Programowanie w Logice

Przykłady programów

Przemysław Kobylański



Przykłady programów

Interpreter prostego języka imperatywnego

```
WYRAŻENIE ::= SKŁADNIK + WYRAŻENIE
WYRAŻENIE ::= SKŁADNIK - WYRAŻENIE
WYRAZENTE ::= SKŁADNTK
SKŁADNIK ::= CZYNNIK * SKŁADNIK
SKŁADNIK ::= CZYNNIK / SKŁADNIK
SKŁADNIK ::= CZYNNIK mod SKŁADNIK
SKŁADNIK ::= CZYNNIK
 CZYNNIK ::= IDENTYFIKATOR
  CZYNNIK ::= LICZBA NATURALNA
 CZYNNIK ::= ( WYRAŻENIE )
```

▼ロト▼部ト▼ミト▼ミト ミ めので

Przykłady programów

Interpreter prostego języka imperatywnego

- ▶ Język **Imperator**¹ jest prostym językiem imperatywnym.
- ▶ Jego składnię opisuje poniższa gramatyka BNF:

```
PROGRAM ::=
```

PROGRAM ::= INSTRUKCJA ; PROGRAM

INSTRUKCJA ::= IDENTYFIKATOR := WYRAŻENIE

INSTRUKCJA ::= read IDENTYFIKATOR

INSTRUKCJA ::= write WYRAŻENIE

INSTRUKCJA ::= if WARUNEK then PROGRAM fi

INSTRUKCJA ::= if WARUNEK then PROGRAM else PROGRAM fi

INSTRUKCJA ::= while WARUNEK do PROGRAM od



Przykłady programów

Interpreter prostego języka imperatywnego

```
WARUNEK ::= KONIUNKCJA or WARUNEK
```

WARUNEK ::= KONIUNKCJA

KONIUNKCJA ::= PROSTY and KONIUNKCJA

KONIUNKCJA ::= PROSTY

PROSTY ::= WYRAŻENIE = WYRAŻENIE PROSTY ::= WYRAŻENIE /= WYRAŻENIE PROSTY ::= WYRAŻENIE < WYRAŻENIE PROSTY ::= WYRAŻENIE > WYRAŻENIE PROSTY ::= WYRAŻENIE >= WYRAŻENIE PROSTY ::= WYRAŻENIE =< WYRAŻENIE

PROSTY ::= (WARUNEK)

¹Nazwę zaproponował MKI.

Interpreter prostego języka imperatywnego

Example (Obliczenie sumy liczb A i B)

```
read A;
read B;
X := A;
Y := B;
while Y > 0 do
        X := X + 1;
        Y := Y - 1;
od;
write X;
```



Przykłady programów

Interpreter prostego języka imperatywnego

Wyrażenie jest albo prostym wyrażeniem będącym zmienną lub liczbą naturalną albo wyrażeniem złożonym będącym sumą, różnicą, iloczynem lub ilorazem dwóch wyrażeń:

```
WYRAŻENIE = id(ID)

WYRAŻENIE = int(NUM)

WYRAŻENIE = WYRAŻENIE + WYRAŻENIE

WYRAŻENIE = WYRAŻENIE - WYRAŻENIE

WYRAŻENIE = WYRAŻENIE * WYRAŻENIE

WYRAŻENIE = WYRAŻENIE / WYRAŻENIE

WYRAŻENIE = WYRAŻENIE mod WYRAŻENIE
```

Przykłady programów

Interpreter prostego języka imperatywnego

- Program reprezentowany jest listą złożoną z termów.
- Każdy term reprezentuje jedną instrukcję.
- ► Jeśli instrukcja jest złożona, to term zawiera jako podterm listę termów reprezentujących zagnieżdżone instrukcje.

```
PROGRAM = []
PROGRAM = [INSTRUKCJA | PROGRAM]

INSTRUKCJA = assign(ID, WYRAŻENIE)
INSTRUKCJA = read(ID)
INSTRUKCJA = write(WYRAŻENIE)
INSTRUKCJA = if(WARUNEK, PROGRAM)
INSTRUKCJA = if(WARUNEK, PROGRAM, PROGRAM)
INSTRUKCJA = while(WARUNEK, PROGRAM)
```



Przykłady programów

Interpreter prostego języka imperatywnego

Warunek jest relacją równości, różności, mniejszości, większości (słabej lub silnej) między wartościami dwóch wyrażeń albo alternatywą lub koniunkcją dwóch warunków:

```
WARUNEK = WYRAŻENIE =:= WYRAŻENIE
WARUNEK = WYRAŻENIE =\= WYRAŻENIE
WARUNEK = WYRAŻENIE < WYRAŻENIE
WARUNEK = WYRAŻENIE > WYRAŻENIE
WARUNEK = WYRAŻENIE =< WYRAŻENIE
WARUNEK = WYRAŻENIE >= WYRAŻENIE
WARUNEK = WARUNEK ; WARUNEK
WARUNEK = WARUNEK , WARUNEK
```

Interpreter prostego języka imperatywnego

```
program1([
    read('N'),
    assign('SUM', int(0)),
    while(id('N') > int(0),
        [assign('SUM', id('SUM') + id('N')),
```

assign('N', id('N') - int(1))]),

Example (Sumowanie liczb od 1 do N)

write(id('SUM'))]).



