Kapitel 6:
Logikorientierte Anfragesprachen

6.1 Grundlagen aus der Logik

Logik ist eine wichtige Grundlage, um über die Semantik von Datenbankanfragesprachen (und später auch über die Semantik von Softwareentwurfssprachen) präzise reden zu können. In einer Logik werden zunächst einmal *Formeln* als syntaktische Gebilde definiert. Auf Formeln kann man dann *Ableitungs-, Beweis- oder Deduktionsregeln* definieren, mit denen man Formeln aus einer gegebenen Formelgrundmenge mechanisch herleiten kann; wir schreiben

$F_1,...,F_n \vdash G$ oder $\frac{F_1,...,F_n}{G}$ für die Ableitung von Formel G aus den Formeln F_1, ..., F_n.

Eine Menge von (schematisierten) Ableitungsregeln nennt man einen *Ableitungs-, Beweis- oder Deduktionskalkül*.
Dieser syntaktischen (*beweistheoretischen*) Seite einer Logik steht die (*modellorientierte*) *Semantik* von Formeln gegenüber. Dazu werden bestimmte Komponenten der Formeln, insbesondere Aussagen-, Funktions- und Prädikatsymbole, in einer (mathematischen) Struktur interpretiert (z.B. den natürlichen Zahlen mit Funktionen wie Addition, Multiplikation, usw. und Prädikaten wie IstPrimzahl oder SindTeilerfremd); auf dieser Basis erhalten ganze Formeln einen Wahrheitswert in der der *Interpretation* zugrundeliegenden Struktur. Wenn eine gegebene Formelmenge G unter einer bestimmten Interpretation ψ in einer Struktur S wahr ist, nennt man die Struktur ein *Modell* der Formelmenge, und wir schreiben $\psi \models G$ oder $\models_\psi G$.

Zwischen syntaktischer Ableitung und semantischer Interpretation besteht typischerweise ein Zusammenhang: idealerweise wünscht man sich, daß für eine gegebene Grundformelmenge G und eine Struktur S, die ein Modell von G ist, jede mit einem Deduktionskalkül ableitbare Formel in S wahr ist (*Korrektheit des Kalküls*) und umgekehrt jede in S wahre Formel aus G syntaktisch ableitbar ist (*Vollständigkeit des Kalküls*). Für bestimmte Strukturen gibt es korrekte und vollständige Kalküle, für andere nicht. Beispielsweise besagt der berühmte Satz von Gödel, daß es für die Struktur der arithmetischen Gleichungen über den natürlichen Zahlen mit den Funktionen Addition und Multiplikation und darauf aufgebauten prädikatenlogischen Formeln keinen vollständigen Ableitungskalkül geben kann.

6.1.1 Aussagenlogik

Definition:
Eine atomare Formel der Aussagenlogik ist eine Aussagenkonstante True oder False oder eine Aussagevariable der Form Ai (i=1, 2, ...).
Die Menge der Formeln der Aussagenlogik ist induktiv wie folgt definiert:

(i) Jede atomare Formel ist eine Formel.

(ii) Wenn F und G Formeln sind, dann auch (F), $\neg$F, F$\wedge$G, F$\vee$G sowie F$\Rightarrow$G als Kurzschreibweise für $\neg$F$\vee$G und F$\Leftrightarrow$G als Kurzschreibweise für (F$\wedge$G)$\vee$($\neg$F$\wedge\neg$G) Formeln. Statt $\neg$F schreibt man häufig auch $\overline{F}$.

Die Formeln der Aussagenlogik sind also Aussageformen über Aussagevariablen.
Um die syntaktische Analyse von Ausdrücken eindeutig zu machen, kann man entweder vollständige Klammerung verlangen (d.h. in (ii) nur (F), ($\neg$F), (F$\wedge$G), (F$\vee$G), (F$\Rightarrow$G), (F$\Leftrightarrow$G) zulassen) oder Präzedenzen festlegen, daß $\neg$ stärker bindet als $\wedge$ oder $\vee$ und diese wiederum stärker als $\Rightarrow$ oder $\Leftrightarrow$.

