SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Reinhard Wilhelm
Jun.-Prof. Dr. Sebastian Hack
Jörg Herter, M.Sc.
Dipl.-Inform. Christoph Mallon

Compiler Construction WS09/10

Exercise Sheet 4

Please hand in the solutions to the theoretical exercises until the beginning of the lecture next Wednesday 2009-11-18, 10:00. Please write the number of your tutorial group or the name of your tutor on the first sheet of your solution. Solutions submitted later will not be accepted.

Exercise 4.1: Viable Prefixes (Points: 2)

The grammar G is given by the productions

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \mid B \\ B &\rightarrow bB \mid c \end{aligned}$$

Which of the following strings are viable prefixes of a right sentinal form (RSF) of G?

- aAbB
- AbbB

Exercise 4.2: LR(0) (Points: 6+4+2)

Let the grammar $G = (\{S', S, A, B\}, \{if, do, otherwise, true, false, ;, id, funcall, (,)\}, P, S')$ with productions P:

$$\begin{aligned} S' &\rightarrow S \\ S &\rightarrow if\ B\ do\ S \mid if\ B\ do\ S\ otherwise\ S \mid A; \\ A &\rightarrow funcall\ id\ (A) \mid funcall\ id \\ B &\rightarrow true \mid false \end{aligned}$$

1. Construct the LR-DFA(G) with the algorithm from the lecture.
2. Mark all inadequate states in the LR-DFA(G).
3. Give a successful run of the LR-DFA(G) on the input word

$$w = if\ false\ do\ funcall\ id(funcall\ id);\ otherwise\ funcall\ id;$$

Exercise 4.3: LL(0) and LR(0) (Points: 2+2+2)

Prove or disprove the following claims:

1. All LL(0) languages are also LR(0) languages.
2. All regular languages are LR(0).
3. Not all LR(0) languages are regular.

Exercise 4.4: LR vs LALR vs SLR (Points: 5)

Order the three classes of grammars regarding their expressive power and discuss the differences of the automata.

Exercise 4.5: Pretty Printer (Project)

In the third phase of your compiler project, you are to add a pretty printer to your parser.
Your parser shall now construct an abstract syntax tree (AST) as an internal representation of the input program. It shall also be able to generate source code from the AST again. The following example illustrates how the format looks like.

```
class HelloWorld {
    public int bar(int a, int b) {
        return c = (a + b);
    }
    public static void main(String[] args) {
        (System.out).println(43110 + 0);
        boolean b = true && (!false);
        if ((23 + 19) == ((42 + 0) * 1))
            b = (0 < 1);
        else if (!true) {
            int x = 0;
            x = (x + 1);
        } else {
            (new HelloWorld()).bar(0, -1);
        }
    }
    public int c;
}
```

Use one tabulator per indentation level and fully parenthesized expresssions, i.e. every subexpression shall be parenthesized, except for literals (0, false, ...), identifiers, and the outermost subexpression! Start a new line for inner statements and increase the indentation, except for blocks as well as if-statements in the else-part of if-statements. The use of whitespace in expressions can be extrapolated from the example.
Output classes, methods, and fields alphabetically sorted. All methods shall be printed before any field is printed. Refer to the example above and the reference file we will provide you with.
Additionally, your parser shall also provide an option that pretty prints the input file using minimal parenthesation.

After this third phase, your compiler can already tokenize, parse input programs, and pretty print them. To enable a segregated test/execution of those three subtasks, implement an additional switch –ast to invoke the default pretty printer and –astmin to invoke the pretty printer using minimal parenthesation. The other switches shall of course stay intact.

Send your solution to the practical exercise to jherter@cs.uni-sb.de until 2009-11-23, 10:00. Please send just one e-mail per project group.