Dlaczego PROLOG

- Prolog was born from a challenge: create a very high level language, even if inefficient according to the standard at the time. At that time efficiency consisted of very fast processing of programs written with great effort.
  So the challenge was to write programs very quickly, and let the machine compute through extensive calculation."
- Prolog narodził się z wyzwania: stworzyć język bardzo wysokiego poziomu, nawet jeśli będzie nieefektywny czasowo.
  W tamtych czasach efektywność polegała na szybkim przetwarzaniu programów napisanych z dużym wysiłkiem.
  Wyzwaniem było pisanie programów, które pisze się bardzo szybko i pozwolenie na wykonanie maszynowe, nawet za cenę czasochłonnych obliczeń.

http://prolog-heritage.org/en

1

Jeśli rozwiązań jest wiele

W odpowiedzi na pytanie:
?- lubi(Kto, _). /* miłośnicy gór
Otrzymujemy odpowiedzi:
jan;
jan;
jerzy;
jerzy;
jozef;
…
Dlaczego podwójnie?

2

Odcięcie

Nakazuje interpreterowi prologu, aby nie uwzględniał alternatywnych możliwści wyboru podczas nawracania.

Odcięcie w regule jest reprezentowane bezargumentowym predykatem ! (znak wykrzyknika)

Np..

bratniadusza(X, Y) :- lubi(X, S), lubi(Y, S), X\=Y. Da 2 wyniki

bratniadusza(X, Y) :- lubi(X, S), lubi(Y, S), X\=Y, !. Da 1wynik

Dzięki temu mechanizmowi

- Program będzie działać szybciej, bo nie będzie starał się spełnić celów, dla których wynik uzgadniania jest z góry znany
- Program zajmuje mniej pamięci, bo nie trzeba zapamiętywać punktów nawracania

3

Użycie odcięcia

```
czytelnik('Bolek').
czytelnik('Lolek').
czytelnik('Tola').

przetrzymana_ksiazka('Tola', ks123).
przetrzymana_ksiazka('Bolek', ks124).
przetrzymana_ksiazka('Bolek', ks125).

usluga_podstawowa(katalog).
usluga_podstawowa(czytelnia).
usluga_dodatkowa(wypozyczanie).
usluga_dodatkowa(pozyczki_miedzybiblioteczne).
```

Już 1 przetrzymana książka wystarcza do odmówienia usług dodatkowych
Predykat ! mówi: nie sprawdzaj już innych książek przetrzymanych przez Bolka, ani alternatywnych definicji predykatu usluga(Osoba, Usluga)
pytanie: czytelnik(X), usluga(X, U). da uslugi poszczególnych czyt.

```
usluga(Osoba, Usluga) :- przetrzymana_ksiazka(Osoba, Ksiazka),
                         !,
                         usluga_podstawowa(Usluga).
usluga(Osoba, Usluga) :- dowolna_usluga(Usluga).
dowolna_usluga(X) :- usluga_podstawowa(X).
dowolna_usluga(X) :- usluga_dodatkowa(X). /* powtórzenie definicji predykatu
    oznacza alternatywne definicje: dowolna_usluga:- usluga_podstawowa(X);
    usluga_dodatkowa(X).
```

4

Ogólnie

Dla predykatu
predykat :- a, b, c, !, d, e, f

Można dokonywać nawrotów między celami a, b, c, gdy jednak uzgodnienie celu c spowoduje przejście do celu d, nawroty mogą się odbywać tylko między celami d, e, f

5

Operacje na liczbach, odcięcie

```
/* sumowanie liczb od 1 do N */
/* wariant z odcięciem  */

suma_do(N,1) :- N=<1, !.
suma_do(N, Wynik) :-
   N1 is N-1,
   suma_do(N1, Wynik1),
   Wynik is Wynik1+N.

/*  można jednak bez odcięcia   */
sumado(1,1).
sumado(N, Wynik):-
   N >=1,
   N1 is N-1,
   sumado(N1, Wynik1),
   Wynik is Wynik1+N1.
```

6

Predykaty wbudowane

- Pomocnicze podczas wykonywania operacji z programem:
  - trace, debug, notrace, nodebug – włączenie/wyłączenie trybu śledzenia wykonania programu
- Wejście/wyjście:
  - read(X)., write(X).
- Operacje na listach
  - member, append, length, reverse
- Koniec pracy
  - halt.
- Inne: /ProgramFiles/swipl/library

7

Thinking in Prolog: Operacje na listach

```
zloz_liste(G, O, [G|O]).    /* tak, to cała definicja predykatu */
Pytania: ?- zloz_liste(G, [2,3], [1,2,3]).    Odp. G = 1.
         ?- zloz_liste(1, [2,3], X).          Odp. X=[1, 2, 3].

/* odwroć   */
odwroc([], []).
odwroc([G|O], Lista) :-
    odwroc(O, Wyn),
    append(Wyn, [G], Lista).

/*  permutacja  */
permutuj([], []).
permutuj(L,[H|T]) :-
   append(V, [H|U], L),
   append(V, U, W),
   permutuj(W, T).
```

8

Thinking in Prolog: Operacje na listach i zbiorach

```
usun(_, [], []). /* cokolwiek usuwasz z pustej listy, zostaje ona pusta */
usun(X, [X|T], M) :-
    usun(X, T, M). /* jeśli elt jest w głowie, wynikiem jest ogon*/
    usun(X, [Y|T],[Y|M]) :- usun(X, T, M). /* jeśli nie jest głową, zostaw go i
     przeszukuj ogon */

iloczyn_zb([], _, []).  /* iloczyn zbioru pustego z czymkolwiek jest zbiorem pustym
iloczyn_zb([X|L1], L2, [X|L3]):-
   member(X, L2), /* pierwszy elt pierwszego zbioru jest w drugim zbiorze?
   !,
   iloczyn_zb(L1, L2, L3).
iloczyn_zb([_|L1], L2, L3) :- iloczyn_zb(L1, L2, L3).

suma_zb([], X, X).
suma_zb([X|L1], L2, L3):-
     member(X, L2), !, suma_zb(L1, L2, L3).
suma_zb([X|L1], L2, [X|L3]):- suma_zb(L1, L2, L3).
```

9

Psychologia

Psychologowie w rozmowach z pacjentami często używają *parafrazowania,* również każdy android będzie nas zapewniał, że nie jest automatem parafrazując nasze pytania

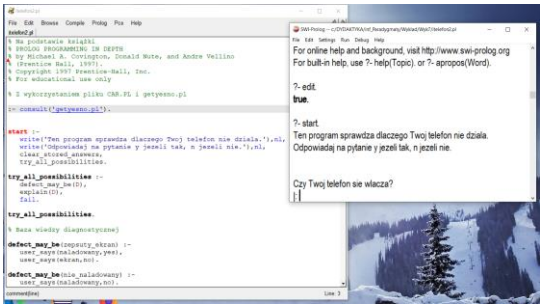
```
podmien(czy,nie).
podmien(ty,[ja,nie]).
podmien(francusku, niemiecku).
podmien(jestes, jestem).
podmien(mowisz, mowie).
podmien(X,X).

zmien([],[]).
zmien([G|O],[X|Y]) :- podmien(G,X), zmien(O,Y).
```

10

System ekspercki

Diagnozowanie uszkodzenia telefonu



11