Κατασκευή front_end Compiler για διανυσματικές γλώσσες

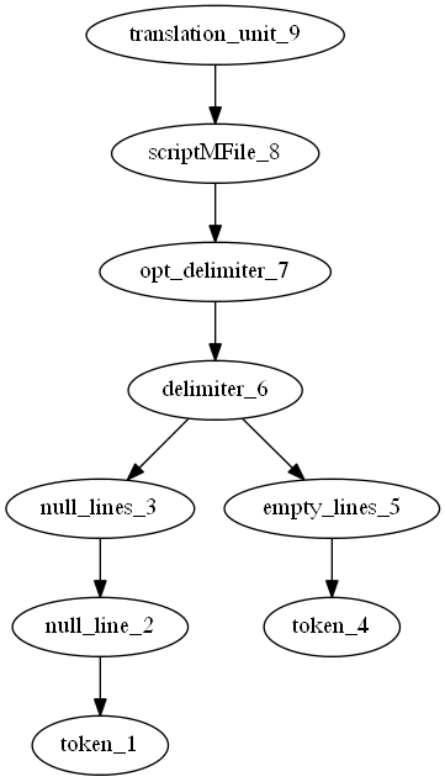
Μεταπτυχιακή Εργασία

Π.Βρουστούρης

2012

Αντικείμενο εργασίας

- Κατασκευή μεταγλωττιστή για M_Files (αρχεία MATLAB), ο οποίος μετατρέπει τον πηγαίο κώδικα που δέχεται στην είσοδο σε μία ενδιάμεση αναπαράσταση (IR) που απεικονίζεται με την μορφή συντακτικού δέντρου.(AST)



Εισαγωγή

Στάδια μεταγλώττισης

- Front_End: Ελέγχει την σύνταξη και τη σημασιολογία του προγράμματος, βρίσκει τα λάθη και δημιουργεί μία ενδιάμεση αναπαράσταση (IR) του πηγαίου κώδικα.
- Middle_End: Παραγωγή βέλτιστης IR (μέσω βελτιστοποίησης του κώδικα) και εξειδίκευσή της με βάση την τελική εφαρμογή.
- Back_End: Μετατροπή της τελικής IR σε γλώσσα μηχανής, κώδικα C, bytecode, ή κάποια άλλη μορφή.

Εισαγωγή (συνέχεια)

Γραμματική χωρίς συμφραζόμενα

- Μπορούμε να αναγνωρίσουμε σωστές συντακτικά προτάσεις → Αναγνωριστές γλώσσας → Πεπερασμένα αυτόματα
- Μπορούμε να παράγουμε σωστές συντακτικά προτάσεις → Παράγωγοι γλώσσας → Κανονικές εκφράσεις
- Μία γραμματική χωρίς συμφραζόμενα είναι παράγωγος προτάσεων που λειτουργεί με κάποιους γραμματικούς κανόνες. (S→aAB)
- Κάποια σύμβολα εμφανίζονται *μόνο* στα *δεξιά* των κανόνων αυτών. Τα σύμβολα αυτά ονομάζονται τερματικά, διότι η παραγωγή μίας συμβολοσειράς που αποτελείται αποκλειστικά από τέτοια σύμβολα, σημαίνει το τέλος της διαδικασίας παραγωγής.

Διαδικασία Scanning

- Front_end → Ανάλυση
- Back_end → Σύνθεση
- Λεκτικός αναλυτής (Scanner). Δέχεται μία ροή χαρακτήρων και παράγει μία ροή από tokens, δηλαδή ακολουθίες χαρακτήρων που μπορούν να θεωρηθούν σαν τερματικά στοιχεία στη χρησιμοποιούμενη γραμματική π,χ IF, AND,";", "+", κτλ. Επίσης σηματοδοτεί τα σχόλια και αγνοεί τα κενά του πηγαίου κώδικα.
- Flex: Software που δέχεται στην είσοδο ένα σύνολο κανονικών εκφράσεων και παράγει έναν λεκτικό αναλυτή (ένα αρχείο c) ο οποίος μπορεί να μετατρέψει ένα σύνολο χαρακτήρων σε ένα σύνολο tokens.

