Effizientere Implementation

Bekannt sind:

- Reihenfolge von Klauseln in Prozeduren
- Reihenfolge von subgoals in Klauseln

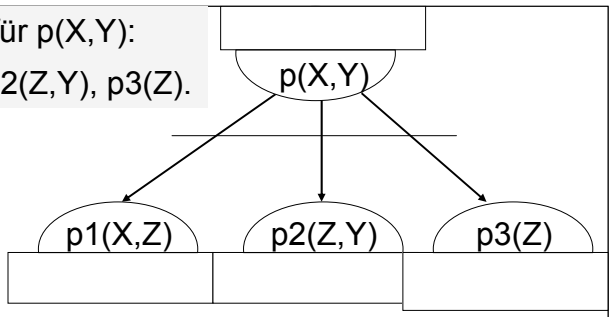
Zur Laufzeit nicht gesamte Listen speichern,
sondern nur Referenz auf jeweils nächsten Eintrag

g_i statt $[g_i, ..., g_n]$,

Modell für Relationen (mit Variablen)

Problemzerlegung für p(X,Y):

p(X,Y) :- p1(X,Z), p2(Z,Y), p3(Z).

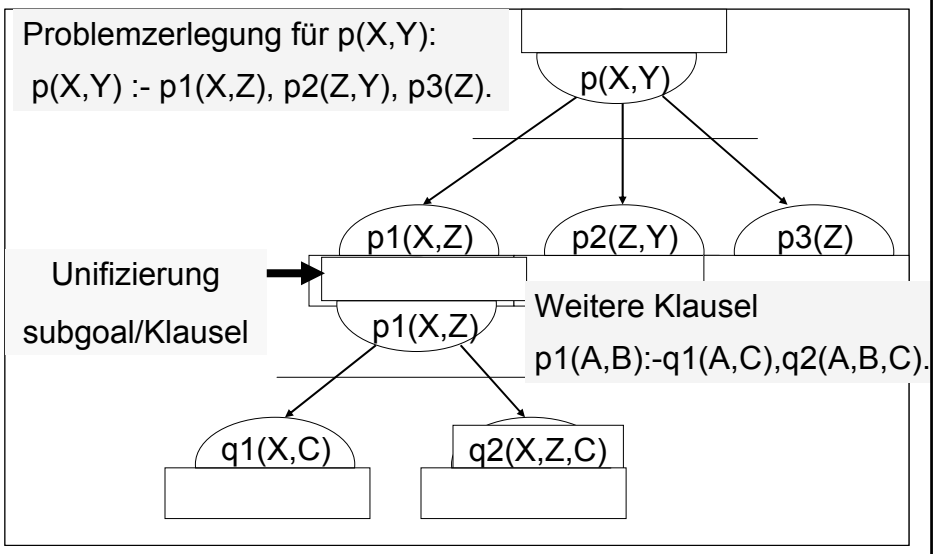


„Ferguson-Diagramm“

Modell für Relationen (mit Variablen)

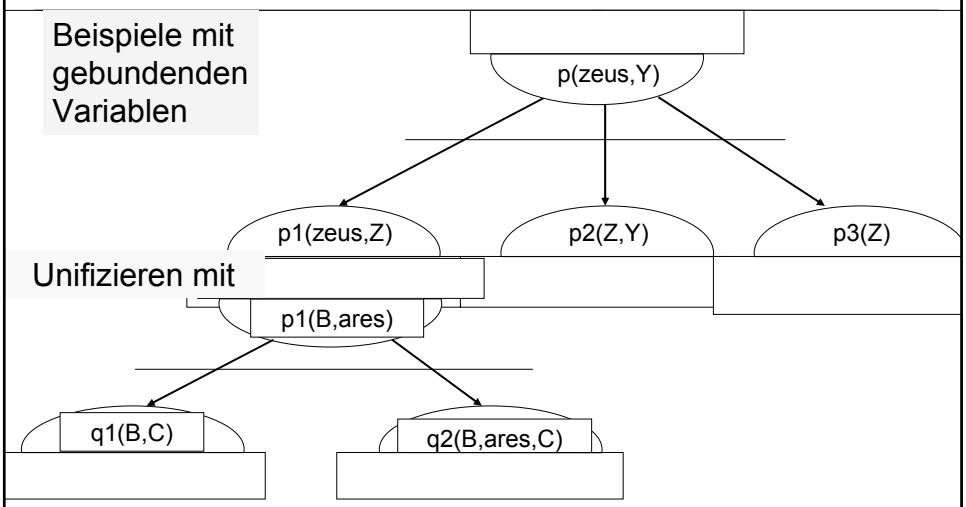
Problemzerlegung für p(X,Y):

