgeleiteten P -Fakten
 - P : aktuelle Extension (da Ableitung auch auf „ältere“ Fakten zugreifen muss)

r1: $T(x, y) :- G(x, y)$
 r2: $T(x, y) :- G(x, z), T(z, y)$



1. Initialisiere IDB

$$T \begin{array}{|c} \hline \\ \hline 1 | 2 \\ \hline 2 | 3 \\ \hline 3 | 1 \\ \hline \end{array}$$

2. Init. Δ IDB

$$\Delta T \begin{array}{|c} \hline \\ \hline 1 | 2 \\ \hline 2 | 3 \\ \hline 3 | 1 \\ \hline \end{array}$$

3.a.i $\Delta T(x, y) := G(x, z), \Delta T(z, y)$

$$\Delta T \begin{array}{|c} \hline \\ \hline 1 | 3 \\ \hline 2 | 1 \\ \hline 3 | 2 \\ \hline \end{array}$$

3.a.ii $\Delta T := \Delta T \setminus T$

3.a.iii $T := T \cup \Delta T$

$$T \begin{array}{|c} \hline \\ \hline 1 | 2 \\ \hline 2 | 3 \\ \hline 3 | 1 \\ \hline 1 | 3 \\ \hline 2 | 1 \\ \hline 3 | 2 \\ \hline \end{array}$$

3.b.i $\Delta T(x, y) := G(x, z), \Delta T(z, y)$

$$\Delta T \begin{array}{|c} \hline \\ \hline 1 | 1 \\ \hline 2 | 2 \\ \hline 3 | 3 \\ \hline \end{array}$$

3.b.ii $\Delta T := \Delta T \setminus T$

3.b.iii $T := T \cup \Delta T$

$$T \begin{array}{|c} \hline \\ \hline 1 | 2 \\ \hline 2 | 3 \\ \hline 3 | 1 \\ \hline 1 | 3 \\ \hline 2 | 1 \\ \hline 3 | 2 \\ \hline 1 | 1 \\ \hline 2 | 2 \\ \hline 3 | 3 \\ \hline \end{array}$$

3.c

...
□

Bottom-up: Seminaive (inkrementell)

1. Initialisiere IDB Prädikate unter Nutzung derjenigen Regeln, die keine IDB Teilziele enthalten
2. Initialisiere die Δ -IDB Prädikate mit den korrespondierenden IDB-Prädikaten
3. In jeder Runde führe für jedes IDB-Prädikat P Folgendes aus:
 - Berechne neues ΔP durch Anwendung jeder Regel für P, indem ein Teilziel als Δ -IDB-Prädikat gesehen wird und die anderen als IDB- oder EDB-Prädikat. (Man muss ja eventuell auf ältere Fakten zurückgreifen)
Mache das für jede mögliche Wahl des Δ -Teilziels.
 - Entferne vom neuen ΔP alle Fakten, die schon in P sind.
 - $P := P \cup \Delta P$.
4. Wiederhole 3. bis keines der IDB-Prädikate sich ändert.

Semantik von reinem Datalog

- 3 verschiedene Ansätze
 - Modelltheoretisch
 - Fixpunkttheoretisch
 - **Beweistheoretisch**

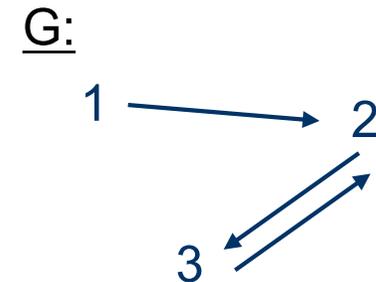
Beweistheoretisch

Ein Fakt ist im Resultat, wenn es einen Beweis (eine Ableitung) aus den Fakten der EDB und den Regeln gibt.

$$(1) \quad T(x, y) :- G(x, y)$$

$$(2) \quad T(x, y) :- G(x, z), T(z, y)$$

- i. $G(1,2)$ in EDB
- ii. $T(1,2)$ gemäß (i) und Regel (1)
- iii. $G(2,3)$ in EDB
- iv. $T(2,3)$ gemäß (iii) und Regel (1)
- v. $T(1,3)$ gemäß (i), (iv) und Regel(2)
- vi.

