



Programowanie deklaratywne

Artur Michalski
Informatyka II rok



Plan wykładu

- Wprowadzenie do języka Prolog
- Budowa składniowa i interpretacja programów prologowych
- Listy, *operatory i operacje arytmetyczne*
- Złożone/abstrakcyjne struktury danych
- Sterowanie mechanizmem nawrotów
- Operacje wejścia/wyjścia w Prologu
- Predefiniowane procedury prologowe
- Styl i technika programowania w Prologu

Notacja operatorów

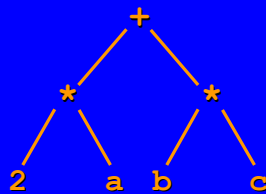
Standardowa reprezentacja operatorów w Prologu

W Prologu operatory reprezentowane są w notacji *prefiksowej*.

Przykład

$+ (* (2, a), * (b, c))$

Wyrażenia tworzą w reprezentacji wewnętrznej struktury drzewiaste (są termami), a operatory pełnią rolę funktorów (są atomami):



Notacja operatorów

Standardowa reprezentacja operatorów w Prologu

Dopuszczalna jest alternatywna reprezentacja operatorów w notacji *infiksowej*.

Przykład

$2*a + b*c$

Wyrażenia w notacji *infiksowej* przekształcane są automatycznie do postaci *prefiksowej* (i na odwrót).

Pierwszeństwo operatorów decyduje o interpretacji wyrażeń w notacji infiksowej.

Notacja operatorów

Definiowanie operatorów w Prologu

Prolog dopuszcza możliwość definiowania nowych operatorów. W celu zdefiniowania operatora korzystamy z specjalnej klauzuli systemowej zwanej również *dyrektywą*.

Postać dyrektywy:

```
op (<pierwszeństwo>, <składnia>, <symbol>)
```

gdzie:

<pierwszeństwo> - klasa pierwszeństwa operatora,

<składnia> - budowa operatora (prefix, infix, suffix),

<symbol> - oznaczenie operatora.

Przykład

```
?- op (600, xfx, has) .
```

Można teraz zdefiniować fakt np.: `piotr has auto.`

Notacja operatorów

Definiowanie operatorów w Prologu c.d.

Zasady:

- Definicja operatora nie określa żadnej operacji, która będzie wykonywana na argumentach operatora
- Operatory definiowane w Prologu są tylko funktorami, służącymi do konstruowania bardziej złożonych struktur
- Klasa pierwszeństwa operatora może przyjmować wartości z zakresu od 1 do 1200
- Oznaczenie operatora musi być nazwą (stałą symboliczną)

Notacja operatorów

Definiowanie operatorów w Prologu c.d.

Budowa składniowa operatora - notacje:

- prefiksowa:

$f\ x\ fy$

- infiksowa:

$xf\ x\ xfy\ yfx$

- postfiksowa:

$xf\ yf$

gdzie:

f - to operator

x - to argument o klasie pierwszeństwa $< f$

y - to argument o klasie pierwszeństwa $\leq f$

Notacja operatorów

Definiowanie operatorów w Prologu c.d.

Zasady pierwszeństwa argumentów i operatora:

Składnik wyrażenia	Klasa pierwszeństwa
argument prosty	0
argument w nawiasach	0
operator	wg definicji (dyrektywa op)
argument złożony	wg pierwszeństwa operatora (funktora) głównego

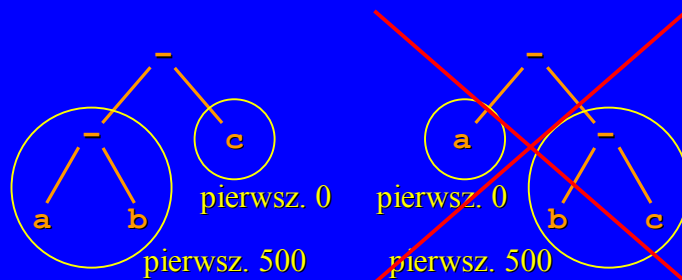
Notacja operatorów

Definiowanie operatorów w Prologu c.d.

Przykład

Załóżmy, że `op(500, yfx, -)`.