p(X,Y) :- p1(X,Z), p2(Z,Y), p3(Z).



Modell für Relationen (mit Variablen)

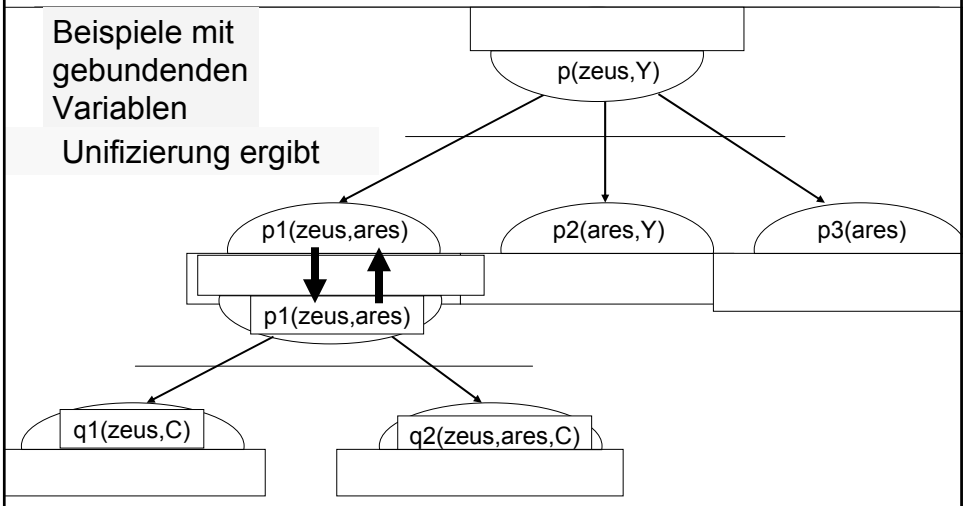
Beispiele mit gebundenden Variablen



Modell für Relationen (mit Variablen)

Beispiele mit gebundenden Variablen

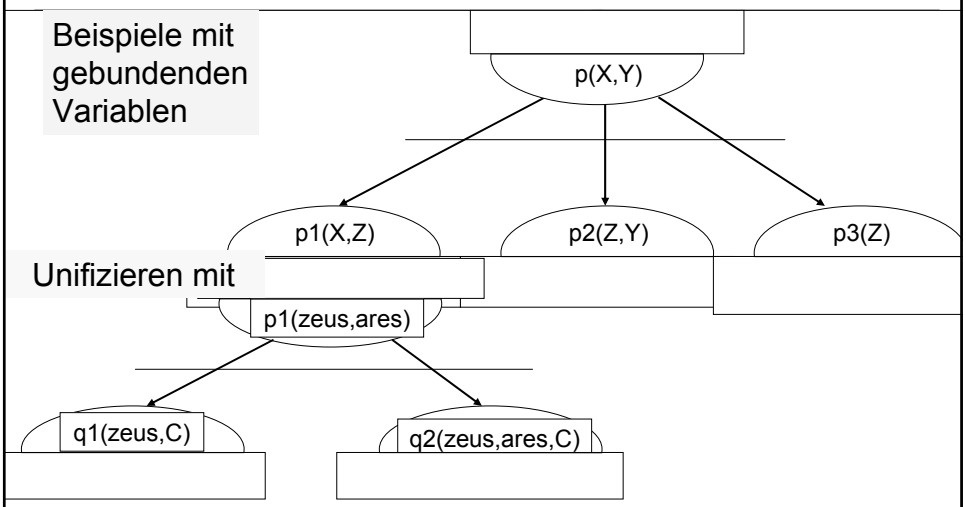
Unifizierung ergibt



Modell für Relationen (mit Variablen)