Definition:
Sei F eine Formel der Aussagenlogik mit atomarer Formelmenge D. Eine *passende Struktur* S zu F ist eine Menge P von Aussagen (die jeweils wahr oder falsch sind), so daß jede Aussagenvariable in D einer dieser Aussagen zugeordnet werden kann. P enthält ggf. die immer wahre Aussage 1 und die immer falsche Aussage 0. Eine *Interpretation* von F ist eine Abbildung ψ: D $\rightarrow$ P, für die gilt ψ(True)=1, ψ(False)=0, ψ(Ai)=1 bzw. 0, falls ψ(Ai) in S wahr bzw. falsch ist.
ψ wird wie wie folgt auf beliebige Formeln F, G, usw. über D fortgesetzt:
(i) ψ((F)) = ψ(F); (ii) ψ(F$\wedge$G)= 1 falls ψ(F)= ψ(G)=1, 0 sonst;
(iii) ψ(F$\vee$G)= 1 falls ψ(F)= 1 oder ψ(G)=1, 0 sonst; (iv) ψ($\neg$F)=1 falls ψ(F)=0, 0 sonst.

Eine Interpretation einer aussagenlogischen Formel ist also im wesentlichen eine Belegung der Aussagevariablen mit Wahrheitswerten.

Definition:
Sei F eine Formel und ψ eine Interpretation von F.
Wenn ψ(F)=1 ist, heißt ψ *Modell* von F. Wir schreiben dann ψ |= F (oder $|=_\psi$ F).
F heißt *erfüllbar*, falls F mindestens ein Modell hat, ansonsten *unerfüllbar*.
F heißt *(allgemein-)gültig* oder *Tautologie*, falls jede Interpretation in einer zu F passenden Struktur ein Modell von F ist.

Satz:
F ist Tautologie genau dann, wenn $\neg$F unerfüllbar ist.

Definition:
G heißt *Folgerung* (Konsequenz) von F1, ..., Fk, geschrieben F1, ..., Fk |= G, wenn für jede Interpretation ψ in einer passenden Struktur gilt: falls ψ ein Modell von F1, ..., Fk ist, dann ist ψ auch ein Modell von G. F und G heißen (semantisch) *äquivalent*, geschrieben F $\equiv$ G, falls für jede Interpretation ψ gilt: ψ(F)= ψ(G).

Satz:
G ist Folgerung von F1, ..., Fk genau dann, wenn (F1 $\wedge$ F2 $\wedge$ ... $\wedge$ Fk) $\Rightarrow$ G eine Tautologie ist.
F und G sind äquivalent genau dann, wenn F Folgerung von G ist und umgekehrt.

Beispiele:

1) Formeln über Variablen T, P:

F1: $T \Rightarrow \neg P$, F2: $P \wedge \neg T$ bzw. zusammengefaßt

F: $(T \Rightarrow \neg P) \wedge (P \wedge \neg T)$

Interpretation: $\psi(T)$ = „7 ist durch 2 teilbar“ (falsch), $\psi(P)$ = „7 ist eine Primzahl“ (wahr)

ψ ist ein Modell von F1 und F2 bzw. von F.

2) Formeln über Variablen S, R, A:

F1: $S \Rightarrow \neg R$, F2: $\neg R \Rightarrow S$, F3: $A \Rightarrow \neg S$, F4: A bzw. zusammengefaßt

F: $(S \Rightarrow \neg R) \wedge (\neg R \Rightarrow S) \wedge (A \Rightarrow \neg S) \wedge (A)$

Interpretation:

$\psi(S)$ = „die Sonne scheint“ (falsch) , $\psi(R)$ = „es regnet“ (wahr), $\psi(A)$ = „es ist April“ (wahr).

ψ ist ein Modell von F1, F2, F3, F4 bzw. von F.

3) $\neg(\neg F) \Leftrightarrow F$ ist eine Tautologie;

$(X \wedge Y) \vee (\neg X \wedge Z)$ ist erfüllbar (z.B. mit X: „dieser Raum ist nichtleer“, Y: „das Licht ist an“ und Z: „das Licht ist aus“), aber keine Tautologie;

$F \wedge \neg F$ ist unerfüllbar.

Ein **Deduktionskalkül** ist eine endliche Menge von Ableitungsregeln der Form $P \vdash K$, wobei P eine endliche Formelmenge, die Menge der Prämissen (oder Hypothesen), ist und K eine Formel, die Konklusion oder Schlussfolgerung. Ein einfacher **Deduktionskalkül für die Aussagenlogik** besteht (z.B.) aus 5 Regeln:

(i)		$\vdash F \Rightarrow (F \Rightarrow F)$	
(ii)		$\vdash (F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$	
(iii)		$\vdash (\neg F \Rightarrow \neg G) \Rightarrow (G \Rightarrow F)$	
(iv)	$F \Leftrightarrow G, H$	$\vdash H[F/G]$	(Ersetzungsregel, Substitution)
(v)	$F \Rightarrow G, F$	$\vdash G$	(Abtrennungsregel, Modus Ponens)

wobei H[F/G] die Formel ist, die man aus H durch syntaktische Substitution von F durch G erhält.

