Datenbanktheorie SS 16

Deduktive Datenbanksysteme

Problem: Transitiver Abschluss ist in PL1 nicht formulierbar (mit zustandsabhängiger Formulierung möglich)
Diskussion:
Typ 5: $q_1(...), ..., q_n(...) : -.$
Typ 6: $q_1(...), ..., q_n(...) : -p_1(...), ..., p_m(...).$
$\Rightarrow$ Übungsaufgabe

Nur Typ 1 und Typ 4: $q(...) : -p_1(...), ..., p_n(...), n \geq 0$ (ist die Hornklauselform und wird bei definiten Datenbanken genutzt)

Definite Datenbanken

$$\begin{aligned} &q(...) : -.(\text{Fakt}) \\ &q(...) : -p_1(...), ..., p_n(...).(\text{deduktive Regel, } p_{1-n} \text{ Teilziele}) \end{aligned} \tag{1}$$

- Mit IBen (Integritätsbedingungen) (+ Typ2, Typ3) (: $-p_1(...), ..., p_n(...)$)
- Typ5 + Typ6 $\Rightarrow$ **Disjunktive Datenbank**
- Definite Datenbank + negative Atome im Rumpf von Hornklauseln erlaubt $\Rightarrow$ Volles Datalog

Formulierung von Anfragen

Klauseln vom Typ: : $-p_1(...), ..., p_n(...)$, geschrieben $? - p_1(...), ..., p_n(...)$

Beispiele:

- $? - ag(X, m).$
  - Bedeutung: Welche Kurse bietet 'm' an?
  - DRC: (x) / ANGEBOT(X, m)
- $? - ag(a3, m).$
  - Bedeutung: Bietet 'm' den Kurs 'a3' an?
  - DRC: () / ANGEBOT(a3, m)
- $? - ag(X, m), bl(X, s, j).$

  - Bedeutung: Gib alle von 'm' angebotene Kurse, die 's' als Wiederholer belegt hat
  - DRC: (x) / ANGEBOT(x, 'm') $\wedge$ BELEGUNG(x, 's', 'y')

- Wie ist (x) / ANGEBOT(x, 'm') $\wedge(\exists$y) BELEGUNG(x, 's', y) formulierbar?
  - Bedeutung: Gib die Dozenten der von s als Wiederholer belegte Kurse
  - Formulierung: $?-Ksm(X).$
    $Ksm(X):-ag(X,m),bl(X,s,y)$
  - Bequemer: $?-ag(X,Y*),bl(X,s,y).$, * Kennzeichnet die Ausgabevariable

In Anfragesprachen werden Vergleichsausdrücke benötigt. Dazu sind in Datalog spezielle vordefinierte Prädikate vorhanden. Für jeden Vergleichsoperator wird die Existenz eines solchen Prädikates angenommen.
Zunächst: Beschränkte Variablen in Regeln. Sei eine Regel r gegeben:

- Jede Variable, die als Argument in einem gewöhnlichen Prädikat im Rumpf von r vorkommt ist beschränkt.
- Jede Variable, die in einem Teilziel $X = c$ oder $c = X$ von r vorkommt, ist beschränkt.
- Eine Variable X ist beschränkt, wenn sie in einem Teilziel $X = Y$ oder $Y = X$ von r vorkommt mit Y ist schon als beschränkt bekannt.

Definition: sicher

Eine Regel heißt sicher, wenn alle in ihr vorkommenden Variablen beschränkt sind.

Beispiele:

- $Kls(X,Y):-bl(Z,s,j),ag(Z,Y),X=Z.$ **sicher**
- $vsj(X,Y):-bl(Y,s,j).$ **nicht sicher** (X ist nicht beschränkt)
- $vs(X,Y):-vs(X,Z),kp(Z,Y).$ **sicher, wenn vs terminiert**
- $kla(Z,Y):-bl(Z,V,j),ag(Z,Y),V\neq s.$ **sicher**

Bemerkung: Falls keine Build-in Prädikate erlaubt sind (/vorkommen):
Eine Regel ist sicher genau dann wenn jede Variable im Kopf der Regel auch im Rumpf der Regel vorkommt.

Definition: Datalog Programm

Ein Datalog-Programm P (ohne IBen(Integritätsbedingungen)) ist eine endliche Menge von Horn-Klauseln mit Jedes $d \in P$ ist entweder

- ein Fakt $q(...)$. ohne Variable
- eine sichere Regel $q(...) :- p_1(...), ..., p_n(...)$. mit $q \in iPraedikat$

Ein $d \in P$ heißt auch **Datalog-Klausel** Alle Fakten zu extensionalen Prädikaten sind als in DB-Relationen gespeichert zu denken.

Beispiel Datenbankzustand:

$$\begin{aligned} &ag(a1, m). \\ &kp(c2, a0). \\ &... \\ &rb(a1, r1, t1) \textit{(Kurs a1 im Raum r1 zu t1)} \\ &rb(a3, r2, t4) \end{aligned} \tag{2}$$

Angebot:

Kursnummer	Dozent
a1	m
...	...