Flex file format

- %{
  <C ορισμοί>
  %}
  <Flex ορισμοί>
  %%
  <Κανονικές εκφράσεις>
  %%
  <Κώδικας Παραγωγής>

Παράδειγμα flex file

```
%{
/* <C declarations> */

# include <stdio.h>
# include <cstdlib>
# include <errno.h>
# include <limits.h>
# include <string>
%}
{%
/* <definitions>*/
# undef yywrap
# define yywrap() 1
#define yyterminate() return token::END
%}

%%
/* Rules */
id              [a-zA-Z][a-zA-Z_0-9]*
int             [0-9]+
blank           [ \t]
%%
/* c code */
set printf( "(STMT set) " );
{NUM}+ printf( "(NUM %s) ", yytext );
{VAR}+ printf( "(VAR %s) ", yytext );
printf( "(OP minus) " );
printf( "(OP plus) " );
printf( "(OP equal) " );
printf( "(END stmt) " );
\n printf( "\n\n" );
<<EOF>> printf( "End of parse\n." );
yyterminate();
```

Διαδικασία Parsing

- Συντακτικός Αναλυτής(Parser): Δέχεται μία σειρά tokens από τον scanner και επαληθεύει ότι αυτή η σειρά αναπαριστά μία «έγκυρη πρόταση» σύμφωνα με τη χρησιμοποιούμενη γραμματική.
- Bison: Software το οποίο παίρνει σαν είσοδο μία ορισμένη γραμματική και παράγει (σε μορφή κώδικα C) έναν καθορισμένο *LR* parser.

Bison file format

- Ορισμοί C και parser
- %%
- Ορισμός τερματικών και μη τερματικών στοιχείων
- %%
- Κανόνες παραγωγής - C υπορουτίνες

Παράδειγμα bison file

```
%{
    #include <stdio.h>
      void yyerror(const char *str )
      {
        fprintf(stderr, "error: %s\n", str );
         }
  int main()
    {
    yyparse();
     return 0
            }
 %}

 %token SET NUMBER VARIABLE OP_MINUS
          OP_PLUS ASSIGN END
%%

statements      :
                | statements statement
                ;
```

```
statement:
SET VARIABLE ASSIGN expression END
{
}
;
expression:  :  NUMBER

                | VARIABLE

                | VARIABLE operator NUMBER

                | VARIABLE operator VARIABLE
                ;

Operator       : OP_MINUS

                | OP_PLUS
                ;
```

Δηλώσεις βασικών κλάσεων

- Διανυσματικές γλώσσες (MATLAB, SVL): Μπορούν να χειριστούν συναθροίσεις από σύνθετους και πολύπλοκους τύπους δεδομένων.
- Όνομα του project: MATFE
- Η κατασκευή του MATFE compiler γίνεται σε αντικειμενοστραφές περιβάλλον. Τα τερματικά και μη τερματικά στοιχεία είναι κόμβοι του αφηρημένου συντακτικού δέντρου που θα κατασκευαστεί. Η γενική κλάση που το περιγράφει είναι η *CASTSyntaxElement*. Τα τερματικά στοιχεία ανήκουν όλα στην κλάση *CTOKEN.* Ορίζουμε επίσης μία κλάση για κάθε μη τερματικό στοιχείο. Όλες αυτές οι κλάσεις είναι υποκλάσεις της *CASTSyntaxElement.*

Δηλώσεις βασικών κλάσεων (συνέχεια)

Στο αρχείο ASTDefines.h ορίζονται:

- enum type TERMINAL_CODE για τα τερματικά στοιχεία.
- enum type NONTERMINAL_CODE για τα μη τερματικά στοιχεία
- enum type NONTERMINAL_PRODUCTION_RULE για τους γραμματικούς κανόνες παραγωγής.

