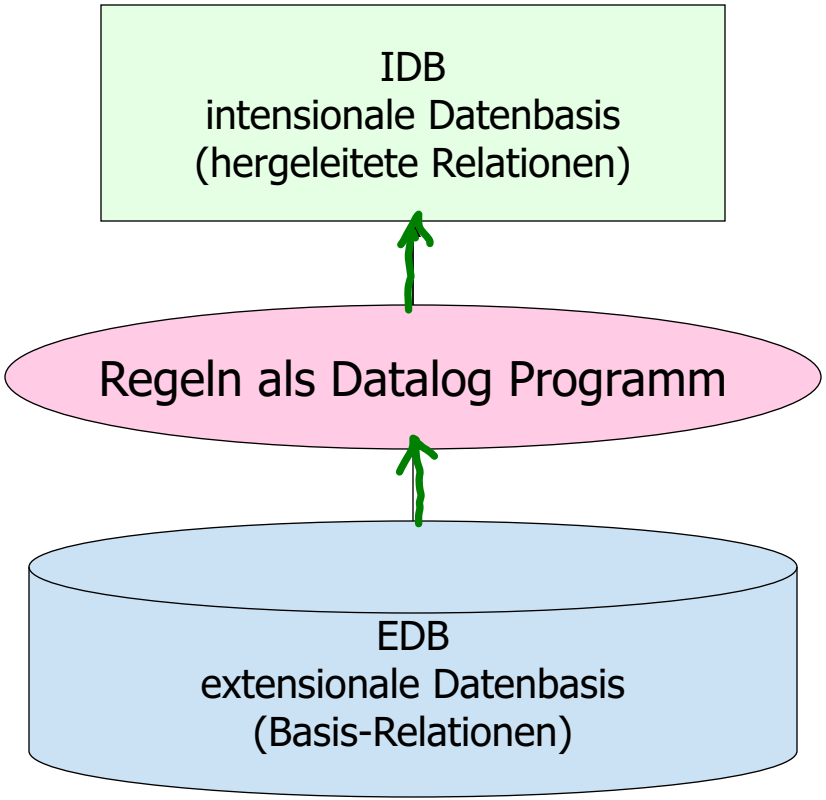
Deduktive Datenbanken

Grundkonzepte einer deduktiven Datenbank



Terminologie

- Die *extensionale Datenbasis* (*EDB*), die manchmal auch Faktenbasis genannt wird.
- Die EDB besteht aus einer Menge von Relationen(Ausprägungen) und entspricht einer „ganz normalen“ relationalen Datenbasis.
- Die *Deduktionskomponente*, die aus einer Menge von (Herleitungs-)*Regeln* besteht.
- Die Regelsprache heißt *Datalog* – abgeleitet von dem Wort *Data* und dem Namen der Logikprogrammiersprache *Prolog*.
- Die *intensionale Datenbasis* (*IDB*), die aus einer Menge von hergeleiteten Relationen(Ausprägungen) besteht.
- Die IDB wird durch Auswertung des Datalog-Programms aus der EDB generiert.

Datalog

Regel:

sokLV(T,S) :- vorlesungen(*V*,*T*,*S*,*P*), professoren(P, „Sokrates“,*R*,*Z*), >(*S*,2).

Äquivalenter Domänenkalkül-Ausdruck:

$\{[t,s] \mid \exists v,p\,([v,t,s,p] \in \text{Vorlesungen} \wedge \exists n,r,z\,([p,n,r,z] \in \text{Professoren} \wedge$

$n = \text{„Sokrates“} \wedge s > 2))\}$

Grundbausteine der Regeln sind *atomare Formeln* **oder** *Literale:*

$q(A_1, \ldots, A_m)$.

q ist dabei der Name einer Basisrelation, einer abgeleiteten Relation oder eines eingebauten Prädikats: <,=,>,…

Beispiel: professoren(*S*, `Sokrates, *R*, *Z*).

Eine Datalog-Regel

$$p(X_1,\ldots,X_m) :- q_1(A_{11},\ldots,A_{1m_1}),\ldots,q_n(A_{n1},\ldots,A_{nm_n}).$$

Jedes $q_j(\ldots)$ ist eine atomare Formel. Die q_j werden oft als *Subgoals* bezeichnet.

X_1, ...,X_m sind Variablen, die mindestens einmal auch auf der rechten Seite des Zeichens :- vorkommen müssen.

Logisch äquivalente Form obiger Regel:

$$p(\ldots) \vee \neg q_1(\ldots) \vee \ldots \vee \neg q_n(\ldots)$$

Wir halten uns an folgende Notation:

- Prädikate beginnen mit einem Kleinbuchstaben.
- Die zugehörigen Relationen – seien es EDB- oder IDB-Relationen – werden mit gleichem Namen, aber mit einem Großbuchstaben beginnend, bezeichnet.

Beispiel Datalog-Programm

Zur Bestimmung von (thematisch) verwandten Vorlesungspaaren

geschwisterVorl(*N* 1, *N* 2) :- voraussetzen(*V*, *N* 1),
voraussetzen(*V*, *N* 2)), *N* 1 < *N* 2.

geschwisterThemen(*T* 1, *T* 2) :- geschwisterVorl(*N* 1, *N* 2),
vorlesungen(*N* 1, *T* 1, *S* 1, *R* 1),
vorlesungen(*N* 2, *T* 2, *S* 2, *R* 2).

aufbauen(*V,N*) :- voraussetzen(*V,N*)
aufbauen(*V,N*) :- aufbauen(*V,M*), voraussetzen(*M,N*).

verwandt(*N,M*) :- aufbauen(*N,M*).
verwandt(*N,M*) :- aufbauen(*M,N*).
verwandt(*N,M*) :- aufbauen(*V,N*), aufbauen(*V,M*).

Voraussetzen: {[Vorgänger, Nachfolger]}
Vorlesungen: {VorlNr, Titel, SWS, gelesenVon]}

Analogie zur EDB/IDB in rel. DBMS

Basis-Relationen entsprechen den EDB.
Sichten entsprechen den IDB:

- „Aufbauen“ als Regeln in einem deduktiven DBMS:

  aufbauen(*V,N*) :- voraussetzen(*V,N*)
  aufbauen(*V,N*) :- aufbauen(*V,M*), voraussetzen(*M,N*).

