Programowanie w Logice

Wprowadzenie

Przemysław Kobylański
na podstawie [CM2003]

Wprowadzenie
Historia

- Język programowania w logice (fr. *PROgrammation en LOGique*).
- Opracowany w roku 1972 przez zespół Alaina Colmerauer na uniwersytecie w Marsylii.
- Artykuł o początkach Prologu: Alain Colmerauer, Philippe Roussele. "The birth of Prolog".
- Serwis o implementacjach Prolog I, Prolog II, Prolog III i Prolog IV: prolog-heritage.org.
- Prolog ma swoje korzenie w rachunku predykatów pierwszego rzędu.
- Związki między algorytmem a logiką opisał w roku 1979 Robert Kowalski z Imperial College w artykule "Algorithm = Logic + Control".

Wprowadzenie
Fakty

```
cenne(złoto).
kobieta(janina).
posiada(jan, złoto).
ojciec(jan, maria).
daje(jan, gazeta, maria).
```

Wprowadzenie
Zapytania

```
lubi(jarek, ryby   ).
lubi(jarek, maria  ).
lubi(jan,   książka).
lubi(jan,   francja).

?- lubi(jarek, pieniądze).
false.
?- lubi(maria, jarek).
false.
?- lubi(jan, książka).
true.
```

Wprowadzenie
Zmienne

```
lubi(jan,   kwiaty).
lubi(jan,   maria ).
lubi(paweł, maria ).

?- lubi(jan, X).
X = kwiaty                          % klawisz Enter
?- lubi(X, maria).
X = jan ;                           % znak średnika
X = paweł ;
false.
```

Wprowadzenie
Koniunkcje

```
lubi(maria, czekolada).
lubi(maria, wino     ).
lubi(jan,   wino     ).
lubi(jan,   maria    ).

?- lubi(jan, maria), lubi(maria, jan).
false.
?- lubi(maria, X), lubi(jan, X).
X = wino ;
false.
```

Wprowadzenie
Reguły

```
lubi(jan, X) :-
    lubi(X, wino).      % Jan lubi lubiących wino.
lubi(jan, X) :-
    lubi(X, wino),      % Jan lubi lubiących wino i
    lubi(X, jedzenie). % jedzenie.
lubi(jan, X) :-
    kobieta(X),         % Jan lubi te kobiety,
    lubi(X, wino).      % które lubią wino.
```

Wprowadzenie
Reguły

```
mężczyzna(albert).
mężczyzna(edward).
kobieta(alicja).
kobieta(wiktoria).
rodzice(edward, wiktoria, albert).
rodzice(alicja, wiktoria, albert).

siostra(X, Y) :-
    kobieta(X),
    rodzice(X, M, O),
    rodzice(Y, M, O).

?- siostra(alicja, edward).
true.
?- siostra(alicja, X).
X = edward ;
X = alicja.
```

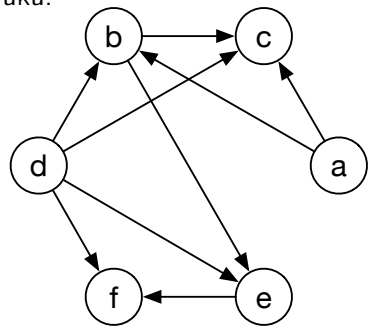
Wprowadzenie
Reguły

```
złodziej(jan).
lubi(maria, czekolada).
lubi(maria, wino).
lubi(jan, X) :-
    lubi(X, wino).
może_ukraść(X, Y) :-
    złodziej(X),
    lubi(X, Y).

?- może_ukraść(jan, X).
X = maria ;
false.
```

Wprowadzenie
Reguły rekurencyjne

Rozpatrzmy problem poszukiwania ścieżki w acyklicznym grafie skierowanym $G = (V, A)$, gdzie V jest skończonym zbiorem węzłów a A jest zbiorem łuków reprezentowanych uporządkowanymi parami węzłów: początkowy i końcowy węzeł łuku.




```
arc(a, b).
arc(a, c).
arc(b, c).
arc(b, e).
arc(d, b).
arc(d, c).
arc(d, e).
arc(d, f).
arc(e, f).
```

Wprowadzenie
Reguły rekurencyjne

Pierwsza wersja:

```
path(X, Y) :- arc(X, Y).
path(X, Y) :- arc(X, A), arc(A, Y).
path(X, Y) :- arc(X, A), arc(A, B), arc(B, Y).
path(X, Y) :- arc(X, A), arc(A, B), arc(B, C), arc(C, Y).
...
```

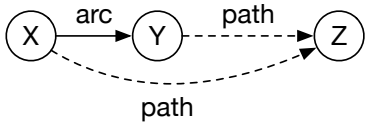
Program składa się z nieskończonej liczby klauzul.