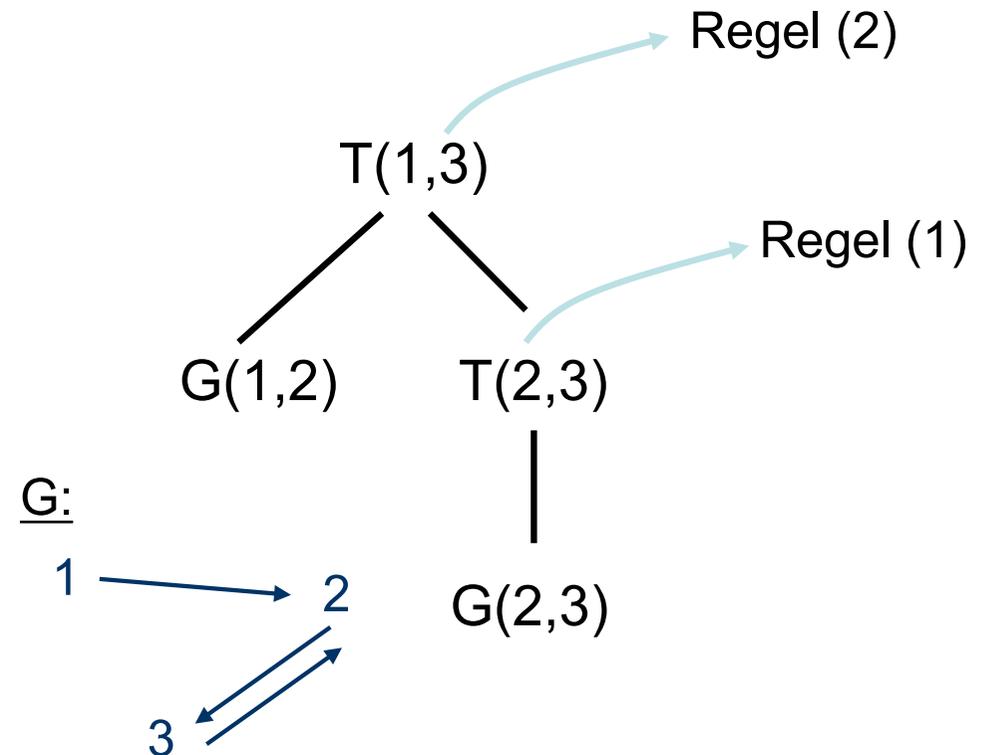


Beweisbäume

Beweisbaum für ein Faktum **A**
von EDB **I** und Datalog
Program **P**:

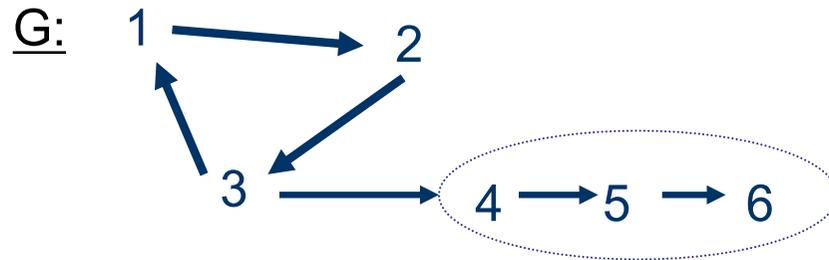
- Jeder Knoten ist ein Fakt
- Jedes Blatt ist Fakt aus **I**
- Wurzel ist **A**
- Vater-Kind-Kante gemäß einer Regel
- (Vater \leftarrow Kind₁, Kind₂, ..., Kind_n)

- (1) $T(x, y) :- G(x, y)$
(2) $T(x, y) :- G(x, z), T(z, y)$



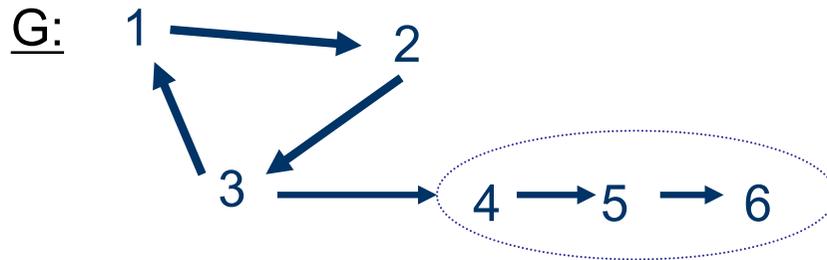
Top-down Evaluation

- Materialisiere nicht alle möglichen Fakten, sondern gehe von konkreter Zielfrage aus. Diese mag sehr viel konkreter sein, z.B. teilinstanziiertes $T(4,x)$ statt $T(x,y)$



Top-down Evaluation

- Materialisiere nicht alle möglichen Fakten, sondern gehe von konkreter Zielfrage aus. Diese mag sehr viel konkreter sein, z.B. teilinstanziiertes $T(4,x)$ statt $T(x,y)$



- Problem: unendliche Zyklen möglich bei Linksrekursion

$$T(x, y) :- G(x, y)$$

$$T(x, y) :- \underline{T(x, z)}, G(z, y)$$

Magic Sets (4. Ansatz: Top-Down+Bottom-up)

Der Magic-Set-Algorithmus überführt eine Datalog-Anfrage Q in eine neue Anfrage Q' mit folgenden Eigenschaften:

- Q' gibt dieselbe Menge an Antworten wie Q
 - Wenn Q' bottom-up ausgewertet wird, produziert es nur solche Fakten, die in einem Top-Down-Verfahren produziert würden.
- Selektionen (spezifische Instanzen) werden von der Anfrage in die Bottom-up-Berechnungen geschoben (mittels Magic-Prädikaten)

Genauere Beschreibung hier ausgelassen. Siehe z.B.:
Abiteboul, Hull, Vianu: Foundations of Databases,
Abschnitt 13.3.

