Εισαγωγή στο BISON

Εργαστήριο Μεταγλωττιστών
Τμήμα Μηχανικών Πληροφορικής
ΤΕΙ Αθήνας

2017

BISON

- Μια γεννήτρια συντακτικών αναλυτών για τις γλώσσες C/C++
- Μετατρέπει την περιγραφή μιας context-free γραμματικής σε ένα LALR συντακτικό αναλυτή γραμμένο
  - LALR = **L**ook-**A**head **L**eft-to-right parse, **R**ightmost-derivation
- *Το bison αποτελεί* μια βελτιωμένη έκδοση του *yacc*

Δομή Προγράμματος BISON

%{

Εισαγωγικό Τμήμα (προαιρετικό)

%}

Δηλώσεις bison } Τμήμα Δηλώσεων

%%

Περιγραφή γραμματικής } Τμήμα Κανόνων

%%

Κυρίως πρόγραμμα C/C++ (προαιρετικό)

Εισαγωγικό Τμήμα

- Μπορεί να περιέχει δηλώσεις macros, συναρτήσεων και μεταβλητών.
- Ότι περιέχει αντιγράφεται χωρίς αλλαγές στην αρχή του παραγόμενου αρχείου .c.
- Είναι προαιρετικό και μπορεί να παραληφθεί αφαιρώντας τα διαχωριστικά %{ και %}

```
%{
        #include <stdio.h>
        #include "token.h"
        void print_token_value (FILE *, int, YYSTYPE);
        extern int lineno;
%}
```

- Μπορούμε επίσης να έχουμε περισσότερα από ένα εισαγωγικά τμήματα ανάμεσα στα οποία παρεμβάλλονται δηλώσεις του *bison*.

Δηλώσεις Bison

- Σε αυτό το τμήμα δηλώνονται τα σύμβολα της γραμματική καθώς και κάποια χαρακτηριστικά τους
  - Δήλωση τερματικών και μη τερματικών συμβόλων
  - Δήλωση αρχικού συμβόλου
  - Καθορισμός προτεραιότητας
- Επίσης δηλώνονται κάποιες παράμετροι που επηρεάζουν το συντακτικό αναλυτή
  - Κυρίως σε σχέση με τα ονόματα των παραγόμενων αρχείων του συντακτικού αναλυτή και των συναρτήσεων που προσφέρει

Δηλώσεις Bison- Γραμματική (1/4)

- ***%token TOKEN*** – Ορίζει το τερματικό σύμβολο *TOKEN*
- ***%start symbol*** – Ορίζει το αρχικό σύμβολο της γραμματικής
  - Αν παραληφθεί, αρχικό σύμβολο θεωρείται το πρώτο μη τερματικό σύμβολο που εμφανίζεται στο τμήμα της περιγραφής της γραμματικής

Άλλες δηλώσεις

- ***%union*** – Ορίζει τους τύπους που μπορούν να πάρουν τα σύμβολα (τερματικά και μη)
- ***%token <intVal> TOKEN*** – Ορίζει το τερματικό σύμβολο *TOKEN*, με τύπο αυτό που προκύπτει από το union.
- ***%type <intVal> expr*** – Ορίζει το μη τερματικό σύμβολο *expr* με τύπο αυτό που προκύπτει από το union.

Δηλώσεις Bison- Γραμματική (2/4)

- ***{ code } symbols*** – Ορίζει ένα τμήμα κώδικα που εκτελείται για τα δοθέντα σύμβολα όταν αυτά σταματήσουν να χρησιμοποιούνται
  - Πολύ χρήσιμο για αποδέσμευση μνήμης σε περιπτώσεις λάθους

```
%union { char *string; }
%token <string> STRING
%destructor { free ($$); } STRING
```

- ***%expect n*** – Δηλώνει ότι αναμένουμε η γραμματική μας να έχει *n* conflicts

Δηλώσεις Bison- Γραμματική (3/4)

- Προτεραιότητες
  - ***%left, %right***
  - **Ορίζουν την προτεραιότητα στα token που ακολουθούν στη γραμμή με αυξανόμενη προτεραιότητα από πάνω προς τα κάτω, π.χ.**

%left ADD, SUB, '(', ')'

