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

Compiler Construction WS09/10

Exercise Sheet 2

Please hand in the solutions to the theoretical exercises until the beginning of the lecture next Wednesday 2009-11-04, 10:00. Please write the number of your tutorial group or the name of your tutor on the first sheet of your solution. Send your solution to the practical exercise to jherter@cs.uni-sb.de until 2009-11-11, 10:00. Please send just one e-mail per project group. Solutions submitted later will not be accepted.

Exercise 2.1: Finite Automata Revisited (Points: 1+3+3+3)

Let $G = (\{S\}, \{(,)\}, P, S)$ with P:

$$S \rightarrow SS \mid (S) \mid \epsilon$$

1. What language $L(G)$ does G generate?
2. Is there a DFA that accepts $L(G)$?
   (Either give such an automaton or prove that no such automaton exists.)
3. The pumping lemma for regular languages describes an essential property of regular languages and thus can be used to show that a language is not regular. However, the pumping lemma does not give a *sufficient* condition for a language to be regular. Show this by giving a language L' that satisfies the conditions of the pumping lemma for regular languages but is not regular.
4. Reconsidering the previous question, can you give a general construction scheme for non-regular languages satisfying the conditions of the pumping lemma for regular languages?

Exercise 2.2: Push-Down Automata (Points: 6)

Let the context-free grammar $(\{S, L, B, C\}, \{a, b, c, d, e, f\}, P, S)$ with productions P

$$\begin{array}{lcl} S & \rightarrow & aBSS \mid bBS \mid cSB \mid dLe \mid f \\ L & \rightarrow & SaL \mid \epsilon \\ B & \rightarrow & aC \mid bC \\ C & \rightarrow & eB \mid \epsilon \end{array}$$

Give a succeeding run of the push-down automaton constructed from this grammar (using the algorithms presented in the lecture) on the input word $dbaabffae$.

Exercise 2.3: Reduced Context Free Grammars (Points: 6+3)

Let (V_N, V_T, P, S) be a context-free grammar. A non-terminal $A \in V_N$ is

- reachable: $\exists \phi_1, \phi_2$ sucht that $S \Rightarrow^* \phi_1 A \phi_2$
- productive: $\exists w \in V_T^*$ sucht that $A \Rightarrow^* w$

A grammar is *reduced* if it has neither unreachable nor unproductive non-terminals.

1. How can we efficiently determine productivity and reachability of non-terminals?
2. How can we transform a grammar into reduced form?

Exercise 2.4: Parser (Project)

In the second phase of your compiler project, you are to implement a parser.
Your parser shall 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. For printing this newly generated source code adhere to the Java formatting conventions. However, do ignore the line size limits of these conventions.

The following example illustrates how the format looks like.

```
class HelloWorld {
	public int c;
	public static void main(String[] args) {
		System.out.println(43110);
		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 bar(int a, int b) {
		return c = a + b;
	}
}
```

Use one tabulator per indentation level and the minimal number of parentheses within expresssions!

Your parser must accept exactly the MiniJava language as defined in the specification available from our web page. Ambiguity in the language can be resolved as you deem fit. However, you should make sure your parser does not crash upon invalid source code but stops with some kind of error message.

After this second phase, your compiler can already tokenize and parse input programs. To enable a segregated test/execution of those two subtasks, implement two switches such that the parameter `--tokens` invokes just the lexer and generates as output the token stream of the input program, while `--ast` invokes the parser.

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