Kapitel 4:
Logikorientierte Anfragesprachen

4.1 Domain-Relationenkalkül (DRK)

Anfragen sind prädikatenlogische Mengenspezifikationen der Form
$\{<x1, ..., xn> \mid F(x1, ..., xn)\}$
wobei F ein prädikatenlogischer Ausdruck über einer Menge von Domain-Variablen ist mit x1, ..., xn als einzigen freien Variablen.

Die Menge der zulässigen prädikatenlogischen Ausdrücke F ist wie folgt präzise definiert:

1) Für Domain-Variablen x1, ..., xn und eine Relation R mit n Attributen ist $<x1,...,xn> \in R$ (auch R(x1, ..., xn) geschrieben) ein zulässiger Ausdruck.
2) Für Domain-Variablen x, y, Konstanten c und Vergleichsoperationen $\theta \in \{=, \neq, <, >, \leq, \geq\}$ sind $x \theta y$ und $x \theta c$ zulässige Ausdrücke.
3) Falls F1 und F2 zulässige Ausdrücke sind, dann sind auch $F1 \wedge F2$, $F1 \vee F2$, $\neg F1$ und (F1) zulässig.
4) Falls F ein zulässiger Ausdruck mit einer freien Domain-Variable x ist, dann sind auch $\exists x: F(x)$ und $\forall x: F(x)$ zulässige Ausdrücke.
5) Falls F ein zulässiger Ausdruck ist, dann ist auch (F) zulässig.
6) Nur die aufgrund von 1) bis 5) erzeugten Ausdrücke sind zulässig.

Semantik des Domain-Relationenkalküls:

Eine *Datenbank* über einem Schema mit Relationen R1(A11, ..., A1n_1), ..., Rm(Am1, ..., Amn$_m$) wird als Formelmenge H aufgefaßt, die für jede Relation Ri ein Prädikatsymbol Ri verwendet und für jedes Tupel <a1, ..., an$_i$> $\in$ val(Ri) eine atomare Formel Ri(a1, ..., an$_i$) enthält. Die Datenbank an sich ist dann ein Modell dieser Formelmenge H. Alternativ kann man H als eine einzige Formel auffassen, die aus der Konjunktion aller dieser atomaren Formeln (über alle Tupel einer Relation und alle Relationen der Datenbank) besteht.

Beispiel:
Für eine Datenbank mit dem Schema
K (KNr, Name, Stadt), P (PNr, Bez), B (Monat, Tag, KNr, PNr, Menge)
und der Ausprägung
val(K) = {<1, Lauer, Merzig>, <2, Schneider, Homburg>},
val(P) = {<1, Papier>}
val(B) = {<7, 16, 1, 1, 100>, <7, 21, 1, 1, 100>, <10, 26, 2, 1, 100>}
ist
H = K(1, Lauer, Merzig) $\wedge$ K(2, Schneider, Homburg) $\wedge$ P(1, Papier) $\wedge$
B(7, 16, 1, 1, 100) $\wedge$ B(10, 26, 2, 1, 100)

Eine Anfrageergebnis für eine Anfrage der Form {<x1, ..., xn> | F(x1, ..., xn)} wird als (potentielle) Folgerung aus der Datenbankformel H bzw. als Beweisziel aufgefasst. Man möchte also zeigen, daß H |= F bzw. H |— F gilt.
Wenn F keine freien Variablen enthält, das Resultat der Anfrage also 1 oder 0 ist, ist die Semantik der Anfrage somit:
1, wenn H |= F und 0 sonst.
Wenn F freie Variablen enthält, was der Regelfall ist, dann ist die Semantik der Anfrage (das Anfrageresultat) die Menge aller möglichen Interpretationen der freien Variablen, also Substitutionen der Variablen durch Individuenkonstanten aus dem Universum U der Datenbankstruktur, so daß F aus H folgt, also:
{<a1, a2, ... an> | a1, a2, ..., an $\in$ U und H |= F[x1/a1, x2/a2, ..., xn/an]}.