Το *left* σημαίνει ότι έχουμε αριστερή προσεταιριστικότητα

Το `1+2-3` **σημαίνει** `(1+2)-3`

%left MUL, DIV

%right EXP , EQ

Το *right* σημαίνει ότι έχουμε δεξιά προσεταιριστικότητα

Το `2^3^4` υπολογίζεται ως `2^(3^4)`

Το `1+2*3^4^5-6` **υπολογίζεται ως**

```
(1 + (2 * (3 ^ (4 ^ 5)))) - 6
```

Δηλώσεις Bison- Γραμματική (4/4)

- Προτεραιότητες
  - **%nonassoc** EQ, UMINUS (=, πρόσημο -)

Το nonassoc σημαίνει ότι δεν υπάρχει προσεταιριστικότητα

%prec UMINUS

Το **%prec** αλλάζει μια ήδη δηλωμένη προτεραιότητα μέσα σε έναν κανόνα, π.χ.

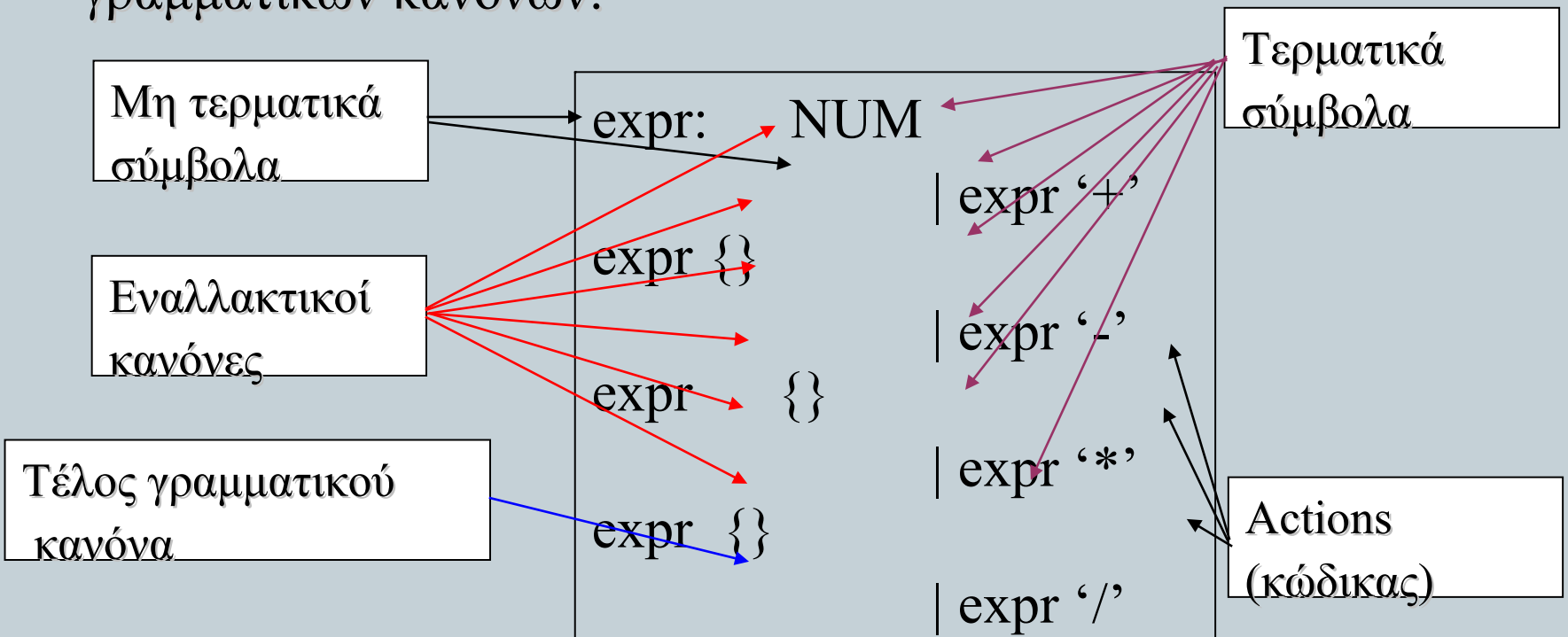
```
arithmetic_expr: '-' arithmetic_expr %prec UMINUS ;
```

