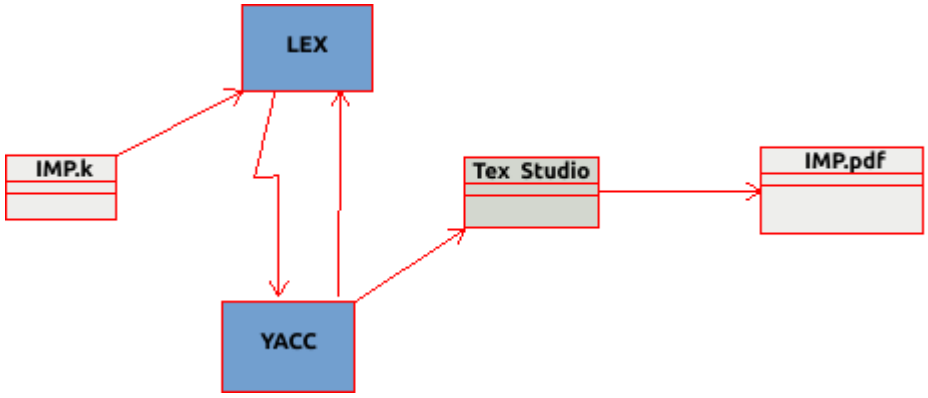
Tools K --> LaTeX

Introducere

Proiectul meu constă in folosirea uneltelor LEX si YACC pentru a putea extrage dintr-un fișier ce conține cod K compilabil comentariile si pentru a crea cateva diagrame pentru reguli si pentru programul in sine in acest mod codul scris va fi mult mai usor de vizualizat.

Odată aplicate regulile din LEX si YACC,va rezulta un fișier cod.tex ce va conține cod scris in LateX ce va putea fi compilat la sfarsit intr-un fișier pdf.



FISIERE :

LAB.k ; prj.l ; prj.y ; fcn.h finish.tex; finish.pdf ;

COMPLILARE :

lex -ll prj.l

yacc -d prj.y

gcc lex.yy.c y.tab.c -lfl -ly -o P

./P LAB.k

daca rezultatul este “Programul a reusit . . . NOOTHINX >.<” atunci s-a creat fisierul finish.tex . Acesta trebuie deschis cu un program de editate text in LaTeX si apoi direct salvat ca fisier pdf --> finish.pdf

Exemplu de cod K:

/* In sintaxă se vor găsit toate elementele din cod ce pot fi citite din programul nostru ca si variabile, operații matematice, operații booleene, blocuri de instrucțiuni etc. Toate acestea vor fi definite la inceputul codului scris si se pot adăuga elemente de acest gen pe parcursul implementării. */

```
module IMP


  syntax AExp ::= Int // builtin
                | Id  // builtin
                > left:
                AExp "+" AExp [left, strict]
                | AExp "-" AExp [left]


endmodule
```

/* In secțiunea de semantică se vor folosi elementele din sintaxă si se vor defini pentru a putea fi efectuate în întregime de către program. */

```
module IMP
imports IMP-SYNTAX
  syntax KResult ::= Bool | Int | String
  configuration
          <T>
              <k> $PGM:Stmt </k>
              <env> .Map </env>
              <store> .Map </store>
              <stack> .List </stack>
```

```
        </T>
  rule I1:Int + I2:Int => I1 +Int I2
  rule <k> int X:Id ; => . ...</k>
       <env> M:Map => M[X <- !L:Int] </env>
       <store> M':Map (.Map => !L |-> 0) </store>

endmodule
```

Exemplu de cod Latex:

```
\documentclass[a4paper,9pt]{article}
\begin{document}
\title{IMP}
\maketitle

\author{\textit{S\^{a}rbu Iulia Iustina, grupa B1}} \\ \\
```

In sintaxă se vor gasit toate elementele din cod ce pot fi citite din programul nostru ca si variabile, operații matematice, operații booleene, blocuri de instrucțiuni etc. Toate acestea vor fi definite la inceputul codului scris si se pot adăuga elemente de acest gen pe parcursul implementării.

```
$module IMP

  syntax AExp ::= Int // builtin
                        | Id  // builtin
                > left:
                AExp "+" AExp [left, strict]
```

| AExp "-" AExp [left]

endmodule$

In secțiunea de semantică se vor folosi elementele din sintaxă si se vor defini pentru a putea fi efectuate în întregime de către program.

$module IMP

imports IMP-SYNTAX

syntax KResult ::= Bool | Int | String

configuration

<T>

<k> $PGM:Stmt </k>

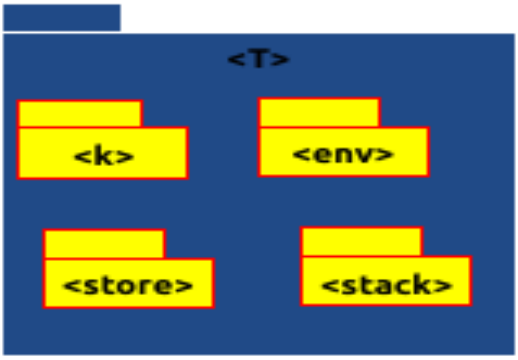
<env> .Map </env>

<store> .Map </store>

<stack> .List </stack>

</T>$

\includegraphics[width=0.5\textwidth]
{program}



//PROGRAM IMAGE EXEMPLE ---->

$rule I1:Int + I2:Int => I1 +Int I2

rule <k> int X:Id ; => . ...</k>

<env> M:Map => M[X <- !L:Int] </env>

<store> M':Map (.Map => !L |-> 0) </store>$

\includegraphics[width=0.5\textwidth]{k}

\includegraphics[width=0.5\textwidth]{store}

\includegraphics[width=0.5\textwidth]{env} $endmodule$

\end{document}