Στο αρχείο ASTSyntaxElements.h ορίζονται:

- Η κλάση CASTSyntaxElement, η κλάση CTOKEN και όλες οι κλάσεις των μη τερματικών στοιχείων.
- Τα πεδία, οι constructors και οι συναρτήσεις που ανήκουν σε αυτές τις κλάσεις.

Δηλώσεις βασικών κλάσεων (συνέχεια)

```
class CASTSyntaxElement {

protected:
MATFE::location *m_location;

public:
CASTSyntaxElement(NONTERMINAL_PRODUCTION_RULE pr, MATFE::location * loc);
~CASTSyntaxElement();
virtual int map_syntax_element(CASTSyntaxElement *, int m=-1)
{return -1;}

TERMINAL_CODE m_TerminalCode;
NONTERMINAL_CODE m_NonTerminalCode;
NONTERMINAL_PRODUCTION_RULE m_NonTerminal_ProductionCode;
static unsigned int m_ObjectSerialNumberCounter;
unsigned int m_NumberOfDescendants;
unsigned int m_ObjectSerialNumber;
vector <CASTSyntaxElement *> m_Descendants;
char* graphstring;
};
```

Δηλώσεις βασικών κλάσεων (συνέχεια)

```
class CTOKEN : public  CASTSyntaxElement{
public:
      CTOKEN(TERMINAL_CODE token, MATFE::location *loc,
         NONTERMINAL_PRODUCTION_RULE pr);
        ~CTOKEN();

protected:

int map_syntax_element(CASTSyntaxElement *, int c=-1);`
};
```

Δηλώσεις βασικών κλάσεων (τέλος)

Παράδειγμα ορισμού κλάσης μη τερματικού στοιχείου

Έστω το non-terminal text_list. Αντίστοιχη κλάση: CTextList

Γραμματικός κανόνας παραγωγής του:

```
text_list      : TEXT
               | text_list TEXT
```

Ορισμός αντίστοιχης κλάσης:

```
class CTextList : public CASTSyntaxElement{
  public:
CTextList (CTOKEN *, MATFE::location *, NONTERMINAL_PRODUCTION_RULE);
CTextList (CTextList *, CTOKEN *, MATFE::location *, NONTERMINAL_PRODUCTION_RULE);
protected:
  virtual int map_syntax_element(CASTSyntaxElement *, int c=-1);
};
```

Μορφή αρχείου MATFE.l

```
%{
 <C++ ορισμοί> π,χ
#include <stdlib.h>
#include <iostream>
#include <string>
#include "ASTDefines.h"
#include "ASTSyntaxElements.h
#include “MATFE.tab.h”
%}
<Flex ορισμοί> π,χ
  #define yyterminate() return token::END
 %%
  <Κανονικές εκφράσεις> π.χ
    HSPACE          [ \t]
    HSPACES         {HSPACE}+
    IDENTIFIER      [a-zA-Z][_a-zA-Z0-9]*
    DIGIT           [0-9]
   INTEGER         {DIGIT}+
%%
```

```
< Κώδικας Παραγωγής> π.χ

{INTEGER}    {
                 BEGIN(QuoteSC);
                 CTOKEN* newtoken;
                  newtoken = new
        CTOKEN(INTEGER_,
                (MATFE::location *)&yylloc,
                 TOKEN_INTEGER);
                 yylval->ast = newtoken;
                return (token::INTEGER);
                 }
```

Μορφή αρχείου MATFE.y

- Ορίζεται στο Bison ένας νέος τύπος “ast” ο οποίος είναι αντικείμενο της CASTSyntaxElement:

```
CASTSyntaxElement * ast;
```