Definition:

Sei H eine endliche Formelmenge – die Hypothesenmenge (z.B. Axiome einer spezifischen Struktur). Die Formel F heißt aus H ableitbar, in Zeichen: $H \vdash F$, wenn es eine endliche Sequenz F0, F1, ..., Fn von Formeln gibt mit Fn=F, so daß für alle Fi gilt: Fi ist Element von H, Fi ist eine der prämissenlosen Formeln (i) bis (iii) des Deduktionskalküls oder Fi ist aus F0, ..., F(i-1) mittels Regel (iv) oder (v) des Deduktionskalküls abgeleitet.

Für $H=\varnothing$ heißt F Theorem des Deduktionskalküls.

Satz:

Der angegebene Deduktionskalkül mit den Regeln (i) bis (v) ist korrekt und vollständig, d.h.

$H \vdash F$ genau dann, wenn $H \models F$

(d.h. $H \Rightarrow F$ ist eine Tautologie bzw. F ist wahr in jeder Struktur, in der H wahr ist)

und – als Sonderfall mit $H=\varnothing$: $\vdash F$ genau dann, wenn $\models F$

(d.h. F ist eine Tautologie bzw. F ist in jeder passenden Struktur wahr)

Beweis: durch Induktion über den Aufbau von F und entsprechende Fallunterscheidungen

Definition:
Eine (endliche) Formelmenge H heißt Axiomensystem für eine Formelmenge M (z.B. alle wahren Theoreme einer mathemat. Struktur), wenn: $\{\psi \mid \psi$ ist Modell von H$\} = \{\psi \mid \psi$ ist Modell von M$\}$.

Alternative Deduktionskalküle könnten z.B. die Brute-Force-Methode verwenden, Wahrheitstafeln für $H \Rightarrow F$ aufzustellen (mit allen Kombinationen möglicher Wahrheitswerte für die Aussagenvariablen und alle Teilformeln), oder eine geeignete Teilmenge der folgenden (aus dem angegebenen Kalkül ableitbaren und so beweisbaren) Äquivalenzen und der Deduktionsregeln (i) bis (v) verwenden:

$\neg\neg F \Leftrightarrow F$	(Doppelnegation)
$F \wedge F \Leftrightarrow F$	(Idempotenz)
$F \vee F \Leftrightarrow F$	
$F \wedge G \Leftrightarrow G \wedge F$	(Kommutativität)
$F \vee G \Leftrightarrow G \vee F$	
$F \wedge (G \wedge H) \Leftrightarrow (F \wedge G) \wedge H$	(Assoziativität)
$F \vee (G \vee H) \Leftrightarrow (F \vee G) \vee H$	
$F \wedge (F \vee G) \Leftrightarrow F$	(Absorption)
$F \vee (F \wedge G) \Leftrightarrow F$	
$F \wedge (G \vee H) \Leftrightarrow (F \wedge G) \vee (F \wedge H)$	(Distributivität)
$F \vee (G \wedge H) \Leftrightarrow (F \vee G) \wedge (F \vee H)$	
$\neg (F \wedge G) \Leftrightarrow (\neg F \vee \neg G)$	(Gesetz von de Morgan)
$\neg (F \vee G) \Leftrightarrow (\neg F \wedge \neg G)$	
$F \vee 1 \Leftrightarrow 1$	(Tautologieregeln)
$F \wedge 1 \Leftrightarrow F$	
$F \vee 0 \Leftrightarrow F$	(Unerfüllbarkeitsregeln)
$F \wedge 0 \Leftrightarrow 0$	
$F \vee (\neg F) \Leftrightarrow 1$	(Tertium non datur)
$F \wedge (\neg F) \Leftrightarrow 0$	(Kontradiktion)
$((F \Leftrightarrow G) \wedge H) \Leftrightarrow ((F \Leftrightarrow G) \wedge H[F/G])$	(Substitution)

Beispiel:
Auf die Frage „Worin besteht das Geheimnis Ihres Lebens?“ antwortet ein Hundertjähriger:
Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.
Immer wenn ich Fisch und Bier zur selben Mahlzeit habe, verzichte ich auf Eiscreme.
Wenn ich Eiscreme habe oder Bier meide, dann rühre ich Fisch nicht an.