Beispiel:
Semantik (Anfrageresultat) von {<k, n> | K(k, n, Merzig)} ("alle Kunden aus Merzig"):
{<1, Lauer>}
Semantik (1 oder 0) von P(1, Speicher) ("gibt es ein Produkt mit PNr 1 und Bez. Speicher"): 0

Beispiele:

1) Finden Sie (die Namen) alle(r) Kunden mit negativem Saldo.
   → {<k,n,st,sa,r> | <k,n,st,sa,r> ∈ Kunden ∧ sa < 0.0} bzw.
   → {<n> | ∃ k,st,sa,r: <k,n,st,sa,r> ∈ Kunden ∧ sa < 0.0}

2) Finden Sie die Namen aller Kunden, die eine unbezahlte Bestellung haben, die vor Anfang Oktober erfolgte.
   → {<n> | ∃ k,st,sa,r: <k,n,st,sa,r> ∈ Kunden ∧
   ∃ b,m,t,p,me,su,stat: <b,m,t,k,p,me,su,stat> ∈ Bestellungen ∧
   m < 10 ∧ stat ≠ 'bezahlt'}

3) Finden Sie die Namen der Homburger Kunden, die seit Anfang September ein Produkt aus Homburg geliefert bekommen haben, jeweils mit der Bezeichnung des entsprechenden Produkts.
   → {<n,bez> | ∃ k,st,sa,r: <k,n,st,sa,r> ∈ Kunden ∧
   ∃ p,g,pr,l,v: <p,bez,g,pr,l,v> ∈ Produkte ∧
   ∃ b,m,t,me,su,stat: <b,m,t,k,p,me,su,stat> ∈ Bestellungen ∧
   st='Homburg' ∧ mw9 ∧ l='Homburg'}

4) Finden Sie die Kunden, von denen mindestens eine Bestellung registriert ist.
   → {<k,n,st,sa,r> | <k,n,st,sa,r> ∈ Kunden ∧
   ∃ b,m,t,p,me,su,stat: <b,m,t,k,p,me,su,stat> ∈ Bestellungen}

5) Finden Sie die Kunden, von denen keine Bestellung registriert ist.
   → {<k,n,st,sa,r> | <k,n,st,sa,r> ∈ Kunden ∧
   ∀ b,m,t,k',p,me,su,stat:
   <b,m,t,k',p,me,su,stat> ∈ Bestellungen ⇒ k≠k'}

6) Finden Sie die Kunden, die alle überhaupt lieferbaren Produkte irgendwann bestellt haben.
   → {<k,n,st,sa,r> | <k,n,st,sa,r> ∈ Kunden ∧
   ∀ p: (∃ bez,g,pr,l,v: <p,bez,g,pr,l,v> ∈ Produkte) ⇒
   (∃ b,m,t,me,su,stat: <b,m,t,k,p,me,su,stat> ∈ Bestellungen) }

Eine Anfragesprache auf der Basis des Domain-Relationenkalküls: Query-by-Example (QBE)

(bzw. MS-Access u.a. als kommerzielle Variante)

Idee:
Bedingungen eines Ausdrucks des Domain-Relationenkalküls werden in Form von "Beispielelementen" in entsprechende Fenster eingetragen, wobei für jedes Relationenschema ein Fenster vorgegeben wird. Gleiche Elemente in verschiedenen Fenstern stehen für "Joinbedingungen".

Beispiel:

Finden Sie die Namen der Homburger Kunden, die seit Anfang September ein Produkt aus Homburg (oder Saarbrücken) geliefert bekommen haben, dessen Preis über 100 DM liegt, jeweils mit der Bezeichnung des entsprechenden Produkts.

Kunden

KNr	Name	Stadt	Saldo	Rabatt
_55	P.	Homburg		

Bestellungen

BestNr	Monat	Tag	KNr	PNr	Menge	Summe	Status
	>= 9		_55	_33			

Produkte

PNr	Bez	Gewicht	Preis	Lagerort	Vorrat
_33	P.		> 100.00	Homburg Saarbrücken	

4.2 Tupel-Relationenkalkül (TRK)

Der Domain-Relationenkalkül hat den Vorteil, daß er unmittelbar aus der Prädikatenlogik 1. Ordnung abgeleitet ist und deren Standardnotation weitgehend übernimmt. Er hat aber auch den Nachteil, daß diese Notation bei Datenbankschemata mit vielen Attributen, etwas ausladend wird, weil sehr viele Domain-Variable eingeführt werden müssen. Eine kompaktere Schreibweise stellt der Tupel-Relationenkalkül bereit, bei dem Variablen, wie z.B. t, für ganze Tupel stehen und auf Attribute mit der Notation t.A Bezug genommen wird.