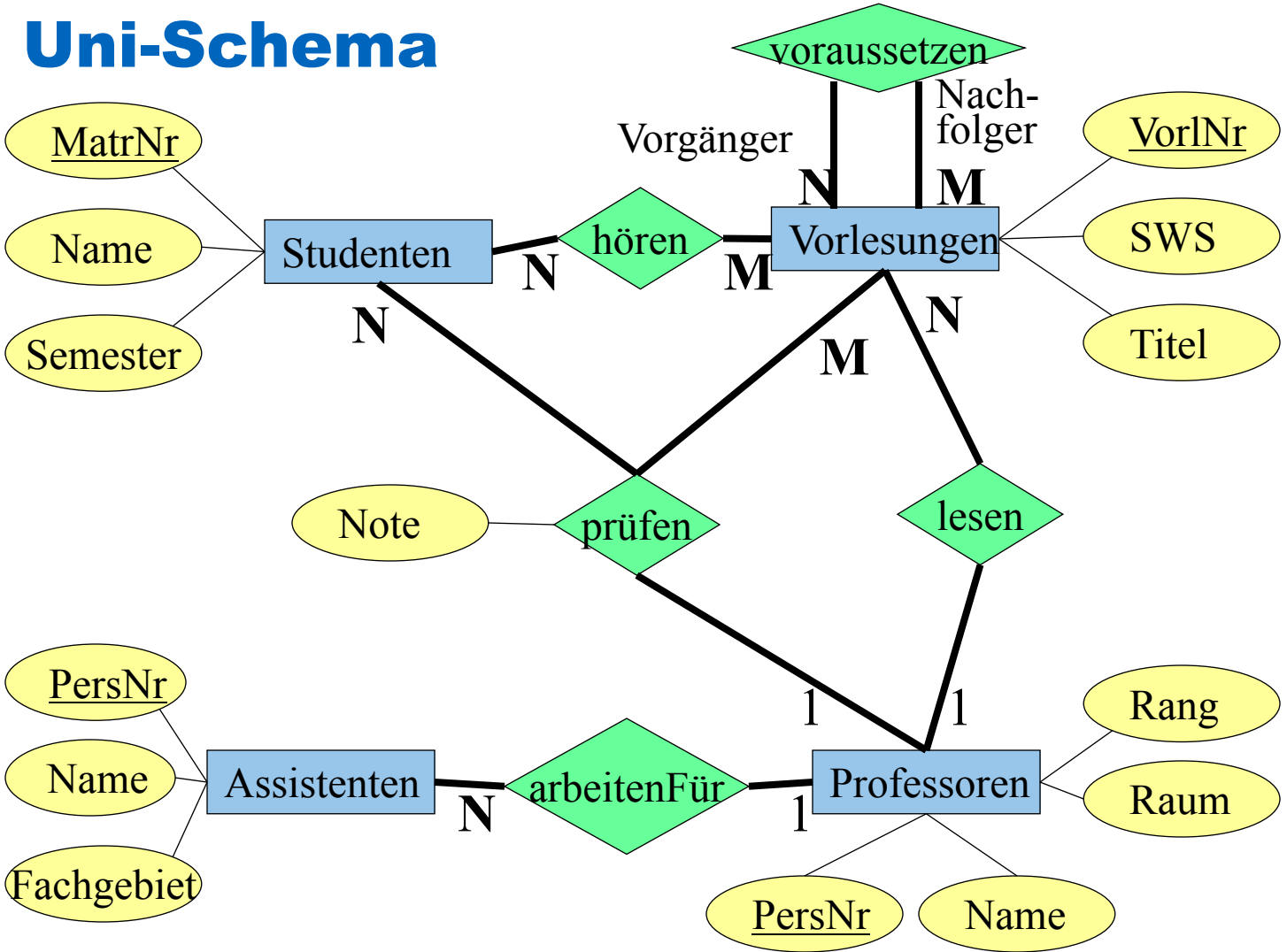
- „Aufbauen“ als Sichtdefinition in DB2:

```
create view aufbauen(V,N) as
  (select Vorgaenger, Nachfolger
   from voraussetzen
      union all
   select a.V, v.Nachfolger
   from aufbauen a, voraussetzen v
   where a.N = v.Vorgaenger)

select * from aufbauen
```

Rekursion in SQL

```
select Vorgänger
from voraussetzen, Vorlesungen
where Nachfolger= VorlNr and
        Titel= `Der Wiener Kreis´
```

Uni-Schema


voraussetzen
Vorgänger
Nach-
folger
N
M
MatrNr
Name
Semester
Studenten
N
hören
M
Vorlesungen
VorlNr
SWS
Titel
N
M
N
Note
prüfen
lesen
1
1
PersNr
Name
Fachgebiet
Assistenten
N
arbeitenFür
1
Professoren
Rang
Raum
PersNr
Name

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen

VorlNr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen

Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören

MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

prüfen

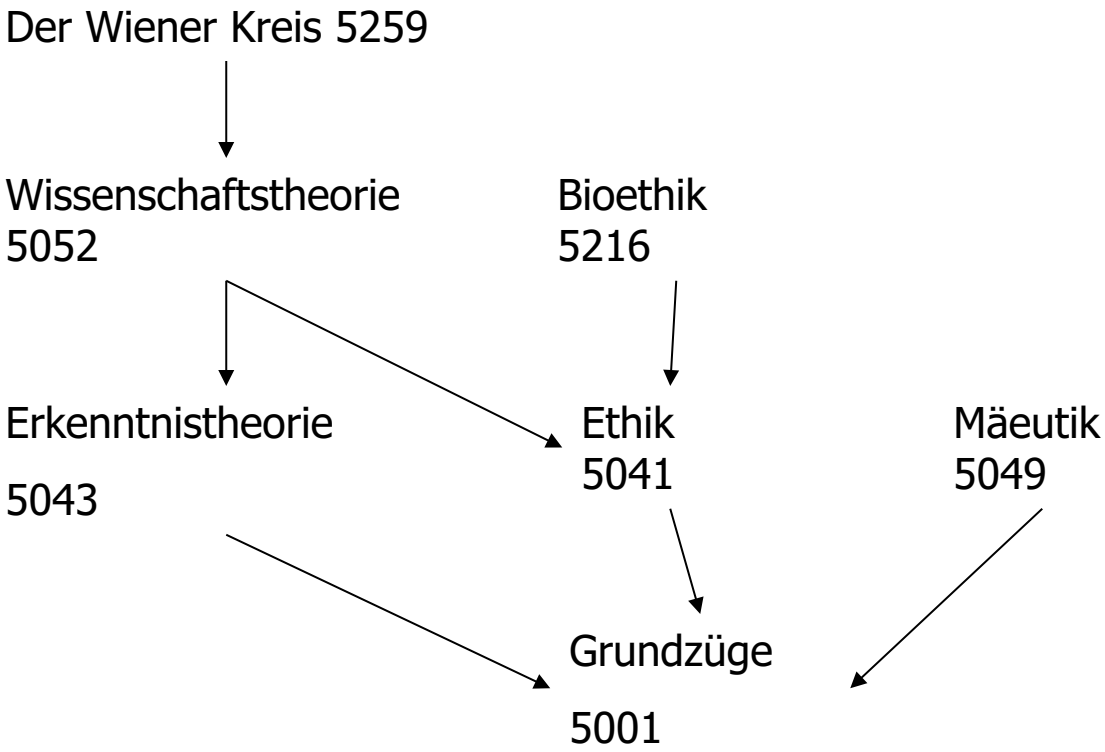
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Assistenten

PerslNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

Vorlesungen			
VorlNr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137


Der Wiener Kreis 5259
Wissenschaftstheorie
5052
Bioethik
5216
Erkenntnistheorie
5043
Ethik
5041
Mäeutik
5049
Grundzüge
5001

Rekursion

```
select v1.Vorgänger
from voraussetzen v1, voraussetzen v2, Vorlesungen v
where v1.Nachfolger= v2.Vorgänger and
        v2.Nachfolger= v.VorlNr and
        v.Titel=`Der Wiener Kreis´
```

Rekursion

select v1.Vorgänger

from voraussetzen v1, voraussetzen v2, Vorlesungen v

where v1.Nachfolger= v2.Vorgänger **and**

v2.Nachfolger= v.VorlNr **and**

v.Titel=`Der Wiener Kreis´

voraussetzen	
Vorgänger	**Nachfolger**
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

voraussetzen	
Vorgänger	**Nachfolger**
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

Vorlesungen			
VorlNr	**Titel**	**SWS**	**gelesen Von**
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

Rekursion

```
select v1.Vorgänger
from voraussetzen v1, voraussetzen v2, Vorlesungen v
where v1.Nachfolger= v2.Vorgänger and
        v2.Nachfolger= v.VorlNr and
        v.Titel=`Der Wiener Kreis´
```

Vorgänger des „Wiener Kreises“ der Tiefe n

```
select v1.Vorgänger
from voraussetzen v1
                ⋮
                voraussetzen vn_minus_1
                voraussetzen vn,

                Vorlesungen v

where v1.Nachfolger= v2.Vorgänger and
                ⋮

                vn_minus_1.Nachfolger= vn.Vorgänger and

                vn.Nachfolger = v.VorlNr and

                v.Titel= `Der Wiener Kreis´
```

Transitive Hülle

$$trans_{A,B}(R) = \{(a,b) \mid \exists k \in \mathrm{N}\ (\exists \Gamma 1, \ldots, \Gamma k \in \mathrm{R}\ (\Gamma 1.A = \Gamma 2.B\ \Lambda \vdots\ \Gamma k\text{-}1.A = \Gamma k.B\ \Lambda\ \Gamma 1.A = a\ \Lambda\ \Gamma k.B = b))\}$$