→ロト 4回ト 4 三ト 4 三 ・ 夕久(*)

Przykłady programów

Interpreter prostego języka imperatywnego

- Bieżący stan obliczeń zapisywać będziemy w postaci listy asocjacji.
- Każda asocjacja jest termem w postaci ID = Wartość, gdzie ID jest nazwą zmiennej a Wartość jest bieżącą wartością tej zmiennej.

```
% podstaw(+Stare, +ID, +Wartość, -Nowe)
podstaw([], ID, N, [ID = N]).
podstaw([ID=_ | AS], ID, N, [ID=N | AS]) :- !.
podstaw([ID1=W1 | AS1], ID, N, [ID1=W1 | AS2]) :-
    podstaw(AS1, ID, N, AS2).

% pobierz(+Asocjacje, +ID, -Wartość)
pobierz([ID=N | _], ID, N) :- !.
pobierz([_ | AS], ID, N) :-
    pobierz(AS, ID, N).
```

Przykłady programów

Interpreter prostego języka imperatywnego

```
Example (Obliczenie ilorazu D i reszty R z dzielenia M przez N)
```

```
program2([
    read('M'),
    read('N'),
    assign('D', int(0)),
    assign('R', id('M')),
    while(id('R') >= id('N'),
        [assign('D', id('D') + int(1)),
            assign('R', id('R') - id('N'))]),
    write(id('D')),
    write(id('R'))]).
```



Przykłady programów

Interpreter prostego języka imperatywnego

```
% wartość(+Wyrażenie, +Asocjacje, -Wartość)
wartość(int(N), _, N).
wartość(id(ID), AS, N) :-
    pobierz(AS, ID, N).
wartość(W1 + W2, AS, N) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N is N1 + N2.
wartość(W1 - W2, AS, N) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N is N1 - N2.
```

Interpreter prostego języka imperatywnego

```
wartość(W1 * W2, AS, N) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N is N1 * N2.
wartość(W1 / W2, AS, N) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N2 = \= 0, N is N1 div N2.
wartość(W1 mod W2, AS, N) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N2 = \= 0,
    N is N1 mod N2.
```



Przykłady programów

Interpreter prostego języka imperatywnego

```
prawda(W1 >= W2, AS) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N1 >= N2.
prawda(W1 =< W2, AS) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N1 =< N2.
prawda((W1, W2), AS) :-
    prawda(W1, AS),
    prawda(W2, AS).
prawda((W1; W2), AS) :-
    (    prawda(W1, AS),
    !
    ;    prawda(W2, AS)).</pre>
```

Przykłady programów

Interpreter prostego języka imperatywnego

```
% prawda(+Warunek, +Asocjacje)
prawda(W1 =:= W2, AS) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N1 =:= N2.
prawda(W1 =\= W2, AS) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N1 =\= N2.
prawda(W1 < W2, AS) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N1 < N2.
prawda(W1 > W2, AS) :-
    wartość(W1, AS, N1), wartość(W2, AS, N2),
    N1 > N2.
```



Przykłady programów

Interpreter prostego języka imperatywnego

```
% interpreter(+Program, +Asocjacje)
interpreter([], _).
interpreter([read(ID) | PGM], ASSOC) :- !,
    read(N),
    integer(N),
    podstaw(ASSOC, ID, N, ASSOC1),
    interpreter(PGM, ASSOC1).
interpreter([write(W) | PGM], ASSOC) :- !,
    wartość(W, ASSOC, WART),
    write(WART), nl,
    interpreter([assign(ID, W) | PGM], ASSOC) :- !,
    wartość(W, ASSOC, WAR),
    podstaw(AS, ID, WAR, ASSOC1),
    interpreter(PGM, ASSOC1).
```

Interpreter prostego języka imperatywnego

```
interpreter([if(C, P) | PGM], ASSOC) :- !,
    interpreter([if(C, P, []) | PGM], ASSOC).
interpreter([if(C, P1, P2) | PGM], ASSOC) :- !,
    (    prawda(C, ASSOC)
    -> append(P1, PGM, DALEJ)
    ; append(P2, PGM, DALEJ)),
    interpreter(DALEJ, ASSOC).
interpreter([while(C, P) | PGM], ASSOC) :- !,
    append(P, [while(C, P)], DALEJ),
    interpreter([if(C, DALEJ) | PGM], ASSOC).

% interpreter(+Program)
interpreter(PROGRAM) :-
    interpreter(PROGRAM, []).
```

4□▶4圖▶4厘▶4厘▶ 厘 約9○

Przykłady programów

Interpreter prostego języka imperatywnego

```
Example (Iloraz i reszta)
```

Przykłady programów

Interpreter prostego języka imperatywnego

Example (Sumowanie)



Przykłady programów

Interpreter podzbioru Prologu w Prologu

- Zakładamy, że predykaty zapisane są w postaci klauzul, których ciała zawierają tylko koniunkcje formuł atomowych (nie ma negacji, alternatyw i implikacji).
- Wszystkie predykaty są zdefiniowane przez nas (nie korzystamy z predykatów wbudowanych, które mogą być napisane np. w języku C).