Kursplan:

Kursnummer	Voraussetzung
c2	a0
...	...

Raumbelegung:

Kursnummer	Raum	Zeit
a1	r1	t1
...	...	
	...	

Belegung:

Kursnummer	Teilnehmer	Wiederholer
a1	s	j
...	...	
	...	

$$\begin{aligned} vs(X,Y) &: -Kp(X,Y). \\ vs(X,Y) &: -vs(X,Z), Kp(Z,Y). \\ stdpl(W,X,Y,Z) &: -bl(X,W,V), rb(X,Y,Z). \\ ueberschneidungen(X,Y) &: -Kp(Z,X), Kp(Z,Y), rb(X,V,T), rb(Y,W,T), X \neq Y. \end{aligned} \tag{3}$$

Deklarative Semantik

Extensionale Prädikate eines Programms (ext. Rel, ext. DB): EDB
Intentionale Prädikate eines Programms (int. Rel, int. DB): IDB

Bedeutung Bedeutung eines Datalog-Programms P: Menge derjenigen Grundatome zu den intentionalen Prädikaten von P, die logisch aus P gefolgert werden können. (Jedes Modell von P ist auch ein Modell von $f \in F$). Mit Zielklausel g(...), $g \in Praed$: Aus P logisch folgerbare Grundatome zu g, die von g(..) subsummiert (überdeckt) werden.

g(a,X)
---
g(a,b)
g(a,c)

In P werden Werte aus Wertebereichen verwendet, ebenso in Darstellung der extensionalen Prädikate als DB-Relation. Daher können wir $Kost_A$ und Dom idefntifizieren. Mithilfer der Herbrand Interpretation kann die Semantik festgelegt werden (ist möglich).

Herbrand-Interpretation /-Modelle Gewöhnliche Interpretation:

$$\begin{aligned} Konst_A = \{a,b\}, Dom = \{\circ, \square\} \\ k(a) = \circ \\ k(b) = \square \\ ext(p(.,.)) = \{(\circ,\square),(\square,\square)\} \end{aligned} \tag{4}$$

eine mögliche Herbrand-Interpretation (passt dazu)

$$\begin{aligned} Konst_A = Dom = \{a, b\} \\ k(a) = a \\ k(b) = b \\ ext(p(.,.)) = \{(a, b), (b, b) \end{aligned} \tag{5}$$

Entsprechende Herbrand-Interpretation. Betrachte alle Paare zu p(.,.), teste gemäß gegebener (gewöhnlicher) Interpretationin ext(p(.,.)).

$$\begin{aligned} Konst_A = \{a, b\}, Dom = \{\circ, \square\} \\ k(a) = \square \\ k(b) = \square \\ ext(p(.,.)) = \{(\circ, \square), (\square, \square) \end{aligned} \tag{6}$$

(a,a) wird zu $(\square, \square) \in ext(p(.,.))$

Herbrand-Interpretation

$$\begin{aligned} Konst_A = Dom = \{a, b\} \\ k(a) = a \\ k(b) = b \\ ext(p(.,.)) = \{(a, a), (a, b), (b, a), (b, b)\} \end{aligned} \tag{7}$$

Bei beiden Interpretationen sind die gleichen Formeln gültig bei Beschränkung auf quantorenfreie Formeln ohne Variablen (und ohne Funktionen).

Beispiel Erste Interpretation: $p(a, b) \wedge p(b, a) \Rightarrow (\square, \square) \in ext(p(.,.)) \wedge ...$ bzw. $(a, b) \in ext(p(.,.)) \wedge (b, a) \in ext(p(.,.))$

Menge von Konstanten und Prädikatensymbole ist endlich, daher ist die Anzahl der möglichen Herbrand-Interpreationen endlich.

Satz von Gödel / Skolem (Herbrand, 1930)

Eine Klauselmenge P hat ein Modell genau dann wenn P hat ein Herbrand-Modell. Daraus folgt, dass ein Verfahren analog zu Wahrheitstabellen in der Aussagenlogik möglich ist.

Beispiel $F = \{p(a) \Rightarrow q(b), p(a) \wedge q(b)\}$, $q(b)$?

p	q	$p(a) \Rightarrow q(b)$erfüllt?	$p(a) \wedge q(b)$erfüllt?	$p(a) \Rightarrow q(b)$und$p(a) \wedge q(b)$erfüllt?
$\{\}$	$\{\}$	✓	-	-
$\{\}$	$\{a\}$	✓	-	-
$\{\}$	$\{b\}$	✓	✓	✓
$\{\}$	$\{a, b\}$	✓	✓	✓
$\{a\}$	$\{\}$	-	✓	-
...	...	...	...	....
$\{b\}$	$\{b\}$	✓	✓	✓

Jedes Modell von F ist auch ein Modell von q(b), d.h. q(b) kann aus F logisch gefolgert werden. Gilt bei Klauselmengen, aber **Vorsicht bei allgemeinen Formeln**.