Δηλώσεις Bison– Παράμετροι

- ***%defines***
  - Παράγει ένα header file με τις δηλώσεις macros για τα σύμβολα της γραμματικής, καθώς και κάποιες επιπλέον δηλώσεις
  - Αν το παραγόμενο αρχείο του συντακτικού αναλυτή είναι το *parser.c*, τότε το header file θα έχει όνομα *parser.h*
- ***%output="file"***
  - *Ορίζει το όνομα του παραγόμενου αρχείου που θα περιέχει τον κώδικα του συντακτικού αναλυτή*
- ***%error-verbose***
  - Χρησιμοποιείται για να πάρουμε πιο αναλυτικά μηνύματα λάθους στην κλήση της *yyerror*

Περιγραφή γραμματικής

- Παράδειγμα περιγραφής μιας context-free γραμματικής μέσω γραμματικών κανόνων:



- Επιτρέπεται και ο κενός κανόνας που κάνει match το κενό string, π.χ. *program: /*empty*/ | statements ;*

Μέρος κυρίου προγράμματος

- Περιέχει κυρίως συναρτήσεις που χρησιμοποιούνται από τον παραγόμενο συντακτικό αναλυτή
- Ότι προστίθεται σε αυτό το τμήμα αντιγράφεται χωρίς αλλαγές στο τέλος του παραγόμενο αρχείο .c που περιέχει τον κώδικα του συντακτικού αναλυτή.
- Το τμήμα αυτό είναι προαιρετικό και όταν παραλειφθεί μπορεί να παραλειφθεί και το δεύτερο σύμβολο “%%”

```
%%

int main(int argc, char **argv) {
            yyparse();
            return 0;
}
```

Παράδειγμα προγράμματος bison με δική του ρουτίνα Λ.Α.

Εισαγωγικό μέρος (3/6)

```
/*  αρχείο BISON  με όνομα calc-0.y (χωρίς flex) για απλές
αριθμητικές πράξεις, άθροισμα και πολ/σμο με postfix notation
(πχ. 5 4 +)  */

%{
        #include <stdio.h>
        int yylex(void);
        void yyerror(char *);
%}
%token INTEGER PLUS MULT NEWLINE

%left      '+'
%left      '*'
```

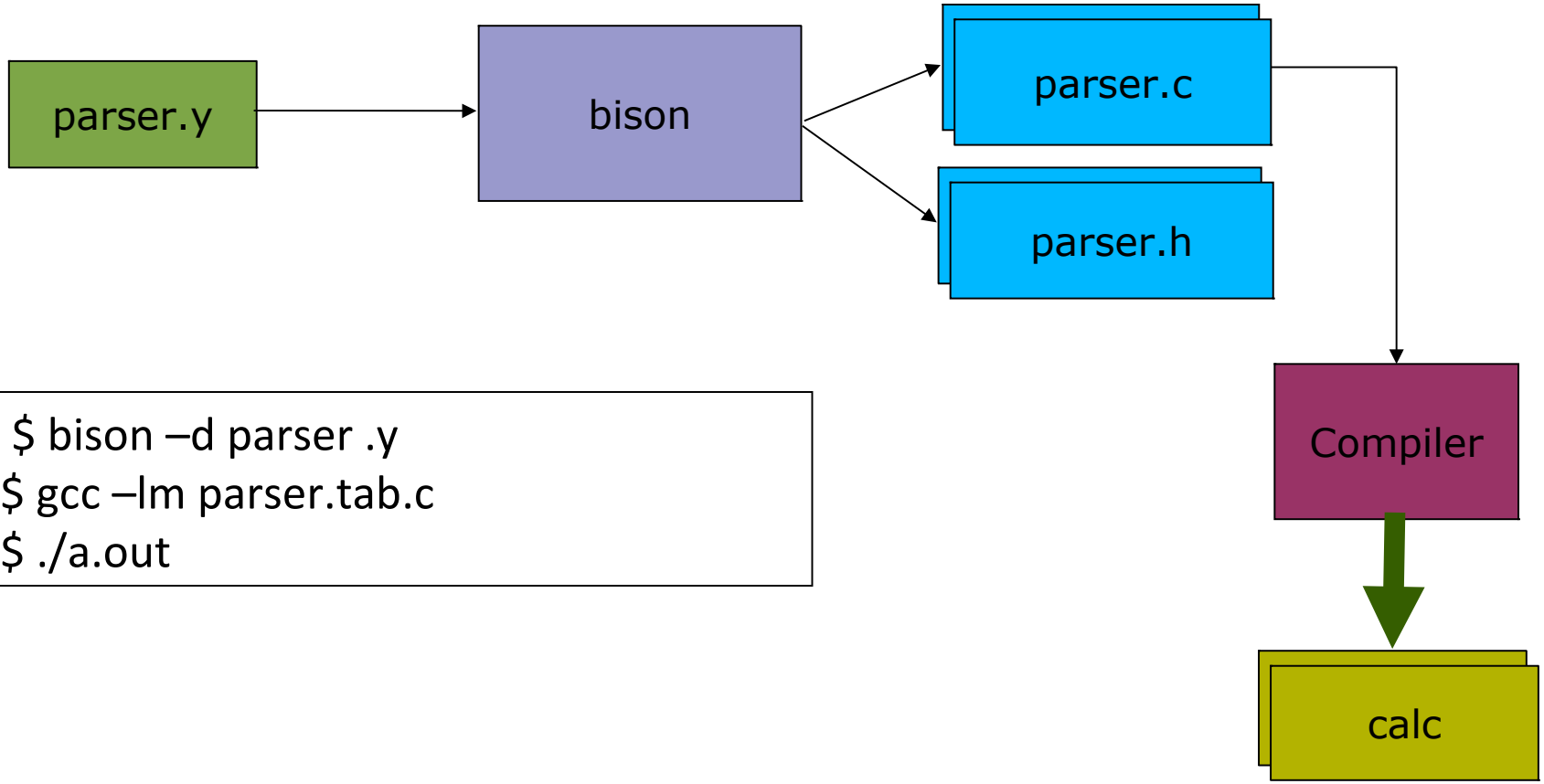
Τμήμα κανόνων (4/6)

```
%%

program:
        program expr NEWLINE { printf("%d\n", $2); }
        | /* empty*/
        ;
expr:
        INTEGER         { $$ = $1; }
        | expr expr PLUS { $$ = $1 + $2; }
        | expr expr MULT { $$ = $1 * $2; }
        ;
%%
```