Anfragen sind prädikatenlogische Mengenspezifikation der Form
$\{t \mid F(t)\}$,
wobei F ein prädikatenlogischer Ausdruck über einer Menge von Tupelvariablen ist mit t als einziger freier Variable, oder der Form
$\{<t1.A1, ..., tn.An> \mid F(t1, ..., tn)\}$,
wobei F ein prädikatenlogischer Ausdruck über einer Menge von Tupelvariablen ist mit t1, ..., tn als (nicht notwendigerweise verschiedenen) einzigen freien Variablen und A1, ..., An Attributnamen sind.

Die Menge der zulässigen prädikatenlogischen Ausdrücke F ist wie folgt präzise definiert:

1) Für eine Tupelvariable r und eine Relation R ist $r \in R$ (auch R(r) geschrieben) ein zulässiger Ausdruck.
2) Für Tupelvariable r, s und Attribute A, B mit dom(A)=dom(B), Konstanten $c \in dom(A)$ und Vergleichsoperationen $\theta \in \{=, \neq, <, >, \leq, \geq\}$ sind r.A θ s.B und r.A θ c zulässige Ausdrücke.
3) Falls F1 und F2 zulässige Ausdrücke sind, dann sind auch $F1 \wedge F2$, $F1 \vee F2$, $\neg F1$ und (F1) zulässig.
4) Falls F ein zulässiger Ausdruck mit einer freien Tupelvariable r ist, dann sind auch $\exists r: F(r)$ und $\forall r: F(r)$ zulässige Ausdrücke.
5) Falls F ein zulässiger Ausdruck ist, dann ist auch (F) zulässig.
6) Nur die aufgrund von 1) bis 5) erzeugten Ausdrücke sind zulässig.

Semantik des Tupel-Relationenkalküls:

Die Semantik des Tupel-Relationenkalküls ist durch eine Übersetzung in den Domain-Relationenkalkül gegeben. Diese ist wie folgt definiert:
In einer Anfrage der Form $\{t \mid F(t)\}$ oder $\{<t.B1, ..., t.Bn> \mid F(t)\}$ wird jeder Term der Form r.Ai mit einer Tupelvariablen r durch eine Domain-Variable ra_i ersetzt und jeder Term der Form $r \in R$ über einer Relation mit Schema R(A1, ..., Am) durch einen Term $R(ra_1, ..., ra_m)$ mit Domain-Variablen $ra_1, ..., ra_m$.
Die Semantik der TRK-Anfrage ist dann gerade die der durch diese Übersetzung entstehenden DRK-Anfrage.

Beispiele:

1) Finden Sie (die Namen) alle(r) Kunden mit negativem Saldo.
   $\rightarrow \{t \mid t \in \text{Kunden} \wedge t.\text{Saldo} < 0.0\}$ bzw. $\{t.\text{Name} \mid t \in \text{Kunden} \wedge t.\text{Saldo} < 0.0\}$

2) Finden Sie die Namen aller Kunden, die eine unbezahlte Bestellung haben, die vor Anfang Oktober erfolgte.
   $\rightarrow \{t.\text{Name} \mid t \in \text{Kunden} \wedge$
   $\exists\, b: b \in \text{Bestellungen} \wedge$
   $t.\text{KNr}=b.\text{KNr} \wedge b.\text{Monat} < 10 \wedge b.\text{Status} \neq \text{'bezahlt'}\}$

3) Finden Sie die Namen der Homburger Kunden, die seit Anfang September ein Produkt aus Homburg geliefert bekommen haben, jeweils mit der Bezeichnung des entsprechenden Produkts.
   $\rightarrow \{k.\text{Name}, p.\text{Bez} \mid k \in \text{Kunden} \wedge p \in \text{Produkte} \wedge$
   $\exists\, b: b \in \text{Bestellungen} \wedge$
   $k.\text{Stadt}=\text{'Homburg'} \wedge b.\text{Monat} \geq 9 \wedge p.\text{Lagerort}=\text{'Homburg'} \wedge$
   $k.\text{KNr}=b.\text{KNr} \wedge b.\text{PNr}=p.\text{PNr}\}$

4) Finden Sie die Kunden, von denen mindestens eine Bestellung registriert ist.
   $\rightarrow \{t \mid t \in \text{Kunden} \wedge \exists\, b: b \in \text{Bestellungen} \wedge b.\text{KNr}=t.\text{KNr}\}$