Rekursion in DB2/SQL99: gleiche Anfrage

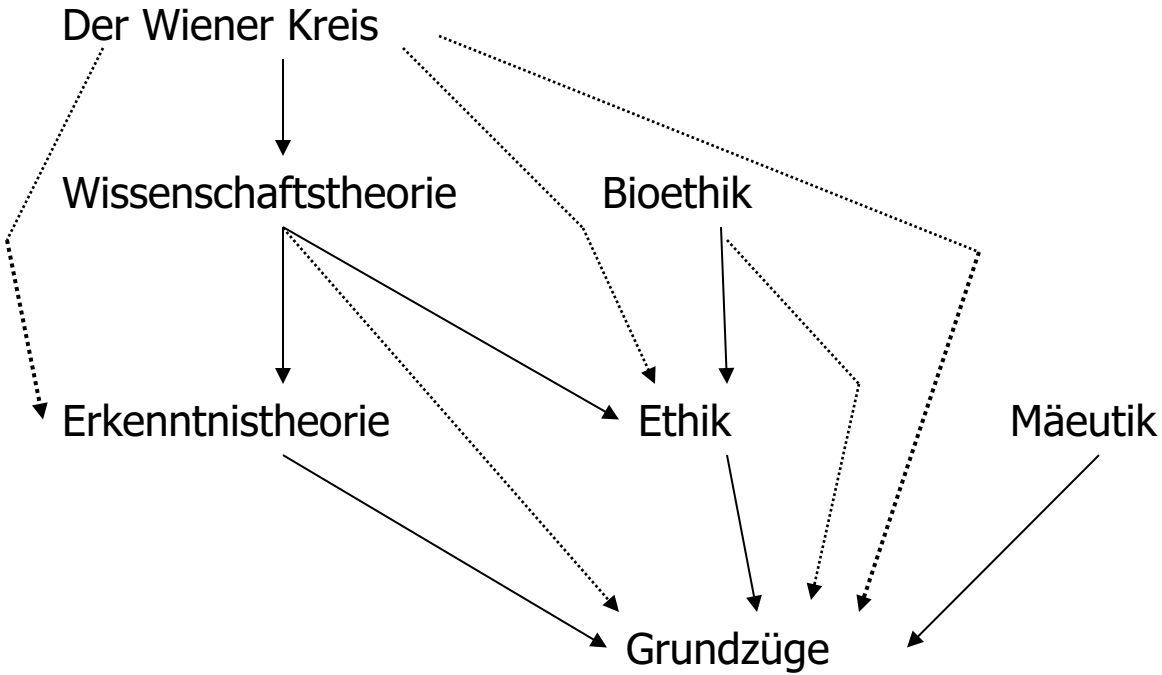
```
with recursive TransVorl (Vorg, Nachf)
as (select Vorgänger, Nachfolger from voraussetzen
        union all
    select t.Vorg, v.Nachfolger
    from TransVorl t, voraussetzen v
    where t.Nachf= v.Vorgänger)

select Titel from Vorlesungen where VorlNr in
    (select Vorg from TransVorl where Nachf in
        (select VorlNr from Vorlesungen
         where Titel= `Der Wiener Kreis´) )
```

- zuerst wird eine temporäre Sicht *TransVorl* mit der **with-**Klausel angelegt

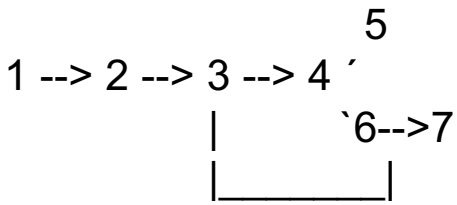
- Diese Sicht *TransVorl* ist rekursiv definiert, da sie selbst in der Definition vorkommt

- Aus dieser Sicht werden dann die gewünschten Tupel extrahiert

- Ergebnis ist natürlich wie gehabt

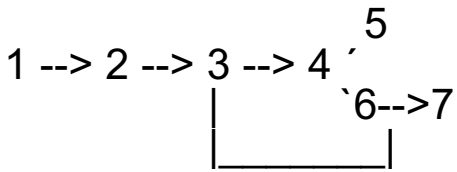

Der Wiener Kreis
Wissenschaftstheorie
Bioethik
Erkenntnistheorie
Ethik
Mäeutik
Grundzüge