Wprowadzenie
Reguły rekurencyjne

Druga wersja:

```
path_1(X, Y) :- arc(X, Y).
path_2(X, Z) :- arc(X, Y), path_1(Y, Z).
path_3(X, Z) :- arc(X, Y), path_2(Y, Z).
path_4(X, Z) :- arc(X, Y), path_3(Y, Z).
...
```

Program składa się z nieskończenie wielu predykatów bardzo podobnie zdefiniowanych.



Wprowadzenie
Reguły rekurencyjne

Trzecia wersja:

```
path(X, Y) :- arc(X, Y).
path(X, Z) :- arc(X, Y), path(Y, Z).
```

Przykładowe zapytania:

```
?- path(a, d).  ?- path(a, c).  ?- path(a, X).  ?- path(X, c).
false.          true ;          X = b ;         X = a ;
                true ;          X = c ;         X = b ;
                false.          X = c ;         X = d ;
                                X = e ;         X = a ;
                                X = f ;         X = d ;
                                false.          false.
```

Wprowadzenie
Jak Prolog znajduje odpowiedź?

```
śmiertelny(X) :-
    człowiek(X).

człowiek(sokrates).

?- śmiertelny(Y).
Y = sokrates
```

Wprowadzenie
Jak Prolog znajduje odpowiedź?

- Program w Prologu jest zbiorem formuł.
- Pytanie jest formułą.
- System sprawdza czy formuła będąca pytaniem jest logicznym wnioskiem z programu.

Wprowadzenie
Jak Prolog znajduje odpowiedź?

- Formuła Q jest logicznym wnioskiem ze zbioru formuł P, jeśli zbiór formuł $P \cup \{\neg Q\}$ jest sprzeczny.
- Zbiór formuł jest sprzeczny, jeśli można wyprowadzić z niego klauzulę pustą (formułę fałszywą).
- **Zasada rezolucji**: z dwóch formuł $p \vee q_1$ oraz $\neg p \vee q_2$ (przesłanek) zawierających komplementarną parę literałów p i $\neg p$, można wyprowadzić formułę $q_1 \vee q_2$ (rezolwentę).

Wprowadzenie

Jak Prolog znajduje odpowiedź?

- Zanegowane pytanie nazywamy celem.
- Podczas działania Prologu zawsze jedną z przesłanek jest aktualny cel a drugą jedna z klauzul programu (rezolucja liniowa).
- Wyprowadzenie rezolwenty można przedstawić graficznie w postaci poziomej kreski oddzielającej dwie przesłanki od wynikającej z nich rezolwenty:

$$\frac{\text{pierwsza przesłanka} \quad \text{druga przesłanka}}{\text{rezolwenta}}$$

- System stara się zredukować w kolejnych krokach cel do klauzuli pustej, którą oznaczamy symbolem $\square$.

Wprowadzenie

Jak Prolog znajduje odpowiedź?

Kolorem czerwonym zaznaczono kolejne cele:

$$\dfrac{\mathit{czlowiek}(\mathit{sokrates}) \qquad \dfrac{\mathit{smiertelny}(X) \vee \neg \mathit{czlowiek}(X) \qquad \neg \mathit{smiertelny}(Y)}{\neg \mathit{czlowiek}(X) \text{ gdy } Y = X}}{\square \text{ gdy } X = \mathit{sokrates}}$$

- Ostatni cel jest pustą klauzulą, zatem zbiór formuł jest sprzeczny.
- Wyliczoną odpowiedzią jest ograniczone do zmiennych występujących w pytaniu złożenie podstawień $Y = X$ i $X = \mathit{sokrates}$ (podstawienie $Y = \mathit{sokrates}$).

Więcej o teoretycznych podstawach działania Prologu będzie na studiach drugiego stopnia (kurs *Metody programowania w logice*).