Programowanie w logice

Operatory

Piotr Fulmański

Wydział Matematyki UŁ

14 marca 2007

Plan prezentacji

Program Prologu składa się z termów.

Term to:

- stała,
- zmienna,
- struktura.

Każdy term zapisywany jest jako ciąg znaków, tworzących cztery kategorie:

- duże litery: A-Z
- małe litery: a-z
- cyfry: 0-9
- znaki specjalne:

Program Prologu składa się z termów.

Term to:

- stała,
- zmienna,
- struktura.

Każdy term zapisywany jest jako ciąg znaków, tworzących cztery kategorie:

- duże litery: A-Z
- małe litery: a-z
- cyfry: 0-9
- znaki specjalne:

Program Prologu składa się z termów.

Term to:

- stała,
- zmienna,
- struktura.

Każdy term zapisywany jest jako ciąg znaków, tworzących cztery kategorie:

- duże litery: A-Z
- małe litery: a-z
- cyfry: 0-9
- znaki specjalne:

Stałą nazywamy konkretne obiekty lub relacje. Istnieją dwa rodzaje stałych: atomy i liczby.
Przykład: `jas, malgosia, posiada, je`

Atomy składają się z:

- symboli, np. ?- lub :- ,
- liter, cyfr znaku podkreślenia (zaczynamy zawsze małą literą),
- dowolnych znaków, jeśli ujęte są w pojedyńczy cudzysłów.

Liczby: -17, 23, 99.9, 123e-3

Stałą nazywamy konkretne obiekty lub relacje. Istnieją dwa rodzaje stałych: atomy i liczby.
Przykład: `jas, malgosia, posiada, je`

Atomy składają się z:

- symboli, np. `?-` lub `:-` ,
- liter, cyfr znaku podkreślenia (zaczynamy zawsze małą literą),
- dowolnych znaków, jeśli ujęte są w pojedyńczy cudzysłów.

Liczby: -17, 23, 99.9, 123e-3

Stałą nazywamy konkretne obiekty lub relacje. Istnieją dwa rodzaje stałych: atomy i liczby.
Przykład: `jas, malgosia, posiada, je`

Atomy składają się z:

- symboli, np. `?-` lub `:-` ,
- liter, cyfr znaku podkreślenia (zaczynamy zawsze małą literą),
- dowolnych znaków, jeśli ujęte są w pojedyńczy cudzysłów.

Liczby: -17, 23, 99.9, 123e-3

Zmienne mają postać atomu złożonego z liter, cyfr lub znaku podkreślenia z zastrzeżeniem, że pierwszym znakiem musi być duża litera.

Czasem interesuje nas tylko czy coś jest prawdą, ale zupełnie nie interesuje nas co.
Czy ktoś lubi Jasia?
W takich sytuacjach możemy użyć zmiennej anonimowej zapisywanej jako jeden znak podkreślenia.

```
?- lubi(_,jas).
```

Zmienne mają postać atomu złożonego z liter, cyfr lub znaku podkreślenia z zastrzeżeniem, że pierwszym znakiem musi być duża litera.

Czasem interesuje nas tylko czy coś jest prawdą, ale zupełnie nie interesuje nas co.
Czy ktoś lubi Jasia?
W takich sytuacjach możemy użyć zmiennej anonimowej zapisywanej jako jeden znak podkreślenia.

```
?- lubi(_,jas).
```

Struktura, inaczej term złożony, to obiekt złożony z innych obiektów.

```
posiada(piotr,auto).
posiada(marcin,auto).
```

Przykład struktury

```
posiada(piotr,auto(nissan,almera)).
posiada(marcin,auto(fiat,punto)).
```

```
?- posiada(X,auto(nissan,Y)).

X = piotr
Y = almera ;
```

Struktura, inaczej term złożony, to obiekt złożony z innych obiektów.

```
posiada(piotr,auto).
posiada(marcin,auto).
```

Przykład struktury

```
posiada(piotr,auto(nissan,almera)).
posiada(marcin,auto(fiat,punto)).
```

```
?- posiada(X,auto(nissan,Y)).

X = piotr
Y = almera ;
```

Struktura, inaczej term złożony, to obiekt złożony z innych obiektów.

```
posiada(piotr,auto).
posiada(marcin,auto).
```

Przykład struktury

```
posiada(piotr,auto(nissan,almera)).
posiada(marcin,auto(fiat,punto)).
```

```
?- posiada(X,auto(nissan,Y)).

X = piotr
Y = almera ;
```

Struktura, inaczej term złożony, to obiekt złożony z innych obiektów.

```
posiada(piotr,auto).
posiada(marcin,auto).
```

Przykład struktury

```
posiada(piotr,auto(nissan,almera)).
posiada(marcin,auto(fiat,punto)).
```

```
?- posiada(X,auto(nissan,Y)).

X = piotr
Y = almera ;
```

Zapis `a-b*c` to inna forma zapisu `-(a,*(b,c))`.