Rekursion in Prolog/Datalog

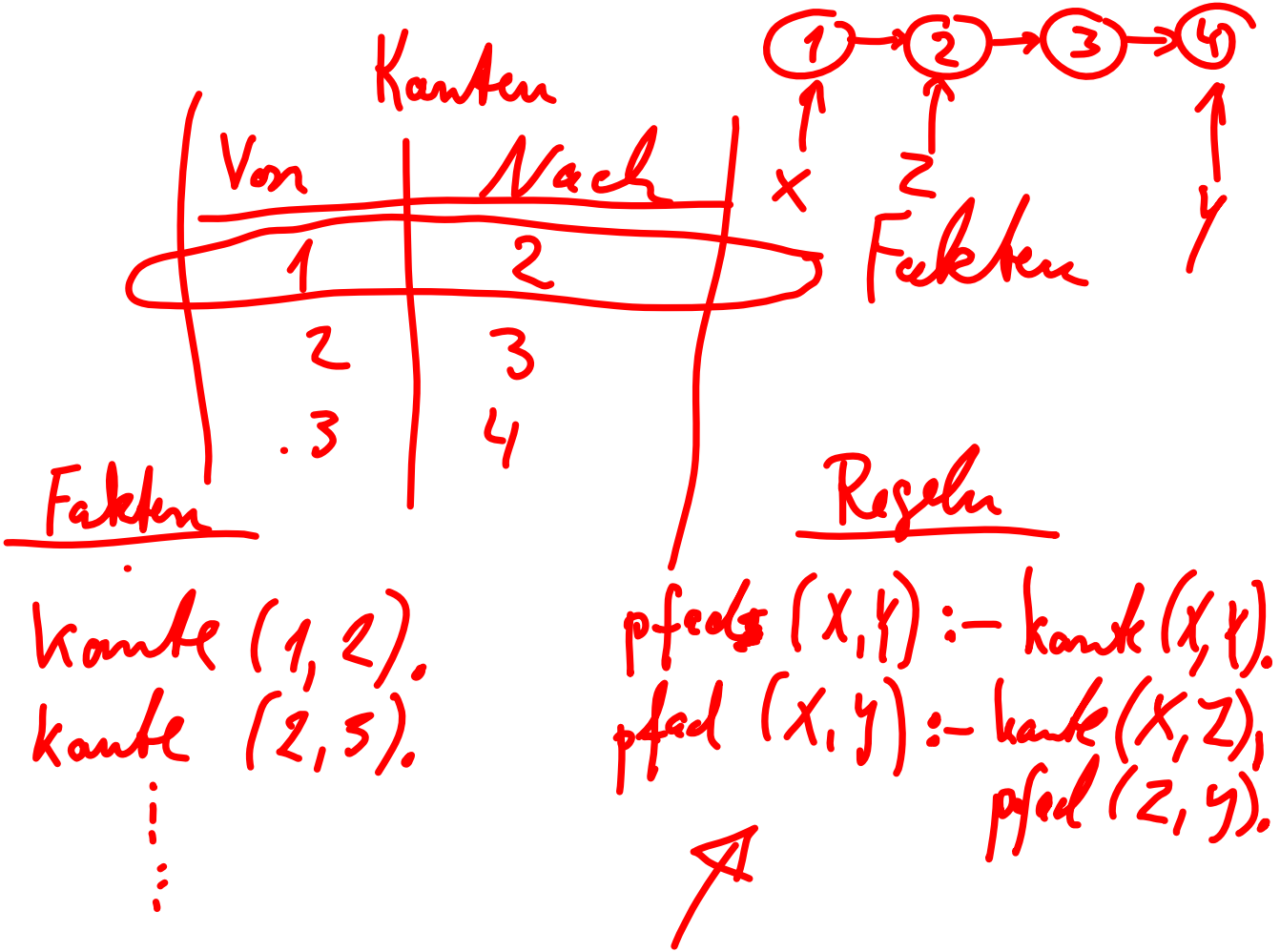


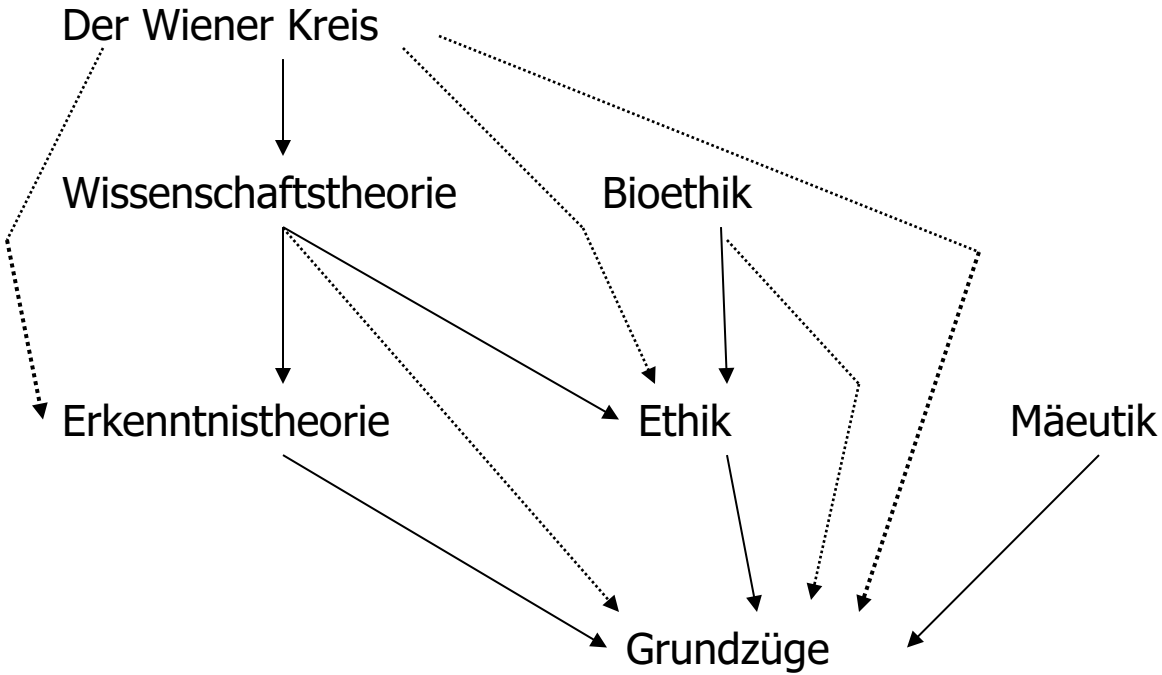

```
kante(1,2).
kante(2,3).
kante(3,4).
kante(4,5).
kante(4,6).
kante(6,7).
kante(3,6).
```

Rekursion in Prolog/Datalog



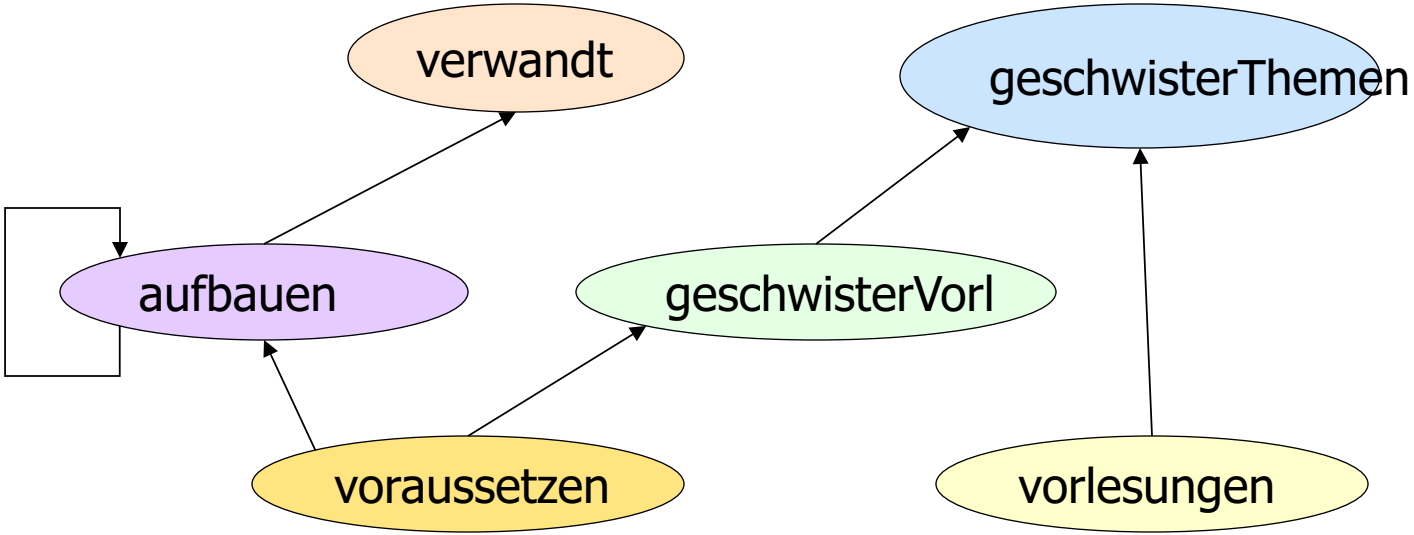

```
kante(1,2).
kante(2,3).
kante(3,4).
kante(4,5).
kante(4,6).
kante(6,7).
kante(3,6).

pfad(V,N) :- kante(V,N).
pfad(V,N) :- kante(V,Z),pfad(Z,N).
```


Kanten
Von
Nach
1
2
2
3
3
4
1
2
3
4
x
z
y
Fakten
Fakten
kante(1,2).
kante(2,3).
Regeln
pfad(X,Y) :- kante(X,Y).
pfad(X,Y) :- kante(X,Z),
pfad(Z,Y).


Der Wiener Kreis
Wissenschaftstheorie
Bioethik
Erkenntnistheorie
Ethik
Mäeutik
Grundzüge

Eigenschaften von Datalog-Programmen

Abhängigkeitsgraph (⟶ = „wird verwendet von“)



- Ein Datalog-Programm ist rekursiv, wenn der Abhängigkeitsgraph einen (oder mehrere) Zyklen hat
- Unser Beispielprogramm ist rekursiv wegen aufbauen → aufbauen

Sicherheit von Datalog-Regeln

Es gibt unsichere Regeln, wie z.B.

ungleich(X, Y) :- X ≠ Y.

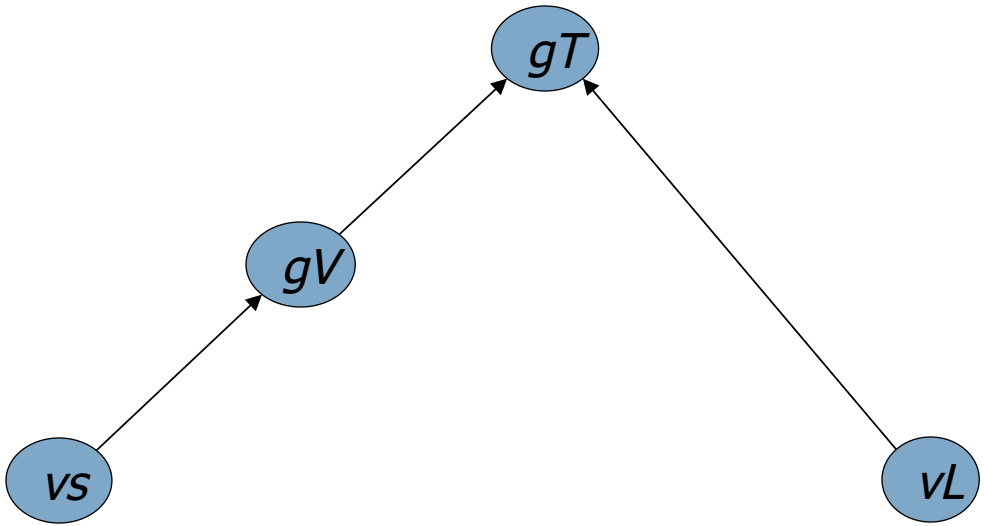
Diese definieren unendliche Relationen.

