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.

Przykład wykorzystania odcięć w Prologu

Obliczanie maksimum

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

Definicja 1	Definicja 2
Max=X, o ile X jest większe lub równe Y lub Max=Y, o ile X jest mniejsze od Y.	*Max=X, o ile X jest większe lub równe Y w przeciwnym przypadku Max=Y.*
Zapis w Prologu:	Zapis w Prologu:
`max(X,Y,X):- X>=Y.`	`max(X,Y,X):- X>=Y,!.`
`max(X,Y,Y):- X<Y.`	`max(X,Y,Y).`

Przykład wykorzystania odcięć w Prologu

```
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),get0(X) – umożliwiają pobranie pojedynczych znaków z bieżącego urządzenia wejściowego

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

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 ozn. 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

Wprowadzenie przez użytkownika elementów listy i zapisanie ich do pliku.

```
pisz_plik :-
    write('Podaj listę:'),
    read(L1),
    tell('n_plik.txt'), /*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

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)).
```


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').

Sprawdzanie typu wartości argumentów (zmiennych lub stałych)

var(X) – sprawdza, czy zmienna X ma przypisaną wartość

Cel var(X) nie zawodzi, gdy X jest zmienną nieukonkretnioną

```
?-var(X).
  true
?-X=50,var(X).
  false
```

nonvar(X) – przeciwieństwo predykatu var(X)

Sprawdzanie typu wartości argumentów (zmiennych lub stałych)

atom(X) – nie zawodzi, gdy X jest atomem Prologu

?-atom(niebo).
true

?-atom(X).
false

?-atom('c:plik.txt').
true

?-atom(123).
false

Sprawdzanie typu wartości argumentów (zmiennych lub stałych)

integer(X) - sprawdza, czy X jest typu całkowitego

number(X) - sprawdza, czy X jest liczbą

atomic(X) - nie zawodzi, gdy X jest liczbą lub atomem

Operacje na strukturach

functor(S,F,N) – umożliwia dostęp do struktury, ustala liczbę argumentów struktury

S – struktura
F – funktor
N – liczba argumentów

?- functor(a(5,2,8),a,3).
true
?- functor(plus(1,2),F,N).
F=plus
N=2

Operacje na strukturach

arg(N,S,A) - umożliwia dostęp do wybranych argumentów struktury

Dwa pierwsze argumenty arg muszą być ukonkretnione
N – numer argumentu struktury
S – struktura

?-arg(4,litery(a,d,r,y,w,q,z,i),A).
A=y

?-arg(2,[a,b,c,d,e,f],A).
A=[b,c,d,e,f]

Operacje na strukturach

X=..L (univ)

L jest listą składającą się z funktora struktury reprezentowanej przez X oraz następującego po nim zbioru argumentów

?-X=..[a,b,c].
X=a(b,c)

?-X=..[suma,2,5,4,3].
X=suma(2,5,4,3)

?- student(jan,kowalski,wmii,poznan)=..L.
L = [student, jan, kowalski, wmii, poznan]

Operacje na strukturach

name(A,L) – rozkłada wyrażenie atomowe na zbiór znaków ujętych w postaci listy

?-name(abcd,L).
L=[97,98,99,100]

?-name(A, [97,98,99,100]).
A=abcd

Wpływ na nawracanie

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

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


Różniczkowanie symboliczne

```
pochodna(X,X,1):-!.
pochodna(C,X,0):-atomic(C).
pochodna(-Z,X,-C):-pochodna(Z,X,C).
pochodna(W+Z,X,A+B):-
        pochodna(W,X,A),pochodna(Z,X,B).
pochodna(W-Z,X,A-B):-
        pochodna(W,X,A),pochodna(Z,X,B).
pochodna(C*Z,X,C*A):-
        atomic(C),C\=X,pochodna(Z,X,A),!.
pochodna(W*Z,X,B*W+A*Z):-
        pochodna(W,X,A),pochodna(Z,X,B).
```


Dodawanie macierzy

```
dodm([L1|O1],[L2|O2],[L3|O3]):-
                dodl(L1,L2,L3),
                dodm(O1,O2,O3).
dodm([],[],[]).
dodl([L1|O1],[L2|O2],[L3|O3]):-
                L3 is L1+L2,
                dodl(O1,O2,O3).
dodl([],[],[]).
```

Systemy ekspertowe

Struktura systemu ekspertowego

- Moduł wnioskowania – wykonuje proces rozumowania w trakcie rozwiązywania problemu postawionego przez użytkownika,
  najważniejszy składnik systemu ekspertowego, jego zadaniem jest wyciąganie wniosków z przesłanek i pytań wprowadzanych przez użytkownika i generowanie odpowiedzi

Systemy ekspertowe

- Baza wiedzy - drugi co do ważności składnik systemu eksperckiego, w bazie wiedzy zawarta jest wiedza dotycząca określonej dziedziny. Wiedza ta zapisana jest za pomocą faktów i reguł

Systemy ekspertowe

- Baza danych zmiennych – tzw. pamięć robocza, stanowi część dynamiczna systemu;
  jest to pomocnicza baza danych, w której przechowywane są wnioski uzyskane przez system podczas jego działania
- Interfejs użytkownika – zapewnia współdziałanie systemu z użytkownikiem, komunikację system-użytkownik; sprowadza się do zadawania pytań, udzielania informacji systemowi oraz odbierania odpowiedzi i wyjaśnień

- Moduł objaśniający – umożliwia udzielanie wyjaśnień, odpowiada na pytania użytkownika (jak? dlaczego? na jakiej podstawie?)