Programski prevodioci

05 Semantička analiza - implementacija

Fakultet tehničkih nauka, Novi Sad
22-23/Z
Dunja Vrbaški

Semantičke vrednosti

Semantička vrednost tokena (terminala)

Koristi se globalna promenljiva
yylval

```
"int"                { yylval... = INT; return _TYPE; }

"+"                  { yylval... = ADD; return _AROP; }

[a-zA-Z][a-zA-Z0-9]* { yylval... = strdup(yytext); return _ID; }
```

INT, ADD - elementi enumeracija, u nastavku
strdup - kopija stringa

Semantička vrednost pojma
(neterminala)

- $$ kome se može dodeliti vrednost u akciji
- Ukoliko nema akcije podrazumevano je da je $$ = $1
  (osim u slučaju praznog pravila – nedefinisano)
- $n se odnose na vrednosti n-te komponente u pravilu

```
exp
  : literal
  | _ID
      {
        $$ = ...
      }
  | function_call
  | _LPAREN num_exp _RPAREN
      { $$ = ... }
  ;
```

Kog tipa su ove vrednosti?

- sematičke vrednosti (naravno) imaju tip
- podrazumevano: int
- može da se promeni; na primer: definiše se da su sve vrednosti tipa double

- međutim, često nije dovoljan jedan tip (kao što nama nije dovoljan)
- potrebno: definisati sve potrebne tipove i definisati kome odgovara koji tip
- može da se uradi na više načina

```
"int"                 { yylval... = INT; return _TYPE; }

"+"                   { yylval... = ADD; return _AROP; }

[a-zA-Z][a-zA-Z0-9]* { yylval... = strdup(yytext); return _ID; }
```

Potrebno nam je da nekad tip bude int, a nekad string

%union deklaracija

```
%union {
  int i;
  char *s;
}
```

- definiše sve moguće tipove
- odgovara uniji u C (Šta je union u C?)

```
%union {
  int i;
  char *s;
}
```

```
%token <i> _TYPE
…
%token <s> _ID
%token <s> _INT_NUMBER
%token <s> _UINT_NUMBER
…
%token <i> _AROP
%token <i> _RELOP

%type <i> num_exp exp literal function_call argument rel_exp
```

minic.y

```
"int"                 { yylval.i = INT; return _TYPE; }

"+"                   { yylval.i = ADD; return _AROP; }

[a-zA-Z][a-zA-Z0-9]* { yylval.s = strdup(yytext); return _ID; }
```

minic.l

Akcije su obično na kraju pravila, ali ne moraju biti.
Računaju se kao komponente prilikom referenciranja za dobijanje semantičkih vrednosti

```
pojam
  : prvi_deo_pravila
      {
          ...
      }
    drugi_deo_pravila
      {
          ...
      }
  ;
```

1

mid-rule action

2

action

```
pojam
  : prvi_deo_pravila
      {
      $<i>$ = ...
      }
    drugi_deo_pravila
      {
      ...$<i>n...
      $$ = ...
      }
  ;
```

1

2

- semantičkoj vrednosti akcije (1) se pristupa preko $$ - zapravo preko $<tip>$
- mora se navesti tip jer nemamo definisano kog tipa je ta akcija
  (za razliku od tokena i pojmova)
- u (2) možemo pristupiti vrednosti te, gore definisane, akcije preko $<tip>n
- u akciji (1) ne možemo koristiti $$ za postavljanje vrednosti celog pravila
- samo poslednju akciju (2) možemo iskoristiti za to

Implementacija TS

- enumeracije za vrstu (kind: FUN, VAR,…) i tip (type) identifikatora
- struktura za elemenat tabele
- funkcije za rad sa tabelom

```
#define SYMBOL_TABLE_LENGTH   64
```

defs.h

```
enum types { NO_TYPE, INT, UINT };

enum kinds { NO_KIND = 0x1, REG = 0x2, LIT = 0x4,
             FUN = 0x8, VAR = 0x10, PAR = 0x20 };
```

defs.h

```
#define err(args...)  sprintf(char_buffer, args), \
                      yyerror(char_buffer)
#define warn(args...) sprintf(char_buffer, args), \
                      warning(char_buffer)
```

defs.h

symtab.h

```
// Element tabele simbola

typedef struct sym_entry {
   char *   name;          // ime simbola
   unsigned kind;          // vrsta simbola
   unsigned type;          // tip vrednosti simbola
   unsigned atr1;          // dodatni attribut simbola
   unsigned atr2;          // dodatni attribut simbola
} SYMBOL_ENTRY;
```

symtab.c

```
SYMBOL_ENTRY symbol_table[SYMBOL_TABLE_LENGTH];
int first_empty = 0;
```