Eine Datalog-Regel ist sicher, wenn alle Variablen im Kopf beschränkt (range restricted) sind. Dies ist für eine Variable X dann der Fall, wenn:

- die Variable im Rumpf der Regel in mindestens einem normalen Prädikat – also nicht nur in eingebauten Vergleichsprädikaten – vorkommt oder
- ein Prädikat der Form X = c mit einer Konstante c im Rumpf der Regel existiert oder
- ein Prädikat der Form X = Y im Rumpf vorkommt, und man schon nachgewiesen hat, dass Y eingeschränkt ist.

Ein zyklenfreier Abhängigkeitsgraph

gV(N1, N2) :- vs(V, N1), vs(V, N2), N1 < N2.
gT(T1, T2) :- gV(N1, N2), vL(N1, T1, S1, R1), vL(N2, T2, S2, R2).



Eine mögliche topologische Sortierung ist: *vs, gV, vL, gT*

Auswertung nicht-rekursiver Datalog-Programme

1. Für jede Regel mit dem Kopf *p(...),* also

   $$p(...) :- q_1(...), ..., q_n(...).$$

   bilde eine Relation, in der alle im Körper der Regel vorkommenden Variablen als Attribute vorkommen. Diese Relation wird im wesentlichen durch einen natürlichen Verbund der Relationen $Q_1, ..., Q_n$, die den Relationen der Prädikate $q_1, ..., q_n$ entsprechen, gebildet. Man beachte, dass diese Relationen $Q_1, ..., Q_n$ wegen der Einhaltung der topologischen Sortierung bereits ausgewertet (materialisiert) sind.
2. Da das Prädikat *p* durch mehrere Regeln definiert sein kann, werden die Relationen aus Schritt 1. vereinigt. Hierzu muss man vorher auf die im Kopf der Regeln vorkommenden Attribute projizieren. Wir nehmen an, dass alle Köpfe der Regeln für *p* dieselben Attributnamen an derselben Stelle verwenden – durch Umformung der Regeln kann man dies immer erreichen.

Auswertung von Geschwister-Vorlesungen und Geschwister-Themen

Die Relation zu Prädikat *gV* ergibt sich nach Schritt 1 aus folg. Relationenalgebraausdruck:

$$\sigma_{N1<N2}\,(Vs1(V, N1) \bowtie Vs2(V, N2))$$

$$Vs1(V, N1) := \rho_{V \leftarrow \$1}(\rho_{N1 \leftarrow \$2}\,(\rho_{Vs1}(Voraussetzen)))$$

Die dadurch definierte Relation enthält Tupel [v, n1, n2] mit:

- Das Tupel [v, n1] ist in der Relation *Voraussetzen* enthalten,
- das Tupel [v, n2] ist in der Relation *Voraussetzen* enthalten und
- $n1 < n2$.

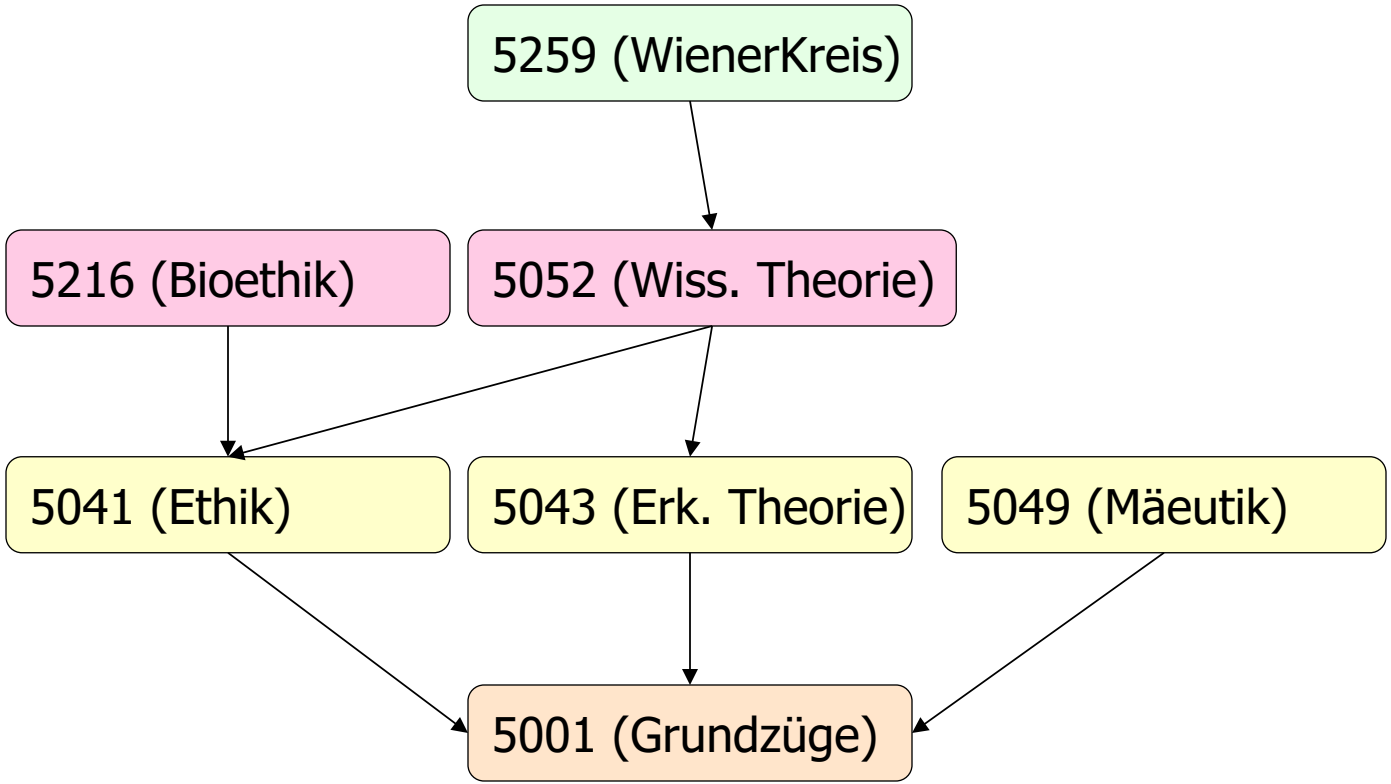
Gemäß Schritt 2. ergibt sich:

$$GV(N1, N2) := \Pi_{N1,\,N2}\,(\sigma_{N1<N2}(Vs1(V, N1) \bowtie Vs2(V, N2)))$$

Analog ergibt sich für die Herleitung von *GT*:

$$GT(T1,T2) := \Pi_{T1,T2}(GV(N1,N2) \bowtie VL1(N1,T1,S1,R1) \bowtie VL2(N2,T2,S2,R2))$$

Veranschaulichung der EDB-Relation *Voraussetzen*



Ausprägung der Relationen *GeschwisterVorl* und *GeschwisterThemen*

GeschwisterVorl	
N1	*N2*
5041	5043
5043	5049
5041	5049
5052	5216

GeschwisterThemen	
T1	*T2*
Ethik	Erkenntnistheorie
Erkenntnistheorie	Mäeutik
Ethik	Mäeutik
Wissenschaftstheorie	Bioethik

Auswertungs-Algorithmus (abstrakt)

Wir betrachten folgende abstrakte Regel:

$$p(X_1,\ldots,X_m) :- q_1(A_{11},\ldots,A_{1m_1}),\ldots,q_n(A_{n1},\ldots,A_{nm_n}).$$

Die Relationen Q_i für die Prädikate q_i seien bereits hergeleitet:

$$Q_i : \{[\$1,\ldots,\$m_i]\}$$

Für jedes Subgoal $q_i(A_{i1},\ldots,A_{im_i})$ bilde folgenden Ausdruck E_i:

$$E_i := \Pi_{V_i}(\sigma_{F_i}(Q_i))$$

Dabei sind die V_i die in $q_i(\ldots)$ vorkommenden Variablen.

