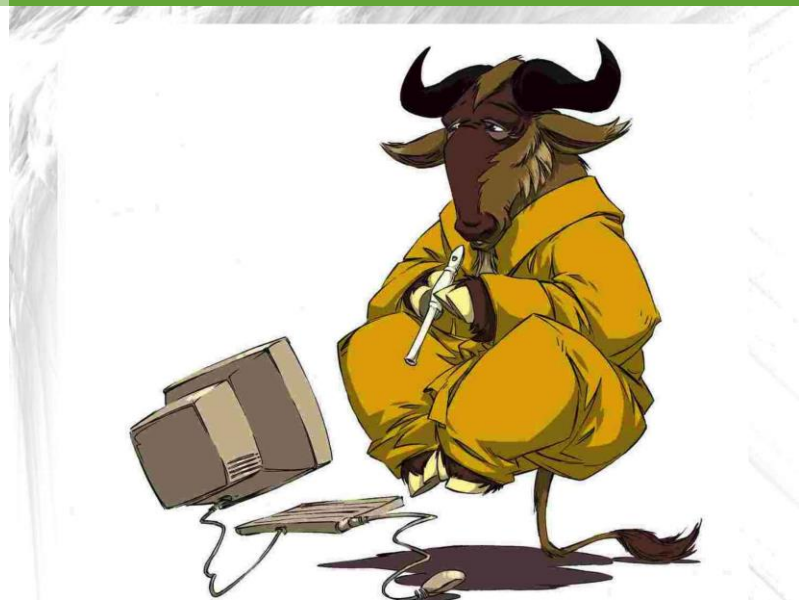
Εισαγωγή στο Bison (Μέρος ΙΙ)

Μεταγλωττιστές,

Χειμερινό εξάμηνο 2018-2019



Τι κάνει το Bison

- Εργαλείο παραγωγής συντακτικού αναλυτή.
- Default LALR(1) γραμματικές.
- Περιέχει σημασιολογικές ρουτίνες.
- Παράγει ένα C αρχείο από ένα .y αρχείο που κατασκευάζουμε εμείς.
- Συνοπτικά χρησιμοποιείται για:
  - Αναγνώριση ή απόρριψη συμβολοσειρών της γλώσσας.
  - Την παραγωγή ενδιάμεσου κώδικα.
  - Την παραγωγή δένδρου συντακτικής ανάλυσης.

Δομή Αρχείου Εισόδου

- Το Bison λαμβάνει ως είσοδο ένα μοναδικό αρχείο προγράμματος με κατάληξη “.y” που αποτελείται από 3 μέρη(όπως και το flex).
  - Ορισμοί.
  - Κανόνες.
  - Συναρτήσεις.
- Δομή:

```
/*Statements block*/
%%
/*Rules block*/
%%
/*User Functions*/
```

Συντακτική Ανάλυση Bottom-Up

- Ο συντακτικός αναλυτής που παράγεται απο τον bison είναι αναλυτής “από κάτω προς τα επάνω” (bottom-up).
- Για την υλοποίηση μιας “από κάτω προς τα επάνω”ανάλυσης, χρησιμοποιούνται τα εξής:
  - μια στοίβα αποθήκευσης των στοιχείων της γλώσσας.
  - η ενέργεια “μετάθεση/ολίσθηση” (shift) που τοποθετεί το επόμενο στοιχείο εισόδου στην στοίβα.
  - η ενέργεια “αναγωγή/ελάττωση“ (reduce) που εφαρμόζεται στην κορυφή της στοίβας όταν έχει εμφανιστεί το δεξί μέλος ενός κανόνα και το αντικαθιστά με το αριστερό μέλος του.
  - οι ενέργειες accept και abort που δηλώνουν την επιτυχημένη ή όχι ανάλυση της εισόδου

Κατηγορήματα

- Κάθε σύμβολο στο Bison συνοδεύεται και από μία τιμή
- Δήλωση του τύπου του κατηγορήματος με χρήση **%type** για μη τερματικά σύμβολα
  - Και τα τερματικά σύμβολα έχουν τιμές
- Οι τιμές των συντιθέμενων κατηγορημάτων N συμβόλων του RHS προσπελαύνονται μέσω των $1-$N
  - Ένα κατηγόρημα για κάθε σύμβολο
- Τα κατηγορήματα του LHS προσπελαύνονται μέσω του $$