5) Finden Sie die Kunden, von denen keine Bestellung registriert ist.
   $\rightarrow \{t \mid t \in \text{Kunden} \wedge \neg(\exists\, b: b \in \text{Bestellungen} \wedge b.\text{KNr}=t.\text{KNr})\}$ oder
   $\rightarrow \{t \mid t \in \text{Kunden} \wedge \forall\, b: b \in \text{Bestellungen} \Rightarrow b.\text{KNr} \neq t.\text{KNr}\}$

6) Finden Sie die Kunden, die alle überhaupt lieferbaren Produkte irgendwann bestellt haben.
   $\rightarrow \{t \mid t \in \text{Kunden} \wedge \quad \forall\, p: p \in \text{Produkte} \Rightarrow$
   $(\exists\, b: b \in \text{Bestellungen} \wedge b.\text{PNr}=p.\text{PNr} \wedge b.\text{KNr}=t.\text{KNr})\}$

Domain-unabhängige Ausdrücke

Problem: Anfragen liefern u.U. unendliche Resultatmengen.
Beispiel: $\{t \mid \neg(\, t \in R)\}$ für eine endliche Relation R.

Lösungsidee:
Eine Formel des Tupelrelationenkalküls heißt *domain-unabhängig* genau dann, wenn für jede mögliche Ausprägung der Datenbank die Resultatmenge der Formel nur von der Datenbank, nicht aber von den zugrundeliegenden Domains abhängt.

Man sollte nur domain-unabhängige Formeln als Anfragen verwenden.

Präzisierung:

Sei F eine geschlossene Formel des Tupelrelationenkalküls. Eine *Interpretation* von F ist eine Abbildung der Relationsnamen, Konstanten und Variablen in F auf Relationsausprägungen, elementaren Konstanten und Tupeln, die aus einem *Universum* von Werten gebildet werden. Wir sagen, daß F erfüllt ist, wenn die Interpretation von F in dem entsprechenden Universum erfüllt ist (bei kanonischer Interpretation der Junktoren und Quantoren).

Sei F(t) eine Formel des Tupelrelationenkalküls mit einer einzigen freien Variable t (in einer Anfrage $\{t \mid F(t)\}$). Die Interpretation von F(t) ist die Menge aller Tupel, die aus Elementen des gewählten Universums gebildet werden, so daß bei Substitution der Tupel für die freie Variable t die Formel F(t) im Universum erfüllt ist.

Wahl des Universums:
- Variante 1 - *unbeschränkte Interpretation*:
  das Universum besteht aus der Vereinigung aller Domains der Datenbank
- Variante 2 - *beschränkte Interpretation*:
  das Universum besteht aus dem *aktiven Domain* der Datenbank und einer Formel F,
  d.h. der Vereinigung aller Attributwerte der Relationsausprägungen der Datenbank
  und aller Konstanten in F

Beispiele:

Sei R eine Relation mit sch(R)={A}, dom(A)={1,2}, val(R)={<1>}.
Der Ausdruck $\{t \mid \neg\, (t \in R)\}$ hat
  als unbeschränkte Interpretation den Wert {<2>} und
  als beschränkte Interpretation den Wert Ø.
Der Ausdruck $\{t \mid \exists\ x : \neg\, (x=t) \wedge x.A=1\}$ hat
  als unbeschränkte Interpretation den Wert {<2>} und
  als beschränkte Interpretation den Wert Ø.
Der Ausdruck $\{t \mid t \in R \wedge \forall\, x : x=t\}$ hat
  als unbeschränkte Interpretation den Wert Ø und
  als beschränkte Interpretation den Wert {<1>}.

Definition:

Eine Formel des Tupelrelationenkalküls heißt *domain-unabhängig* genau dann, wenn ihre unbeschränkte Interpretation für jede mögliche Wahl von Domains, die den aktiven Domain umfassen, mit ihrer beschränkten Interpretation übereinstimmt.

Sichere Ausdrücke

Die "Sicherheit" von Ausdrücken ist eine hinreichende, syntaktisch prüfbare Bedingung für Domain-Unabhängigkeit.

Idee:
Die Variable x in der Formel F(x) heißt beschränkt, wenn für jede Interpretation von F gilt:
wenn F(x) wahr ist, dann liegt die Interpretation von x im aktiven Domain von F.
Die Variable x in der Formel F(x) heißt unbeschränkt, wenn für jede Interpretation von F gilt:
wenn die Interpretation von x nicht im aktiven Domain von F liegt, muß F(x) wahr sein.
Durch syntaktische Einschränkungen von Formeln können
- freie Variablen beschränkt werden
(so daß in $\{t \mid F(t)\}$ die Formel F(t) nur für t aus dem aktiven Domain erfüllbar ist),
- durch Existenzquantoren gebundene Variablen beschränkt werden
(so daß in $\exists x : G(x)$ die Formel G(x) nur für x aus dem aktiven Domain erfüllbar ist) und
- durch Allquantoren gebundene Variablen unbeschränkt werden
(so daß in $\forall x : G(x)$ die Formel G(x) für Interpretationen von x, die nicht im aktiven Domain liegen, immer wahr ist).

