Prolog
Składnia i struktury danych

Składnia

- Programy w Prologu budowane są za pomocą termów.
- Term może być stałą, zmienną, lub strukturą.
- Każdy term zapisywany jest jako ciąg znaków. Mozna używać dużych lub małych liter, cyfr oraz znaków specjalnych

Składnia | Stałe

- Stałe określają obiekty lub relacje.
- Wyróżnia się dwa typy stałych. Atomy I liczby.

Składnia | Atomy

Atom jest ciąggiem znaków utworzonym z

- małych i dużych liter, cyfr i znaku podkreślenia z zastrzeżeniem, że pierwszym znakiem musi być mała litera, np

  a, aLA, x_y_z, abc10

- dowolnego ciągu znaków ujętego w apostrofy, np

  'To jest atom'

- symboli

  ?#-

Składnia | Liczby

W prologu dostępne są liczby całkowite I rzeczywiste np:

-16, 57, 99.9, 123e-8

Liczba "e" w prologu oznacza potęgę 10 więc 123e8 oznacza $123 * 10^8$

Składnia | Zmienne

Zmienna jest ciągiem znaków utworzonym z małych i dużych liter, cyfr i znaku podkreślenia zastrzeżeniem, że pierwszym znakiem musi być duża litera lub znak podkreślenia, np.

X, Kto, _123, X_1_2, _

Pojedynczy znak podkreślenia, określa się zmienną anonimową. Korzystamy z niej zawsze wtedy, gdy interesuje nas tylko czy coś jest prawdą, ale interesuje nas co, np

Czy ktoś lubi Marka?

?- lubi(_,Marek).

Składnia | Zmienne

Należy pamiętać, że wielokrotnym wystąpieniom zmiennej anonimowej w jednym wyrażeniu mogą być przypisane różne wartości np.

```
?- a(1,2)=a(X,Y).
X = 1, Y = 2
?- a(1,2)=a(_,_).
Yes
```

Składnia | Struktury

Struktury w prologu nazywane są termami złożonymi czyli obiektami złożonymi z innych obiektów czyli atomów, liczb, zmiennych oraz innych struktur.

Przykład: f(arg_1,...,arg_n)

W tym przypadku arg_1, … , arg_n są termami, zas f jest atomem (nazwą relacji)

Składnia | Struktury

Dzięki takim wyrażeniom możemy lepiej opisać interesujący nas problem np.

posiada(piotr,auto).

posiada(jurek,auto).

Pozwala to stwierdzić że obiekty "piotr" I "jurek" związane są relacją "posiada" z obiektem "auto".

Czyli Piotr I Jurek mają auto.

Nie wiadomo jakie jest to auto i czy to nie jest to samo auto.

Składnia | Struktury

Możemy zapisać fakt używając struktury w środku innej struktury np.

posiada(piotr,auto(nissan)).

posiada(jurek,auto(bmw)).

Teraz mamy możliwość zapytać bardziej szczegółowo np. jaka jest marka auta

?- posiada(piotr,auto(X)).

X = nissan

Operatory

Operatory w Prologu posiadają cechy takie jak w prawdziwej matematyce czyli:

- Priorytet
- Pozycja
- Łączność

Operatory | Priorytet

Priorytet czyli kolejność wykonywania działań.

W Prologu każdy operator posiada przypisaną nieujemną liczbę która określa jego priorytet. Im mniejsza liczba tym wyższy priorytet, np znak + ma priorytet 500, znak * ma priorytet 400.

Priorytet termów to 0.

Operatory | Pozycja

Pozycja określa miejsce występowania operatora względem operandów.