- Κάθε τερματικό και μη τερματικό στοιχείο είναι τύπου <ast>

Μορφή αρχείου MATFE.y (συνέχεια)

```
%%  <Ορισμοί C++ και parser>
π.χ  %define namespace "MATFE«
#define YY_DECL \
          MATFE::MATParserClass::token_type  \
           yylex (MATFE::MATParserClass::semantic_type* yylval, \
           MATFE::MATParserClass::location_type* yylloc, \
             MATFE_driver& driver)
        }
  %union{
    char* text;
          CASTSyntaxElement *ast;
          double imaginary;
             };
%%
<Ορισμοί τερματικών και μη τερματικών στοιχείων> π.χ
%token <ast> FOR
%token <ast> RB
%token <ast> LB
%left <ast> AND OR

%type <ast> translation_unit
%type <ast> scriptMFile
%type <ast> opt_delimiter
%type <ast> f_def_line
```

```
.Κανόνες παραγωγής -  C++ υπορουτίνες> π.χ

f_def_line     : FUNCTION f_output ASSIGN IDENTIFIER f_input
                    {
                     $$ = new CFDefLine((CTOKEN*)$1,
                     (CFOutput*)$2, (CTOKEN*)$3,
                     (CTOKEN*)$4, (CFInput*)$5,
                     (MATFE::location *)&yyloc,

          FDefLine__FUNCTION_FOutput_ASSIGN_IDENTIFIER_FInput);
                cout << "f_def_line stage(1st rule)"<< endl;
                 cout << "--------------------------" << endl;
                  }
                     | FUNCTION IDENTIFIER f_input
                    {
                     $$ = new CFDefLine((CTOKEN*)$1,
                     (CTOKEN*)$2, (CFInput*)$3,
                     (MATFE::location *)&yyloc,
                      FDefLine__FUNCTION_IDENTIFIER_FInput);
                      cout << "f_def_line stage(2nd rule)"<< endl;
                      cout << "--------------------------" << endl;
                     }
```

Υλοποίηση κλάσεων- Κατασκευή του AST

Υλοποίηση CASTSyntaxElement
CastSyntaxElement.h

```
class CASTSyntaxElement {
public:
        CASTSyntaxElement(NONTERMINAL_PROD
        UCTION_RULE pr, MATFE::location * loc);
        ~CASTSyntaxElement();
        virtual int map_syntax_element(CASTSyntaxElement *,int
        m=-1){return -1;}
        TERMINAL_CODE m_TerminalCode;
        NONTERMINAL_CODE m_NonTerminalCode;

        NONTERMINAL_PRODUCTION_RULE
        m_NonTerminal_ProductionCode;
        static unsigned int m_ObjectSerialNumberCounter;
        unsigned int m_NumberOfDescendants;
        unsigned int m_ObjectSerialNumber;
        vector <CASTSyntaxElement *> m_Descendants;


protected:
        MATFE::location *m_location;
};
```

CastSyntaxElement.cpp

```
CASTSyntaxElement::CASTSyntaxElement(NONTERMINAL_P
        RODUCTION_RULE pr, MATFE::location * loc) {

    m_NonTerminal_ProductionCode=pr;
    m_location=loc;
    m_ObjectSerialNumberCounter++;
    m_ObjectSerialNumber=m_ObjectSerialNumberCounter;

}

CASTSyntaxElement::~CASTSyntaxElement() {
  m_ObjectSerialNumberCounter--;
}
```

Υλοποίηση κλάσεων- Κατασκευή του AST (συνέχεια)

