DEFINIRANJE SEMANTIKE

Ivan Bratko
FRI, Univerza v Ljubljani

DEFINIRANJE SINTAKSE IN SEMANTIKE

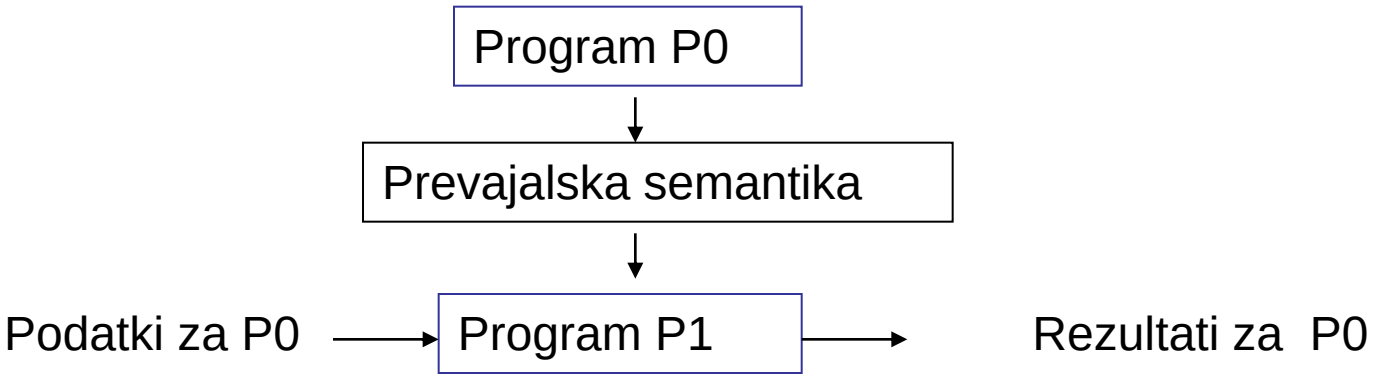
- Sintaksa - razmeroma neproblematično
  - Gramatike
  - BNF zapis
  - Sintaksni diagrami
- Semantika - bolj problematično
- Številni pristopi
  - Atributne gramatike
  - Prevajalska semantika
  - Operacijska semantika
  - Denotacijska semantika
  - Aksiomatska semantika
  - Naravna semantika (natural semantics)

ATRIBUTNE GRAMATIKE

- Neterminalni simboli gramatike: dodamo atribute
- Atributi so nosilci pomena
- Npr. DCG v prologu (Definite Clause Grammar)
- Definiranje pomena naravnega jezika v DCG

PREVAJALSKA SEMANTIKA

- Prevajalska semantika = Translational semantics
- Program P0 v jeziku L0
- Program P1 v jeziku L1
- Pomen P0 = P1, pri čemer P1 da enake rezultate kot P0

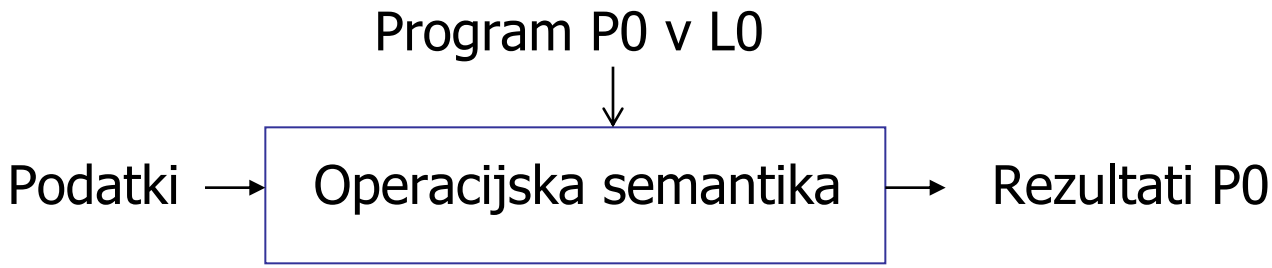


PREVAJALSKA SEMANTIKA

- Prednost: prevajalska semantika pokaže, kako naj bo narejen prevajalnik za L0
- Slabost: Zelo odvisna od konkretnega ciljnega jezika