Beispiel: $\{p(a), (\exists X)(\neg p(X))\}$ Formelmenge, keine Klauselmenge
Modell (vgl. Übung):

$$\begin{aligned} Dom &= \{0, 1\} \\ k(a) &= 0 \\ ext(p(.)) &= \{(0)\} \end{aligned} \tag{8}$$

Aber: Es gibt kein durch `ext` bestimmtes Herbrand-Modell:

1. $ext(p(.)) = \{(a)\}, Konst = Dom = \{a\}$
2. $ext(p(.) = \{\})$

Herbrand-Modell muss genügend viele Elemente enthalten, damit der Satz von Gödel / Skolem gelten kann. **Skolemisierung** bedeutet, dass man alle Existenzquantoren durch Funktionen ersetzt:

$$(\forall x_1, ..., x_n)(\exists y)(F) \rightsquigarrow (\forall x_1, ..., x_n)(F[f(x_1, ...x_n)/y]) \tag{9}$$

Bemerkung: Skolemisierung

Jede Formel der PL1 Logik kann man in einer erfüllbarkeitsäquivalente Formel in Skolemform umformen:

1. Pränexnormalform

2. Umformungen à la $(\forall x_1, ..., x_n)(\exists y)(F) \rightsquigarrow (\forall x_1, ..., x_n)(F[f(x_1, ...x_n)/y])$ mit jeweils einem neuen Funktionssymbol

Dies ist eine Art "Materialisierung" der durch den Existenzquantor gebundenen Variablen.

Beispiel (von oben) $\{p(a), \neg p(y)^1\}$ erfüllbar $\iff \{p(a), (\exists X)(\neg p(X))\}$ erfüllbar

Vorsicht: Semantische Äquivalenz von Formeln und ihren Skolem-Normalformen im Allgemeinen bicht gegeben.

Skolem-NF: $(\exists X)(p(X)) : p(a)$ Interpretation I mit

$$\begin{aligned} Dom &= \{1, 2\} \\ k(a) &= 1 \\ ext(p(.)) &= \{(2)\} \end{aligned} \tag{10}$$

$\rightsquigarrow \models_I (\exists X)(p(X))$ Belegung von X mit allen Elementen aus Dom, d.h. auch mit 2. Aber $\not\models_I p(a)$, da $(1) \notin ext(p(.))$.

Im Kontext von Datalog

- Herbrand-Universum: Konst
- Herbrand-Basis (HB): Menge aller Grundatome
  ($\rightsquigarrow EDB^2, IDB^3$(Fakten in der Datenbank und solche die Ableitbar sind) $\subset HB$)
- Herbrand-Interpretation: (Konst, id, ext), d.h. jedes Konstantensymbol wird als es selbst interpretiert. (Verglichen mit relationaler Interpretation, dort k ist Bijektion)
- Jede Herbrand-Interpretation ist eindeutig bestimmt durch *ext* (Extension, Ausprägung), da Konst und id unveränderlich sind
- Jedes Prädikat ist eindeutig bestimmt durch die Angabe der Grundatome, für die es "wahr" liefert.
- extentional: genaue Antwort in Tupel
- intentional: Beispielsweise Formeln als Antworten

[1] neue Variable
[2] gegeben durch Datenbankzustand
[3] muss ausgerechnet / gefolgert werden

Definition: Herbrand-Interpretation

Einfache Definition: Eine Herbrand-Interpretation ist eine Teilmenge der Herbrand-Basis.

Beispiel Kursverwaltung

$$\begin{aligned} \text{Aus DB-Rel. KURSPLAN} &\rightsquigarrow kp \in ePräd \\ vs(X,Y) : -kp(X,Y) &\rightsquigarrow vs \in iPräd \end{aligned} \tag{11}$$

Gesucht: Durch Programm P bestimmte Grundatome (Fakten).

Definition: Grundatom

Ein Grundatom f ist eine logische folgerung einer Menge D von Datalog Klauseln (z.B. $D \vDash f$) $\diamondsuit_{Def}$. Jedes Herbrand Modell von D ist auch ein Modell von f.
Da f ein Grundatom ist gilt $D \vDash f \implies f$ ist in jedem Herbrand.Modell von D enthalten. Das heißt $f \in \bigcap\{I | IHerbrand - ModellvonD\}$.
Sei $f \in \bigcap\{I | IHerbrand - ModellvonD\}$, dann ist f ein Grundatom und jedes Modell von D auch in Modell von f.

Definition: Mege aller Konsequenzen

$cons(D) =_{def} \{f \in HB_D | D \vDash f\}$

Satz 1.1

$cons(D) = \bigcap\{I | IHerbrand - ModellvonD\}$
Aufgrund der Eigenschaften unserer Regel ist der Schnitt / cons(D) ein Modell, dies gilt es zu beweisen:

Da $cons(D) \subseteq HB_D$, ist $cons(D)$ eine Herbrand-Interpretation.

Satz 1.2

$cons(D)$ ist ein Herbrand-Modell von D.