Τμήμα κύριου προγράμματος (5/6)

```
yylex() {
        char num = 0; char c;
        c = getchar();

        // Ignores spaces and tabs
        while (c == ' ' || c == '\t') { yylval = 0; c = getchar(); }

        // Processes  all digits
        while (c >= '0' && c <= '9')
        {               yylval = (yylval * 10) + (c - '0');
                                num = 1;  c = getchar();         }
        if (num) { ungetc(c, stdin); return INTEGER; }
        if (c == '+') return PLUS;
        if (c == '*') return MULT;
        if (c == '\n') return NEWLINE;
        yyerror("invalid character"); }

void yyerror(char *s) {
        fprintf(stderr, "%s\n", s); }

int main(void)  {
        yyparse();
        return 0; }
```

Παραγωγή εκτελέσιμου




```
$ bison –d parser .y
$ gcc –lm parser.tab.c
$ ./a.out
```

Flex & Bison

- ***Flex***
  - Κάνει include το header file που παράγεται από το bison για να δει τα tokens και τους τύπους τους
  - Μέσα στα actions γράφουμε κώδικα που επιστρέφει στο bison ένα-ένα τα tokens που αναγνωρίστηκαν
  - Επίσης, φροντίζουμε να παρέχουμε τις τιμές για τα tokens που έχουν και κάποιο δηλωμένο τύπο
    - Π.χ. για τον ακέραιο 15 θα πρέπει να δώσουμε στο bison το token INTEGER, αλλά και να του παρέχουμε αριθμητική τιμή 15
- ***Bison***
  - Στον πρόλογο δηλώνουμε τη συνάρτηση yylex που είναι υπεύθυνη για την λεξικογραφική ανάλυση (και παρέχεται από το flex)
  - Εφαρμόζουμε τους κανόνες της γραμματικής ανάλογα με τα tokens που επιστρέφονται από την yylex
  - Στα actions χρησιμοποιούμε και τις τιμές που έχουν τα tokens με συγκεκριμένους τύπους
    - Π.χ. εκτελούμε τον κανόνα για τον ακέραιο και κατόπιν στο action παίρνουμε και την αριθμητική τιμή του

Παράδειγμα (1/6)