Wyróżniamy trzy pozycje:

- Infixowe - operator występuje pomiędzy operandami np +, -, * , /
- Prefixowe - operator występuje przed operandem np. - traktowany jako znak liczby
- Postfixowe - operator występuje za operandem np. znak silni " ! "

Operatory | Łączność

W prologu łączność określana jest za pomocą atomów (wzorców) postaci a f b, gdzie a i b mogą przyjąć wartość x lub y, natomiast f określa położenie operatora. Znak y powinno się rozumieć jako: na tej pozycji występuje term o priorytecie nie większym od priorytetu operatora, natomiast x rozumiemy jako na tej pozycji występuje term o priorytecie mniejszym od priorytetu operatora. Zauważmy, że nie jest możliwe zagnieżdżanie operatorów nie posiadających łączności (np. is, wzorzec xfx). Podobnie zachowuje się wzorzec fx, nie pozwalając napisać np. --3.

Operatory

Używając zapytania "*current_op*" możemy sprawdzić cechy operatorów wbudowanych.

```
?- current_op(Priorytet, Lacznosc, +).
Priorytet = 500,
Lacznosc = fx ;
```

Operatory

Czasami łatwiej jest zapisać niektóre funkcje za pomocą operatorów.

Jest to forma składni która ułatwia czytanie niektórych struktur.

Jeśli byśmy chcieli zapisać wyrażenie "x + y * z" w normalnej postaci struktury musielibyśmy napisać "+(x,*(y,z))".

Oba zapisy są dopuszczalne w Prologu i oznaczają to samo.

Operatory

Zapis 3+4 nie powoduje wykonania operacji arytmetycznych. Jest to po prostu inny sposób zapisania wyrażenia

Przykład:

3+4.

?- +(3,4).

true

? 3+4.

True

Operatory

W prologu według tej zasady możemy zapisać lubi(jas, malgosie)
za pomocą operatora "lubi":

jas lubi malgosie

Operatory

Przykład:

?- op(500, xfx, lubi).

? - jas lubi malgosie.

true

Relacje równości

Rodzaje relacji równości/różności w Prologu:

- X=Y, prawdziwy, gdy term X i Y unifikują się;
- X is E, prawdziwy, gdy X unifikuje się z wartością wyrażenia E;
- E1 =:= E2, prawdziwy, gdy wartości wyrażeń E1 i E2 są równe;
- E1 =\= E2, prawdziwy, gdy wartości wyrażeń E1 i E2 są różne;
- T1 == T2, prawdziwy, gdy termy T1 i T2 są identyczne (identyfikują się leksyklanie, łącznie z nazwami zmiennych);
- T1 \== T2, prawdziwy, gdy termy T1 i T2 nie są identyczne

Relacje równości | Przykłady

?- X=5,X=5.
X = 5

?- X=5,X=2+3.
false

?- X=5,X is 2+3.
X = 5

?- 2+3 = 2+3.
true

2+3 =:= 2+3.
true

?- 5 =:= 2+3.
true

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

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

?- X\==Y.
true

Arytmetyka

Prolog posiada wbudowane predykaty do porównywania liczb.

Argumentami mogą być też zmienne, stałe I struktury.

Predykaty porównywania potrafią porównać termy traktując je jak wyrażenia arytmetyczne.

Arytmetyka

W Prologu wyróżniamy następujące operatory porównania:

- X=:=Y warunek zachodzi jeśli X I Y są tymi samymi liczbami
- X=\=Y warunek zachodzi jeśli X I Y są różnymi liczbami
- X < Y
- X > Y
- X =< Y
- X >= Y

Arytmetyka

Prolog posiada następujące operatory arytmaetyczne:

- + dodawanie
- - różnica
- * mnożenie
- / dzielenie
- // dzielenie całkowite
- X mod Y reszta z dzielenia

Operatory

Aby wykonać operacje arytmetyczne musimy użyć operatora wbudowanego " is "

Przykład:

?- X is 2+3*5.

X = 17

Obliczanie celu

Zatwierdzenie zapytania powoduje uruchomienie procesu mającego na celu wykazanie, że istnieje ciąg podstawień i przekształceń pozwalający przypisać zapytaniu wartość logiczną *prawda*. Poszukiwanie takiego dowodu nazywane jest obliczaniem celu.