Beispiele mit gebundenden Variablen

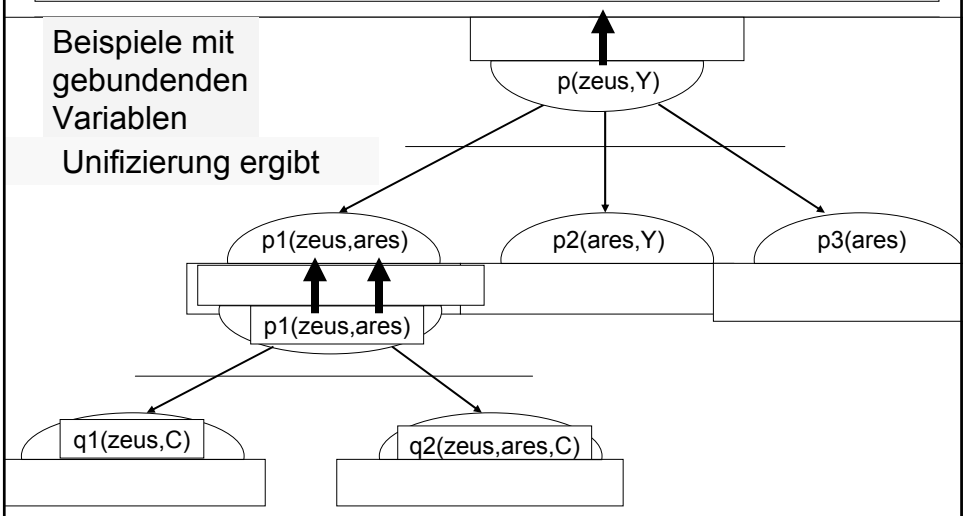
Unifizieren mit



Modell für Relationen (mit Variablen)

Beispiele mit gebundenden Variablen

Unifizierung ergibt



Algorithmus mit Variablen

```
PROCEDURE solve(unsolved_goals: GOALLIST);
                VAR k:KLAUSEL, g: GOAL;
BEGIN
 IF unsolved_goals = NIL THEN HALT(Result)
 ELSE g:= top(unsolved_goals);
   FORALL k ∈ klauseln(g) DO
                (* Klauseln für g nacheinander rekursiv probieren*)
     IF unify (g , head(k), σ) THEN
       solve(concatenate( σ(subgoals(k)), σ(tail(unsolved_goals)) ))
                (* subgoals der Klausel k weiter verfolgen *)
     END
   END (*FORALL*)
 END (*IF*)
END solve;
```

(für mehrfache Antworten: später)

σ ist die bei der Unifikation verwendete Substitution

Algorithmus mit Variablen

unify(g , head(k), σ) bewirkt

- Prüfung auf Unifizierbarkeit von g und head(k)
- Bindung von Variablen gemäß σ
  durch Aktionen im Laufzeitsystem:
  - Für jede Klausel k wird Speicherplatz (frame) angelegt,
  - darin Speicherplatz für Argumente (environment) mit Verweisen auf Bindungen.
  - Beim Backtracking löschen der frames,die jünger als jüngster Backtrack-Punkt sind.