Παραδείγματα

- Υπάρχει δυνατότητα για πρόσβαση και στα κληρονομούμενα κατηγορήματα.
  - Πρόσβαση μέσω των $0,$-1….
- Παράδειγμα

```
declaration : class type nanelist;
class: GLOBAL {$$=1;}
| LOCAL {$$=2;}
;
type: REAL {$$=1;}
| INTEGER {$$=2;}
;
namelist : NAME {mksymbol($0,$-1,$1);}
| namelist NAME {mksymbol($0,$-1,$2);}
;
```

Συγκρούσεις

- Το Bison μπορεί να δημιουργεί LALR(1) συντακτικούς αναλυτές.
- Δεν μπορούν να χειριστούν διφορούμενες γραμματικές.
- Το Bison μπορεί:
  - Shift-Reduce collision → Shift
  - Reduce-Reduce → Προτιμάται ο κανόνας που έχει γραφτεί
  - πρώτος
- Ούτε γραμματικές που απαιτούν περισσότερα του ενός lookahead symbols.
- Αν χρησιμοποιήσουμε την επιλογή bison –v παράγεται ένα αρχείο (.output) με περιγραφές των συγκρούσεων.
- Χειροκίνητη επίλυση συγκρούσεων.
  - Με την χρήση των δηλώσεων προτεραιότητας προσεταιριστικότητας.

Είδη Συγκρούσεων

- Shift/Reduce:
  - Γραμματικές Τελεστών.
  - Dangling ELSE.
  - Nested lists of items.
  - Ο ΣΑ παρέχει μεθόδους επίλυσης.
- Reduce/Reduce:
  - Limited lookahead.
  - Overlap of alternatives.
  - Συνήθως, επιλύονται με τροποποίηση της γραμματικής.

Παράδειγμα Συγκρούσεων

- Παράδειγμα:
  - exp: exp ‘+’ exp
  - | exp ‘-’ exp
  - | exp ‘*’ exp,
- 3*4+5→(3*4)+5 ή 3*(4+5)?
- Επίλυση με χρήση προτεραιότητας και προσαιτεριστικότητας.
- Χρήση των %left, %right, %nonassoc στις δηλώσεις του πρώτου μέρους.
  - %left ‘+’,’-’
  - %left ‘*’,’/’
  - %right ‘=‘
- Χρήση του %prec για την προτεραιότητα των κανόνων.

Dangling Else(Shift/Reduce)

- Κανόνας γραμματικής:
  - stmt: IF cond stmt
    | IF cond stmt ELSE stmt …
- Μια δομή else αντιστοιχεί στην πιο κοντινή δομή if.
- Επιβολή αυτής της κατεύθυνσης στον ΣΑ.
  - Τροποποίηση της γραμματικής
  - Χρήση του THEN
  - Χρήση ενός fake token

Dangling Else

- Χρήση του THEN:

```
%nonassoc THEN
%nonassoc ELSE
stmt: IF expr THEN stmt
        | IF expr stmt ELSE stmt
        ;
```

Dangling Else

- Επίλυση με fake token:

```
%nonassoc LOWER_THAN_ELSE
%nonassoc ELSE
stmt : IF cond stmt %prec LOWER_THAN_ELSE
        | IF cond stmt ELSE stmt
```

Overlap (Reduce/Reduce)

- Υπάρχουν δύο εναλλακτικοί κανόνες με το ίδιο LHS.

```
person: girls
        | boys
        ;
girls: ALICE
        | BETTY
        | CHRIS
        | DARRYL
        ;
boys: ALLEN
        | BOB
        | CHRIS
        | DARRYL
        ;
```

Θα πάρετε μια reduce/reduce σύγκρουση για CHRIS και DARRYL επειδή bison δεν μπορεί να πει εάν προορίζονται να είναι κορίτσια ή αγόρια.

