Université libre de Bruxelles

Project - Part 1
Lexical Analyser

Aldar Saranov, Przemyslaw Gasinski

Aldar.Saranov@ulb.ac.be
Przemyslaw.Gasinski@ulb.ac.be

INFO-F403 Introduction to language theory and compiling
(M-INFOS/F277)

Gilles Geeraerts

October 2016

Regular Expressions

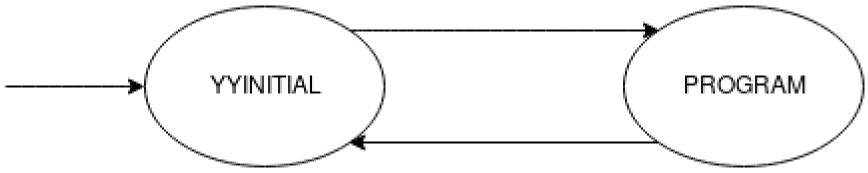
Eventually the following REs were implemented.

```
single_endline = \r|\n|\r\n
space = " "+
endline = ( {single_endline}+ ({space}+ {single_endline})* )+
comment = {endline} ("c"|"C"|"*"|"d"|"D"|"!")(.*)
varname = ([a-zA-Z])([a-zA-Z]|[0-9])*
number = [0-9]+
```

Not to mention the state REs and transition rules.

States

YYINITIAL : The initial state that contains every RE needed.
PROGRAM : State used to recognise the name of the program and separate it from other variable names.



Encountered problems

Endline merging

It makes sense to merge adjacent “endlines” into one in order to remove token redundancy. Initial proposition was to use the following:

```
single_endline = \r|\n|\r\n
space = " "+
endline = ({single_endline}+({space}+|{single_endline})+)+
```

The problem of such REs was that it “eated” ending spaces.

```
[space] c this is a comment
```

Such string could not be recognised as comment because its prefix space was included into preceding merging “endlines”. Considering the fact that several spaces could be placed between the “endlines” we had to implement such REs for recognition:

```
single_endline = \r|\n|\r\n
space = " "+
endline = ({single_endline}+({space}+{single_endline})*)+
```

These REs assure the “endline” merging while keeping spaces before comments.

ProgName recognition

Due to the fact that we had to store the list of declared variables, we needed to exclude the name of the program out of it. In order to achieve this, a separate state was introduced to differentiate the string matched by the `varname` RE. Upon reading the `PROGRAM` keyword, we switch to the `PROGRAM` state which has a different RE for `varname`. Although we consider the program name as a variable in our lexical analyser, we do not add it to the list of known variables while in this particular state. Therefore, instead of calling :

```
preprocessor.newVariable(token(LexicalUnit.VARNAME, yyline,
yycolumn, yytext()));
```

Where `token` returns the newly created `Symbol` and passes it onto `newVariable`, we omit the `newVariable` method and simply call :

```
token(LexicalUnit.VARNAME, yyline, yycolumn, yytext());
```