Modellierung:
Formel G: $(\neg B \Rightarrow F) \wedge ((F \wedge B) \Rightarrow \neg E) \wedge ((E \vee \neg B) \Rightarrow \neg F)$

Vereinfachung, sprich Deduktion (mit Unterstreichung als Negation):

$G \Leftrightarrow (B \vee F) \wedge (\underline{F} \wedge \underline{B} \vee \underline{E}) \wedge ((\underline{E} \vee \neg \underline{B}) \vee \underline{F})$	nach Definition der Implikation
$\Leftrightarrow (B \vee F) \wedge (\underline{F} \vee \underline{B} \vee \underline{E}) \wedge ((\underline{E} \wedge B) \vee \underline{F})$	nach dem Gesetz von de Morgan
$\Leftrightarrow (B \vee F) \wedge (\underline{F} \vee \underline{B} \vee \underline{E}) \wedge (\underline{E} \vee \underline{F}) \wedge (B \vee \underline{F})$	nach dem Distributivitätsgesetz
$\Leftrightarrow ((B \vee F) \wedge (B \vee \underline{F})) \wedge ((\underline{F} \vee \underline{B} \vee \underline{E}) \wedge (\underline{E} \vee \underline{F}))$	wegen Kommutativität und Assoziativität
$\Leftrightarrow ((B \vee (F \wedge \underline{F})) \wedge ((\underline{F} \vee \underline{E}) \vee (\underline{B} \wedge 0))$	nach Distributivitätsgesetz und Tautologieregel
$\Leftrightarrow ((B \vee 0) \wedge ((\underline{F} \vee \underline{E}) \vee 0)$	wegen Kontradiktion und Unerfüllbarkeit
$\Leftrightarrow B \wedge (\underline{F} \vee \underline{E})$	wegen Unerfüllbarkeitsregel

Interpretation: Trinke zu jeder Mahlzeit Bier, und iss niemals Fisch mit Eiscreme!

3.2 Prädikatenlogik 1. Ordnung

Motivation: Man möchte kompliziertere Zusammenhänge in einer formalen Logik präzisieren können, die in der Aussagenlogik nicht ausdrückbar sind.
Beispiele:
1) Es gibt unendlich viele Primzahlen:
   $\forall q\ \exists p\ \forall x,y\ (\ p>q \wedge ((x>1 \wedge y>1) \Rightarrow (x \cdot y \neq p))\ )$ bzw.
   $\forall q\ \exists p\ \forall x,y\ (\text{greater}(p,q) \wedge ((\text{greater}(x,1) \wedge \text{greater}(y,1)) \Rightarrow (\neg\ \text{equal}(\text{mult}(x,y),p)))\ )$
2) Das kgV zweier teilerfremder Primzahlen ist ihr Produkt:
   $\forall x,y\ (\text{ggT}(x,y)=1 \Rightarrow \text{kgV}(x,y)=x \cdot y)$
3) Invarianten von Programmen (mit dem Gesamtziel der Programmverifikation)
4) Datenbankanfragen (siehe Kapitel 4) und Datenbankinvarianten (siehe Kapitel 7)

Definition:
Gegeben seien Variablen xi (i=1,2, ...), Prädikatsymbole Pi der Stelligkeit ki (i=1,2, ...), d.h. Prädikatsymbole mit ki Argumenten, und Funktionssymbole fi der Stelligkeit li(i=1,2, ...).
0-stellige Prädikatsymbole sind Aussagen, 0-stellige Funktionssymbole sind Konstanten.
Terme sind induktiv definiert:

(i) Jede Variable ist ein Term.
(ii) Wenn t1, ..., tk Terme sind und fi ein k-stelliges Funktionssymbol ist, dann ist auch fi(t1, ..., tk) ein Term.

Eine *atomare Formel* hat die Form Pi(t1, ..., tk) mit einem k-stelligen Prädikatsymbol Pi und Termen t1, ..., tk.
Formeln der Prädikatenlogik 1. Ordnung sind induktiv definiert:

(i) Jede atomare Formel ist eine Formel.
(ii) Für Formeln F und G sind auch $\neg$F, F$\wedge$G, F$\vee$G und (F) Formeln.
(iii) Für eine Variable xi und eine Formel F sind auch $\forall$xi (F) und $\exists$xi (F) Formeln.
$\forall$ und $\exists$ heißen All- bzw. Existenzquantor, xi ist eine quantifizierte (gebundene) Variable, und F ist der Rumpf der Formel.

Zur eindeutigen Syntaxanalyse kann man entweder vollständige Klammerung verlangen oder Präzedenzen festlegen, so daß Junktoren ($\neg$, $\wedge$, $\vee$) stärker binden als Quantoren und die Präzedenzen der Aussagenlogik gelten. Statt $\forall x\ \forall y\ \forall z$ ... schreiben wir auch kurz $\forall x,y,z$, und Analoges gilt für $\exists$.