Behandlung der Negation in Datalog



Datalog mit Negation

- Erweiterung von Datalog mit Negation möglich
 1. Definition einer „sicheren Regel“ ist zu erweitern
 2. Negation hierarchisch auswertbar gestalten (Stratifikation)
- Ad 1: Jede Variable eines negativen Atoms muss in einem (eventuell jeweils anderen) positiven Atom des Regelrumpfs vorkommen
- D.h., Negation nur als „Filter“
 - **Verboten:** $T(x, y) :- \neg H(x, y) ;$
 - **Erlaubt:** $T(x, y) :- \neg K(x, y, w), G(x, w), R(w, y)$

Ad 2: Stratifikation

- Negiertes Atom nicht in Schleife eines IDB-Prädikats
 - $P(\mathbf{x}) \text{ :- } Q(\mathbf{x}), \neg P(\mathbf{x})$ Verboten
 - $Q(1), Q(2)$
- Problem: Semantik von solchen IDB-Prädikaten P nicht als kleinster Fixpunkt berechenbar (da nicht existent)
- In obigen Beispiel „eiert“ die aktuelle Extension von P hin und her zwischen zwei Extension (keine Monotonie; kein Fixpunkt)
 - Initial: $P = \{\}$
 - Runde 1 : $P = \{1,2\}$
 - Runde 2: $P = \{\}$
 - Runde 3: $P = \{1,2\}$ etc.

Stratum

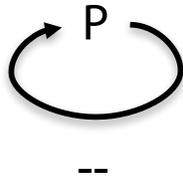
- Stratum eines Prädikats P = maximale Anzahl von Negationen, die man auf IDB-Prädikate anwenden muss, um P zu evaluieren.
- Stratifiziertes Datalog: alle Prädikate haben endliches Stratum
- In obigem Beispiel
 $P(\mathbf{x}) \quad :- \quad Q(\mathbf{x}), \neg P(\mathbf{x})$
würde man die Extension von P unendlich oft negieren

Stratum-Graph

- Knoten = IDB-Prädikate.
- Kante $A \rightarrow B$ falls Prädikat A abhängt von Prädikat B (B kommt in Rumpf eine Regel vor, deren Kopf A enthält).
- Kante ist mit “-” (für Negation) markiert, falls B negiertes Teilziel.
- Das **Stratum** eines Knotens (Prädikats) ist die maximale Zahl von “-“-Kanten auf einem Pfad von diesem Knoten.
- Ein Datalog-Programm ist **stratifiziert**, falls alle IDB-Prädikate endliche Strata haben, sprich im Stratum-graph kein Zykel mit Negation vorkommt.

Beispiel

$P(x) \leftarrow Q(x) , \text{ NOT } P(x)$



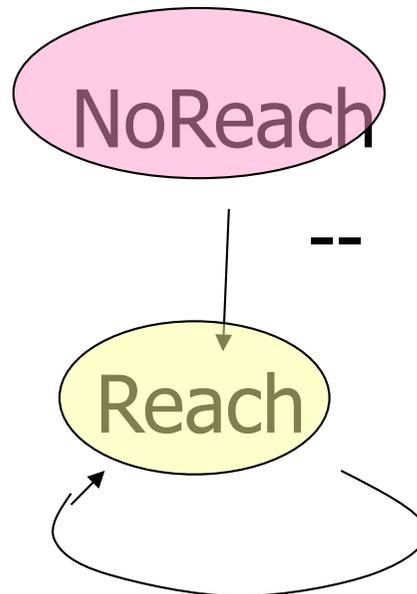
Man bekommt unendlichen Pfad: $P \rightarrow P \rightarrow P \rightarrow P \rightarrow P \rightarrow \dots$
mit negierten markierten Labels

Weiteres Beispiel

- $EDB = \text{Source}(\mathbf{x}) , \text{Target}(\mathbf{x}) , \text{Arc}(\mathbf{x}, \mathbf{y}) .$
- Regeln, die Ziele ($\text{Target}(\mathbf{x})$) beschreiben, die von keiner Source ($\text{Source}(\mathbf{x})$) erreichbar sind:
 - $\text{Reach}(\mathbf{x}) \leftarrow \text{Source}(\mathbf{x})$ Erlaubt
 - $\text{Reach}(\mathbf{x}) \leftarrow \text{Reach}(\mathbf{y}) , \text{Arc}(\mathbf{y}, \mathbf{x})$
 - $\text{NoReach}(\mathbf{x}) \leftarrow \text{Target}(\mathbf{x}) , \text{NOT Reach}(\mathbf{x})$

Stratum 1:

≤ 1 Negations-Kante auf jedem Pfad von NoReach.



Stratum 0:

Keine Negations-Kante von allen Pfaden ausgehend von Reach.

Auswertung stratifizierter Programme

- Werte nacheinander Stratum- i -Prädikate aus, indem alle EDB-Prädikate und Stratum- (j) -Prädikate der unteren Strata $j < i$ als EDB-Prädikate gesehen werden
- $i = 0, 1, \dots, m$