Beweis

z.Z. Jedes $d \in D$ ist gültig in dieser Interpretation, also $cons(D)$ ($\vDash_{cons(D)} d$).

Beweis Falls d ein Grundatom ist gehört d zu jedem Herbrandmodell von D, daraus folgt $d \in cons(D)$.
Sei d eine Regel $q(...) : -p_1(...), ..., p_m(...)$. Sei ϱ eine Belegung für die Variable von d. *Annahme:* $(\forall 1 \leq i \leq m)(||p_i(.)||^{\varrho} \in cons(D))$, sonst d gültig unter $cons(D)$. Für jedes Herbrand-Modell I von D gilt, dann $||p_i(.)||^{\varrho} \in I, i = 1, ..., m$ und damit $||q_(.)||^{\varrho} \in cons(D)$, da d eine Horn-Klausel ist. (Der Schluss ist z.B. für $d = q_1(...), q_2(...) : -p_1(...), ..., p_n(...)$ nicht möglich.)

Damit $D \vDash_{cons(D)} ||q(...)||^{\varrho} \in cons(D)$. Wir erhalten, dass $cons(D)$ ein Herbrand-Modell von D ist. Da d beliebig gewählt wird folgt der Satz.

$cons(D)$ ist offensichtlich eindutig bestimmt und das kleinste Herbrand-Modell von D. Damit: Seantik eines Datalog-Programms P ist gegeben durch das kleinste Herbrand-Modell von P oder (äquivalent) durch $cons(P) = \{f \in HB_P | O \vDash f\}$.

Beispiel $r \in ePräd, p, q \in iPräd, Konst = \{1, 2, 3\}$.

Fakten:

$$\begin{aligned} &r(1). \\ &p(X) : -q(X). \\ &q(X) : -r(X). \end{aligned} \tag{12}$$

Herbrand-Modelle:

$$\begin{aligned} &\{r(1), p(1), q(1)\} \\ &\{r(1), p(1), q(1), q(2), p(2)\} \\ &\{r(1), p(1), q(1), q(2), p(2), p(3)\} \end{aligned} \tag{13}$$

Fixpunkt-Semantik

Deklarative Semantik liefert keinen brauchbaren Algorithmus. Die Fixpunkt-Sematik führt direkt zu einem algorithmus für die Schrittweise Berechnung des kleinsten Herbrand-Modells.

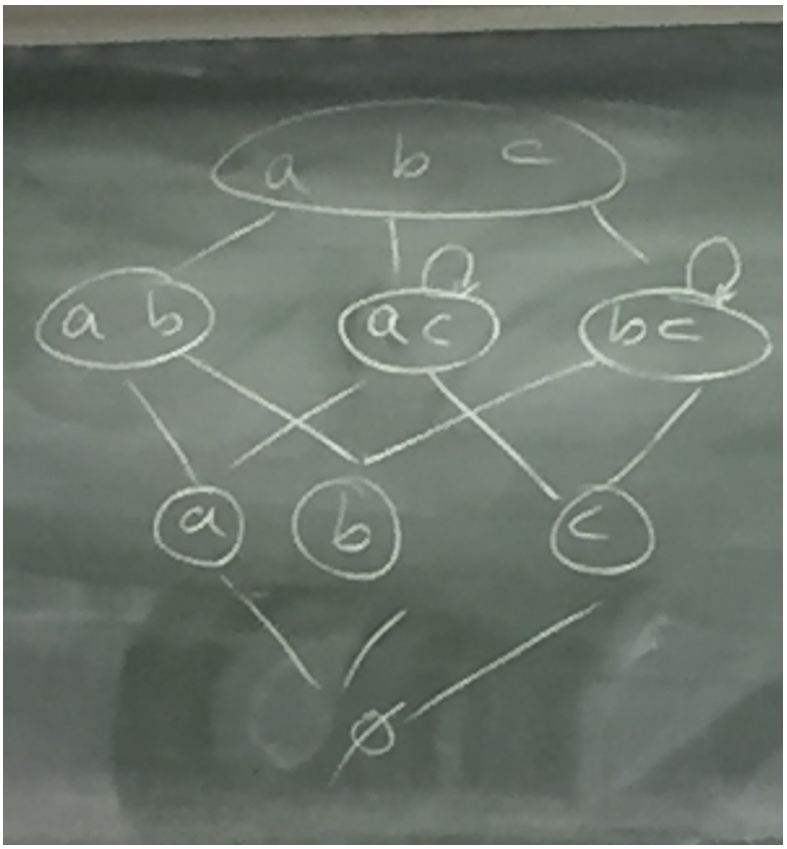

Figure 1:

$$\begin{aligned} &c \in ac \\ &\tau(c) \subseteq \tau(ac) \\ &\tau(c) = c! \\ &\tau(c) = ac \end{aligned} \tag{14}$$

$$\begin{aligned} &c \in bc \\ &\tau(c) \subseteq \tau(bc) \\ &\tau(c) = c! \\ &\tau(c) = bc \end{aligned} \tag{15}$$

Dies ist ein Wiederspruch

P Datalog-Programm τ_P liefert Fakten von P vereinigt mit Fakten, die in einem Schritt aus der Argumentenmenge von τ_P mit den Regeln von P abgeleitet werden können.