Interpreter podzbioru Prologu w Prologu

```
Example (Konkatenacja list na liście)
```

```
app([], X, X).
app([X | L1], L2, [X | L3]) :-
    app(L1, L2, L3).

app([], []).
app([L1 | L2], L3) :-
    app(L1, L4, L3),
    app(L2, L4).
```



Przykłady programów

Interpreter podzbioru Prologu w Prologu

```
?- clause(app(A, B), C).
A = B, B = [],
C = true;
A = [_G4870|_G4871],
C = (app( G4870, G4874, B), app( G4871, G4874)).
```

Przykłady programów

Interpreter podzbioru Prologu w Prologu

- ► Predykat clause(H, B) dostarcza głowe H i ciało B klauzuli.
- ► H jest unifikowane z głową a B z ciałem klauzuli.
- ► Fakt ma ciało równe true.

```
?- clause(app([], A, B), C).
A = B,
C = true.
?- clause(app(A, B, C), D).
A = [],
B = C,
D = true;
A = [_G4888|_G4889],
C = [_G4888|_G4892],
D = app(_G4889, B, _G4892).
```



Przykłady programów

Interpreter podzbioru Prologu w Prologu

- ► Interpreter Prologu w Prologu zapiszemy w postaci predykatu udowodnij(Cel), gdzie Cel jest zadanym celem do udowodnienia.
- Celem może być warunek true, formuła atomowa lub koniunkcja dwóch celów.

```
udowodnij(true) :- !.
udowodnij((G1, G2)) :- !,
    udowodnij(G1),
    udowodnij(G2).
udowodnij(A) :-
    clause(A, B),
    udowodnij(B).
```

Interpreter podzbioru Prologu w Prologu

```
?- udowodnij(app(X, Y, [1, 2, 3])).
X = [],
Y = [1, 2, 3];
X = [1],
Y = [2, 3];
X = [1, 2],
Y = [3];
X = [1, 2, 3],
Y = [];
false.
```

4□▶ 4□▶ 4 亘 ▶ 4 亘 ▶ 9 0 ○

Przykłady programów

Unifikacja i reprezentacja termów

Idea algorytmu unifikacji termów:

Przykłady programów

Interpreter podzbioru Prologu w Prologu

```
?- udowodnij(app([1, 2, 3], X, [1, 2, 3, 4 | Y])).
X = [4|Y].
?- udowodnij(app([[1, 2], [3], [4, 5]], X)).
X = [1, 2, 3, 4, 5].
```



Przykłady programów

Unifikacja i reprezentacja termów

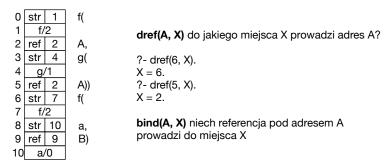
Reprezentacja termu w WAM (ang. Warren Abstract Machine):

A str A' pod adresem A jest referencja do struktury pamiętanej pod adresem A'

A ref A' pod adresem A jest referencja do adresu A'

A fun/n pod adresem A jest n-argumentowy funktor **fun**

Dwa termy f(A, g(A)) i f(a, B) zajmują 11 komórek pamięci:



Unifikacja i reprezentacja termów

:- dynamic store/2.

Example (Unifikacja w WAM)

Reprezentacja komórek pamięci w postaci dynamicznych faktów:

```
store(0, str(1)).
                     % f(
store(1, f/2).
store(2, ref(2)).
                     % A,
store(3, str(4)).
                     % g(
store(4, g/1).
store(5, ref(2)).
                           A))
                     % f(
store(6, str(7)).
store(7, f/2).
store(8, str(10)).
                    % a,
store(9, ref(9)).
                        B)
store(10, a/0).
```



Przykłady programów

Unifikacja i reprezentacja termów

```
Example (Unifikacja w WAM cd.)
```

Przykłady programów

Unifikacja i reprezentacja termów



Przykłady programów

Unifikacja i reprezentacja termów

Example (Unifikacja w WAM cd.)

Unifikacja i reprezentacja termów

Example (Unifikacja w WAM cd.)

Przykładowe wywołanie:

?- unify(0, 6). % odpowiada f(A, g(A)) = f(a, B) true.

Jaki jest wynik unifikacji? Jakie wartości przyjęły zmienne?

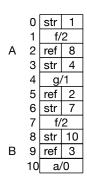


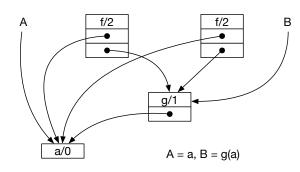
Przykłady programów

Unifikacja i reprezentacja termów

Example (Unifikacja w WAM cd.)

Wynik unifikacji:







Przykłady programów

Unifikacja i reprezentacja termów

Example (Unifikacja w WAM cd.)

Zmiany zachodzące w komórkach pamięci:

