Programmierkurs Prolog

SS 1998

Thorsten Joachims

Universität Dortmund

LS VIII - Prof. K. Morik

Uebersicht

- Aufbau von Expertensystemen
- Produktionenregeln
- Backward-Chaining
- Forward-Chaining
- Erklärungen
- Unsicheres Wissen
- Interaktive XPS-Shell

Produktionsregeln

```
if flur_nass and bad_trocken
     then problem_in_kueche.
if kueche_trocken and flur_nass
     then bad_undicht.
if kuechenfenster_zu or kein_regen
     then kein_wasser_von_draussen.
if problem_in_kueche and
     kein_wasser_von_draussen
     then kueche_undicht.

fakt(flur_nass).
fakt(bad_trocken).
fakt(kuechenfenster_zu).
```

Operatordefinitionen

```
op(+Precedence, +Assoc, +Name)
```

• Praeferenz	Bindung (0 bis 1200)	
• Assoc	---------------------------	
	xfx	infix
	xfy	infix
	yfx	infix
	fx	prefix
	fy	prefix
	xf	postfix
	yf	postfix
• x	steht fuer ein Argument mit echt kleinerer Praeferenz	
• y	steht fuer ein Argument mit gleicher oder kleinerer Praeferenz	

Backward-Chaining

```
meta_bc(Goal) :-
     fakt(Goal).

meta_bc(Goal) :-
     If Praem then Goal,
     meta_bc(Praem).

meta_bc(Goal1 and Goal2) :-
     meta_bc(Goal1),
     meta_bc(Goal2).

meta_bc(Goal1 or Goal2) :-
     meta_bc(Goal1)
     ;
     meta_bc(Goal2).
```

Forward-Chaining

```
meta_fc :-
     neuer_fakt(P),!,
     write(‚Derived: ‚),write(P),nl,
     assert(fakt(P)),
     meta_fc
     ;
     write(‚Done.‘).
neuer_fakt(Goal) :-
     if Praem then Goal,
     not(fakt(Concl)),
     composed_fakt(Praem).
composed_fakt(Goal) :-
     fakt(Goal).
composed_fakt(Goal1 and Goal2) :-
     composed_fakt(Goal1),
     composed_fakt(Goal2).
composed_fakt(Goal1 or Goal2) :-
     composed_fakt(Goal1)
     ;
     composed_fakt(Goal2).
```

Forward vs. Backward

Produktionenregeln schliesen von

Eingabeinformation --> abgeleitete Information

Beispiele:

- Daten --> Zielen
- Hinweisen --> Hypothesen
- Beobachtungen --> Diagnosen

Forward: “*Data driven*”

Backward: “*Goal driven*”

Was ist besser?

- viele Daten, wenige Ziele
- viele Ziele, wenige Daten

Wie-Erklärungen

Wie (how) wurde die Antwort abgeleitet?

Wird nach der Diagnose gefragt.

Beispiel:

Die Küche ist undicht, da das Problem in der Küche liegt und kein Wasser von draußen gekommen ist.
Das Problem ist in der Küche, da der Flur nass ist aber das Bad trocken ist.
Das Wasser ist nicht von draußen gekommen, da das Fenster geschlossen ist.

Backward-Chaining II
Erstellen des Beweisbaumes

```
meta_bc(Goal,Goal) :-
     fakt(Goal).

meta_bc(Goal, Goal <= PraemP) :-
     If Praem then Goal,
     meta_bc(Praem).

meta_bc(G1 and G2, P1 and P2) :-
     meta_bc(G1,P1),
     meta_bc(G2,P2).

meta_bc(G1 or G2, P) :-
     meta_bc(G1,P)
     ;
     meta_bc(G2,P).
```

Warum-Erklärungen

Warum (why) soll ein Fakt eingegeben werden?

Wird während der Konsultation gefragt.

Beispiel:

Ziel: Küche undicht?
Frage: Ist das Bad trocken?
Benutzer: Warum die Frage?
Antwort: Weil der Flur nass ist und wenn das Bad trocken ist, dann ist das Problem in der Küche.
Wenn das Problem in der Küche ist und kein Wasser von draußen gekommen ist, dann ist die Küche undicht.

Unsicheres Wissen

Erweiterung der Produktionenregeln und Fakten durch “*Sicherheitsfaktoren*”

```
fakt(Aussage,Sicherheit)
```

```
if Prämissen then Konclusion : Sicherheit
```

Beispiel:

```
fakt(kein_regen,0.8).


if
     flur_nass and bad_trocken
then
     problem_in_kueche : 0.9.
```

Verrechnung der Unsicherheitsfaktoren

Eine einfache Möglichkeit:

- c(P1 and P2) = min(c(P1), c(P2))
- c(P1 or P2) = max(c(P1), c(P2))
- c(if P1 then P2 : C) => c(P2) = c(P1) * C

Zusatzbedingung:

Keine zwei Regeln haben die gleiche Konklusion.

Backward-Chaining III
Schließen unter Unsicherheit

```
meta_bc(Goal,C) :-
    fakt(Goal,C).

meta_bc(Goal, C) :-
    If Praem then Goal : C1,
    meta_bc(Praem,C2),
    C is C1 * C2.

meta_bc(G1 and G2, C) :-
    meta_bc(G1,P1,C1),
    meta_bc(G2,P2,C2),
    minimum(C1,C2,C).

meta_bc(G1 or G2, C) :-
    meta_bc(G1,C1),
    meta_bc(G2,C2),
    maximum(C1,C2).
```