Παράδειγμα: Υλοποίηση κλάσης CAssignment

CAssignment.h

```
class CAssignment: public CASTSyntaxElement {
   public:
    CAssignment(CReference *, CTOKEN * , CExpression*,
    MATFE::location *, NONTERMINAL_PRODUCTION_RULE);

     CAssignment(CSassigneeMatrix *, CTOKEN *, CExpression*,
     MATFE::location *, NONTERMINAL_PRODUCTION_RULE);

    CAssignment(CMassigneeMatrix *, CTOKEN *,  CReference *,
    MATFE::location *,  NONTERMINAL_PRODUCTION_RULE);


protected:
 virtual int map_syntax_element(CASTSyntaxElement *, int c=-1);

         };
```

```
CAssignment::CAssignment(CReference *ref, CTOKEN* token,
                         CExpression * exp, MATFE::location *loc,
                         NONTERMINAL_PRODUCTION_RULE pr)
                        :CASTSyntaxElement(pr, loc)
{
  m_NonTerminalCode = assignment_;
  int position = map_syntax_element(ref);
  cout << "Reference pos:" <<position << endl;
  m_Descendants.push_back(ref);
  m_Descendants.push_back(token);
  m_Descendants.push_back(exp);
  m_NumberOfDescendants =3;
}
CAssignment::CAssignment(CSassigneeMatrix *csa,
CTOKEN*   token, CExpression* exp, MATFE::location *loc,
                         NONTERMINAL_PRODUCTION_RULE pr)
:CASTSyntaxElement(pr,loc){
  m_NonTerminalCode = assignment_;
  int position = map_syntax_element(csa);
  cout << "CSassigneeMatrix pos:" <<position << endl;
  m_Descendants.push_back(csa);
  m_Descendants.push_back(token);
  m_Descendants.push_back(exp);
  m_NumberOfDescendants =3; }
```

Υλοποίηση κλάσεων- Κατασκευή του AST (συνέχεια)

```
CAssignment::CAssignment (CMassigneeMatrix *msa, CTOKEN* token,
                                    CReference* ref,MATFE::location *loc,
                                    NONTERMINAL_PRODUCTION_RULE pr)
:CASTSyntaxElement(pr,loc){
  m_NonTerminalCode = assignment_;
  int position = map_syntax_element(msa);
  cout << "CMassigneeMatrix pos" <<position << endl;
  m_Descendants.push_back(msa);
  m_Descendants.push_back(token);
  m_Descendants.push_back(ref);
  m_NumberOfDescendants=3;
};
```

Υλοποίηση κλάσεων- Κατασκευή του AST (συνέχεια)

```
int CAssignment::map_syntax_element(CASTSyntaxElement
*el, int m)
{
   cout << "In CAssignment::map_syntax_element() -- " << endl
        << "NTPC : " << m_NonTerminal_ProductionCode <<
        endl <<
       "el->m_NonTerminalCode :" << el->m_NonTerminalCode
        << endl;
        switch (m_NonTerminal_ProductionCode)
    {
         case Assignment__Reference_ASSIGN_Expr:
        if ( el->m_NonTerminalCode = = reference_ ){
            return 0;}
       else if ( el->m_NonTerminalCode = = token_){
              return 1;}
       else if ( el->m_NonTerminalCode == expr_ ){
              return 2;}
      break;
```

```
case Assignment_SAssigneeMatrix_ASSIGN_Expr:
        if ( el->m_NonTerminalCode = = s_assignee_matrix{
          return 0;}
         else if ( el->m_NonTerminalCode = = token_){
          return 1;}
          else if ( el->m_NonTerminalCode == expr_ ){
        return 2;}
         break;
case Assignment__MAssigneeMatrix_ASSIGN_Reference:
        if ( el->m_NonTerminalCode == m_assignee_matrix_ ){
           return 0;}
        else if ( el->m_NonTerminalCode == token_){
        return 1;}
        else if ( el->m_NonTerminalCode == reference_ ){
        return 2;}
        break;
        default:
    cout << endl << "Error in Assignment::map_syntax_element"
           << endl << "System Exits !!!";
          exit(0);
        }
  return -1;
 }
```