Eine Variable x in einer Formel F heißt *gebunden*, wenn x in einer Teilformel der Form $\forall$x (F) oder $\exists$x (F) vorkommt; ansonsten heißt x *frei*. Formeln ohne freie Variablen heißen geschlossen.
Beispiel: In $\forall x\ (P(x,y) \wedge \exists z\ (Q(z,x)))$ sind x und z gebunden, und y ist frei.

Als notationelle Konvention für Formeln verlangen wir, daß keine Variable an mehr als einen Quantor gebunden wird und daß keine Variable sowohl frei als auch gebunden vorkommt. Dies ist durch einfache Umbenennung von Variablen (Bereinigung) immer möglich.
Zur einfacheren Lesbarkeit werden häufig stilistische Konventionen für Bezeichner eingeführt: Variablen werden mit x, y, z, u, v, w, ... bezeichnet, Konstanten mit a, b, c, ..., Funktionssymbole der Stelligkeit ≥ 1 mit f, g, h, ... und Prädikatsymbole mit P, Q, R, S, ...

Semantik von prädikatenlogischen Formeln

Definition:
Gegeben sei eine Menge von Funktions- und Prädikatsymbolen (z.B. alle in einer Formel oder Formelmenge vorkommenden Symbole) mit entsprechenden Stelligkeiten. Eine dazu passende *Struktur* ist ein Tripel S = (U, fun, pred) mit

- einem *Universum* (einer *Trägermenge*) U von Individuen (z.B. ganzen Zahlen oder Zeichenketten),
- einer Menge fun von *Funktionen* fi: $U \times ... \times U \rightarrow U$ der Stelligkeit li, so daß jedes Funktionssymbol der Stelligkeit li eine Funktion derselben Stelligkeit existiert,
- einer Menge pred von *Prädikaten* Pi: $U \times ... \times U \rightarrow \{0,1\}$ der Stelligkeit ki, so daß jedes Prädikatsymbol der Stelligkeit ki ein Prädikat derselben Stelligkeit existiert.

Die *Interpretation* ψ einer Formel F in einer passenden Struktur S = (U, fun, pred) ist eine Abbildung, die das Universum U festlegt und jedem Funktions- und Prädikatsymbol in F eine passende Funktion aus fun bzw. ein passendes Prädikat aus pred zuordnet. Dies ist also eine Abbildung ψ von atomaren Formeln ohne Variablen als Argumenten auf wahre oder falsche Prädikate in der gegebenen Struktur.
ψ wird wie wie folgt auf beliebige Formeln F, G, usw. und Terme über denselben Funktions- und Prädikatsymbolen fortgesetzt:

(i) $\psi((F)) = \psi(F)$; (ii) $\psi(F \wedge G) = 1$ falls $\psi(F) = \psi(G) = 1$, 0 sonst;
(iii) $\psi(F \vee G) = 1$ falls $\psi(F) = 1$ oder $\psi(G) = 1$, 0 sonst; (iv) $\psi(\neg F) = 1$ falls $\psi(F) = 0$, 0 sonst,
(v) $\psi(f(t1, ..., tk)) = \psi(f)(\psi(t1), ..., \psi(tk))$ für ein Funktionssymbol f und Terme t1, ..., tk,
(vi) $\psi(x) = a$ mit einem beliebigen $a \in U$ für eine (freie) Variable x
(vii) $\psi(P(t1, ..., tk)) = \psi(P)(\psi(t1), ..., \psi(tk))$ für ein Prädikatsymbol P und Terme t1, ..., tk,
(viii) $\psi(\forall x (F)) = 1$ falls $\psi(F[x/a]) = 1$ für alle $a \in U$, wobei F[x/a] die Formel ist, die man aus F erhält, wenn man alle Vorkommen von x durch a ersetzt, und
(ix) $\psi(\exists x (F)) = 1$ falls $\psi(F[x/a]) = 1$ für ein beliebiges $a \in U$.

Beispiel 1:
F: $\forall q \exists p \forall x,y$ (greater(p,q) $\wedge$ ((greater(x,1) $\wedge$ greater(y,1)) $\Rightarrow$ ($\neg$ equal(mult(x,y),p))))
Eine passende Struktur ist z.B. S = (N_0, {+, ·}, {>, =}).
Interpretation ψ:
add $\mapsto$ +, mult $\mapsto$ ·, greater $\mapsto$ >, equal $\mapsto$ =, q $\mapsto$ 1, p $\mapsto$ 2, x $\mapsto$ 3, y $\mapsto$ 4
ψ (F) = für jede natürliche Zahl q gibt es eine natürlich Zahl p, so daß für alle Zahlen x und y gilt:
p ist größer als q und falls x und y beide größer als 1 sind, ist x·y von p verschieden
(oder einfacher: zu jeder Zahl gibt es eine Primzahl, die größer ist als die Zahl)