Das Selektionsprädikat F_i setzt sich aus einer Menge konjunktiv verknüpfter Bedingungen zusammen.

Auswertungs-Algorithmus (abstrakt)

Falls in $q_i(...,c,...)$ eine Konstante *c* an der *j*-ten Stelle vorkommt, füge die Bedingung

$$\$j = c$$

hinzu.

Falls eine Variable X mehrfach an Positionen *k* und l in $q_i(...,X,...,X,...)$ vorkommt, füge für jedes solches Paar die Bedingung

$$\$k = \$l$$

hinzu.

Auswertungs-Algorithmus (abstrakt)

Für eine Variable *Y*, die nicht in den normalen Prädikaten vorkommt, gibt es zwei Möglichkeiten:

Sie kommt nur als Prädikat

$$Y = c$$

für eine Konstante *c* vor. Dann wird eine einstellige Relation mit einem Tupel

$$Q_Y := \{[c]\}$$

gebildet.

Sie kommt als Prädikat

$$X = Y$$

vor, und *X* kommt in einem normalen Prädikat $q_i(\ldots,X,\ldots)$ an *k*-ter Stelle vor. In diesem Fall setze

$$Q_Y := \rho_{Y \leftarrow \$k}\left(\Pi_{\$k}\left(Q_i\right)\right) .$$

Auswertungs-Algorithmus (abstrakt)

Nun bilde man den Algebra-Ausdruck

$$E := E_1 \bowtie \ldots \bowtie E_n$$

und wende anschließend

$$\sigma_F(E)$$

an, wobei F aus der konjunktiven Verknüpfung der Vergleichsprädikate

$$X \ \Phi \ Y$$

besteht, die in der Regel vorkommen. Schließlich projizieren wir noch auf die im Kopf der Regel vorkommenden Variablen:

$$\Pi_{X_1,\ldots,X_m}(\sigma_F(E))$$

Beispiel: nahe verwandte Vorlesungen

Wir wollen diese Vorgehensweise nochmals am Beispiel demonstrieren:

(r_1) *nvV(N1,N2) :- gV(N1,N2).*

(r_2) *nvV(N1,N2) :- gV(M1,M2),vs(M1,N1),vs(M2,N2).*

Dieses Beispielprogramm baut auf dem Prädikat *gV* auf und ermittelt nahe verwandte Vorlesungen, die einen gemeinsamen Vorgänger erster oder zweiter Stufe haben. Für die erste Regel erhält man folgenden Algebra-Ausdruck:

$$E_{r_1} := \Pi_{N1,N1}(\sigma_{true}(GV(N1,N2)))$$

Für die zweite Regel ergibt sich gemäß dem oben skizzierten Algorithmus:

$$E_{r_2} := \Pi_{N1,N2}(GV(M1,M2) \bowtie Vs1(M1,N1) \bowtie Vs2(M2,N2)).$$

Daraus ergibt sich dann durch die Vereinigung

$$NvV := E_{r_1} \cup E_{r_2}$$

die Relation *NvV*, die durch das Prädikat *nvV* definiert ist. Die Leser mögen bitte die Auswertung dieses Relationenalgebra-Ausdrucks an unserer Beispiel-Datenbasis durchführen.

Auswertung rekursiver Regeln

a(V,N) :- vs(V,N).
a(V,N) :- a(V,M), vs(M,N).

Aufbauen	
V	N
5001	5041
5001	5043
5001	5049
5041	5216
5041	5052
5043	5052
5052	5259
5001	5216
5001	5052
5001	5259
5041	5259
5043	5259

Auswertung rekursiver Regeln

Betrachten wir das Tupel [5001, 5052] aus der Relation *Aufbauen*. Dieses Tupel kann wie folgt hergeleitet werden:

1. *a* (5001, 5043) folgt aus der ersten Regel, da *vs* (5001, 5043) gilt.
2. *a* (5001, 5052) folgt aus der zweiten Regel, *da*
   a) *a* (5001, 5043) nach Schritt 1. gilt und
   b) *vs* (5043, 5052) gemäß der EDB-Relation *Voraussetzen* gilt.

Naive Auswertung durch Iteration

$$A(V,N) = Vs(V,N) \cup \Pi_{V,N}(A(V,M) \bowtie Vs(M,N))$$

A := {}: /*Initialisierung auf die leere Menge */
repeat

$A' := A;$

$A := Vs(V,N);$ /* erste Regel */

$A := A \cup \Pi_{V,N}(A'(V,M) \bowtie Vs(M,N));$ /*zweite Regel */

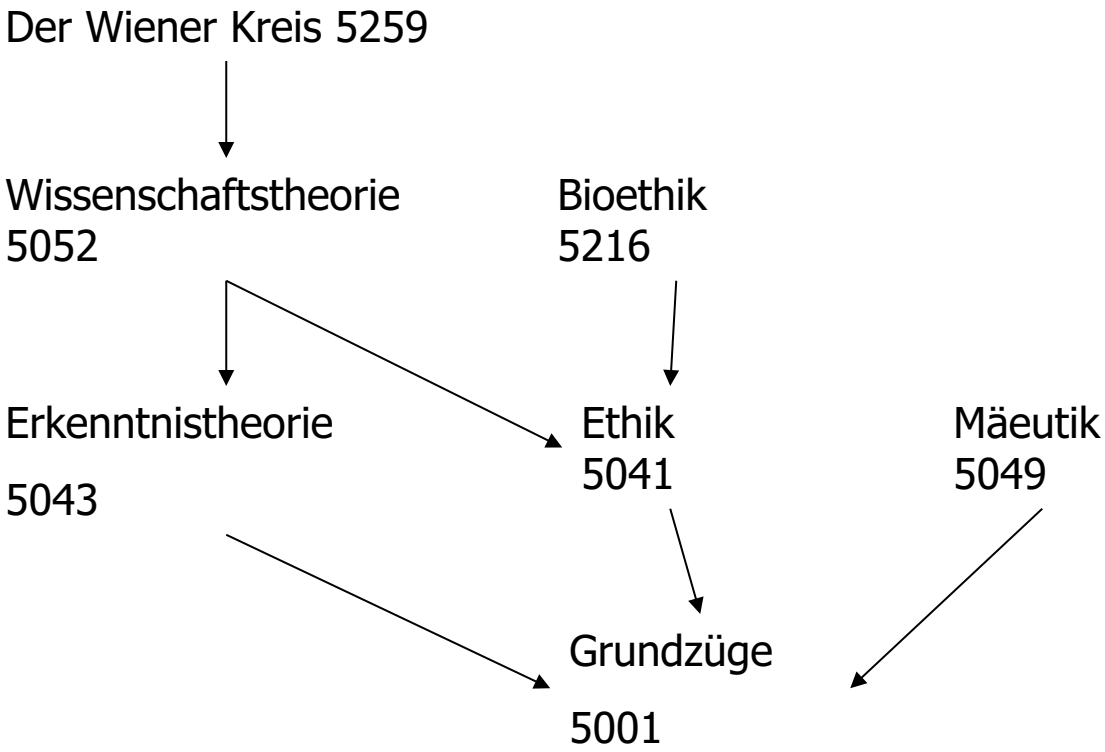
until A‘ = A
output A;

Naive Auswertung durch Iteration

1. Im ersten Durchlauf werden nur die 7 Tupel aus *Voraussetzen* nach *A* „übertragen“, da der Join leer ist (das linke Argument *A‘* des Joins wurde zur leeren Relation {} initialisiert).
2. Im zweiten Schritt kommen zusätzlich die Tupel [5001,5216], [5001,5052], [5041,5259] und [5043,5259] hinzu.
3. Jetzt wird nur noch das Tupel [5001,5259] neu generiert.
4. In diesem Schritt kommt kein neues Tupel mehr hinzu, so dass die Abbruchbedingung *A‘* = *A* erfüllt ist.