Uwaga

Operator arytmetyczny nie powoduje wykonania jakichkolwiek obliczeń.
1+2 to nie jest to samo co 3
1+2 to inny zapis termu `+(1,2)`

Kiedy własny operator

```
lubi(jas,malgosie).
jas lubi malgosie.
```

Zapis `a-b*c` to inna forma zapisu `-(a,*(b,c))`.

Uwaga

Operator arytmetyczny nie powoduje wykonania jakichkolwiek obliczeń.
1+2 to nie jest to samo co 3
1+2 to inny zapis termu `+(1,2)`

Kiedy własny operator

```
lubi(jas,malgosie).
jas lubi malgosie.
```

Zapis `a-b*c` to inna forma zapisu `-(a,*(b,c))`.

Uwaga

Operator arytmetyczny nie powoduje wykonania jakichkolwiek obliczeń.
1+2 to nie jest to samo co 3
1+2 to inny zapis termu `+(1,2)`

Kiedy własny operator

```
lubi(jas,malgosie).
jas lubi malgosie.
```

Operator jest atomem wiążącym kilka operandów, podobnie jak struktura, z tym wyjątkiem, że możemy zapisać go także w jednej z następujących postaci:

- prefixowej, np. `+1`,
- postfixowej, np. `x!`,
- infixowej, np. `1*2`.

Definiując nowy operator, oprócz zdecydowania się na jedną z wcześniej opisanych postaci, ustalić musimy jego priorytet i łączność

- priorytet: od 1 do 1200 (im mniej tym większy)
- łączność: lewo- lub prawostronna
  (1-2-3 interpretowane jest jako ((1-2)-3) ponieważ - jest operatorem lewostronnie łącznym);
  łączność definiujemy za pomocą jednego z atomów: xf, yf, xfx, xfy, yfx, fx lub fy.

Operator jest atomem wiążącym kilka operandów, podobnie jak struktura, z tym wyjątkiem, że możemy zapisać go także w jednej z następujących postaci:

- prefixowej, np. `+1`,
- postfixowej, np. `x!`,
- infixowej, np. `1*2`.

Definiując nowy operator, oprócz zdecydowania się na jedną z wcześniej opisanych postaci, ustalić musimy jego priorytet i łączność

- priorytet: od 1 do 1200 (im mniej tym większy)
- łączność: lewo- lub prawostronna
  (1-2-3 interpretowane jest jako ((1-2)-3) ponieważ - jest operatorem lewostronnie łącznym);
  łączność definiujemy za pomocą jednego z atomów: xf, yf, xfx, xfy, yfx, fx lub fy.

```
xf    postfix   (ang. non-associative)
yf    postfix   lewostronny
xfx   infix     n-a
xfy   infix     prawostronny
yfx   infix     lewostronny
fx    prefix    n-a
fy    prefix    prawostronny
```

```
lubi(jas,malgosie).

?- [p1].
% p1 compiled 0.00 sec, 852 bytes

Yes
?- lubi(jas,malgosie).

Yes
?- jas lubi malgosie.
ERROR: Syntax error: Operator expected
ERROR: jas
ERROR: ** here **
ERROR: lubi malgosie .
?- op(500, xfx, lubi).

Yes
?- jas lubi malgosie.
```

```
?- X is 2+3*5.

X = 17

Yes
?- op(100,yfx,+).

Yes
?- X is 2+3*5.

X = 25

Yes
?-
```

Język prolog posiada wsparcie do wykonywania prostych operacji arytmetycznych przy pomocy wbudowanych predykatów. Predykaty te nazywamy systemowymi. Pomimo pozornego podobieństwa do reszty predykatów, te są zaszyte bezpośrednio w kodzie środowiska, przez co działają szybciej i bardziej efektywnie niż predykaty napisane w czystym Prologu. W sytuacjach gdy mamy wybór pomiędzy predykatami wbudowanymi, a predykatami dodanymi warto wybierać predykaty wbudowane.