Ableitung in einem Schritt

Definition: Substitution

Eine Substitution ist eine endliche Menge der Form

$$\{X_1/t_1, \cdots, X_n/t_n\}, X_1, ..., X_n unterschiedliche Variablen, t_1, ..., t_n Terme, X_i \neq t_i \tag{16}$$

Falls alle t_i Konstanten ist dies eine **Grundsubstitution**.

Sei θ eine Substitution, t ein Term (Variable oder Konstante), so gilt

$$t\theta =_{def} \begin{cases} t_i, & \text{falls } t/t_i \in \theta \\ t, & \text{sonst} \end{cases} \tag{17}$$

Sei L ein Literal, $L\theta$ bezeichnet dasjenige Literal, das aus L entsteht, indem alle Variablen X_i mit L_i für die $X_i/t_i \in \theta$ gilt, simultan durch t_i ersetzt werden. Analog $d\theta$ für eine Datalog-Klausel d.

Beispiel

$$L = p(X, Y, a), \theta = \{X/a, Y/X\} L\theta = p(a, X, a) \tag{18}$$

Sei D eine Menge von Datalog Klauseln. In einem Schritt aus D ableitbare Menge von Fakten:

$$fakt_1(D) = \{f \in HB_D | (\exists Regel L_0 : -L_1, ..., L_n)(\exists Sicht\theta)(\{L_1\theta, \cdots, L_n\theta\} \subseteq D \wedge f = L_0\theta)\} \tag{19}$$

(Annahme: built-in Prädikate geeignet berücksichtigt ⇒ Algorithmus FAKT_1)

Definiere $\tau_P | 2^{HB} \Rightarrow 2^{HB}$. $\tau_P(I) =_{def} P_F \cup fakt_1(P_R \cup I)$. P_f = Menge der Fakten von P. P_P Menge von Regeln von P.

Satz 1.3

Für jedes Datalog-Programm P ist τ_P eine monotone Transformation auf $(2^{HB}, \subseteq)$.

Beweis

Seien $I_1, I_2 \in HB_P$ mit $I_1 \subseteq I_2$. z.Z.: $\tau_P(I_1) \subseteq \tau_P(I_2)$.
Sei $f \in \tau_P(I_1)$. Falls $f \in P_F$, dann gilt auf $f \in \tau_p(I_2)$. $f \in fakt_1(P_R \cup_1)$ da $I_1 \subseteq I_2$, gilt $P_R \cup I_1 \subseteq P_R \cup I_2$ und somit $f \in fakt_1(P_R \cup I_2)$ wg. Monotonie von Fakt_1 auf $(2^{HB}, \subseteq)$.

Beispiel

$$\begin{aligned} P =& p(1) \\ =& p(2). \\ =& r(1). \end{aligned} \tag{20}$$

$$\begin{aligned} q(X) &: -s(X), r(X). \\ s(X) &: -p(X). \end{aligned}$$

$\rightsquigarrow^{T_p(\emptyset)}$

$$\begin{aligned} & p_1 \\ & p_2 \\ & r_1 \end{aligned} \tag{21}$$

$\rightsquigarrow^{T_p(\cdots)}$

$$\begin{array}{l} p_1 \\ p_2 \\ r_1 \\ s(1)(\text{fakt\_1}) \\ s(2)(\text{fakt\_1}) \end{array} \tag{22}$$

$\rightsquigarrow^{T_p(\cdots)}$

$$\begin{array}{l} p_1 \\ p_2 \\ r_1 \\ s(1) \\ s(2) \\ q(1)(\text{fakt\_1}) \end{array} \tag{23}$$

Satz 1.4

Für jedes Datalog-Programm P gilt $cons(P) = lf_p(\tau_p)$ (lf = least fixpunkt).

Beweis

:

1) $cons(P) ist ein Fixpunkt von \tau_P$.
$\tau_p(cons(P)) = P_F \cup fakt\_1(P_R \cup cons(P))$. cons(P) ist kleinstes Herbrand-Modell von P, d.h. alle Regeln sind gültig unter $cons(P) \Rightarrow fakt\_1(P_R \cup cons(P) = cons(P) \backslash P_F)$. (und kein Fakt kann aus cons(P) entfernt werden, ohne dass die Modelliereigenschaft verloren geht.) $\Rightarrow \tau_P(cons(P)) = P_F \cup cons(P) \backslash P_F = cons(P)$

2) $cons(P)$ ist ~~minimaler Fixpunkt von τ_P. Annahme: $(\exists I \subseteq cons(P))(\tau_P(I) = I)$~~. cons(P) ist minimales Herbrand-Modell $\Rightarrow$ I ist kein Herbrand-Modell. Da $P_F \subseteq I$ wg. Annahme $\tau_P(I) = I$ folgt midestens eine Regel ist nicht erfüllt, d.h. $(\exists h_0, \cdot, h_n \in P_R)(\exists Substitution \theta)(\{h_1\theta, \cdots, L_n(\theta)\} \subseteq I \wedge L_1\theta \in I)$. ~~Aber~~ (1

$L_0 \in fakt_1(P_R \cup I)$ nach Definitio von fakt_1. Da auch $L_0\theta \in P_F$ wegen $P_F \subseteq I, folgt\tau_P(I) \neq I$. Noch Fixpunkttheorem (Kuaster / Tarski) ist minimaler Fixpuknt von τ_P kleinster Fixpunkt von τ_P.