Beispiel 2:
F: K (1, Lauer, Merzig) $\wedge$ K(2, Schneider, Homburg) $\wedge$ P(1, Papier) $\wedge$ B(7, 16, 1, 1, 100)
mit Prädikatsymbolen K, P und B und Konstanten 1, Lauer, usw.
Eine passende Struktur ist z.B. eine Datenbank mit dem folgenden Datenbankschema (d.h. eine Menge von Relationen über passenden kartesischen Produkten des Universums $N_0 \cup \Sigma^*$, der Menge aller natürlichen Zahlen und Zeichenketten über einem Standardalphabet Σ):
Kunden (KNr, Name, Stadt), Produkte (PNr, Bez), Bestellungen (Monat, Tag, KNr, PNr, Menge).
ψ (F) = 1, falls es die entsprechenden vier Tupel in der Datenbank gibt, 0 sonst.

Erweiterungen der Prädikatenlogik 1. Ordnung:

Eine Variation der Prädikatenlogik 1. Ordnung ist die *mehrsortige (typisierte) Variante*, bei der in einer passenden Struktur das Universum aus einer endlichen Menge von *Trägermengen (Sorten)* U1, ..., Um besteht und Funktionen und Prädikate mehrsortig sind mit einer *Signatur*, die neben der Stelligkeit die Sorten der Funktions- und Prädikatargumente festlegt. Eine solche Struktur nennt man auch eine mehrsortige Algebra.

Beispiel: U= {U1, U2} mit U1=R^n, U2=R und

Funktionen Addition: U1 × U1 → U1, Multiplikation: U1 × U2 → U1,
Skalarprodukt: U1 × U1 → U2, Vektorprodukt: U1 × U1 → U1, Nullvektor: → U1 sowie
Prädikaten Orthogonal: U1 × U1 → {0,1} usw.

Der Übergang zu einer mehrsortigen Logik verändert nicht die Ausdruckskraft der Prädikatenlogik 1. Ordnung. Man gewinnt jedoch an Ausdruckskraft, wenn man Prädikate schachteln kann, also Prädikate selbst Prädikate als Argument haben können, oder wenn man nicht nur Individuen, sondern auch Prädikate quantifizieren (also an Quantoren binden) darf. Solche Erweiterungen führen auf die *Prädikatenlogik höherer Ordnung*, die wir hier nicht weiter betrachten.

Ein Beispiel einer Formel höherer Ordnung ist die folgende Spezifikation der transitiven Hülle H einer binären Relation R:

$$\forall x,y\ (\ H(x,y) \Leftrightarrow \forall P\ (\ (R(x,y) \Rightarrow P(x,y)) \wedge \forall u,w,z\ (P(u,w) \wedge P(w,z) \Rightarrow P(u,z)))$$
$$\Rightarrow (H(x,y) \Rightarrow P(x,y))\ )$$

Intuitive Interpretation :
H ist kleinste binäre Relation, die R enthält und transitiv abgeschlossen ist.
Also gilt für jede binäre Relationen P, dass sie, wenn sie R enthält und transitiv abgeschlossen ist, eine Obermenge von H sein muss.

Erfüllbarkeit, Unerfüllbarkeit, Gültigkeit, Modell

Definition:
Sei F eine Formel der Prädikatenlogik 1. Ordnung und ψ eine Interpretation von F.
Wenn $\psi(F)=1$ ist, heißt ψ *Modell* von F. Wir schreiben dann $\psi \models F$ (oder $\models_\psi F$).
F heißt *erfüllbar*, falls F mindestens ein Modell hat, ansonsten *unerfüllbar*.
F heißt *(allgemein-)gültig* oder *Tautologie*, falls jede Interpretation in einer zu F passenden Struktur ein Modell von F ist.

Satz:
F ist Tautologie genau dann, wenn $\neg$F unerfüllbar ist.

Definition:
G heißt *Folgerung* (Konsequenz) von F1, ..., Fk, geschrieben F1, ..., Fk $\models$ G, wenn für jede Interpretation ψ in einer passenden Struktur gilt: falls ψ ein Modell von F1, ..., Fk ist, dann ist ψ auch ein Modell von G. F und G heißen (semantisch) *äquivalent*, geschrieben F $\equiv$ G, falls für jede Interpretation ψ gilt: $\psi(F)= \psi(G)$.

Satz:
G ist Folgerung von F1, ..., Fk genau dann, wenn $(F1 \wedge F2 \wedge ... \wedge Fk) \Rightarrow G$ eine Tautologie ist.
F und G sind äquivalent genau dann, wenn F Folgerung von G ist und umgekehrt.