Algorithmus mit Variablen

Problem:

Effiziente Methode zur Bindung von Variablen beim Klauselaufruf und zum Auflösen von Bindungen beim Backtracking

Implementierungsidee:

Gegenseitige Referenzen der Variablen in Baumform

Anbindung an konstante/komplexe Strukturen an der Wurzel des Baumes

Dereferenzierung: Verfolgen einer Kette von Referenzen

Implementation: frames

Für jeden Klauselaufruf wird im Laufzeitsystem ein Speicherbereich reserviert: „frame“

Er enthält:

- Klausel einschließlich der Argumente.
  Effiziente Variante: „structure sharing“, in frame nur
  - Verweis auf template der Klausel
  - Argumente der Klausel („environment“).
- Verweis auf nächstes subgoal der aktuellen Klausel.
- Verweis auf Vater-Klausel der aktuellen Klausel.
- (Verweis auf jüngsten Backtrack-Punkt.)

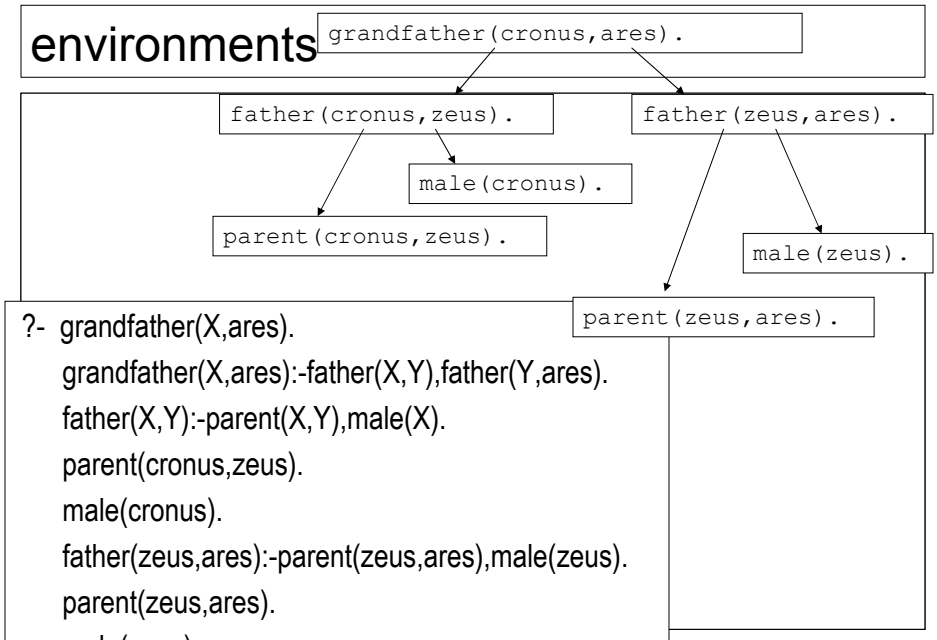
Implementation: frames

Frame enthält zusätzlich für Backtrack-Punkt :

- Nächste Klausel beim Backtracking.
- Davor liegender Backtrack-Punkt.
- Trail-Information (Protokoll der zu löschenden Variablenbindungen)

Dafür Reorganisation bei jedem Backtracking.

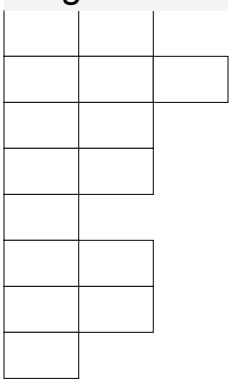
environments




```
?- grandfather(X,ares).
   grandfather(X,ares):-father(X,Y),father(Y,ares).
   father(X,Y):-parent(X,Y),male(X).
   parent(cronus,zeus).
   male(cronus).
   father(zeus,ares):-parent(zeus,ares),male(zeus).
   parent(zeus,ares).
   male(zeus).
```


