Programowanie
w logice

PROLOG

Predykat odcięcia „cut” („!”)

„Cut” – bezargumentowy predykat jest interpretowany logicznie jako zawsze prawdziwy i służy do ograniczania nawrotów.

Realizacja tego predykatu, występującego jako jeden z podcelów w ciele klauzuli, uniemożliwia nawrót do któregokolwiek z poprzedzających go podcelów przy próbie znajdowania rozwiązań alternatywnych.

Cut

Wszystkie zmienne, które zostały ukonkretnione podczas realizacji poprzedzających odcięcie podcelów w ciele klauzuli, zachowują nadane im wartości w trakcie realizacji występujących po predykacie odcięcia warunków.

Odcięcie nie ma wpływu na nieukonkretnione zmienne występujące w następujących po nim podcelach.

Wpływ na nawracanie

repeat – generowanie wielu rozwiązań danego problemu poprzez „wymuszanie” nawrotów

Przykład.

```
a(1).
a(2).
a(3).
a(4).
?-repeat, a(X),write(X), X==3,!.
123
X=3
```

Przykład wykorzystania odcięć

Obliczanie maksimum

Klauzula **max(X,Y,Max)** ma zwracać spośród dwóch wartości **X** i **Y** wartość większą jako **Max**.

Definicja 1
Max=X, o ile
X jest większe lub równe Y
lub
Max=Y, o ile
X jest mniejsze od Y.

Zapis w Prologu:

```
max(X,Y,X):- X>=Y.
max(X,Y,Y):- X<Y.
```

Definicja 2
Max=X, o ile
X jest większe lub równe Y
w przeciwnym przypadku
Max=Y.

Zapis w Prologu:

```
max(X,Y,X):- X>=Y,!.
max(X,Y,Y).
```

Przykład wykorzystania odcięć

```
silnia(0,1) :- !.
silnia(X,Y) :- N is X-1,
               silnia(N,B),
               Y is X*B.
```

Użycie odcięcia powoduje, że w przypadku, gdy realizacja warunku brzegowego kończy się sukcesem (dla X=0), próba realizacji procedury rekurencyjnej w ogóle nie zostanie podjęta.

Predykat „fail”

„fail” powoduje niepowodzenie wykonywania klauzuli. Wykonanie tego predykatu zawsze zawodzi. Najczęściej używany w celu wymuszenia nawrotów.

Użyty w kombinacji z „cut” (!,fail) zapobiega użyciu innej klauzuli przy próbie znalezienia rozwiązań alternatywnych, co oznacza niepowodzenie wykonywania całej procedury.

Predykaty obsługi wejścia/wyjścia

Czytanie i pisanie znaków:

get(X) – umożliwiają pobranie pojedynczych znaków z bieżącego urządzenia wejściowego

```
?-get(X).
|: a
X=97.
```

put(X) – powoduje wypisanie do bieżącego urządzenia wyjściowego znaku, którego reprezentację w kodzie ASCII stanowi zmienna X

```
?- put(104),put(101),put(108),put(108),put(111).
hello
```

Predykaty obsługi wejścia/wyjścia

Czytanie i pisanie termów:

write(X) – powoduje wypisanie termu (jeśli X jest ukonkretniona z prologowym termem) do bieżącego urządzenia wyjściowego (domyślnie monitor)

```
?-write('hallo').
hallo
?-write("hallo").
[104,97,108,108,111]
```

Predykaty obsługi wejścia/wyjścia

display(X) – równoważny predykatowi write z różnicą dotyczącą traktowania operatorów

```
?-display(a*b+c*d).
+(*(a,b),*(c,d))

?-write(a*b+c*d).
a*b+c*d
```

Predykaty obsługi wejścia/wyjścia

read(X)

w przypadku, gdy zmienna X jest nieukonkretniona, spowoduje ukonkretnienie tej zmiennej termem wczytanym z bieżącego urządzenia wejściowego

Predykaty obsługi wejścia/wyjścia

Czytanie i pisanie do plików:

tell(X) – ze zmienną X ukonkretnioną nazwą pliku kojarzy bieżące urządzenie wejściowe z plikiem o podanej nazwie, przygotowując go do operacji pisania (otwarcie pliku)

Jeśli X oznacza nazwę pliku istniejącego, to poprzednia jego zawartość zostanie usunięta.

W przypadki pliku nie istniejącego, zostanie on utworzony.

append(X) – otwarcie pliku do zapisu, bez usunięcia zawartości pliku (dopisanie)

told – zamknięcie pliku

Przykład

Wprowadzenie przez użytkownika elementów listy i zapisanie ich do pliku.

```
pisz_plik :-
    write('Podaj listę:'),
    read(L1),
    tell('plik.txt'), /*lub append*/
    write(L1),
    write(.),
    nl,
    told.
```

Predykaty obsługi wejścia/wyjścia

see(X) - ze zmienną X ukonkretnioną nazwą pliku kojarzy bieżące urządzenie wejściowe z plikiem o podanej nazwie, przygotowując go do operacji czytania (otwarcie pliku)

seen - zamknięcie pliku

Przykład

Odczytanie elementów listy liczbowej z pliku, obliczenie i wyświetlenie ich sumy

```
czytaj_plik :- write('Czytam z pliku...'),
        nl,
        see('przyk.txt'),
        read(L),
        seen,
        write('suma elementow listy z pliku wynosi:'),
       sumlist(L,Suma),
       write(Suma).
```

Predykaty obsługi wejścia/wyjścia

Predykaty dynamicznej zmiany pamięci:

asserta(X) – umożliwia dołączenie do bazy danych – **na początek** – klauzuli, którą jest ukonktretniona zmienna X

assertz(X) – umożliwia dołączenie do bazy danych – **na koniec** – klauzuli, którą jest ukonktretniona zmienna X

Predykaty obsługi wejścia/wyjścia

retract(X) – usunięcie z bazy danych pierwszej klauzuli dającej się uzgodnić z argumentem predykatu

np. asserta(student(adam,kowalski,s12345)).
retract(film(ziemia_obiecana,wajda)).

Predykaty obsługi wejścia/wyjścia

consult(X) - umożliwia rozszerzenie prologowej bazy danych o zbiór klauzul zawartych w określonym pliku lub wprowadzanych bezpośrednio z klawiatury.

Klauzule odczytywane z danego pliku są dołączane na koniec bazy danych.

Np. consult('dane.txt').

Literatura

- W. Clocksin, C. Mellish, „Prolog. Programowanie”
- E.Gatnar, K.Stąpor, „Prolog”
- G.Brzykcy, A.Meissner, „Programowanie w prologu i programowanie funkcyjne”
- M. Ben-Ari, “Logika matematyczna w informatyce”