Definition:
Eine Formelmenge H heißt ein Axiomensystem für eine Formelmenge M (z.B. alle wahren Theoreme einer mathematischen Struktur), falls gilt: $\{\psi \mid \psi$ ist Modell von H$\} = \{\psi \mid \psi$ ist Modell von M$\}$.

Beispiel:
Formeln F1, ..., F5 bzw. Formel $F= F1 \wedge ... \wedge F5$:
$F1 = \forall x,y,z \ (G(f(f(x,y),z), f(x,f(y,z))))$
$F2 = \forall x \ (G(f(x,e),x))$
$F3 = \forall x,y \ (G(x, y) \Leftrightarrow G(y, x))$
$F4 = \forall x \ \forall y \ \forall z \ (G(x,y) \wedge G(y,z) \Rightarrow G(x,z))$
$F5 = \forall x \ \exists y \ (G(f(x,y), e))$

Interpretation A (Modell):
U = Z (Menge der ganzen Zahlen), ψ(G) = Gleichheit, ψ(f) = Addition, ψ(e) = 0, ψ(F)=1 (wahr)
Interpretation B (kein Modell):
U = Z (Menge der ganzen Zahlen), ψ(G) = Gleichheit, ψ(f) = Multiplikation, ψ(e) = 1, ψ(F)=0
Interpretation C (Modell):
U = R (Menge der reellen Zahlen), ψ(G) = Gleichheit, ψ(f) = Multiplikation, ψ(e) = 1, ψ(F)=1
Interpretation D (kein Modell):
$U = N_0$ (Menge der natürlichen Zahlen), $\psi(G) = \geq$, ψ(f) = Addition, ψ(e) = 1, ψ(F)=0
Interpretation E (kein Modell):
$U = \Sigma^*$, ψ(G) = Gleichheit der Länge, ψ(f) = Stringkonkatenation, $\psi(e) = \varepsilon$, ψ(F)=0

Deduktionskalküle

Es gibt Deduktionskalküle für die Prädikatenlogik 1. Ordnung, die korrekt und vollständig sind, z.B. den Resolutionskalkül oder den Tableaukalkül. Diese spielen z.B. in der Logikprogrammierung (z.B. in der Programmiersprache Prolog) oder beim automatischen Theorembeweisen eine Rolle. Es handelt sich jedoch nur um sog. Semientscheidungsverfahren: bei einer semantischen Tautologie als Eingabe terminiert das Ableitungsverfahren mit dem Resultat "gültig" (rein technisch beweist der Resolutionskalkül die Unerfüllbarkeit der negierten Eingabe), bei einer Nichttautologie (deren Negation erfüllbar ist) aber muß das Verfahren nicht terminieren.

Mittels dieser Kalküle (oder auch auf anderem Wege) kann man die Gültigkeit der folgenden - praktisch sehr nützlichen - Umformungsregeln beweisen:

alle aussagenlogischen Äquivalenzen
Wenn $F \Leftrightarrow G$ und F als Teilformel in H vorkommt, dann ist $H \Leftrightarrow H[F/G]$
$\neg \forall(F) \Leftrightarrow \exists x (\neg F)$
$\neg \exists (F) \Leftrightarrow \forall x (\neg F)$
$(\forall x (F)) \wedge G \Leftrightarrow \forall x (F \wedge G)$ falls x in G nicht frei vorkommt
$(\forall x (F)) \vee G \Leftrightarrow \forall x (F \vee G)$ falls x in G nicht frei vorkommt
$(\exists x (F)) \wedge G \Leftrightarrow \exists x (F \wedge G)$ falls x in G nicht frei vorkommt
$(\exists x (F)) \vee G \Leftrightarrow \exists x (F \vee G)$ falls x in G nicht frei vorkommt
$(\forall x (F)) \wedge (\forall y (G)) \Leftrightarrow \forall x (F \wedge G[y/x])$, falls x in G nicht frei vorkommt
$(\exists x (F)) \vee (\exists y (G)) \Leftrightarrow \exists x (F \vee G[y/x])$, falls x in G nicht frei vorkommt
$\forall x (\forall y (F)) \Leftrightarrow \forall y (\forall x (F))$
$\exists x (\exists y (F)) \Leftrightarrow \exists y (\exists x (F))$
$\forall x (F) \Rightarrow F[x/a]$ für alle Konstanten a
$((\forall x (F)) \wedge (F \Rightarrow G)) \Rightarrow (\forall x (G))$