Jak wiadomo z ćwiczeń, możemy zapisać następujące zdanie:
`(X is 1+2).`
w konsekwencji czego zostanie wykonanych tutaj kilka czynnosci:

- zostanie wyliczona wartość wyrażenia 1+2, co doprowadzi nasze zapytanie do postaci
  `X is 3`
- `is` zostanie zinterpretowane jako podstawienie `X:=3`
- zostanie zwrócona wartość `Yes` informaująca o tym, że `X` i 3 mają teraz taką samą wartość.

Jeśli napiszemy
`3 is 1+2`
wówczas `is` nie zadziała jako „operator” przypisania a spowoduje jedynie wykonanie porównania wartości z wyrażeniem. Zwróconą wartością będzie `Yes`.

Jeśli napiszemy

```
1+2 is 1+2
```

wówczas zwrócona zostanie wartość `No`. Wynika to z faktu, że oba argumenty są wyrażeniami. Wyrażenie z prawej strony zostanie obliczone dając w wyniku liczbę 3. Wyrażenie z lewej strony pozostanie bez zmian, przez co nie będzie reprezentowane przez wartość 3.

Do porównywania zamiast `is` możemy używać takż operatora `=:=`, który w odróżnieniu od `is` pozwala na umieszczenie wyrażeń arytmetycznych w miejscu obydwu swoich argumentów. Możemy wiec napisać:
`(3+2 =:= 4+1).`
co zostanie poprawnie zinterpretowane i zwróci wartość `'Yes'`.

Prolog udostępnia także inne predykaty wyglądające jak operatory służące do porównywania wyrażeń arytmetycznych. Są nimi:

- mniejszy <
- większy >
- mniejszy równy <=
- większy równy >=

Mogą one przyjmować wyrażenia arytmetyczne zarówno w pierwszym jak i drugim argumencie. W związku z tym da się wyliczyć wartość zapytań takich jak:

```
(1+2 < 2+3).
(1+2*3 >= 3+3).
(1 < 4).
(N < N+1).
```

a jeśli wartości `X`, `Y` i `Z` są określone można również zapisać i wyliczyć

```
(X >= Y+Z).
(X+1 < Y*2+X).
```

Algorytm Euklidesa

```
nwd(X,0,X).
nwd(X,Y,Nwd) :- Y > 0, R is X mod Y, nwd(Y,R,Nwd).
```

Do zapamiętania

- Czym są w Prologu stałe, zmienne struktury? Podaj przykład.
- Co to jest operator?
- Co określa operator? Podaj przykład własnego operatora i jego użycia.
- Jak działa predykat `is`; czym różni się od operatora `=:=`?
- Podaj przykład prostego programu obliczeniowego (np. silnia) i omów jego działanie.

Do zapamiętania

- Czym są w Prologu stałe, zmienne struktury? Podaj przykład.
- Co to jest operator?
- Co określa operator? Podaj przykład własnego operatora i jego użycia.
- Jak działa predykat `is`; czym różni się od operatora `=:=`?
- Podaj przykład prostego programu obliczeniowego (np. silnia) i omów jego działanie.

Do zapamiętania

- Czym są w Prologu stałe, zmienne struktury? Podaj przykład.
- Co to jest operator?
- Co określa operator? Podaj przykład własnego operatora i jego użycia.
- Jak działa predykat `is`; czym różni się od operatora `=:=`?
- Podaj przykład prostego programu obliczeniowego (np. silnia) i omów jego działanie.

Do zapamiętania

- Czym są w Prologu stałe, zmienne struktury? Podaj przykład.
- Co to jest operator?
- Co określa operator? Podaj przykład własnego operatora i jego użycia.
- Jak działa predykat `is`; czym różni się od operatora `=:=`?
- Podaj przykład prostego programu obliczeniowego (np. silnia) i omów jego działanie.

Do zapamiętania

- Czym są w Prologu stałe, zmienne struktury? Podaj przykład.
- Co to jest operator?
- Co określa operator? Podaj przykład własnego operatora i jego użycia.
- Jak działa predykat `is`; czym różni się od operatora `=:=`?
- Podaj przykład prostego programu obliczeniowego (np. silnia) i omów jego działanie.