Overlap (Reduce/Reduce)

- Επίλυση με προσθήκη εξτρά κανόνα.

```
person: girls
        | boys
        | either
        ;
girls: ALICE
        | BETTY
        ;
boys: ALLEN
        | BOB
        ;
either: CHRIS
        | DARRYL
        ;
```

Συνοψίζοντας

- Βρείτε το σφάλμα shift/reduce στο name.output.
- Προσδιορίστε τον κανόνα reduce.
- Προσδιορίστε τον (τους) σχετικό κανόνα shift.
- Δείτε που θα κάνει reduce ο κανόνας reduce.

Ανάνηψη Λαθών

- Διαδικασία κατά την οποία ανιχνεύεται ένα συντακτικό σφάλμα και συνεχίζεται η ανάλυση για την ανίχνευση περισσότερων λαθών.

- Τεχνικές ανάνηψης:
  - Error Recovery
  - Error Repair
  - Error Correction

Ανάνηψη Λαθών

- Το Bison προσφέρει έναν αποδοτικό μηχανισμό για ανάνηψη από λάθη.
- Χρήση του ειδικού **error** token.
- Σύμβολα που ακολουθούν το **error** αποτελούν σημεία συγχρονισμού.
- Μόλις βρεθεί ένα λάθος, αφαίρεση συμβόλων από τη στοίβα.
  - Μέχρι να μπορεί να εισαχθεί το σύμβολο error.
- Έπειτα, αναζητείται ένα σημείο συγχρονισμού και συνεχίζεται η κανονική διαδικασία ανάλυση.

Τι θα κάνει το Bison

- error: Όταν ανιχνευθεί σφάλμα, σηματοδοτεί μέχρι που θα πρέπει να αγνοηθεί η είσοδος, ώστε να συνεχίσει η συντακτική ανάλυση.
- stmt: error ‘;’ ; // είσοδος y = d; x = a b;

Στοίβα: stmt ‘x’ ‘=‘ ‘a’, lookahead: ‘b’
Ανιχνεύεται σφάλμα
εξάγονται τα ‘x’, ‘=‘, ‘a (Στοίβα: stmt)
shift του error (Στοίβα: stmt error)
διαγραφή ‘b’, lookahead: ‘;’
Στοίβα: stmt error ‘;’ => Στοίβα: stmt stmt

Μεταβλήτες/Συναρτήσεις Ανάνηψης Λαθών

- yyerrok: Επαναφέρει τον αναλυτή στην κανονική κατάσταση αναφοράς λαθών.
- yyerror(): Συνάρτηση εκτύπωσης μηνύματος σφάλματος.
- YYRECOVERING: Επιστρέφει μη μηδενική τιμή όταν ο αναλυτής βρίσκεται σε κατάσταση ανάνηψης από σφάλμα.
- %error-verbose: οδηγία για πιο κατατοπιστικά μηνύματα
- yyclearin: Macro το οποίο αγνοεί το πιο πρόσφατο lookahead token.
  - Χρησιμοποιείται κυρίως σε interactive ΣΑ.

Σημεία Συγχρονισμού

- Που τοποθετούνται τα error token?
- Tradeoff μεταξύ ακρίβειας και πλήθους συμβόλων που αγνοούνται.
- Καταλληλότερα τα σημεία όπου τερματίζονται λίστες εντολών.
- Παράδειγμα:

```
loop_statement : DO ID ASSIGN iter_space body ENDDO
                | DO error ENDDO {yyerrok;}
                ;
```

Παράδειγμα Calculator

```
%{
    #include <stdio.h>
    void yyerror (char const *s);
    extern int num_lines;
%}
   %token TK_NUM TK_ID TK_BADID
   %left '-' '+'
   %left '*' '/'
   %left TK_NEG     /* negation--unary minus */
   %right '^'    /* exponentiation */

%% /* The grammar follows.  */
   input:    /* empty */
         | input line
   ;
   line:     ';'
         | exp ';'
   ;
```

```
exp: TK_NUM | TK_ID
          | exp '+' exp            {printf("exp + exp\n");}
          | exp '-' exp            {printf("exp - exp\n");}
          | exp '*' exp            {printf("exp * exp\n");}
          | exp '/' exp            {printf("exp / exp\n");}
          | '-' exp  %prec TK_NEG    {printf("-exp\n");}
          | exp '^' exp            {printf("exp ^ exp\n");}
          | '(' exp ')'            {printf("( exp )\n");}
   ;
%%
int main(void) {
          return yyparse();
}

void yyerror (char const *s) {
          printf("%d: %s\n", num_lines, s);
}
```