OPERACIJSKA SEMANTIKA

- Semantika je interpreter za jezik L0



- Prednosti:
  - Daje intuitiven opis jezika (kaj naredijo posamezni konstrukti jezika?)
  - Privlačno za programerja - semantika (interpreter) = algoritem
  - Prototipno programiranje, izvedljiva specifikacija

- Slabost: Omejitve, ki sledijo iz konkretne implementacije (računalnik in jezik interpreterja)

DENOTACIJSKA SEMANTIKA

- Pomen programa so matematični objekti v *semantični domeni* programa
- Semantična domena odvisna od tematike programa, vendar naj bi vsebovala matematične objekte
- *Semantične funkcije* preslikajo sintaksno strukturo programa v semantično domeno (semantika se navezuje na sintakso)
- Program → Sintaksno drevo —Semantične funkcije→ Pomen
- Denotacijski princip definiranja semantičnih funkcij: pomen celote je kombinacija pomenov sestavnih delov

DENOTACIJSKA SEMANTIKA: PRIMER

- Sintaktična domena: binarna števila **B**
- Semantična domena: naravna števila **N**
- Semantična funkcija num: **B** $\longrightarrow$ **N**
- Sintaksa

  B ::= 0 | 1 | B 0 | B 1

- Semantična funkcija

  num [0] = 0, num [1] = 1

  num [B 0] = 2 * num [B]

  num [B 1] = 2 * num [B] + 1

DENOTACIJSKA SEMANTIKA BINARNIH ŠTEVIL V DCG

```
% Sama sintaksa

bnum  -->  bdigit.

bnum  -->  bnum, bdigit.

bdigit -->  [0].

bdigit  -->  [1].
```

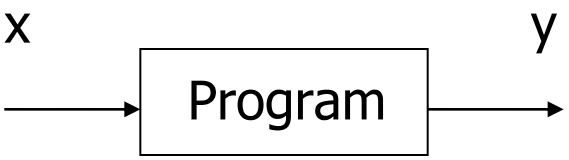
BINARNA ŠTEVILA - DODAMO POMEN

```
bnum( N)  -->  bdigit( N).          % N = pomen bin. števila

bnum( N)  -->  bnum( N1), bdigit( N2), { N is 2*N1 + N2}.

bdigit( 0) -->  [0].

bdigit( 1)  -->  [1].

?- bnum( N, [1,0,1],[]).
N = 5

?- bnum( 5, B, []).
B = [1,0,1]
```

DENOTACIJSKA SEMANTIKA: NOTACIJA

Program ⟶ Matematični objekt (običajno funkcija)

Pomen



y = Pomen [Program] (x)

Pomen [Program] : X ⟶ Y

Včasih oklepaji zapisani kot:

Oglati oklepaji - sintaktični argumenti; Pomen⟦ Program⟧

PROGRAMSKI JEZIK ZA ZELO ENOSTAVEN RAČUNALNIK (primer, Meyer)

- Računalnik: register + izhodni trak
- Podatek = začetno stanje registra
- Rezultati = končna vsebina izhodnega traku

Primer programa

```
[ begin,
    dte,          % Double to even: podvoji vsebino registra
    dto,          % Double to odd: podvoji vsebino registra, prištej 1
    print,
    dto,
    print,
    dto,
    halve,
    dte,
    print,
  end]
```

Npr.: Začetna vrednost registra = 0, rezultat je izhodni trak: 1, 3, 6

SINTAKSA

```
% Sintaksa jezika za ta racunalnik

program  -->  [begin], instructs, [end].

instructs  -->  instr.

instructs  -->  instr, instructs.

instr  -->  [ dte].          % Double to even: reg := 2*reg
instr  -->  [ dto].          % Double to odd: reg := 2*reg + 1
instr  -->  [ halve].        % reg := reg div 2
instr  -->  [ print].        % Print reg on output tape
```

DENOTACIJSKA SEMANTIKA

- Pomen programa določi semantična funkcija:

  $M_{program}[p]$: N --> N*

  $M_{program}$: Program --> (N --> N*)

- Pomen ukaza (instr) določa semantična funkcija:

  $M_{instr}[i]$: State --> State

  M_{instr}: instr --> (State --> State)

POMEN UKAZA

```
Minstr[i: instr] ( (reg,tape): state)  =
    case i of
        dte:        ( 2*reg, tape)
        dto:        ( 2*reg+1, tape)
        halve:      ( reg div 2, tape)
        print:      ( reg, tape + reg)
    end
```

