

# Klauseln

- Fakten

`kind(anton, berta).`

- Regeln

`mutter(M,X) :- kind(X,M), weiblich(M).`

- Anfragen

`?- mutter(M,anton).`

# Schreibung

Bisher kennen wir:

- Konstanten klein  
anton
- Prädikate klein  
kind
- Zahlen  
25
- Variablen groß  
Person

Dürfen Buchstaben a-z, A-Z, Ziffern 0-9 und Underscore \_ enthalten

# Datenstrukturen

In Prolog gibt es nur eine Datenstruktur: **Term**

- Einfache Terme
  - Konstanten (constants, atomic terms)
    - Atome
    - Zahlen
  - Variablen
- Strukturen (zusammengesetzte Terme)

# Atome

Eines der folgenden:

- Aus Zeichen a-zA-Z0-9\_ aufgebaut, beginnend mit einem Kleinbuchstaben.  
Beispiel: ein\_2ter\_Versuch
- Nur aus Sonderzeichen aufgebaut.  
Beispiele: := []
- In einfache Anführungszeichen eingeschlossen.  
Beispiel: 'Hans K. Mayer vom 5. Stock'  
Ggf. Escapes verwenden.  
Beispiel: 'Ein ''a\' häßlich geschrieben;\nZweite Zeile'

# Variable

muss sein

- Aus Zeichen a-zA-Z0-9\_ aufgebaut
- Mit einem Großbuchstaben oder \_ beginnend

Beispiele: X Ein\_2ter\_Versuch \_Zahl

„Anonyme Variable“

- \_ ist ein Platzhalter für eine (jeweils neue) Singleton Variable.
- $p(\_, \_)$  entspricht  $p(A, B)$ , nicht  $p(A, A)$ .

# Strukturen (Zusammengesetzte Terme)

haben die Form

- $f(t_1, \dots, t_n)$

wobei

- $f$  ein Atom ist. **Funktor**
- $t_1, \dots, t_n$  Terme sind. **Argumente**

Jedes Argument ist ein Term, also jeweils eines der folgenden:

- Atom
  - Zahl
  - Variable
  - Zusammengesetzter Term
- $n$  heißt die **Stelligkeit** des Funktors  $f$ .

## Beispiele für zusammengesetzte Terme

```
buch(Signatur, Autor, Titel)
```

```
buch(xyz379, 'Clocksin, Mellish', 'Programming in Prolog')
```

```
komplexe_Zahl(3,5)      (zur Darstellung der komplexen Zahl  $3 + 5i$ )
```

# Teilterme

## Definition

Die Argumente  $t_1, \dots, t_n$  eines zusammengesetzten Terms  $f(t_1, \dots, t_n)$  nennt man auch seine **unmittelbaren Teilterme**.

## Definition

Der Begriff des **Teilterms** eines Terms  $t$  ist folgendermaßen induktiv (d.h. durch Rekursion) definiert:

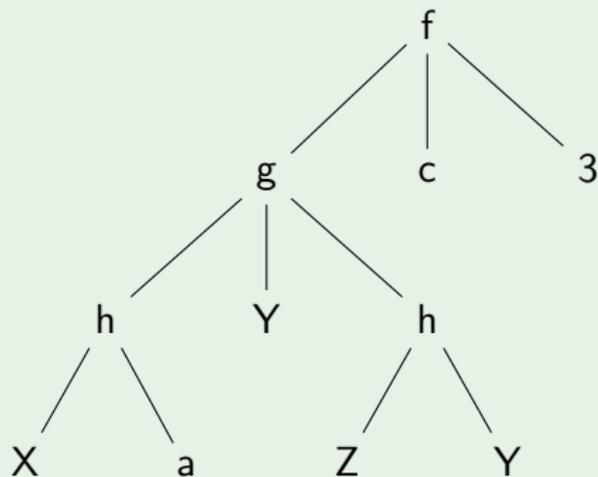
- $t$  ist ein Teilterm von  $t$ .
- Jeder Teilterm eines unmittelbaren Teilterms von  $t$  ist ein Teilterm von  $t$ .