environments

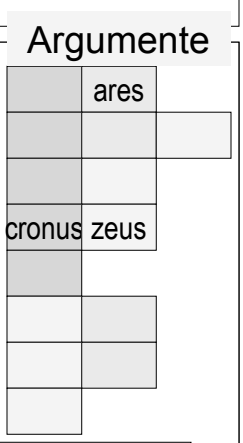
Argumente



```
?- grandfather(X,ares).
   grandfather(X,ares):-father(X,Y),father(Y,ares).
   father(X,Y):-parent(X,Y),male(X).
   parent(cronus,zeus).
   male(cronus).
   father(zeus,ares):-parent(zeus,ares),male(zeus).
   parent(zeus,ares).
   male(zeus).
```

environments

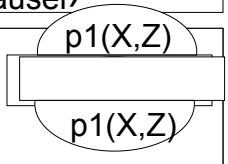
```
?- grandfather(X,ares).
   grandfather(X,ares):-father(X,Y),father(Y,ares).
   father(X,Y):-parent(X,Y),male(X).
   parent(cronus,zeus).
   male(cronus).
   father(zeus,ares):-parent(zeus,ares),male(zeus).
   parent(zeus,ares).
   male(zeus).
```



Aneinandergebundene Variable in einer Klasse
Implementation als Baum von Referenzen
Endpunkte: referierter Term (z.B. in Konstanten-Tabelle)

Unify($Argument^i_{goal}$,$Argument^i_{klausel}$)

T1 := Dereferenziere($Argument^i_{goal}$)
T2 := Dereferenziere($Argument^i_{klausel}$)



Unifikation erfolgreich, falls T1 und T2 unifizierbar.

Als „Seiteneffekte“ dabei Variablen binden:
Falls T1 und T2 Variable: T1 und T2 aneinander binden.
    Prinzip der Rückwärtsreferenz:
    Jüngere Variable an ältere Variable binden
Sonst: Variable Ti an den nicht-variablen Term binden
    Falls jüngere Variable älter als jüngsterBacktrack-Punkt:
    Bindung im „trail“ protokollieren.

Backtracking

Rücksetzen der frames bis zum letzten Backtrack-Punkt.

Dabei entfallen Bindungen der dortigen Variablen automatisch

Bindungen für ältere Variablen müssen explizit gelöst werden gemäß Protokoll im „trail“.

Fortsetzung mit der im Backtrack-Punkt als Alternative vorgemerkten nächsten Klausel

Backtracking

?- grandfather(X,ares).
grandfather(X,ares):-father(X,Y),father(Y,ares).
father(X,Y):-parent(X,Y),male(X).
parent(uranus,cronus).
male(uranus).
father(cronus,ares):-parent(cronus,ares),male(cronus).
parent(cronus,ares). Fehlschlag

Argumente

	ares
uranus	cronus

Backtracking

?- grandfather(X,ares).
grandfather(X,ares):-father(X,Y),father(Y,ares).
father(X,Y):-parent(X,Y),male(X).

Argumente

	ares

~~Algorithmus mit Variablen~~ und mehrfachen Antworten

(0) (Start) $\mathcal{L}$: = [[Ausgangsproblem(e)]] .

(1) Falls $\mathcal{L}$ = [[ ], ...,[ ]] : REPORT(Result), weiter bei **(4)** .

(2) Sei L_i erste nicht-leere Subgoal-Liste aus $\mathcal{L}$ = [L_1,...,L_m] .
Sei g_{i1} erstes Element aus L_i = [g_{i1}, ..., $g_{i,ni}$] :
- g_{i1} aus L_i entfernen: L'_i := [g_{i2}, ..., $g_{i,ni}$] .
- Falls keine unifizierbaren Klauseln für g_{i1}: weiter bei **(4)** .

(3) Sei k die nächste unifizierbare Klausel für g_{i1} mit Unifikator: σ .
Falls k Fakt: weiter bei **(1)** .
Falls k Regel: g_{i1} :- g_1, ..., g_n :
$\mathcal{L}$:= σ([[g_1, ..., g_n], L_1,.., L'_i,..., L_m]) .
Falls weitere Klauseln für g_{i1} existieren: *Backtrack-Punkt* setzen
Weiter bei **(2)** .