Definition:

Sei F(t) eine Formel des Tupelrelationenkalküls mit freier Variable t. Zu jeder Tupelvariablen in F läßt sich ein Schema (d.h. eine Menge Attributen) aus F und dem Datenbankschema ableiten. Die *Beschränktheit* und die *Unbeschränktheit* der Attribute von Tupelvariablen in einer Formel F sind wie folgt induktiv definiert.

1) In $t \in R$ sind alle Attribute von t beschränkt.
2) In t.A=c mit einer Konstanten c ist t.A beschränkt.
3) In r.A=s.B $\wedge$ F ist r.A beschränkt, wenn s.B in F beschränkt ist,
und s.B beschränkt, wenn r.A in F beschränkt ist..
4a) In F $\wedge$ G ist t.A genau dann beschränkt, wenn t.A in F oder in G beschränkt ist.
4b) In F $\wedge$ G ist t.A genau dann unbeschränkt, wenn t.A in F und in G unbeschränkt ist.
5a) In F $\vee$ G ist t.A genau dann beschränkt, wenn t.A in F und in G beschränkt ist.
5b) In F $\vee$ G ist t.A genau dann unbeschränkt, wenn t.A in F oder in G unbeschränkt ist.
6a) In $\neg$F ist t.A beschränkt genau dann, wenn t.A in F unbeschränkt ist.
6b) In $\neg$F ist t.A unbeschränkt genau dann, wenn t.A in F beschränkt ist.
7a) In $\exists$ x: F(x) ist t.A. beschränkt genau dann, wenn t.A in F beschränkt ist.
7b) In $\exists$ x: F(x) ist t.A. unbeschränkt genau dann, wenn t.A in F unbeschränkt ist.
8a) In $\forall$ x: F(x) ist t.A beschränkt genau dann, wenn t.A in F beschränkt ist.
8b) In $\forall$ x: F(x) ist t.A unbeschränkt genau dann, wenn t.A in F unbeschränkt ist.
9) Nur die gemäß 1) bis 8) beschränkten bzw. unbeschränkten Attribute von Tupelvariablen sind beschränkt bzw. unbeschränkt.

Eine Formel des Tupelrelationenkalküls ist *sicher* (engl.: safe) genau dann, wenn
a) alle Attribute aller freien Tupelvariablen beschränkt sind,
b) für jede Teilformel der Form $\exists$ x: P(x) alle Attribute von x in P beschränkt sind,
c) für jede Teilformel der Form $\forall$ x: P(x) alle Attribute von x in P unbeschränkt sind.

Satz:

Jede sichere Formel des Tupelrelationenkalküls ist domain-unabhängig.

Beispiele (für das Schema R(A,B) und z.B. die Ausprägung R={<2,2>}):

- $\{t \mid \neg(t \in R) \vee t.A=2 \vee t.B=2\}$ ist unsicher.
- $\{t \mid \neg(t \in R) \wedge t.A=2\}$ ist unsicher.
- $\{t \mid \neg(t \in R) \wedge t.A=2 \wedge t.B=2\}$ ist sicher.

- $\{t \mid \exists s: (s \in R \wedge s.A=t.A) \}$ ist unsicher.
- $\{t \mid \exists s: (s \in R \vee s=t) \}$ ist unsicher.
- $\{t \mid \exists s: (s \in R \wedge s=t) \}$ ist sicher.
- $\{t \mid \neg(\exists s: (s \in R \wedge s=t)) \}$ ist unsicher.

- $\{t \mid \forall s: (s \in R \wedge s=t) \}$ ist unsicher.
- $\{t \mid \forall s: (s \in R \Rightarrow s=t) \}$ ist unsicher.
- $\{t \mid \forall s: (s \in R \Rightarrow (s=t \wedge t.A=2 \wedge t.B=2)) \}$ ist unsicher.
- $\{t \mid t \in R \wedge \forall s: (s \in R \Rightarrow (s=t \wedge t.A=2 \wedge t.B=2)) \}$ ist sicher.
- $\{t \mid \forall s: (s \in R \wedge (s=t \wedge t.A=2 \wedge t.B=2)) \}$ ist unsicher.
- $\{t \mid \forall s: (s \in R \Rightarrow \neg(s=t)) \}$ ist unsicher.