Vorgehensweise bei Fixpunktberechnung

bottom-up

Bezeichnung: Forward-chaining

Da "$\Leftarrow$" die natürliche Richtung für die Anwendung von Regeln ist. Da Regeln sicher sind ist $L_i\theta$ stets ein Fakt θ_i. Betrachte Eignung der Methode zur Anfrageauswertung, z.B. ?- vs(c4, y). "Alle Kurse, die Voraussetzung von Kurs c4 sind"

Ineffizient Berechnung des lf_p und anschließende Siche zu vs(c4, y) passende Fakten $\Rightarrow$ Vermeide möglichst Berechnung nicht relevanter Fakten (später).

Starte mit Zielklausel, Suche ein Regel die zu einem Atomder Klausel passt. Ersetze das Atom durch angepassten Rumpf der gewählten Regel, neue Anfrage, iterieren.

Bezeichnung: Backward-chaining

Beispiel

$$\begin{aligned} P =& p(1) \\ =& p(2). \\ =& r(1). \end{aligned} \tag{24}$$

$$\begin{aligned} q(X) &: -s(X), r(X). \\ s(X) &: -p(X). \end{aligned}$$

Prozedurale Semantik

Beweistheoretische Sicht, übertragen auf Datalog Programm P aufgefasst als Theorie 1. Stufe

Semantik: $\{f \in HB_P | P \vdash f\}$
Wie können Ableitungen (Beweise) von Fakten systematisch durchgeführt werden?

Regeln:

$$\begin{aligned}(1) & vs(X,Y) :- Kp(X,Y).\\ (2) & vs(X,Y) :- vs(X,Z), kp(Z,Y).\\ & P = \{Kp(c4,a3), \cdots, kp(c2,a0), (1), (2)\}\end{aligned} \tag{25}$$

Zeige $P \vdash vs(c4, a0)$.

$$\begin{aligned}(1) &= (\forall X)(\forall Y)(kp(X,Y) \Rightarrow vs(X,Y))\\ (2) &= (\forall x)(\forall Y)(\forall Z)(vs(X,Z) \wedge kp(Z,Y) \Rightarrow vs(X,Y))\end{aligned} \tag{26}$$

Beweis

Idee

- Erzeuge alle Beweismuster (Suchbäume)
- Suche Belegungen für die Variable, so dass alle Blätter zu Fakten aus P werden
- jede solche Belegung erzeugt einen Beweisbaum.

Zunächst Suche nach "passenden" Köpfen von Regeln erfordert Definition.

Definition: Unifizierbar

Seien L_1 und L_2 heißen **unifizierbar**, wenn ($\exists$ Substitution Θ)($L_1\Theta = L_2\Theta$). Θ heißt dann **Unifikator**.

Beispiel $L_1 = vs(X, Z)$
$L_2 = cs(X, Y)$
$\Theta = \{Z/Y\}$ und $\Theta = \{Y/Z\}$ sind Unifikatoren von dem Paar L_1, L_2, aber auch $\Theta = \{X/a, Y/a, Z/a\}$. Die ersten beiden sind spezifischer als das letzte

$L_1 = q(X, Y, c) L_2 = q(W, b, Z)$ Unifikation z.B.:

$$\begin{aligned}\Theta_1 &= \{X/WY/bZ/c\}\\ \Theta_2 &= \{X/T, W/T, y/b, Z/c\}\\ \Theta_3 &= \{X/a, W/a, Y/b, Z/c\}\end{aligned} \tag{27}$$

Nicht unifizierbar: $L_1 = q(X, c, U)L_2 = q(W, G, Z)$ oder $L_1 = q(X, a, X)L_2 = q(b, Y, Y)$

Definition: Komposition

Sei $\Theta = \{X_1/t_1, \cdots, X_n/t_n\}, \varsigma = \{Y_1/n_1, \cdots, Y_m/t_m\}$ Substitutionen.
Die Komposition $\Theta\varsigma$ von Θ und ς erhält man aus

$$X_1/t_1\varsigma, \cdots, X_m/t_m\varsigma, Y_1/n_q, \cdots, Y_m/n_m \tag{28}$$

Durch Streichen von Elementedn der Form Z/Z sowie Y_i/n_i mit $Y_i = X_j$ für ein jj $\in \{1, ..., n\}$

Beispiel $\theta = \{X/a, Y/W\}\varsigma = \{X/bmY/V, W/Z\}$
$\Theta\varsigma = \{X/a, YZ, W/Z\}$

Definition: allgemeinere Substitution

Seien Θ, ς Substitutionen, Θ heißt allgemeiner als $\varsigma\diamondsuit(\exists\text{Substitution}\delta)(\Theta\delta = \varsigma)$. Seine L_1, L_2 Literale. Ein allgemeinster Unifikator (mgu) von $L_1 in L_2$ ist ein Unifikator von $L_1 und L_2$, der allgemeiner als alle anderen Unifikatoren ist.