(Naive) Auswertung der rekursiven Regel *aufbauen*

Schritt	A
1	[5001,5041], [5001,5043], [5001,5049], [5041,5216], [5041,5052], [5043,5052], [5052,5259]
2	[5001,5041], [5001,5043], [5001,5049], [5041,5216], [5041,5052], [5043,5052], [5052,5259] [5001,5216], [5001,5052], [5041,5259], [5043,5259],
3	[5001,5041], [5001,5043], [5001,5049], [5041,5216], [5041,5052], [5043,5052], [5052,5259] [5001,5216], [5001,5052], [5041,5259], [5043,5259], [5001,5259]
4	wie in Schritt 3 (keine Veränderung, also Terminierung des Algorithmus


Der Wiener Kreis 5259
Wissenschaftstheorie
5052
Bioethik
5216
Erkenntnistheorie
5043
Ethik
5041
Mäeutik
5049
Grundzüge
5001

Inkrementelle (semi-naive) Auswertung rekursiver Regeln

Die Schlüsselidee der *semi-naiven Auswertung* liegt in der Beobachtung, dass für die Generierung eines neuen Tupels *t* der rekursiv definierten IDB-Relation *P* eine bestimmte Regel

$$p(...) :- q_1(...), ..., q_n(...).$$

für Prädikat *p* „verantwortlich“ ist. Dann wird also im iterativen Auswertungsprogramm ein Algebra-Ausdruck der Art

$$E(Q1 \bowtie ... \bowtie Qn)$$

iterativ ausgewertet. Es reicht aber aus

$$E(\Delta Q_1 \bowtie Q_2 \bowtie ... \bowtie Q_n) \cup E(Q_1 \bowtie \Delta Q_2 \bowtie ... \bowtie Q_n) \cup ... \cup E(Q_1 \bowtie Q_2 \bowtie ... \bowtie \Delta Q_n)$$

auszuwerten.

Betrachte Tupel $t = [5001,5259]$

Dieses Tupel wurde aus dem folgenden Join gebildet:

$$\underbrace{[5001,5052]}_{t_1 \in A} \bowtie \underbrace{[5052,5259]}_{t_2 \in Vs}$$

Programm zur semi-naiven Auswertung von *aufbauen*

$A := \{\ \}; \Delta Vs := \{\ \};$

$\Delta A := Vs(V, N);$ /* erste Regel */

$\Delta A := \Delta A \cup \Pi_{V,N}(A(V; M) \bowtie Vs(M, N));$ /*zweite Regel */

A: = ΔA;

repeat

$\Delta A' := \Delta A;$

$\Delta A := \Delta Vs(V, N);$ /* erste Regel, liefert ∅ */

$\Delta A := \Delta A \cup$ /* zweite Regel */

$\Pi_{V,N}(\Delta A'(V, M) \bowtie Vs(M, N)) \cup$

$\Pi_{V,N}(A(V, M) \bowtie \Delta Vs(M, N));$

$\Delta A := \Delta A - A;$ /* entferne "neue" Tupel, die schon vorhanden waren */

A := A ∪ ΔA;

until $\Delta A = \emptyset;$

Illustration der semi-naiven Auswertung von *Aufbauen*

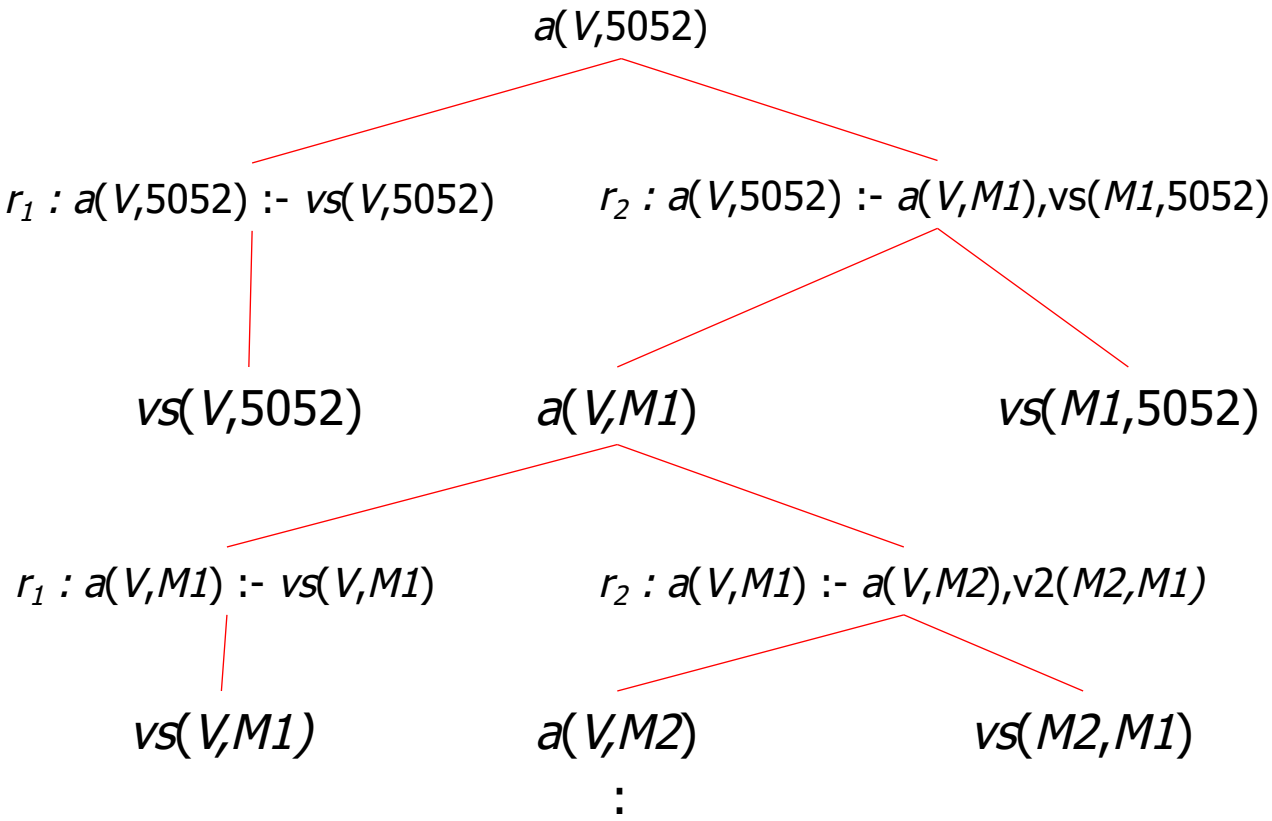
Schritt	ΔA
Initialisierung (Zeile 2. und 3.)	(sieben Tupel aus VS) [5001,5042], [5001,5043] [5043,5052], [5041,5052] [5001,5049], [5001,5216] [5052,5259]
1. Iteration	(Pfade der Länge 2) [5001,5216], [5001,5052] [5041,5259], [5043,5259]
2. Iteration	(Pfade der Länge 3) [5001,5259]
3. Iteration	∅ (Terminierung)