Κατασκευή Interface

File driver.h

```
#ifndef DRIVER_H
#define DRIVER_H
#include <string>
#include "MATFE.tab.h"
#include "ASTDefines.h"
#include "ASTSyntaxElements.h"
#include <iostream>
#include <fstream>
extern FILE* yyin;
YY_DECL;
```

```
class MATFE_driver
{
  public:
        MATFE_driver();
        virtual ~MATFE_driver();
        int result;
        std::string file;
        bool debug_mode;
        bool trace_scanning;
       bool trace_parsing;
        // Handling the scanner.
        void scan_begin();
        void scan_end();
        // Run the parser. Return 0 on success.
        int parse (const std::string& f);
        // Error handling.
        void error (const MATFE::location& l, const std::string&
        m);
        void error (const std::string& m);
       // debug
        void printInfo(const char* msg);
        void set_debug_mode(bool flag);
        string to_string();
       };
       #endif // DRIVER_H
```

Κατασκευή Interface (συνέχεια)

File driver.cpp

```
#include "driver.h"
#include <cstdio>
#include <iostream>
#include <fstream>
using namespace std;

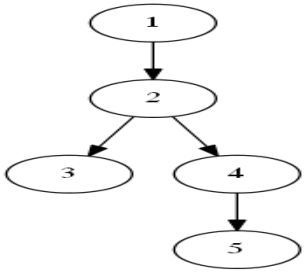
MATFE_driver::MATFE_driver()
: trace_scanning(false), trace_parsing(false){
        }
MATFE_driver::~MATFE_driver () {
        this->debug_mode = false;
         }
int MATFE_driver::parse (const std::string &f)
{
        file=f;
       scan_begin();
        printInfo("Creating parser object...");
        MATFE::MATParserClass parser(*this);
        parser.set_debug_level(trace_parsing);
        printInfo("Created.");
        printInfo("Initiating parsing...");
        int res = parser.parse();
        scan_end();
        return res; }
```

```
void MATFE_driver::error (const MATFE::location& l, const
        std::string& m){
        std::cerr <<"Driver error: "<< l << ": " << m << std::endl;
}
void MATFE_driver::scan_begin (){
        yyin = fopen(file.c_str(),"r");
}
void MATFE_driver::scan_end (){
   fclose(yyin);
}
void MATFE_driver::error (const std::string& m){
        std::cerr << m << std::endl;
}
void MATFE_driver::printInfo(const char* msg) {
   if (! this->debug_mode) return;
       cout << msg << endl;
}
void MATFE_driver::set_debug_mode(bool flag) {
   this->debug_mode = flag;
}
string MATFE_driver::to_string() {
   string s("");
   s = s + file;
        return s;
}
```

Οπτικοποίηση IR

- Χρήση Graph Visualization Software (Graphviz) – αρχείων DOT.
- Παράδειγμα αρχείου εισόδου του Graphviz

  digraph G {1 ->2; 2->3; 2 ->4; 4 ->5;}



Οπτικοποίηση IR (συνέχεια)

Παράδειγμα:

Έστω το αποτέλεσμα μεταγλώττισης ενός στοιχειώδους αρχείου MATLAB:

```
null_line          |  token
null_lines         |  null_line
delimiter          |  null_lines
empty_lines        |  token
delimiter          | empty_lines
opt_delimiter      | delimiter
scriptMFile        | opt_delimiter
translation_unit   | scriptMFile
```

Αντίστοιχη είσοδος στο Graphviz:

```
digraph G {
null_line_2 -> token_1;
null_lines_3 -> null_line_2;
delimiter_6 -> null_lines_3;
empty_lines_5 -> token_4;
delimiter_6 -> empty_lines_5;
opt_delimiter_7 -> delimiter_6;
scriptMFile_8 -> opt_delimiter_7;
translation_unit_9 -> scriptMFile_8;
}
```

Οπτικοποίηση IR (συνέχεια)

Στα πεδία της CASTSyntaxElement* προσθέτουμε την

char* graphstring;

όπου καταχωρούμε το τερματικό στοιχείο που αντιστοιχεί στην NONTERMINAL_CODE m_NonTerminalCode;

Η αντιστοίχιση γίνεται μέσω ενός πίνακα char * nonterminals του αρχείου ASTDefines.cpp. ως εξής:

this→graphstring = nonterminals[this→m_NonTerminalCode];

(κώδικας που περιέχεται στους constructors κάθε μη τερματικού στοιχείου).