Beispiel Θ_2 ist allgemeiner als Θ_3; betrachte $\delta = \{T/a\}$, es gilt $\Theta_2\delta = \Theta_3$. Θ_1 ist allgemeiner als $\Theta_2, \delta = \{W/T\}$. mgu ist i.A. nicht eindeutig bestimmt. $L_1 = p(X, X)L_2 = p(V, W)$ mgu:

$$\begin{array}{l}\{X/W, V/W\}\\ \{X/V, W/V\}\\ \{V/X, W/X\}\end{array} \tag{29}$$

Beispiel $L_1 = q(X, Y, c)$[4] $L_2 = q(W, b, Z)$[5]

Suchbaum

(Beweismuster zu einem Programm P)

[4] t1, t2, t3
[5] k1 k2 k3

- Wurzel ist mit einem "Ziel" $g = p(t_1, \cdots, t_n), p \in iPr \dot{\cdot} \dot{\cdot} ad$, bekannt.
- Knoten entlang eines Pfases von der Wurzel aus sind abwechselnd mit Atome und Regeln benannt. (Ziel-, Regelknoten)
- Alle Blattknoten sind mit Atomen benannt
- Sei k ein mit einem Atom $a(s_1, ..., s_l)$ benannten Knoten (keine Blattknoten), dann ist der unmittelbare Nachfolger von k mit einer Regel von P benannt, deren Kopf mit $q(sq, ..., s_l)$ unifizierbar ist
- sei k' ein mit einer Regel $r = L_0 : -L_1, ..., L_m$ benannte Knoten der unmittelbare Vorgänger von k' sei mit $q(s_1, ..., s_l)$ benannt. Dann hat k' m unmittelbare Nachfolger, die mit $I_1 mgu(q(s_1, ..., s_l), I_0), ..., I_m mgu(q(s_1, ..., s_l), I_0)$ benannt sind. Dabei sind $I_0, .., I_m$ Atome, die aus $L_1, ..., L_m$ durch eventuelle Umbenennung von Variablen hevorgehen (die Variablen seien stets so umbenannt, dass sie im Baum eindeutig sind).

Beispiel

Suchbaum $\rightarrow$ Beweisbaum

Gegeben Sei der Suchbaum S und eine Grundsubstitution Θ mit folgenden Eigenschaften:

- Θ ordnet jeder Variable in S ein Konstantensymbol aus der Menge der in P vorkommenden Konstantensymbole zu.
- Für jedes Blatt von S gilt, dass Θ, angewand auf die Benennung des Blattes, einen Fakt aus P_F liefert (build-in-Prädikate seien entsprechend berücksichtigt)

Beweisbaum B entsteht aus S durch Anwendung von Θ auf alle Benennungen von Zielknoten. B repräsentiert einen Beweis für $g\Theta$, g benennung der Wurzel von S.

Beispiel:

$$\Theta = \{X/c4, Y/a0, Zc2, Z'/a3\} \tag{30}$$
$$kp(c4, a3), kp(a3, c2), kp(c2, a0) \text{ Fakten} \tag{31}$$
$$\tag{32}$$

Bezeichnung : Voller Beweisbaum

Systematische Erzeugung der Suchbäume

Sukzessive alle Bäume der Tiefe 1,2, ... Erzeugen (breadth first)

Definition: Tiefe eines Baums maximale Anzahl von Zielknoten auf einem Pfad von einem Blattknoten zur Wurzel. Entsprechend Knoten der Tiefe i, Ebene i eines Baumes. Zusätzlich: Spezielle Suchbäume (Tiefe 0) für Fakten aus P.

Beispiel : Ziel $g = vs(c4, y)$

- **Tiefe 1**: vs(c4, y), Θ, kein Beweisbaum
- **Tiefe 2**: vs(c4, y)
  kp(c4,y), $\{y/a3\}$

Offensichtlich

- Zu jedem endlichen Beweise, der mit Grundatomen "Startet", kann ein entsprechender Beweisbaum auf den beschriebenen Weg erhalten werden.
- Zu jedem Fakt $f \in cons(P)$ gibt es einen endlichen Beweis, der mit Grundatomen startet, siehe Fixpunktberechnung $\rightarrow$ voller Beweisbaum

$\Rightarrow$ Methode Suchbaum / Beweisbaum ist vollständig.
Da ein Beweisbaum mit $g\Theta$ als Benennung der Wurzel einen Beweis für $g\Theta$ darstellt
$\Rightarrow$ Methode Suchbaum / Beweisbaum ist korrekt.