Reprezentacja wyrażenia `a-b-c` to `(a-b)-c`, a nie `a-(b-c)`.



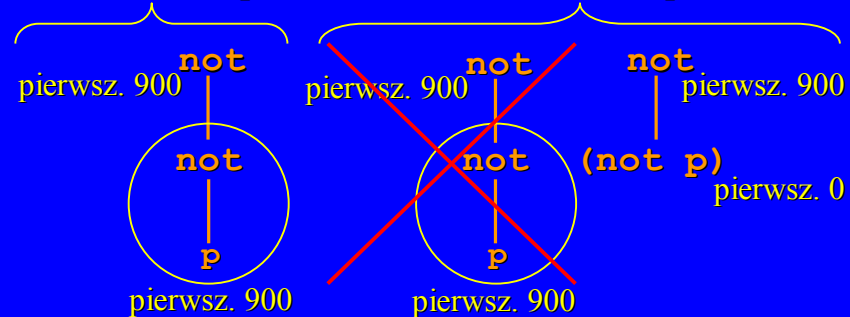
Notacja operatorów

Definiowanie operatorów w Prologu c.d.

Przykład

`op(900, fy, not) .`
`not not p`

`op(900, fx, not) .`
`not (not p)`



Notacja operatorów

Definiowanie operatorów w Prologu c.d.

Przykład

Wyrażenie rachunku zdań (prawo de Morgana):

$$\sim (A \& B) \Leftrightarrow \sim A \vee \sim B$$

Zapis w Prologu (standardowy, prefiksowy):

equivalence (not (and (A, B)), or (not (A) , not (B))) .

Zapis w Prologu (nowe operatory):

?- op (800 , xfx , <=>) .

?- op (700 , xfy , v) .

?- op (600 , xfy , &) .

?- op (500 , fy , ~) .

Można teraz zdefiniować term: $\sim (A \& B) \Leftrightarrow \sim A \vee \sim B$.

Notacja operatorów

Definiowanie operatorów w Prologu c.d.

Uwagi

- Definicje nowych operatorów nie określają żadnych nowych działań na argumentach, lecz wzbogacają notację o nowe sposoby tworzenia złożonych form reprezentacji
- Operator o najwyższej klasie pierwszeństwa jest głównym operatorem wyrażenia złożonego; operatory o niższej klasie pierwszeństwa mają natomiast silniejsze wiązanie
- Specyfika operatora zależy zarówno od jego położenia względem argumentów, jak i od zasad pierwszeństwa operatora oraz jego argumentów

Operacje arytmetyczne w Prologu

Operacja przypisania w Prologu

= (znak równości) operacja uzgadniania (unifikacji) termów
is operacja przypisania (ewaluacji wyrażenia)

Przykład

?- **X=1+2.**

Odpowiedź:

X=1+2

term z funktorem +
oraz argumentami 1 i 2

?- **X is 1+2.**

Odpowiedź:

X=3

predefiniowana procedura
ewaluacji wyrażenia **is**

Operacje arytmetyczne w Prologu

Podstawowe operacje arytmetyczne w Prologu

Oznaczenie operatora	Operacja arytmetyczna
+	dodawanie
-	odejmowanie
*	mnożenie
/	dzielenie rzeczywiste
//	część całkowita z dzielenia
mod	reszta całkowita z dzielenia
rem	część ułamkowa z dzielenia
** lub ^	potęgowanie

Operacje arytmetyczne w Prologu

Podstawowe operacje porównania w Prologu

Oznaczenie operatora	Operacja porównania
>	większy
<	mniejszy
>=	większy lub równy
=<	mniejszy lub równy
==	równy
==\	różny
\=	termy się nie unifikują

Operacje arytmetyczne w Prologu

Operacja równości (==) a unifikacja (=)

Operator = dokonuje uzgodnienia dwóch obiektów i jeśli jest ono możliwe, prowadzi do wiązania zmiennych w tych obiektach (bez obliczania wartości wyrażeń!).

Operator == powoduje obliczenie wartości argumentów bez wiązania zmiennych (muszą być one już związane!).