- Γραμματική για έναν απλό υπολογιστή αριθμητικών εκφράσεων
  - Υποστηρίζει εκφράσεις που χωρίζονται με χαρακτήρες τέλους γραμμής
  - Κάθε έκφραση είναι μια αριθμητική έκφραση ή μια εντολή if
- Λεκτικός αναλυτής
  - Σύμβολα +, -, *, /, (,), =, \n, ακέραιους, μεταβλητές
- Συντακτικός αναλυτής
  - Εκχωρήσεις σε μεταβλητές
  - Εκφράσεις που περιέχουν μεταβλητές και αριθμητικές εκφράσεις

Παράδειγμα - Flex (2/6)

```
%{
    #include "parser.h" /* <- will be generated from parser.y */
%}

/* Flex options */
%option noyywrap
%option yylineno

/* Flex macros */
id      [a-zA-Z][a-zA-Z_0-9]*
integer [0-9]+

%%

"+"     { return '+'; }
"-"     { return '-'; }
"*"     { return '*'; }
"/"     { return '/'; }
"("     { return '('; }
")"     { return ')'; }
"="     { return '='; }
[\n]+   { return '\n'; }

{integer}   { return INTEGER; }
{id}        { return ID; }

[ \t]+  {}

.       { fprintf(stderr, "Cannot match character `%s' with any rule\n", yytext); }
```

To header file *parser.h* δημιουργείται από τον bison

Μετατρέπει πολλαπλά \n σε ένα

Οι τιμές των συμβολικών ονομάτων INTEGER, ID είναι ορισμένες στο header file *parser.h*

Παράδειγμα Εισαγωγικού Τμήματος Bison (3/6)

```
%{
    #include <stdio.h>
    int yyerror (char* yaccProvidedMessage);
    int yylex (void);

    extern int yylineno;
    extern char* yytext;
    extern FILE* yyin;
%}

%start program

%token ID INTEGER

%right      '='
%left       ','
%left       '+' '-'
%left       '*' '/'
%nonassoc   UMINUS
%left       '(' ')'

%%
```

Καλείται από το Bison με ένα μήνυμα όταν "ανακαλύψει" κάποιο λάθος

Αρχικό σύμβολο γραμματικής

Δήλωση των τερματικών συμβόλων που χρησιμοποιούνται από τον Bison (και τον flex)

Ορισμός προτεραιοτήτων και προσεταιριστικότητας

Παράδειγμα Τμήματος Κανόνων (4/6)

```
%%
program:
        program expr NEWLINE
        | /* empty*/
        ;
expr:
       arith_expr
       | if_expr
       ;
arith_expr:
     ID | INTEGER
        | expr '+' expr
        | expr '-' expr
        | expr '*' expr
        | expr '-' expr
        |'-' expr %prec UMINUS
        ;
if_expr:
        'if' ID 'then' ID '=' arith_expr
        |'if' ID 'then' ID '=' arith_expr 'else'
expr
        ;
%%
```

ε κανόνας

Αλλαγή προτεραιότητας κανόνα

Παράδειγμα μέρους κυρίου προγράμματος (5/6)

```
%%

int yyerror (char* yaccProvidedMessage)
{
    fprintf(stderr, "%s: at line %d, before token: %s\n", yaccProvidedMessage, yylineno, yytext);
    fprintf(stderr, "INPUT NOT VALID\n");
}

//*************************************************************************

int main(int argc, char** argv)
{
    if (argc > 1) {
        if (!(yyin = fopen(argv[1], "r"))) {
            fprintf(stderr, "Cannot read file: %s\n", argv[1]);
            return 1;
        }
    }
    else
        yyin = stdin;

    yyparse();
    return 0;
}
```

Την παρέχουμε εμείς στο bison

Παράγεται από το bison

Παράδειγμα Τμήματος Κανόνων (6/6)
αναγνώριση αναθέσεων αντί της if

```
program:    assignments expressions
            | /* empty */
            ;

expression: INTEGER
            | ID
            | expression '+' expression
            | expression '-' expression
            | expression '*' expression
            | expression '/' expression
            | '(' expression ')'
            | '-' expression %prec UMINUS
            ;

expr:       expression '\n'

expressions: expressions expr
            | expr
            ;

assignment: ID '=' expression '\n'
            ;

assignments: assignments assignment
            | /* empty */
            ;
```