Funkcije za rad sa TS

```
// Vraca indeks prvog sledeceg praznog elementa.
// greska ako prekoracimo
int get_next_empty_element(void);


// Vraca indeks poslednjeg zauzetog elementa.
int get_last_element(void);
```

```
// Ubacuje novi simbol (jedan red u tabeli)
// i vraca indeks ubacenog elementa u tabeli simbola
// ili -1 u slucaju da nema slobodnog elementa u tabeli.
int insert_symbol(char *name, unsigned kind, unsigned type,
                  unsigned atr1, unsigned atr2);
```

```
// Ubacuje konstantu u tabelu simbola (ako vec ne postoji).
int insert_literal(char *str, unsigned type);
```

registri se ubacuju prilikom inicijalizacije

```
// Ubacuje novi simbol (jedan red u tabeli)
// i vraca indeks ubacenog elementa u tabeli simbola
// ili -1 u slucaju da nema slobodnog elementa u tabeli.
int insert_symbol(char *name, unsigned kind, unsigned type,
                  unsigned atr1, unsigned atr2);
```

```
// Ubacuje konstantu u tabelu simbola (ako vec ne postoji).
int insert_literal(char *str, unsigned type);
```

Obratiti pažnju:

`insert_symbol`

- Ubacuje simbol, informacije o identifikatoru (promenljiva, funkcija, parameter).
- get_next_empty_element + set vrednosti
- Ne proverava se ovde da li već postoji taj identifikator u tabeli.
- To je zadatak za proces analize, ovo je samo SP

`insert_literal`

- Proverava da li već postoji simbol u tabeli
- Sam literal ne nosi puno semantičkih informacija, samo vrednost i tip.
- Nema potrebe da unosimo istu konstantu nekoliko puta u tabelu.
- Dodatno ima i proveru opsega

```
// Vraca indeks pronadjenog simbola ili vraca -1.
int lookup_symbol(char *name, unsigned kind);
```

```
// set i get metode za polja tabele simbola
void     set_name(int index, char *name);
char*    get_name(int index);
void     set_kind(int index, unsigned kind);
unsigned get_kind(int index);
void     set_type(int index, unsigned type);
unsigned get_type(int index);
void     set_atr1(int index, unsigned atr1);
unsigned get_atr1(int index);
void     set_atr2(int index, unsigned atr2);
unsigned get_atr2(int index);
```

```
// Inicijalizacija tabele simbola
// koristi se pre parsiranja
void init_symtab(void);

// Brise sve elemente tabele simbola.
// korisiti se posle parsiranja
void clear_symtab(void);

// Brise elemente tabele od zadatog indeksa
// koristi se kad se završi parsiranje funkcije
void clear_symbols(unsigned begin_index);

// Ispisuje sve elemente tabele simbola.
void print_symtab(void);
```

Implementacija semantičkih pravila

- prilikom parsiranja
- realizuju se u akcijama pravila u .y fajlu
- koristi se TS

```
int var_num = 0;
```

Broj identifikatora u trenutnom opsegu (funkciji)

```
int fun_idx = -1;
```

Indeks simbola funkcije koja se trenutno parsira

```
int fcall_idx = -1;
```

Indeks simbola funkcije čiji poziv parsiramo

P: svi lokalni identifikatori iste funkcije (lokalne promenljive i parametri) moraju biti međusobno različiti

```
variable
  : _TYPE _ID _SEMICOLON
      {
        if(lookup_symbol($2, VAR|PAR) == NO_INDEX)
           insert_symbol($2, VAR, $1, ++var_num, NO_ATR);
        else
            err("redefinition of '%s'", $2);
      }
  ;
```

```
int f(int x) {
   int a;
   int a; //error
   int x; //error
   a = x + x;
   return a;
}
```

```
#define NO_INDEX                -1
```

defs.h

```
enum kinds { NO_KIND = 0x1, REG = 0x2, LIT = 0x4,
             FUN = 0x8, VAR = 0x10, PAR = 0x20 };
```

```
...symbol_table[i].kind & kind)...
```

symtab.c

P: svi globalni identifikatori moraju biti međusobno različiti
P: opseg vidljivosti lokalnih id-a je od definicije do kraja funkcije u kojoj su definisani

```
function
  : _TYPE _ID
      {
        fun_idx = lookup_symbol($2, FUN);
        if(fun_idx == NO_INDEX)
          fun_idx = insert_symbol($2, FUN, $1, NO_ATR, NO_ATR);
        else
           err("redefinition of function '%s'", $2);
      }
    _LPAREN parameter _RPAREN body
      {
        clear_symbols(fun_idx + 1);
        var_num = 0;
      }
  ;
```

```
int f(int x) {
   int a;
   a = x + x;
   return a;
}

int f() {  //error
   int a;
   a = 5;
   return a;
}
```

Inicijalno:

```
int var_num = 0;
```

```
int fun_idx = -1;
```

```
NO_ATR, NO_ATR  ?
```

```
parameter
  : /* empty */
      { set_atr1(fun_idx, 0); }
  | _TYPE _ID
      {
        insert_symbol($2, PAR, $1, 1, NO_ATR);
        set_atr1(fun_idx, 1);
        set_atr2(fun_idx, $1);
      }
  ;
```

STRING SIMBOLA	VRSTA SIMBOLA	TIP SIMBOLA	ATRIBUT SIMBOLA	ATRIBUT SIMBOLA
main	FUN	povratni tip funkcije	broj parametara	tip parametra
p	PAR	tip parametra	redni broj parametra	-
x	VAR	tip promenljive	redni broj promenljive	-
100	LIT	tip literala	-	-
%0	REG	tip sadrzaja registra	-	-

Ne realizuje se nijedno pravilo samo se menja TS:

- dodaje se element za parameter
- menja se element za funkciju (dodat kroz prvu akciju u pravilu za function)