- $\{t \mid t \in K \wedge \forall p: p \in P \Rightarrow (\exists b: b \in B \wedge b.PNr=p.PNr \wedge b.KNr=t.KNr) \}$ ist sicher.

Satz:
Eine Anfrage des Tupelrelationenkalküls ist sicher, wenn
- die Anfrage die Form hat $\{t \mid t \in R \wedge F(t)\}$,
- jede existenzquantifizierte Teilformel die Form hat $\exists x: x \in R \wedge F(x)$ und
- jede allquantifizierte Teilformel die Form hat $\forall x: s \in R \Rightarrow F(x)$.

Für den Domain-Relationenkalkül sind Domain-Unabhängigkeit und Sicherheit analog definiert.

4.3 Mächtigkeit relationaler Anfragesprachen

Anfragesprachen existierender Datenbanksysteme basieren überwiegend entweder auf dem Tupel-Relationenkalkül (z.B. Ingres/Quel) oder dem Domain-Relationenkalkül (z.B. DB2/QMF oder das GUI von MS Access) oder einer Kombination von Relationenalgebra und Tupel-Relationenkalkül (SQL).

Satz:

Jede Anfrage, die sich mit der Relationenalgebra ausdrücken lässt, lässt sich sowohl mit dem sicheren Tupel-Relationenkalkül als auch dem sicheren Domain-Relationenkalkül ausdrücken.

Jede Anfrage, die sich mit dem sicheren Tupel-Relationenkalkül ausdrücken lässt, lässt sich sowohl mit der Relationenalgebra als auch dem sicheren Domain-Relationenkalkül ausdrücken.

Jede Anfrage, die sich mit dem sicheren Domain-Relationenkalkül ausdrücken lässt, lässt sich sowohl mit der Relationenalgebra als auch dem sicheren Tupel-Relationenkalkül ausdrücken.

Beweis: siehe Vorlesung

Bemerkung:
Relationenalgebra, sicherer Tupel-Relationenkalkül und sicherer Domain-Relationenkalkül haben also dieselbe Ausdrucksmächtigkeit.

4.4 Datalog: Deduktionsregeln als Anfragesprache

In DBS werden Informationen traditionellerweise ausschließlich *extensional* - in Form von *Daten (Fakten)* - repräsentiert. In wissensbasierten Systemen und in modernen, um deduktive Fähigkeiten erweiterten DBS können Informationen zusätzlich auch *intensional* - in Form von *Regeln* - repräsentiert werden. Mit Hilfe der Regeln können aus den Fakten weitere Informationen hergeleitet werden.

Beispiel mit rein extensionaler Repräsentation:

Flüge (F)

FlugNr	Abflugort	Zielort	...
LH58	Frankfurt	Chicago	
AA371	Chicago	Phoenix	
DA77	Phoenix	Yuma	
AA70	Frankfurt	Dallas	
AA351	Dallas	Phoenix	
UA111	Chicago	Dallas	

Flugverbindungen (V)

Abflugort	Zielort
Frankfurt	Chicago
Frankfurt	Dallas
Frankfurt	Phoenix
Frankfurt	Yuma
...	...

Beispiel mit intensionaler Repräsentation der Flugverbindungen:

als Fixpunktgleichung in der Relationenalgebra:

$V = F \cup (V \,|x|\, [V.Zielort=F.Abflugort] F)$

mit der Semantik, daß V die bzgl. $\subseteq$ kleinste Relation ist, für die Fixpunktgleichung erfüllt ist.

als Menge von (Prolog-artigen) Regeln:

Flugverbindungen (a,z) :- Flüge (a,z)
Flugverbindungen (a,z) :- Flugverbindungen (a,o), Flüge (o,z)

als Formeln des Domain-Relationenkalküls:

$\forall$ a, z: Flüge (a,z) $\Rightarrow$ Flugverbindungen (a,z)
$\forall$ a, z, o: Flugverbindungen (a,o) $\wedge$ Flüge (o,z) $\Rightarrow$ Flugverbindungen (a,z)

bzw.

