

Übersicht

Operatoren

sind Funktoren, an die nicht direkt eine Klammer folgt.

- ◆ Infix-, Präfix-, Postfix-Schreibweise
- ◆ Präzedenz, Assoziativität, Position
- ◆ Vordefinierte und selbstdefinierte Operatoren
- ◆ Operator- vs. Funktor-Argument-Schreibweise
- ◆ Regelklauseln als Terme!

Operatoren – 1

Infix-Schreibweise

In Operator-Schreibweise sind gewisse Ausdrücke besser lesbar als in Funktor-Argument-Schreibweise.

- ◆ Zum Beispiel Infix-Schreibweise in Arithmetik

$-(12, -(3, 2))$

$+(*(2, 3), 4)$

$12-3-2$

$2*3+4$

- ▶ Allerdings bringen Operatoren zusätzlich das Problem der Präzedenz und Assoziativität mit sich!

Operatoren – 2

Operatoren: Man nehme ...

Zu jeder Definition eines Operators gehört

- ◆ **Präzedenz** (*precedence; priority*)
 - ◆ Zahl zwischen 0 und 1200
 - ◆ Je kleiner die Zahl, desto höher die Präzedenz (!).
- ◆ **Position**
 - ◆ Präfix-Operator: Operator steht *vor* Argument.
 - ◆ Infix-Operator: Operator steht *zwischen* Argumenten.
 - ◆ Postfix-Operator: Operator steht *nach* Argument.
- ◆ **Assoziativität** (*associativity*)
 - ◆ linksassoziativ: $a \cdot b \cdot c = (a \cdot b) \cdot c$
 - ◆ rechtsassoziativ: $a \cdot b \cdot c = a \cdot (b \cdot c)$
 - ◆ nicht schachtelbar: $a \cdot b \cdot c$ ist kein zulässiger Term!
- ◆ **Name** – Atom

Operatoren – 3

Position und Assoziativität

f: Funktor

x: nicht-assoziativ

y: assoziativ

Angabe	Bedeutung
$f x$	Präfix, nicht schachtelbar
$f y$	Präfix, rechtsassoziativ
$x f$	Postfix, nicht schachtelbar
$y f$	Postfix, linksassoziativ
$x f x$	Infix, nicht schachtelbar
$x f y$	Infix, rechtsassoziativ
$y f x$	Infix, linksassoziativ

Operatoren – 4

Vordefinierte Operatoren nach ISO-Standard

Präzedenz	Position/Assoz.	Operatoren
1200	xfx	:- -->
1200	fx	:- ?-
1100	xfy	;
1050	xfy	->
1000	xfy	','
900	fy	\+ spy
700	xfx	= \= == \== @< @=< @> @>= is := =\= < > >= > =..
500	yfx	+ - /\ \/
400	yfx	* / // rem mod << >>
200	xfx	^
200	fy	\ -

Operatoren – 5

Selbstdefinierte Operatoren

Selbstdefinierte Operatoren brauchen Deklaration...

```
:- op(600, xfx, vater).
```

Präzedenz ↑
Infix, nicht assoziativ ↑
Name →

```
:- op(600, xfx, vater).  
hans vater gabi.  
hans vater kevin.
```

Mit `current_op/3` lassen sich die gegenwärtig definierten Operatoren ausgeben.

```
?- current_op(600, A, F).  
A = xfx  
F = vater  
yes
```

Operatoren – 6

Operator vs. Funktor-Argument-Notation

Die Operator-Schreibweise ist ein syntaktischer Zucker (*syntactic sugar*).

- Für jede Operator-Schreibweise eines Terms gibt es eine Schreibweise in der Funktor-Argument-Form.
- Das eingebaute Prädikat `write_canonical/1` schreibt die Funktor-Argument-Notation jedes Terms heraus:

```
?- write_canonical(12/3/2).  
/(/(12,3),2)
```

- Operatoren-Prädikate sind auch in der Funktor-Argument-Form aufrufbar:

```
?- =(A,a).  
A = a
```

Operatoren – 7

Regelklauseln sind Operator-Terme

Regel-Klauseln in Funktor-Argument-Notation

- Die Symbole

```
:- falls  
, und  
; oder
```

```
frau(Jemand) :-  
person(Jemand),  
weiblich(Jemand).
```

sind vordefinierte zweistellige Operatoren.

Regel in Operatorschreibweise

```
:- (frau(Jemand), ', ' (person(Jemand), weiblich(Jemand))).
```

Regel in Funktor-Argument-Notation

- Die Weglassung der Hochkommata bei ", " in Operatorschreibweise ist echte Spezialsyntax!
- Der Punkt am Schluss von Fakten und Regeln ist kein Operator! Er ist das Zeichen, das Term-Enden markiert!

Operatoren – 8