## Beispiel

Die Teilterme von  $f(g(a), X)$  sind  $f(g(a), X)$ ,  $g(a)$ ,  $a$  und  $X$ .

## Strukturbaum eines Terms

Der Strukturbaum des Terms  $f(g(h(X,a),Y,h(Z,X)),c,3)$



## Operatoren: Infix-, Präfix-, Postfix-Schreibweise

### Deklaration eines Funktors als Operator

`:- op(Praezedenz,Spezifikator,Operatorname).`

- 2-stelliger Funktor als **Infix-Operator**
- 1-stelliger Funktor als **Präfix-** oder **Postfix-Operator**

### Beispiel

- `:- op(500,yfx,+).`
- Nun kann man statt `+(1,2)` auch `1+2` schreiben.

### Abfrage

`?- current_op(Pr,Sp,Op).`

# Präzedenz (Priorität)

## Präzedenz

- Zahl zwischen 1 und 1200
- Je kleiner die Präzedenz ist, desto stärker bindet der Operator.

## Beispiel

- $:- \text{op}(500, yfx, +)$ .
- $:- \text{op}(400, yfx, *)$ .
- $*$  bindet stärker als  $+$ . „Punkt vor Strich“
- Bei  $(2*3)+(4*5)$  dürfen die Klammern weggelassen werden.
- Bei  $(2+3)*4$  dürfen die Klammern nicht weggelassen werden.

# Spezifikator

?-  $op(Pr, Sp, Op)$ .    oder    :-  $op(Pr, Sp, Op)$ .

## Spezifikator

- Eines der folgenden Atome:  $fx$   $fy$   $xf$   $yf$   $xfx$   $xfy$   $yfx$
- Symbolisiert Position von Funktor und Argumenten.
- $f$  symbolisiert den Funktor.
- $x$  und  $y$  symbolisieren die Argumente.
- $x$  oder  $y$  gibt Assoziativität an.
- $x$ : Klammern um das Argument dürfen weggelassen werden, wenn die Präzedenz des Hauptfunktors des Arguments kleiner als die Präzedenz von  $Op$  ist.
- $y$ : Klammern um das Argument dürfen weggelassen werden, wenn die Präzedenz des Hauptfunktors des Arguments kleiner oder gleich der Präzedenz von  $Op$  ist.

# Spezifikator

## Beispiel

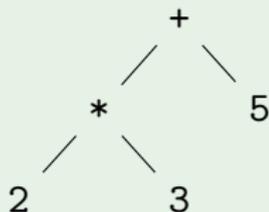
- $:- \text{op}(400, \text{yfx}, *) .$
- $:- \text{op}(400, \text{yfx}, /) .$
- Beim Term  $(2*3)/4$  dürfen die Klammern weggelassen werden.
- $*$  und  $/$  sind linksassoziativ.
- Ausdrücke damit ohne Klammern sind linksgeklammert zu verstehen:
- $a/b/c*d = ((a/b)/c)*d$ .

# Operatorname

Der Operatorname muss ein Atom sein.

## Beispiel für einen Term mit Operatoren

- $2*3+5$  ist Operatorschreibweise für den Term
- $+(*(2,3),5)$  (kanonische Schreibweise)
- Der Strukturbaum:



- Beispiel einer Anfrage:  
?- X is 2\*3+5.  
X = 11

## Beispiel für einen Term mit Operatoren

### Verschiedene Schreibweisen für einen Term

- $-(-(6, 1), 3)$
- $(6-1)-3$
- $6-1-3$

# Logische und prozedurale Sichtweise

nachkomme(N,X) :- kind(N,X).

nachkomme(N,X) :- kind(K,X), nachkomme(N,K).

## Logische Sichtweise

- nachkomme(N,X), wenn kind(N,X).
- nachkomme(N,X), wenn kind(K,X) und nachkomme(N,K).

## Beweisprozedur

- Um nachkomme(N,X) zu beweisen, genügt es, kind(N,X) zu beweisen.
- Um nachkomme(N,X) zu beweisen, genügt es, kind(K,X) und nachkomme(N,K) zu beweisen.