$\forall$ a, z: $\langle a,z\rangle \in$ Flüge $\Rightarrow$ $\langle a,z\rangle \in$ Flugverbindungen
$\forall$ a, z, o: $\langle a,o\rangle \in$ Flugverbindungen $\wedge$ $\langle o,z\rangle \in$ Flüge $\Rightarrow$ $\langle a,z\rangle \in$ Flugverbindungen

Datalog

Definitionen:

Eine *Hornklausel* über einer endlichen Menge von Prädikatsymbolen ist eine logische Formel der Form

$\forall$ X1, ..., Xn:
P1 (Z11, Z12, ..., Z1k_1) $\wedge$ ... $\wedge$Pm (Zm1, Zm2, ..., Zmk_m) $\Rightarrow$ P0 (Z01, Z02, ..., Z0k_0)

mit k_i-stelligen Prädikaten Pi, Variablen X1, ..., Xn und
Konstanten oder Variablen Zij, so daß Zij entweder eine Konstante ist oder eine der Variablen X1, ..., Xn.

Die Formel
P0 (Z01, Z02, ..., Z0k_0)
wird als "Kopf" (engl.: head) der Hornklausel bezeichnet,
die Formel
P1 (Z11, Z12, ..., Z1k_1) $\wedge$ ... $\wedge$Pm (Zm1, Zm2, ..., Zmk_m)
als "Rumpf" (engl.: body).

Gegeben sei eine endliche Menge von Prädikaten.
Ein *Datalog-Programm* besteht aus
- einer Menge von Fakten der Form Pi (V1, V2, ..., Vk)
  mit einem k-stelligen Prädikat Pi und Konstanten V1, ..., Vk und
- einer Menge von Regeln in Form von Hornklauseln über den gegebenen Prädikaten.

Ein *Datalog-Programm mit Negation* besteht aus
- einer Menge von Fakten wie in einem einfachen Datalog-Programm und
- einer Menge von Regeln in Form von Klauseln vom selben Typ wie bei Datalog,
  bei denen jedoch Prädikate im Rumpf negiert sein können.

Eine *Anfrage* (Ziel; engl: goal) für ein Datalog-Programm ist ein Ausdruck der Form P(Z1, ..., Zk) mit einem k-stelligen Prädikat P und Konstanten oder Variablen Z1, ..., Zk, so daß mindestens eines der Zj eine (freie) Variable ist.

Eine Regel r heißt *rekursiv*, wenn ihr Kopfprädikat auch in ihrem Rumpf vorkommt oder - allgemeiner – wenn es Regeln r1, r2, ..., rn gibt mit r1 = rn = r, so daß für alle i ein Rumpfprädikat von ri im Kopf von r(i+1) steht. Man nennt dann auch das Kopfprädikat von r rekursiv.

Eine rekursive Regel heißt *linear*, wenn das Kopfprädikat genau einmal unter den Rumpfprädikaten auftaucht und keines der anderen Rumpfprädikate als Kopfprädikat einer anderen rekursiven Regel auftritt.

Ein Datalog-Programm heißt linear, wenn es nur lineare oder nichtrekursive Regeln hat.

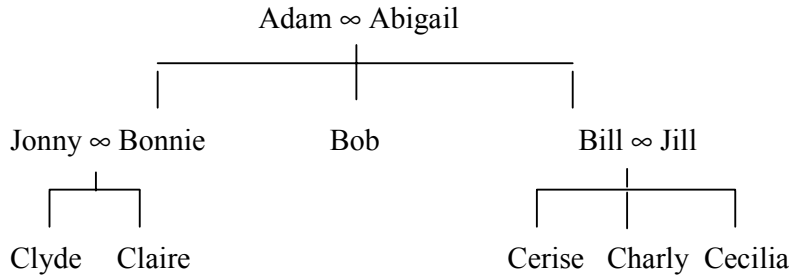
Satz:

Die Menge der Anfragen, die sich mit Datalog ohne Rekursion und ohne Negation ausdrücken lassen, ist eine echte Untermenge der Anfragen, die sich mit dem Domain-Relationenkalkül ausdrücken lassen.

Die Menge der Anfragen, die sich mit Datalog mit Negation, aber ohne Rekursion ausdrücken lassen, ist identisch mit der Menge der Anfragen, die sich mit dem Domain-Relationenkalkül ausdrücken lassen.

Die Menge der Anfragen, die sich mit Datalog mit Rekursion und mit Negation ausdrücken lassen, ist eine echte Obermenge der Menge von Anfragen, die sich mit dem Domain-Relationenkalkül ausdrücken lassen.