POMEN ZAPOREDJA UKAZOV

$M_{instructs}$: Instructs --> (State --> State)

$M_{instructs}$[c:instructs] ((reg,tape): state) =
if empty(tail(c)) then
M_{instr}[head(c)] ((reg,tape))
else
$M_{instructs}$[tail(c)] (M_{instr}[head(c)] ((reg,tape)))

DODAMO POMEN V DCG

```
program( (R0 --> OutTape) )  -->
    [begin], instructs(( (R0,[]) --> (R,OutTape) ) ), [end].

instructs( Minstr )  -->
            instr( Minstr ).

instructs( ( State0 --> State ) )  -->
            instr( ( State0 --> State1 ) ),
            instructs( ( State1 --> State ) ).
```

POMEN, NAD.

```
instr( ( (R0,Tape) --> (R,Tape) ) )  -->
        [ dte], { R is 2*R0}.

instr( ( (R0,Tape) --> (R,Tape) ) )  -->
        [ dto], { R is 2*R0 + 1}.

instr( ( (R0,Tape) --> (R,Tape) ) )  -->
        [ halve], { R is R0 // 2}.

instr( ( (R,Tape0) --> (R,Tape) ) )  -->
        [ print], { conc( Tape0, [R], Tape)}.
```

POMEN PROGRAMOV Z ZANKO

- Primer programa:

```
[ begin,
    dte,
    while, reg_below( 100), do,          % Reg < 100
        begin, dte, dto, print, end,
  end]
```

V POMEN EKSPLICITNO UVEDEMO FUNKCIJE

- Predstavitev funkcije
- Funkcija je objekt oblike:

  fun(X, Y, Constraints)

- Funkcija preslika X --> Y
- Omejitve so navadno podane kot zaporedje prologovih ciljev

PRIMER FUNKCIJE

- F = fun(X, Y, Y is 2*X+1)

- ?- F = fun(X, Y, Y is 2*X+1),
  apply(3, Y, F).
  Y = 7

PREDIKAT APPLY

- Definicija pomožnega predikata apply za klic funkcij

- apply(Arg, Res, F) : uporabi funkcijo F na Arg, rezultat je Res

```
apply( X, Y, fun( X, Y, Constraints))  :-
   Constraints.   % Izvedi omejitve Constraints na argumentih X in Y
```