Παράδειγμα Calculator

- **Είσοδος:**
- inv 1234;
- 234inv + 234;
- 1234 + a;
- (2345);
- (inv;
- 1243 + 234;
- 234);
- 234 * 234;

- **Έξοδος:**
- syntax error

Παράδειγμα Calculator με Χρήση **error**

```
%{
    #include <stdio.h>
    void yyerror (char const *s);
    extern int num_lines;
%}
   %error-verbose
   %token TK_NUM TK_ID TK_BADID
   %left '-' '+'
   %left '*' '/'
   %left TK_NEG     /* negation--unary minus */
   %right '^'    /* exponentiation */
%% /* The grammar follows.  */
   input:    /* empty */
         | input line ;
   line:     ';'
         | exp ';‘
         | exp error exp ';' {yyerror("missing arithmetic operator or ';'");
yyerrok;}
         | error ';‘ {yyerrok;} ;
```

```
exp: TK_NUM | TK_ID
   | TK_BADID          {yyerror("invalid ID"); yyerrok;}
         | exp '+' exp          {printf("exp + exp\n");}
         | exp '-' exp          {printf("exp - exp\n");}
         | exp '*' exp          {printf("exp * exp\n");}
         | exp '/' exp          {printf("exp / exp\n");}
         | '-' exp  %prec TK_NEG    {printf("-exp\n");}
         | exp '^' exp          {printf("exp ^ exp\n");}
         | '(' exp ')'          {printf("( exp )\n");}
         | '(' exp error        {yyerror("Missing operator ')'"); yyerrok;}
   ;
%%
int main(void) {
            return yyparse();
}

void yyerror (char const *s) {
            printf("%d: %s\n", num_lines, s);
```

Παράδειγμα Calculator με Χρήση **error**

- **Είσοδος:**
- inv1234;
- 234inv + 234;
- 1234 + a;
- (2345);
- (inv;
- 1243 + 234;
- 234);
- 234 * 234;

- **Έξοδος:**
- syntax error, unexpected TK_NUM
- missing arithmetic operator or ';'
- invalid ID
- exp + exp
- (exp)
- syntax error, unexpected ';‘
- Missing operator ')'
- exp + exp
- syntax error, unexpected ')'
- exp * exp

Πίνακα Συμβόλων (Μέρος ΙΙ)

- Βασικές λειτουργίες
- Εισαγωγή (insert):
  - Κάθε φορά που αναγνωρίζεται ένα όνομα δημιουργείται μία νέα εγγραφή για αυτό στον πίνακα συμβόλων, εφόσον δεν υπάρχει ήδη κατά τη δήλωση μιας νέας μεταβλητής ή συνάρτησης.
- Αναζήτηση (lookup):
  - Αναζήτηση ενός ονόματος στο τρέχον επίπεδο εμβέλειας ή σε περιέχουσα εμβέλεια.
- Απενεργοποίηση (hide):
  - Απενεργοποίηση των μεταβλητών που δεν χρειάζονται.

Παράδειγμα

```
input(x);
g = 12.4;
print(typeof(x));
function foo(x, y)
{
        print(x + y);
        local p = y;
        print(p);
        function h(a)
        {
                return a + x + y;
        }
        y = h(x);
}
```

lookup(input), lookup(x), ins(x)
lookup(g), ins(g)
lookup(print), lookup(typeof), lookup(x)
lookup(foo), ins(foo), lookup(x), ins(x), lookup(y), ins(y)
lookup(print), lookup(x), lookup(y)
lookup(p) ins(p) lookup(y)
lookup(print), lookup(p)
lookup(h) ins(h) lookup(a) ins(a)
lookup(a) , lookup(x), lookup(y),
hide(a)
lookup(y), lookup(h), lookup(x)
hide(foo.x, foo.y, foo.p, foo.h)

References

- http://web.iitd.ac.in/~sumeet/flex__bison.pdf

ΕΡΩΤΗΣΕΙΣ/ΑΠΟΡΙΕΣ