Οπτικοποίηση IR (συνέχεια)

Η αναδρομική συνάρτηση GraphEmmiter μετατρέπει την ενδιάμεση αναπαράσταση σε αρχεία εσόδου Graphviz. Την εισάγουμε στον MATFE_driver σε δύο μορφές (με ένα και δύο ορίσματα).

```
void MATFE_driver::ASTGraphEmmiter(CASTSyntaxElement *parent, ofstream* out)
{
   for (unsigned int i=0; i < (parent->m_Descendants.size()); i++)
    {
      (*out)  << parent→graphstring << parent→m_ObjectSerialNumber    << ' ';
      (*out) << "→ ";
     (*out)  << parent→m_Descendants[i]→graphstring  <<
       parent→m_Descendants[i]→m_ObjectSerialNumber<<";" << endl;
       ASTGraphEmmiter(parent->m_Descendants[i], out);
   }
}
```

Οπτικοποίηση IR (συνέχεια)

- ```
  void MATFE_driver::ASTGraphEmmiter(CASTSyntaxElement *parent)
  {
  ofstream graphout;
  graphout.open("mygraph.grv");
  graphout << "digraph G {" << endl;
  ASTGraphEmmiter(parent, &graphout);
  graphout << "}" << endl;
  graphout.close();
  }
  ```

- Με την κλήση της ASTGraphEmmiter στους constructors της CTransUnit (κλάση που αντιστοιχεί στο non-terminal translation_unit) και στο σώμα του αντικειμένου parse του MATFE driver, το αρχείο εισόδου του graphviz σχηματίζεται αναδρομικά.

- Η κλήση αυτή γίνεται ως εξής:
  ```

Οπτικοποίηση IR (συνέχεια)

- Ορίζω, στην κλάση CTransUnit, μία νέα μεταβλητή instance τύπου CTransUnit

```
static CTransUnit* instance;
```

- Στην μεταβλητή αυτή καταχωρώ την graphstring σε κάθε constructor της CTransUnit.

```
this->graphstring = nonterminals[this->m_NonTerminalCode];
CTransUnit::instance = this;
```

- Τέλος στο αντικείμενο parse του driver καλώ την ASTGraphEmmiter στην μεταβλητή instance