Bottom-Up oder Top-Down Auswertung

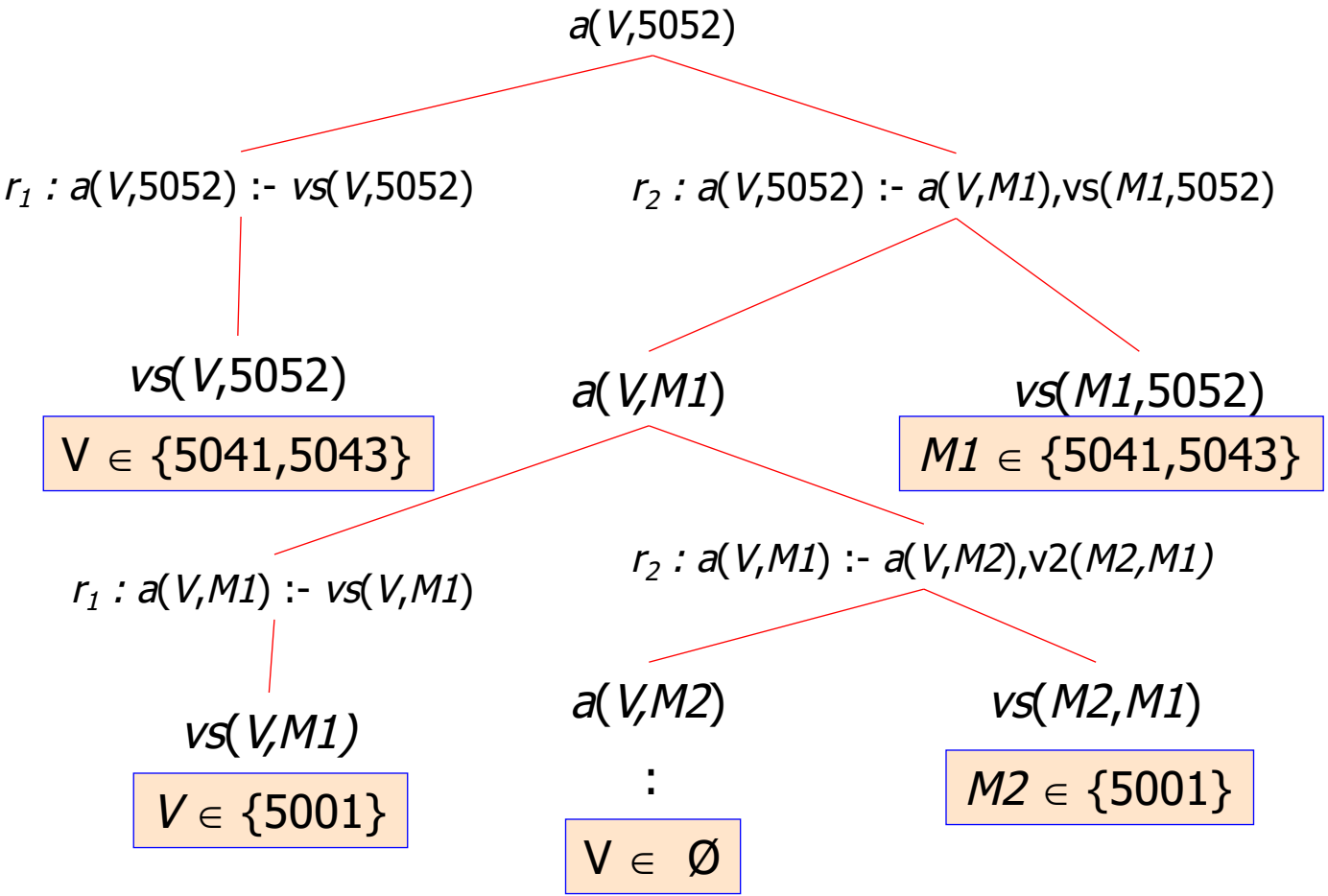
(r_1) *a* (*V*,*N*) :- *vs* (*V*,*N*).
(r_2) *a* (*V*,*N*) :- *a* (*V*,*M*),*vs*(*M*,*N*).

query (*V*) :- *a*(*V*,5052).

Rule/Goal-Baum zur Top-Down Auswertung



Rule/Goal-Baum mit Auswertung



Negation im Regelrumpf

indirektAufbauen(V,N) :- aufbauen(V,N), ¬voraussetzen(V,N)

Stratifizierte Datalog-Programme

Eine Regel mit einem negierten Prädikat im Rumpf, wie z.B.

$$r \equiv p\,(...) :- q_1\,(...),\ ...,\ \neg q_i\,(...),\ ...,\ q_n\,(...).$$

kann nur dann sinnvoll ausgewertet werden, wenn Q_i schon vollständig materialisiert ist. Also müssen zuerst alle Regeln mit Kopf $q_i\,(...)$:- ... ausgewertet sein. Das geht nur, wenn q_i nicht abhängig von p ist.

Also darf der Abhängigkeitsgraph keine Pfade von q_i nach p enthalten. Wenn das für alle Regeln der Fall ist, nennt man das Datalog-Programm *stratifiziert.*

Auswertung von Regeln mit Negation

$$iA(V,N) :- a(V,N), \neg vs(V,N).$$

$$iA(V,N) = \Pi_{V,N}\left(A(V,N) \bowtie \overline{Vs}(V,N)\right)$$
$$= A(V,N) - Vs(V,N)$$

$$Komplement - Bildung$$

$$\overline{Q}_i := \underbrace{\left(DOM \times \ldots \times DOM\right)}_{k-mal} - Q_i$$

Ein etwas komplexeres Beispiel

$$grundlagen(V) :- voraussetzen(V, N).$$

$$spezialVorl(V) :- vorlesungen(V, T, S, R), \neg grundlagen(V).$$

$$Grundlagen(V) := \Pi_V\left(Voraussetzen(V, N)\right)$$

$$SpezialVorl(V) := \Pi_V\left(Vorlesungen(V, T, S, R) \bowtie \overline{Grundlagen}(V)\right)$$

Hierbei ist $\overline{Grundlagen}(V)$ als $DOM - Grundlagen(V)$ definiert.

$$SpezialVorl(V) := \Pi_V\left(Vorlesungen(V, T, S, R)\right) - Grundlagen(V)$$

Ausdruckskraft von Datalog

Die Sprache Datalog, eingeschränkt auf nicht-rekursive Programme aber erweitert um Negation, wird in der Literatur manchmal als *Datalog* $\neg_{non\text{-}rec}$ bezeichnet

Diese Sprache *Datalog*$\neg_{non\text{-}rec}$ hat genau die gleiche Ausdruckskraft wie die relationale Algebra – und damit ist sie hinsichtlich Ausdruckskraft auch äquivalent zum relativen Tupel- und Domänenkalkül

Datalog mit Negation und Rekursion geht natürlich über die Ausdruckskraft der relationalen Algebra hinaus – man konnte in Datalog ja z.B. die transitive Hülle der Relation *Voraussetzen* definieren.

Datalog-Formulierung der relationalen Algebra-Operatoren

Selektion

$$\sigma_{SWS>3}(Vorlesungen),$$

$$query(V,T,S,R) :- vorlesungen(V,T,S,R), S>3.$$

$$query(V,S,R) :- vorlesungen(V,"Mäeutik",S,R).$$

Projektion

$$query(Name,Rang) :- professoren(PersNr,Name,Rang,Raum).$$

Join

$$\Pi_{Titel,Name}(Vorlesungen \bowtie_{gelesenVon=PersNr} Professoren)$$

$$query(T,N) :- vorlesungen(V,T,S,R), professoren(R,N,Rg,Ra).$$

Datalog-Formulierung der relationalen Algebra-Operatoren

Kreuzprodukt

$$query(V1, V2, V3, V4, P1, P2, P3, P4) :- vorlesungen(V1, V2, V3, V4), professoren(P1, P2, P3, P4).$$

$$\text{Professoren} \times \text{Vorlesungen}$$

Vereinigung

$$\Pi_{PersNr,Name}(Assistenten) \cup \Pi_{PersNr,Name}(Professoren)$$

$$query(PersNr, Name) :- assistenten(PersNr, Name, F, B).$$
$$query(PersNr, Name) :- professoren(PersNr, Name, Rg, Ra).$$

Datalog-Formulierung der relationalen Algebra-Operatoren

Mengendifferenz

$$\Pi_{VorlNr}(Vorlesungen) - \Pi_{Vorgänger}(Voraussetzen)$$

$$vorlNr(V) :- vorlesungen(V, T, S, R).$$
$$grundlagen(V) :- voraussetzen(V, N).$$
$$query(V) :- vorlNr(V), \neg grundlagen(V).$$