Satz 1.5

Sei P ein Datalog-Programm. Die Suchbaum / Beweisbaum Methode, angewand auf alle Ziele $q(X_1, \cdots, X_{Stelligkeit(q)})$, q intentionales Prädikatesymbol von P, liefert cons(P) als Ergebnis

Frage: Abbruchkriterium bei der Erzeugung von Suchbäumen

Es gilt

- Beweisbäume sind isomorph zu den Suchbäumen, aus denen sie erzeugt wurden
- Zu jedem Fakt $f \in cons(P)$ gibt es einen vollen Beweisbaum, in dem auf jeder Zielknotenebene ein neuer Fakt (neues Teilziel) auftritt

  Beweis Sei B ein voller Beweisbaum, der diese Eigenschaft nicht erfüllt. Dann gibt es ein i, so dass in B auf Ebene i nur Fakten auftreten, die schon auf vorhergehenden Ebenen auftreten. Die Teilbäume von B mit Wurzel auf Ebene i können "höher gehängt" werden $\Rightarrow$ Es gibt einen äquivalenten, vollen Beweisbaum mit geringerer Tiefe.
- Es gibt nur endlich viele verschiedene Grundatome, die aus den Prädikaten und Konstantensymbolen gebildet werden können. Konkret: Seien ap(p) die Anzahl verschiedener Prädikatensymbole die in p vorkommen
- ak(p) die Anzahl verschiedener Konstantensymbole, die in P vorkommen

Dann ist $max.fakt(P) = ap(P) * ak(P)^{max\_st(P)}$ eine obere Schranke für die Anzahl verschiedener Grundatome. **Damit:**

Satz 1.6

Die Suchbaum / Beweisbaum Methode bleibt vollständig für ein Programm P, wenn nur Bäume mit max. Tiefe max_fakt(P) betrachtet werden. Statt breadth first andere Vorgehensweise denkbar:

- Depth first mir Abbruch bei max_fakt(P) und backtracking
- Dynamische Abbruchbedingungen, abhängig von erzeugten Faktenmenge (limit 10)
- Erkennen und Ausnutzen identischer (Teil-) Zielknoten ($\rightsquigarrow$ gerichteter, azyklischer Graph)

Zusammenfassen von Such und Beweisbäumen, Berücksichtigung von Fakten aus P_F $\Rightarrow$ frühzeitiges Erkennen von Suchbäumen zu denen keine Beweisbäume existieren

Sei g das Ziel (Wurzelbenenung), dann erzeugt ein Baum der mit der leeren Menge endet, genau ein Ergebnis.

$g\Theta_1, ..., \Theta_n, (\Theta_1, ..., \Theta_n$die in dieser Reihenfolge angewandten Unifikatoren (von der Wurzel aus) (33)

$\Theta = \Theta_1, ..., \theta_n$ heißt Antwortsubstitution. (34)

Resolutionsmethode

Für allgemeine Klauselformen entwickelte Methode zum automatischen Beweisen.

Literatur J Allen Robinson: A machine-oriented logic base on the resolution principle. Journal of the ACM 12, S.23-41, 1965

In Logikprogrammierung (Horn-Klauseln): SLD-Resolution (Linear Resolution with Selected Function for Definite Clauses).

- S (Selection function): Legt fest, wie Literale aus einem Ziel (Klausel mit einem oder mehreren negierten Literalen) auszuwählen sind
- L (Linear): In jedem Schritt wird genau ein Literal ausgewählt für die Unifikation mit dem Kopf einer Regel einer einem Fakt
- D (Definit): Beschränkung auf Programme mit definiten Klauseln

Bemerkung Methode steht für ganze Klasse von Algortihmen. Methode auch bekannt unter LUSH (Linear resolution with Unrestricted Selection function for Horn clauses)

Übergang Such / Beweibaum $\rightsquigarrow$ Resolution Nachweis der Unerfüllbarkeit einer Klausel

Seien $E = \{\neg M_1, \cdots, \neg M_i, \cdots, \neg M_K\}, r = L_0 : -L_1, \cdots, L_m$ eine Regel und $S(E) = \neg M_i$. Es existiere $mgu(M_i, L_0) = 0$.

Resolvente von E und r: $\{\neg M_1\Theta, \cdots, \neg M_{i-1}\Theta, \neg L_1\Theta, \cdots, Ł_m\Theta\neg M_{i+1}\Theta, \cdots, \neg M_K\Theta\}$

- Unerfüllbarkeit: Anwendung der Methode ende mit $\square$ (leere Klausel, entspricht Falsch)
- Ergebnis: Sei $g = \{\neg p(t_1, \cdots, t_n)\}$ das Ziel, $\{p(t_1, \cdots, t_n)\Theta/\Theta$ Antwortsubstitution für eine Ableitung von g, die mit $\square$ endet $\}$

  Ohne Beweis.

0.1 Satz 1.7

Die SLD-Resolution liefert korrekte und vollständige Ergebnisse für alle Datalog-Programme.