Obliczanie celu

Każdy predykat wchodzący w skład zapytania staje się (pod)celem, który Prolog stara się spełnić jeden po drugim. Jeśli identyczne zmienne wstępują w kilku podcelach, wówczas, związane jest z nimi identyczne podstawienie.

Obliczanie celu

Jeśli cel pasuje do głowy reguły, wówczas mają miejsce odpowiednie podstawienia wewnątrz reguły1 i tym samym otrzymujemy nowy cel, zastępujący niejako cel początkowy. Jeśli cel ten składa się z kilku predykatów, wówczas zostaje on podzielony na kilka podceli, przy czym każdy z nich traktujemy jak cel pierwotny. Proces zastępowania wyrażenia przez inne wyrażenie nazywamy rezolucją i można opisać go następującym algorytmem.

Obliczanie celu

1. Dopóki zapytanie nie jest puste, wykonuj:

(a) Wybierz term z zapytania2 .

(b) Znajdź fakt lub regułę unifikującą się z termem3 . Jeśli nie ma żadnego faktu lub reguły, zwróć FAIL, w przeciwnym razie kontynuuj. i. Jeśli znaleziono fakt, usuń go z zapytania. ii. Jeśli znaleziono regułę, zastąp term ciałem reguły.

2. Zwróć SUCCESS.

Obliczanie celu

Stosowanie unifikacji i rezolucji pozwala na wykazanie prawdziwości lub jej braku, według następujących zasad 1. Jeśli cel jest zbiorem pustym, zwróć prawdę. 2. Jeśli nie ma głów reguł lub faktów unifikujących się z rozważanym wyrażeniem, zwróć fałsz. 3. W przypadku niepowodzenia (otrzymanie wartości fałsz), wróć do takiego miejsca, w którym stosując rezolucję możesz uzyskać inne wyrażenie i ponów cały proces. Zasada ta nazywana jest nawracaniem (ang. backtracking).

Obliczanie celu | Przykład

a(b,c).

a(c,d).

aa(X,Y) :- a(X,Z), a(Z,Y).

?- aa(b,A)

Krok 1. Rezultatem unifikacji dla aa(b,A) oraz aa(X,Y) jest podstawienie:

X = a I A = Y

Rezolucja: zastępując aa(b,A) przez a(X,Z), a(Z,Y) i stosując

uzyskane podstawienie otrzymujemy nowe zapytanie:

a(b,Z),a(Z,Y).

Obliczanie celu | Przykład

```
a(b,c).
a(c,d).
aa(X,Y) :- a(X,Z), a(Z,Y).
?- aa(b,A)

?- a(b,Z),a(Z,Y).
```

Krok 2. Z uzyskanego w poprzednim kroku zapytania wybieramy atom a(b,Z) i w wyniku unifikacji z faktem a(b,c) otrzymujemy podstawienie

Z=c

Rezolucja: ponieważ unifikacja dotyczyła faktu więc rozpatrywany atom z zapytania zostaje usunięty (zastąpiony przez element pusty) po czym do otrzymanego w ten sposób wyrażenia Stosujemy unifikację w wyniku czego otrzymujemy kolejne Zapytanie:

a(c,Y).

Obliczanie celu

Innymi słowy można powiedzieć, że unifikacja jest, podobnie jak w „tradycyjnym" programowaniu, przypisywaniem wartości do zmiennych, natomiast rezolucja sposobem przekonstruowywania zapytania.

Obliczanie celu | Przykład

```
a(b,c).
a(c,d).
aa(X,Y) :- a(X,Z), a(Z,Y).
?- aa(b,A)

?- a(b,Z),a(Z,Y).
?- a(c,Y).
```

Krok 3. W uzyskanym w poprzednim kroku zapytaniu występuje tylko jeden atom a(c,Y) i w wyniku unifikacji z faktem a(c,d) otrzymujemy podstawienie

Y=d

Rezolucja: ponieważ unifikacja dotyczyła faktu więc rozpatrywany atom z zapytania zostaje usunięty w wyniku czego otrzymujemy puste zapytanie, co oznacza koniec procesu.