Beispiele:



Fakten

Mann (Adam), Mann (Jonny), Mann (Bob), Mann (Bill), Mann (Clyde) ,Mann (Charly)

Frau (Abigail), Frau (Bonnie), Frau (Jill), Frau (Claire), Frau (Cerise), Frau (Cecilia)

Ehepaar (Adam, Abigail), Ehepaar (Jonny, Bonnie), Ehepaar (Bill, Jill)

Elternteil (Adam, Bonnie), Elternteil (Adam, Bob), Elternteil (Adam, Bill),
Elternteil (Abigail, Bonnie), Elternteil (Abigail, Bob), Elternteil (Abigail, Bill),
Elternteil (Jonny, Clyde), Elternteil (Jonny, Claire),
Elternteil (Bonnie, Clyde), Elternteil (Bonnie, Claire),
Elternteil (Bill, Cerise), Elternteil (Bill, Charly), Elternteil (Bill, Cecilia),
Elternteil (Jill, Cerise), Elternteil (Jill, Charly), Elternteil (Jill, Cecilia)

SelbeGeneration (Adam, Abigail),
SelbeGeneration (Adam, Adam), SelbeGeneration (Abigail, Abigail)

SpieltMit (Clyde, Charly)

Regeln

Elternteil (X, Y) $\Rightarrow$ Vorfahren (X, Y)
Elternteil (X, Y) $\wedge$ Vorfahren (Y, Z) $\Rightarrow$ Vorfahren (X, Z)

Elternteil (X, Y) $\wedge$ Mann (X) $\Rightarrow$ Vater (X, Y)
Elternteil (X, Y) $\wedge$ Frau (X) $\Rightarrow$ Mutter (X, Y)

Elternteil (X, Y) $\wedge$ Elternteil (X, Z) $\wedge$ Y$\neq$Z $\Rightarrow$ Geschwister (Y, Z)
Elternteil (X, Y) $\wedge$ Elternteil (U, W) $\wedge$ Geschwister (X, U) $\wedge$ Frau (W) $\Rightarrow$ Cousine (Y, W)
Elternteil (X, Y) $\wedge$ Elternteil (U, W) $\wedge$ SelbeGeneration (X, U) $\Rightarrow$ SelbeGeneration (Y, W)

Geschwister (X, Y) $\Rightarrow$ SpieltMit (X, Y)
SpieltMit (Clyde, Y) $\Rightarrow$ SpieltMit (Claire, Y)
TRUE $\Rightarrow$ SpieltMit (Cecilia, Y) (leerer Rumpf)

Elternteil (X, Y) $\Rightarrow$ Verwandt (X, Y)
Geschwister (X, Y) $\Rightarrow$ Verwandt (X, Y)
Verwandt (X, Y) $\Rightarrow$ Verwandt (Y, X)
Verwandt (X, Y) $\wedge$ Verwandt (Y, Z) $\Rightarrow$ Verwandt (X, Z) (nichtlinear)

Mann (X) $\wedge$ $\neg$Ehepaar (X, Y) $\Rightarrow$ Ledig (X) (mit Negation)
Frau (Y) $\wedge$ $\neg$Ehepaar (X, Y) $\Rightarrow$ Ledig (Y) (mit Negation)

Beispiele für Anfragen

Ledig (X)
Cousine (Clyde, X)
SelbeGeneration (Clyde, X)

Bemerkung:

Im Gegensatz zu Prolog ist Datalog wirklich nichtprozedural (z.B. ist die Reihenfolge der Regeln irrelevant) und soll große, persistente Sammlungen von Fakten und Regeln verarbeiten.

Ergänzende Literatur zu Kapitel 4:

P. Kandzia, H.-J. Klein, Theoretische Grundlagen relationaler Datenbanksysteme, BI-Verlag, Reihe Informatik, Band 79, 1993

S. Abiteboul, R. Hull, V. Vianu, Foundation of Databases, Addison-Wesley, 1995

P.C. Kanellakis, Elements of Relational Database Theory, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Elsevier, Amsterdam, 1991

J.D. Ullman, Principles of Database and Knowledge-Based Systems Vol. 1 and Vol.2, Computer Science Press, 1989

S. Ceri, G. Gottlob, L. Tanca, Logic Programming and Databases, Springer-Verlag, 1990

A.B. Cremers, U. Griefahn, R. Hinze, Deduktive Datenbanken, Vieweg-Verlag, 1994