PREDIKAT ZA ITERACIJO

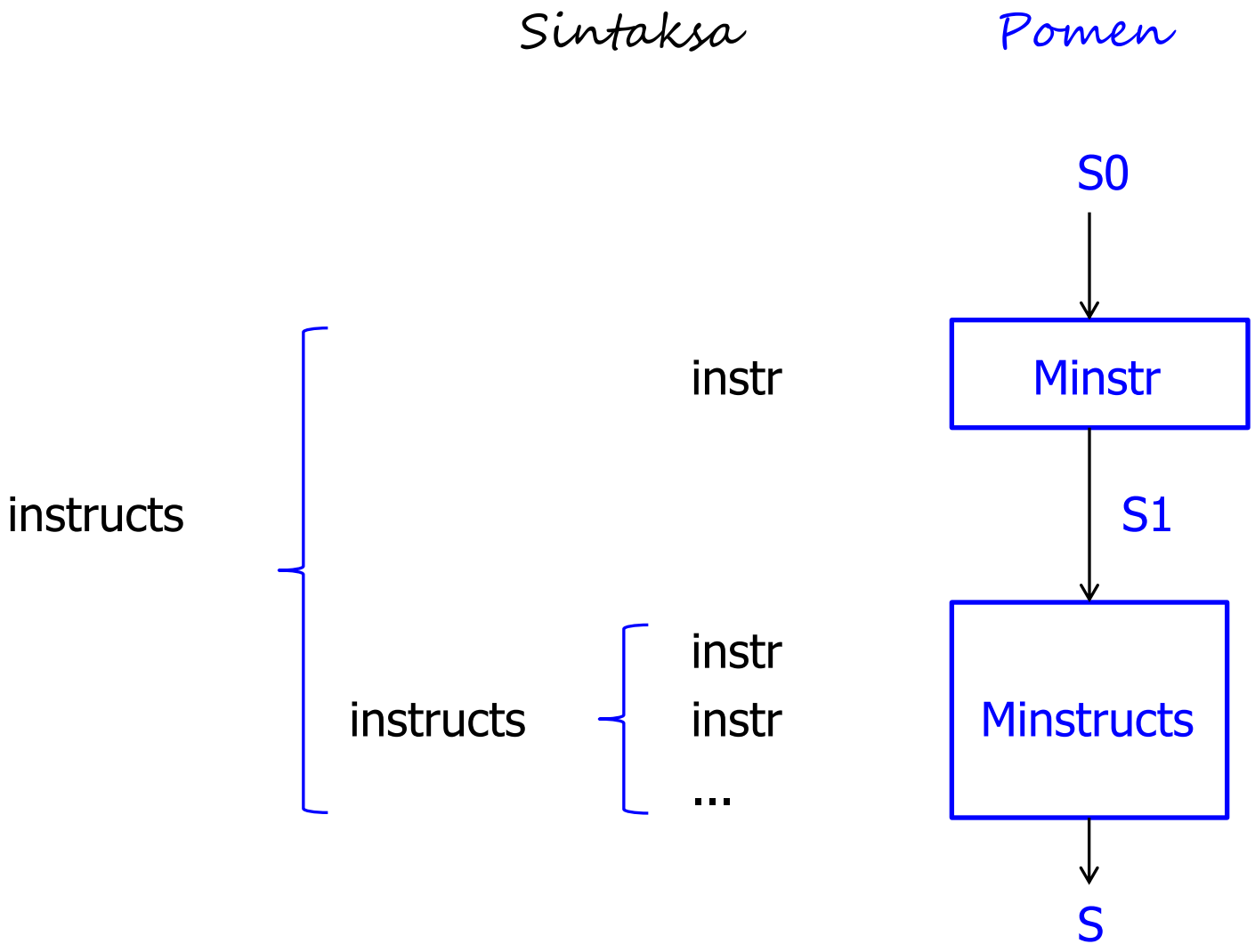
while Condition do BodyOfLoop

```
% loop( State0, Meaning_of_Condition, Meaning_of_Body, State)

loop( State0, Mcond, _, State0)  :-        % State unchanged
   apply( State0, false, Mcond), !.        % Condition false - exit loop

loop( S0, Mcond, MBody, S)  :-             % Here condition is true
   copy_term( MBody, MBodyCopy),           % Copy Mbody (body meaning)
   apply( S0, S1, MBody),                  % Execute body of loop
   loop( S1, Mcond, MBodyCopy, S).         % Iterate
```

SEMANTIKA ZA JEZIK ENOSTAVNEGA RAČUNALNIKA Z ZANKO

```
program( fun( R0, Tape, apply( (R0,[]), (_,Tape), Minstructs ) ) )  -->
   [begin], instructs( Minstructs), [end].

instructs( Minstr )  -->  instr( Minstr).

instructs( fun( S0, S,                    % See diagram next slide
            ( apply( S0, S1, Minstr), apply( S1, S, Minstructs) ) ) )  -->
   instr(Minstr), instructs( Minstructs).

instr( fun( (R0,Tape), (R,Tape), R is 2*R0 ) )  -->  [ dte].
```

POMEN SESTAVLJENEGA UKAZA instructs



SEMANTIKA JEZIKA Z ZANKO

```
instr( fun( S0, S, loop( S0, Mcond, Minstructs, S) ) )   -->
   [ while], cond( Mcond), [do, begin], instructs( Minstructs), [end].

cond( fun( (Reg, _), TruthVal,
         ( Reg < X, !, TruthVal = true; TruthVal = false) ) )   -->
   [ reg_below(X) ].        % Register value less than X
```

...

ENOSTAVEN ALGORITMIČNI JEZIK

- Primer programa:

```
begin
   a  :=  a + b
   b  :=  a - b
   a  :=  a - b
end
```

- Kaj naredi ta program?

PRIMER PROGRAMA, SEZNAMSKI ZAPIS

- Seznamski zapis za DCG gramatiko:

```
program1(
    [ begin,
        a, :=, a+b,
        b,  :=,  a-b,
        a, :=, a-b,
      end]).
```

MALO BOLJ ZANIMIV PROGRAM

```
begin
  r := a
  q := 0
  while b =< r do
        begin
            q := q + 1
             r := r - b
        end
   end
```

- Kaj naredi ta program?