```
ASTGraphEmmiter(CTransUnit::instance);
```

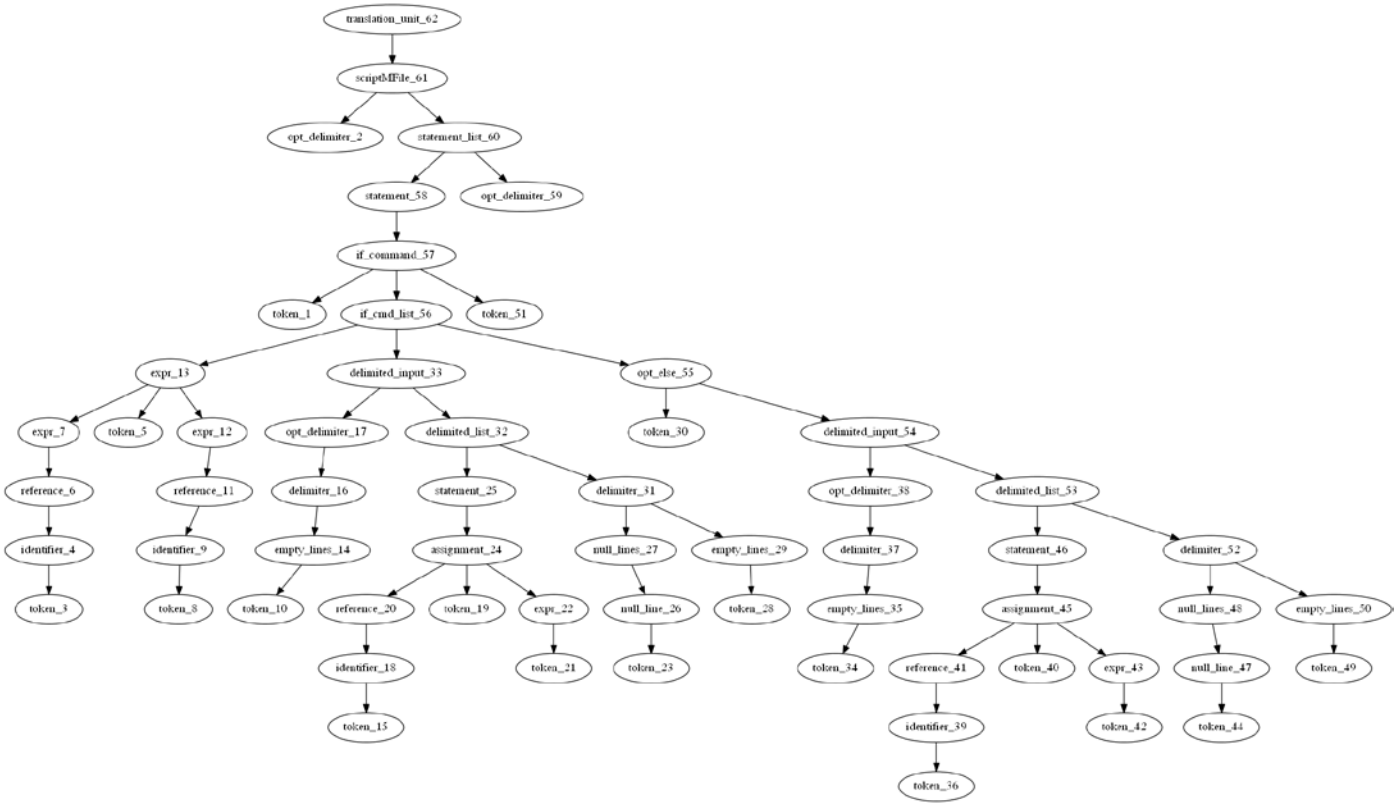
Οπτικοποίηση IR (συνέχεια)

Παράδειγμα: Μεταγλώττιση του ακόλουθου τμήμα MATLAB κώδικα:

```
if r == c
x=3;
else
x=5;
end
```

Οπτικοποίηση IR (τέλος)

Παραγόμενο AST



Η συνάρτηση main

```
#include <iostream>
#include "driver.h"
#include "MATFE.tab.h“
using namespace std;

int main (int argc, char *argv[])
{
    MATFE_driver driver;
    driver.set_debug_mode(false);
    char** argv_init = argv;
    cout << "Start scanning..." << endl;
    for (int i = 0; i < argc; i++)
            argv = argv_init+i;
        if (*argv == std::string ("-p")) {
            driver.trace_parsing = false;
        }
        else if (*argv == std::string ("-s")) {
            driver.trace_scanning = false;
        }
else {
            string filename = string(*argv);
          cout << "Scanning file: "<< filename <<   endl;
            if (!driver.parse(filename))
           {
             std::cout << "OK. End of file" <<   std::endl;
            }

       }
    cout << "It was a nice parsing!" << endl;
    return 0;
}
```

Επίλογος

Προτάσεις για κατασκευή back_end

- Μετατροπή της παραγόμενης IR σε system level C για την κατασκευή ενός kernel.
- Ανάπτυξη ενός hardware μοντέλου (μέσω VHDL ή Verilog) για τη δημιουργία ενός ενσωματωμένου συστήματος που μπορεί να εφαρμόσει μέρος της λογικής του MATLAB.
- Μετατροπή της IR σε Native Assembly για την κατασκευή μιας αυτόνομης εφαρμογής ή ενός driver.