

SUPRALGOL²⁰¹⁶

Introduction to language theory and compiling

Project – Part 2

Gilles GEERAERTS

Ismael JECKER

Thi-Van-Anh NGUYEN

November 10, 2015

No proper program contains an indication which as an operator-applied occurrence identifies an operator-defining occurrence which as an indication-applied occurrence identifies an indication-defining occurrence different from the one identified by the given indication as an indication-applied occurrence.

ALGOL 68 Report

For this second part of the project, you will write the *parser* of your SUPRALGOL²⁰¹⁶ compiler. More precisely, you must:

1. Transform the SUPRALGOL²⁰¹⁶ grammar (see Figure 1) in order to: (a) Remove unreachable and/or unproductive variables, if any; (b) Remove left-recursion and apply factorisation where need be; (c) Make the grammar non-ambiguous by taking into account the priority and the associativity of the operators. Table 1 shows these priorities and associativities: operators are sorted by decreasing order of priority (with two operators in the same row having the same priority).
2. Give the *action table* of an LL(1) parser for the transformed grammar. You must justify this table by giving the details of the computations of the relevant First and Follow sets.
3. Write, in Java, a recursive descent LL(1) parser for this grammar. Your parser should use the scanner that you have designed in the first part of the project in order to extract the sequence of tokens from the input. For this part of the project, the output of your parser must consist only of *the leftmost derivation* of the input string, if it is correct; or an error message if there is a syntax error. The leftmost derivation can be given as a sequence of rule numbers (do not forget to number your rules accordingly in the report!)

You must hand in:

- A PDF report containing the modified grammar, the action table, with all the necessary justifications, choices and hypothesis
- The source code of your parser;
- The SUPRALGOL²⁰¹⁶ example files you have used to test your analyser;
- All required files to evaluate your work (like a `Main.java` file calling the parser, etc).

You must structure your files in four folders:

- `doc` contains the JAVADOC and the PDF report;

[1]	<Program>	→ begin <Code> end
[2]	<Code>	→ ϵ
[3]		→ <InstList>
[4]	<InstList>	→ <Instruction>
[5]		→ <Instruction> ; <InstList>
[6]	<Instruction>	→ <Assign>
[7]		→ <If>
[8]		→ <While>
[9]		→ <For>
[10]		→ <Print>
[11]		→ <Read>
[12]	<Assign>	→ [VarName] := <ExprArith>
[13]	<ExprArith>	→ [VarName]
[14]		→ [Number]
[15]		→ (<ExprArith>)
[16]		→ - <ExprArith>
[17]		→ <ExprArith> <Op> <ExprArith>
[18]	<Op>	→ +
[19]		→ -
[20]		→ *
[21]		→ /
[22]	<If>	→ if <Cond> then <Code> fi
[23]		→ if <Cond> then <Code> else <Code> fi
[24]	<Cond>	→ <Cond> <BinOp> <Cond>
[25]		→ not <SimpleCond>
[26]		→ <SimpleCond>
[27]	<SimpleCond>	→ <ExprArith> <Comp> <ExprArith>
[28]	<BinOp>	→ and
[29]		→ or
[30]	<Comp>	→ =
[31]		→ >=
[32]		→ >
[33]		→ <=
[34]		→ <
[35]		→ /=
[36]	<While>	→ while <Cond> do <Code> od
[37]	<For>	→ for [VarName] from <ExprArith> by <ExprArith> to <ExprArith> do <Code> od
[38]	<Print>	→ print([VarName])
[39]	<Read>	→ read([VarName])

Figure 1: The SUPRALGOL²⁰¹⁶ grammar.

Operators	Associativity
-, not	right
*, /	left
+, -	left
>, <, >=, <=, =, /=	left
and	left
or	left

Table 1: Priority and associativity of the SUPRALGOL²⁰¹⁶ operators (operators are sorted in decreasing order of priority).

- test contains all your example files;
- dist contains an executable JAR **that must be called part2.jar**;
- more contains all other files.

Your implementation must contain:

1. your scanner (from the first part of the project);
2. your parser;
3. an executable public class Main that reads the file given as argument and writes on the standard output stream the leftmost derivation.

The command for running your executable must be:

```
java -jar part2.jar inputFile
```

You will compress your folder (in the *zip* format—no *rar* or other format), **which is called:**

```
Part2_Surname1_Surname2.zip
```

where Surname1 and Surname2 are the (family) names of the students of the group (in alphabetical order), and you will submit it on the Université Virtuelle before **November, 27th**. A printed copy of your report only will be handed in to the Secrétariat étudiants (Mrs. Maryka Peetroons, NO8.104) for the same date. You are allowed to work in group of maximum two students.