Technische Universität München *Summer Semester 08*
Fakultät für Informatik *Vesal Vojdani*
Prof. Dr. H. Seidl *vojdanig@in.tum.de*

Compiler Construction

Exercise Sheet 6

Deadline: 4. June 2008, at the lecture, in room 02.07.053, or by e-mail.

Exercise 1: *Push-Down Automata* *6 Points*

Let $\Sigma = \{a, b\}$. Give the context-free grammar for the following languages and the corresponding non-deterministic push-down automata for one of them:

a) $\{u \mid u \in \Sigma^*;\ \#_a u = \#_b u\}$;
b) $\{u \mid u \in \Sigma^*;\ \#_a u \neq \#_b u\}$;
c) $\{uv \mid u, v \in \Sigma^*;\ |u| = |v|;\ u \neq v\}$,

where $\#_x w$ is the number of times the symbol x occurs in w, and $|w|$ is the length of the word w.

Exercise 2: *Extended context-free grammar* *5 Points*

Sometimes the notation of context-free grammars are extended with constructs that provide some of the convenient notation of regular expressions. We introduce the use of square and curly brackets in production bodies, e.g. $A \to X\ [Y]\ Z$ and $A \to X\ \{Y\}\ Z$, respectively. The square brackets mean that the content is optional ($Y?$), while the curly brackets says that the content can be repeated zero or more times (Y^*). Show that any language than can be generated by this a grammar using these extensions can also be generated with a context free grammar without using the extensions.

Exercise 3: *Finite automata cannot count* *5 Points*

The above exercise indicates that any regular language can be expressed by a context-free grammar. A more direct proof would be based on translating the transitions of the DFA to productions. A more interesting fact is that context-free languages are more powerful than regular ones, where the classic example is $\{a^n b^n | n \in \mathbb{N}\}$. Prove this by making the contradictory assumption that you have a finite automata with k states which recognizes this language. Think about what must happen after reading more than k occurrences of the character a in some input string.

(Prove this yourself, or if you run into trouble, look it up in a book or recall it from some other lecture. I still want to you to prove it!)

Exercise 4: *The CYK algorithm* *10 Points*

The goal of this exercise is to get close to finding the Cocke-Younger-Kasami algorithm for any context-free language, which works best on grammars in *Chomsky Normal Form*. This is a long exercise, focus mainly on working out the examples; general algorithms are appreciated, but not necessary, the grading will be gentle. :)

A grammar is in Chomsky Normal Form (CNF), when each of its productions is in one of the following forms:

$$A \to t \qquad \text{for terminal symbol } t \tag{1}$$

$$A \to BC \qquad \text{for non-terminals } B \text{ and } C \tag{2}$$

Assume you are given a grammar in CNF, then the algorithm to recognize the string $a_1 a_2 \cdots a_n$ works using dynamic programming by filling in a $n \times n$ matrix T such that T_{ij} is the set of non-terminals that derive the substring $a_i a_{i+1} \cdots a_j$. Most of our work is therefore concerned with bringing context-free grammars to CNF.

a) Consider the following grammar, which is in CNF:

$$S \to SA \qquad S \to AB \qquad A \to BA \qquad A \to a \qquad B \to b$$

Create the table to recognize the word *babba* starting from the diagonal, which corresponds to application of the productions of type (1), and working towards getting S into the corner T_{1n} using the productions of type (2) by combining all possible splits of the string into two-substrings. Note that there are two parses of this string, which you should detect.

b) Show how to transform a given grammar to a grammar in CNF, which generates the same language, except the empty string ε, which CNF grammars can't generate.) You can try right away to solve this, or use the following steps to eliminate problematic productions step-by-step.

c) Give a method to eliminate all ε-productions ($A \to \varepsilon$) from a given grammar. Again, we aim for an equivalent grammar, except the ability to generate the empty string. It helps to first find the set of all non-terminals A such that $A \to^* \varepsilon$. Try your algorithm on

$$S \to aSB \mid B \qquad\qquad B \to bS \mid \varepsilon$$

d) Having eliminated ε-productions, now get rid of single productions ($A \to B$). Here, it helps to keep track of a set of pairs of non-terminals such that $A \to^* B$ by a sequence of single production. Try this on

$$S \to S + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (S) \mid t$$

e) When you know how to get rid off ε- and single productions, you only need a few more tweaks to get any grammar to CNF. Figure these out and then bring the grammar from d) to CNF using your method.