Przykład

?- 1+2==2+1.

Yes

?- 1+A=B+2.

A=2

B=1

?- 1+2=2+1.

No

?- 1+A==B+2.

ERROR:Arguments are not
sufficiently instantiated

Różne rodzaje operacji równości w Prologu

Rodzaje relacji równości/różności w Prologu:

- $X=Y$ spełniony, gdy termy X i Y unifikują się
- $X \text{ is } E$ spełniony, gdy X unifikuje się z wartością wyrażenia E (ewaluacja E potem proces uzgadniania z X)
- $E1 == E2$ spełniony, gdy wartości wyrażeń arytmetycznych $E1$ i $E2$ są równe
- $E1 \neq E2$ spełniony, gdy wartości wyrażeń arytmetycznych $E1$ i $E2$ są różne
- $T1 == T2$ spełniony, gdy termy $T1$ i $T2$ są identyczne (unifikują się leksykalnie włącznie z nazwami zmiennych)
- $T1 \neq T2$ spełniony, gdy termy $T1$ i $T2$ nie są identyczne

Różne rodzaje operacji równości w Prologu

Zastosowanie relacji równości/różności w Prologu:

Przykład

?- $f(a,b) == f(a,b)$.

Yes

?- $f(a,b) == f(a,X)$.

No

?- $f(a,X) == f(a,Y)$.

No

?- $X \neq Y$.

Yes

?- $t(X, f(a,Y)) == t(X, f(a,Y))$.

Yes

Operacje arytmetyczne w Prologu

Zastosowanie operacji arytmetycznych

Przykład

Największy wspólny dzielnik dwóch liczb: dla dwóch liczb całkowitych, dodatnich X i Y , największy wspólny dzielnik D :

- równa się X , jeżeli X i Y są równe,
- równa się największemu wspólnemu dzielnikowi X i $Y-X$, jeżeli $X < Y$,
- równa się największemu wspólnemu dzielnikowi Y i $X-Y$, jeżeli $Y < X$.

Zapis w Prologu:

```
nwd(X,X,X) .  
nwd(X,Y,D) :- X<Y, Y1 is Y-X, nwd(X,Y1,D) .  
nwd(X,Y,D) :- Y<X, nwd(Y,X,D) . lub  
nwd(X,Y,D) :- Y<X, X1 is X-Y, nwd(X1,Y,D) .
```

Operacje arytmetyczne w Prologu

Zastosowanie operacji arytmetycznych c.d.

Przykład

Wyznaczyć długość listy termów. Długość listy:

- równa się 0, jeżeli lista jest pusta,
- równa się długości jej ogona powiększonej o 1, jeżeli lista jest niepusta.

Zapis w Prologu:

```
length([],0) .  
length([_|Tail],N) :- length(Tail,N1) ,  
N is 1+N1 .
```

Operacje arytmetyczne w Prologu

Zastosowanie operacji arytmetycznych c.d.

ciąg dalszy przykładu – rozważmy inny (błędny) zapis w Prologu:

```
length([],0).  
length([_|Tail],N):- length(Tail,N1),  
                    N = 1+N1.
```

↑
unifikacja!

Zapytanie: ?- length([a,b,[c,d],e],N).

N = (1+(1+(1+(1+0))))

Właściwe zapytanie (w tej wersji length!):

?- length([a,b,[c,d],e],N), L is N.

N=(1+(1+(1+(1+0))))

L=4

Operacje arytmetyczne w Prologu

Zastosowanie operacji arytmetycznych c.d.

Podsumowanie

- Operacje z użyciem operatorów arytmetycznych wymagają zastosowania predefiniowanych procedur ewaluacji wyrażenia
- Ewaluacja wyrażenia jest możliwa tylko gdy zastosujemy operator **is**
- Operatory porównania również prowadzą do ewaluacji porównywanych wyrażeń
- W trakcie ewaluacji wyrażenia wszystkie argumenty muszą mieć przypisaną (związaną) wartość liczbową
- Realizacja procedury **is** przebiega w dwóch krokach: ewaluacja wyrażenia, a następnie *unifikacja* zmiennej do wartości tego wyrażenia