## Logische und prozedurale Sichtweise

```
nachkomme(N,X) :- kind(N,X).
```

```
nachkomme(N,X) :- kind(K,X), nachkomme(N,K).
```

### Prozedurale Sichtweise

- Um das Ziel `nachkomme(N,X)` zu lösen, löse `kind(N,X)`.
- Um `nachkomme(N,X)` zu lösen,  
löse `kind(K,X)` und `nachkomme(N,K)`.

# Listen

- In Prolog können wir mehrere Terme zu einer Liste zusammenfassen, z.B. die Terme a, b und c zur Liste [a,b,c].
- a, b und c sind die *Elemente* der Liste [a,b,c].
- Eine Liste ist aufgebaut aus ihren Elementen und der leeren Liste [] mit dem zweistelligen Funktor ./2 wie folgt:<sup>1</sup>  
[a,b,c] = .(a, .(b, .(c, [])))
- Für Listen wird statt der Schreibweise mit Funktor die einfachere *Listenschreibweise* verwendet, z.B.
  - [a,b,c] statt .(a, .(b, .(c, [])))
  - [X|Xs] statt .(X,Xs)
  - [X,Y|Xs] statt .(X, .(Y, Xs))
  - [X,Y,Z|Xs] statt .(X, .(Y, .(Z, Xs)))
- Listen sind ganz normale Terme und die Listenschreibweise ist nur eine abkürzende Schreibweise.

---

<sup>1</sup>In SWI-Prolog ab Version 7 heißt der Funktor [ ]/2 statt ./2

# Induktive Definition des Begriffs der Liste

## Definition

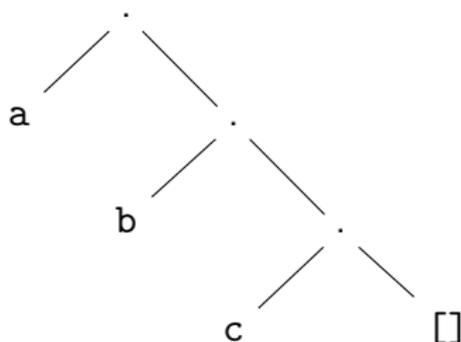
- 1  $[]$  ist eine Liste.
- 2  $[X|Xs]$  ist eine Liste, wenn  $X$  ein Term und  $Xs$  eine Liste ist.

## Beispiel

- Nach 1. ist  $[]$  eine Liste.
- Daher ist nach 2. auch  $[c|[]]$ , also  $[c]$  eine Liste.
- Daher ist nach 2. auch  $[b|[c]]$ , also  $[b,c]$  eine Liste.
- Daher ist nach 2. auch  $[a|[b,c]]$ , also  $[a,b,c]$  eine Liste.

## Der Strukturbaum einer Liste

Der Strukturbaum der Liste [a,b,c] ist<sup>2</sup>



---

<sup>2</sup>In SWI-Prolog ab Version 7 wird statt dem Punkt `.` das Prolog-Atom `[]` als Name des Funktors verwendet.

## Beispiele für Anfragen zu Listen

?- functor([a,b,c],F,N).

F = ' . ',

N = 2.

?- arg(1,[a,b,c],X).

X = a.

?- arg(2,[a,b,c],X).

X = [b, c].

?- [a,b,c] =.. X.

X = [ ' . ', a, [b, c] ].

## Beispiele für Anfragen zu Listen

?- [X|Xs] = [a,b,c].

X = a,

Xs = [b, c].

?- [X,Y|Xs] = [a,b,c].

X = a,

Y = b,

Xs = [c].

?- [X,Y,Z|Xs] = [a,b,c].

X = a,

Y = b,

Z = c,

Xs = [].

# Programmieren mit Listen

Typischerweise zwei Fälle entsprechend der induktiven Definition des Begriffs der Liste behandeln:

- 1 Leere Liste []
- 2 Nichtleere Liste [X|Xs]

## Beispiel: Programm zur Bestimmung der Länge einer Liste

```
listenlaenge([],0).  
listenlaenge([_|Xs],N) :-  
    listenlaenge(Xs,M),  
    N is M+1.
```