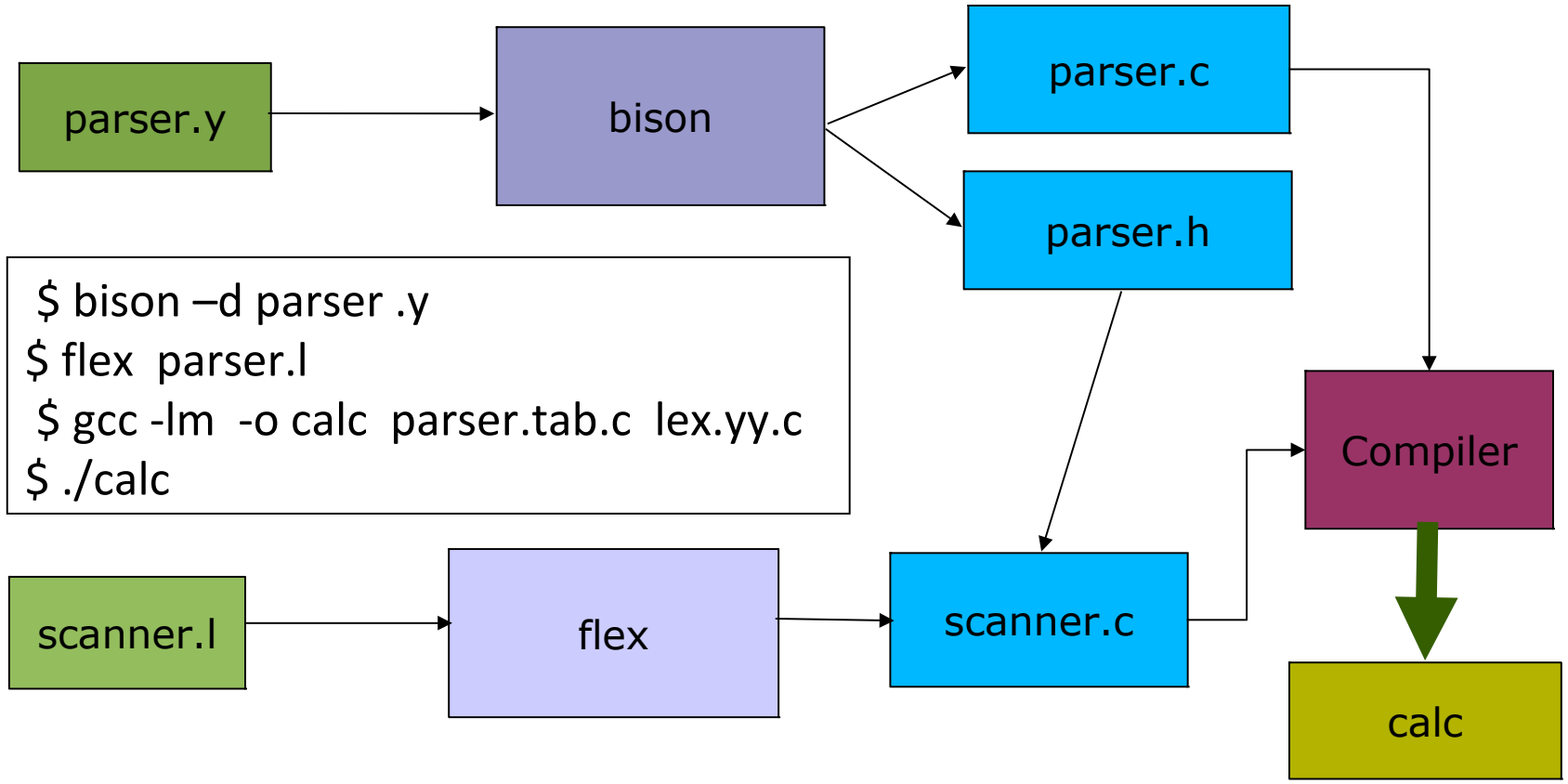
ε κανόνας

Αλλαγή προτεραιότητας κανόνα

Δημιουργία μη κενής λίστας expressions

Δημιουργία λίστας assignments που μπορεί να είναι κενή

Παράδειγμα – Παραγωγή εκτελέσιμου (6/6)



References

- Bison Home Page
  - http://www.gnu.org/software/bison
- Bison Manual
  - http://www.gnu.org/software/bison/manual/html_mono/bison.html
- Bison for Windows
  - http://gnuwin32.sourceforge.net/packages/bison.htm