SINTAKSA JEZIKA

```
program  -->  [begin], instructs, [end].

instructs  -->  instr.

instructs  -->  instr, instructs.

instr  -->  var, [ := ], expr.              % Prireditveni stavek

instr  -->  [ print(X)], {atom(X)}.         % Izpiši vrednost spremenljivke X

instr  -->  [while], cond, [do, begin], instructs, [end].     % While zanka
```

SINTAKSA, NAD.

```
cond  -->  expr, [ < ], expr.      % Expression1 < Expression2

cond  -->  expr, [ =< ], expr.     % Expression1 =< Expression2

var  -->  [ X ], { atom(X)}.       % Spremenljivka je atom

expr  -->  [ Expr].                % Katerikoli legalni izraz v prologu
                                   % (pričakujemo aritm. izraz)
```

SEMANTIKA

```
program( fun( S0, S, apply( S0, S, Minstructs ) ) )  -->
   [begin], instructs( Minstructs), [end].

instructs( Minstr )  -->  instr( Minstr).

instructs( fun( S0, S, ( apply( S0, S1, Minstr), apply( S1, S, Minstructs) ) ) )
    -->
   instr(Minstr), instructs( Minstructs).
```

SEMANTIKA, PRIREDITVENI STAVEK

- Predstavitev stanja: seznam elementov oblike Var = Value
  Vsebuje lahko tudi "izhodni trak" printout = [Val1, Val2, ...]

- Primer: State = [a = 9, b = 2, q = 0, r = 9, printout = [0, 9]]

```
instr( fun( S0, [ X = Value | S1],      % Dodaj novo vrednost sprem. X
        ( apply( S0, Value, Mexpr),     % Izrač. vrednost izraza
          del( X = _, S0, S1) ) ) )     % Briši staro vrednost sprem. X
  -->
  var( X), [ := ], expr( Mexpr).
```

SEMANTIKA, PRINT(X)

```
instr( fun( S0, [ printout = L1 | S1],
          ( member( X=V, S0),
            del( printout = L0, S0, S1),
            conc( L0, [V], L1) ) ) )
   -->
   [ print( X)].              % Print value of variable X on output
```

SEMANTIKA, WHILE ZANKA

```
instr( fun( S0, S,
        loop( S0, Mcond, Minstructs, S) ) )
   -->
   [ while], cond( Mcond), [do, begin], instructs( Minstructs), [end].
```

SEMANTIKA, POGOJNI IZRAZ

```
cond( fun( S, TruthVal,
        ( apply( S, Val1, ME1),
          apply( S, Val2, ME2),
          ( Val1 < Val2, !, TruthVal = true; TruthVal = false) ) ) )
   -->
   expr( ME1), [ < ], expr( ME2).          % Expression1 < Expression2
```

SEMANTIKA, VAR, EXPR

```
var( X)  -->  [ X ], { atom(X)}.

expr( fun( S, Value, eval( Expr, S, Value) ) )  -->
   [ Expr].          % Expr je aritm. izraz v prologu, npr. x + 2*y + 3
```

EVALUACIJA IZRAZOV

- Vradnost izraza za dano stanje programskih spremenljivk

```
?-  eval( x + 2*y + 3, [ x=2, y = 1, z=9], Value).
Value = 7
```

EVALUACIJA IZRAZOV

```
% eval( Expr, State, Value):  Expr in state State evaluates to Value

eval( N, _, N)  :-
   number( N), !.          % The value of a number is the number itself

eval( X, State, Val)  :-
   atom( X),  !,                    % A program variable
   member( X = Val, State).         % Current value of X

eval( E1 + E2, State, Val)  :-      !,        % The sum of E1 and E2
   eval( E1, State, V1),
   eval( E2, State, V2),
   Val is V1 + V2.

...
```

SEMANTIKA PROGRAMSKIH JEZIKOV: POVZETEK

- Razni pristopi:
  - prevajalska semantika,
  - operacijska semantika,
  - denotacijska semantika,
  - aksiomatska semantika, ...

- Denotacijska semantika
  - pomeni stavkov so matematični objekti, npr. funkcije
  - semantika se navezuje na sintaksno strukturo
  - pomen stavčne fraze je defniran kot kombinacija pomenov podfraz
  - za pomen programa potrebujemo stanje računalnika (ali del stanja)

- Denotacijsko semantiko lahko vgradimo v DCG gramatiko