(4) Backtracking: Rücksetzen zum jüngsten *Backtrack-Punkt* :
$\mathcal{L}$ zurücksetzen auf Stand vor *Backtrack-Punkt* , weiter bei **(3)**
Falls kein *Backtrack-Punkt* existiert: EXIT(no).

Weitere Implementationsprobleme

- Für Vorwärtsreferenzen bei komplexeren Strukturen.
- Für Eingriffe in Beweisablauf (cut).
- Effiziensteigerung (vorzeitige Speicherfreigabe), z.B.
  - last call Optimierung („lco“)
  - deterministische Klauseln („dco“)

Später mehr dazu

Prolog-Compiler:
Übersetzung in optimiertes Programm
(WAM = Warren abstract machine)

Darstellung von Funktionen

Eine n-stellige Funktion f(x1,...,xn) ist als
(n+1)-stellige Relation r(x1,...,xn, f(x1,...,xn)) darstellbar

```
addiere(Summand1,Summand2,Summe)
nachfolger(Zahl,Nachfolger)
```

Der Funktionswert muss nicht an der letzten Stelle stehen.

```
multipliziere(Produkt,Faktor1,Faktor2)
```

Relationen und Funktionen

Modellierung von Aufgabenbereichen erfordert Strukturierung

- Beschreibung von Objekten
- Beschreibung von Zusammenhängen
- Beschreibung von Verfahren

Formale Modelle sind entscheidend für die
- Entwicklung
- Beschreibung
- Bewertung
- Validierung

von Konzepten und Verfahren

Modellierung (z.B. Sprachverarbeitung)

Strukturen z.B.:
- Wörter
- Satzstruktur (Syntax, Grammatik)
- Bedeutungsstruktur (Akteure, Handlung, ...)

Hans kauft das rote Fahrrad von Fritz.
Fritz verkauft sein rotes Fahrrad an Hans.
Fritz schenkt Hans sein rotes Fahrrad.
Hans klaut das rote Fahrrad von Fritz.

Verfahren z.B.
- Wörter erkennen
- Sätze erkennen
- Bedeutung erfassen

Modellierung

Sprache kann beschreiben

- Objekte (z.B. Substantive: Fahrrad)
- Eigenschaften (z.B. Adjektive: rot)
- Beziehungen (z.B. Verben: besitzt)

Hans besitzt ein rotes Fahrrad.

Prädikatenlogisch:

- Prädikate/Relationen beschreiben Eigenschaften und Beziehungen
- Terme beschreiben Objekte

Modellierung: Eigenschaften

Das rote Fahrrad ist drei Jahre alt, leicht reparaturbedürftig und in sehr gutem Zustand. Es kostet 100 € .

Attribut-Werte-Paare:

{ [Farbe,rot], [Alter, 3Jahre], [Zustand, sehr gut], [Funktionsfähigkeit, leicht reparaturbedürftig], [Preis, 100 €] }

Vektor von Attribut-Werten:

[rot, 3, sehr gut, leicht reparaturbedürftig, 100]

$\in W_{Farbe} \times W_{Alter} \times W_{zustand} \times W_{Funktionsfähigkeit} \times W_{Preis}$

Modellierung: Relation (Eigenschaften)

Fahrrad-Angebote

$$\subseteq W_{Farbe} \times W_{Alter} \times W_{zustand} \times W_{Funktionsfähigkeit} \times W_{Preis}$$

Farbe	Alter	Zustand	Fkts-fähig.	Preis
rot	3	sehr gut	Leicht rep.	100
rot	2	mittel	Ersatzteile	20
gelb	2	gut	ja	100

[blau, 1, sehr gut, sehr gut, 10] $\notin$ Fahrrad-Angebote

Fahrrad-Angebote(blau, 1, sehr gut, sehr gut, 10) = falsch