Achtung: Es gelten jedoch *nicht*
$\exists x (\forall y (F)) \Leftrightarrow \forall y (\exists x (F))$
$(\forall x (F)) \vee (\forall y (G)) \Leftrightarrow \forall x (F \vee G[y/x])$
$(\exists x (F)) \wedge (\exists y (G)) \Leftrightarrow \exists x (F \wedge G[y/x])$

Beispiel:

Folgende Zusammenhänge seien wahr:

i) Informatikstudenten können programmieren.
ii) Studenten mit guten Mathematiknoten studieren Informatik.
iii) Studenten, die mit einem Informatiker (jemandem, der Informatik studiert) verwandt sind, haben gute Mathematiknoten.
iv) Alle saarländischen Studenten sind mit Heinz Becker verwandt.
v) Die Verwandtschaftsbeziehung ist symmetrisch.
vi) Heinz Becker hat Informatik studiert.

Modellierung:

Universum: Menge aller Studenten
Prädikate:
KannProgrammieren P(x)
StudiertInformatik I(x)
HatGuteMathenoten M(x)
SindVerwandt V(x,y)
IstSaarländer S(x)
Funktionen und Konstanten:
HeinzBecker hb()
Formeln:

i) $\forall x (I(x) \Rightarrow P(x)) \wedge$
ii) $\forall x (M(x) \Rightarrow I(x)) \wedge$
iii) $\forall x \ \forall y ((V(x,y) \wedge I(y)) \Rightarrow M(x)) \wedge$
iv) $\forall x (S(x) \Rightarrow V(x,hb)) \wedge$
v) $\forall x \ \forall y ((V(x,y) \Leftrightarrow V(y,x)) \wedge$
vi) $I(hb)$

Vereinfachung, sprich Deduktion:

(iii) $\wedge$ (iv) $\wedge$ (vi):

$\forall x \ \forall y (I(hb) \wedge (S(x) \Rightarrow V(x,hb)) \wedge (V(x,y) \wedge I(y)) \Rightarrow M(x)))$
$\vdash \ \forall x \ (I(hb) \wedge (S(x) \Rightarrow V(x,hb)) \wedge (V(x,hb) \wedge I(hb)) \Rightarrow M(x)) \ )$
$\vdash \ \forall x \ (S(x) \Rightarrow M(x)) \ (*)$

(*) $\wedge$ (ii) $\wedge$ (i):

$\forall x \ ((S(x) \Rightarrow M(x)) \ \wedge \ (M(x) \Rightarrow I(x)) \ \wedge \ (I(x) \Rightarrow P(x)))$
$\vdash \ \forall x \ (S(x) \Rightarrow P(x))$

Fazit: alle Saarländer können programmieren!

6.2 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	

6.2 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 $r.A\ \theta\ s.B$ und $r.A\ \theta\ c$ 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 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.
   $\rightarrow \{t \mid t \in Kunden \wedge t.Saldo < 0.0\}$ bzw. $\{t.Name \mid t \in Kunden \wedge t.Saldo < 0.0\}$

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

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.
   $\rightarrow \{k.Name, p.Bez \mid k \in Kunden \wedge p \in Produkte \wedge$
   $\exists\, b: b \in Bestellungen \wedge$
   $k.Stadt='Homburg' \wedge b.Monat \geq 9 \wedge p.Lagerort='Homburg' \wedge$
   $k.KNr=b.KNr \wedge b.PNr=p.PNr\}$

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

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

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

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.

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 | 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. 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: $<a,z> \in$ Flüge $\Rightarrow$ $<a,z> \in$ Flugverbindungen
$\forall$ a, z, o: $<a,o> \in$ Flugverbindungen $\wedge$ $<o,z> \in$ Flüge $\Rightarrow$ $<a,z> \in$ 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.

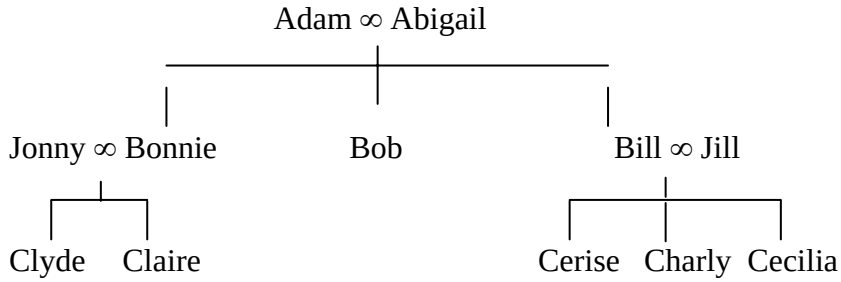
Satz:

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 6:

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
Uwe Schöning: Logik für Informatiker, Spektrum Akademischer Verlag, 1995