**MA 3219 – Computability Theory**

**Frank Stephan.** Departments of Computer Science and Mathematics, National University of Singapore, 3 Science Drive 2, Singapore 117543, Republic of Singapore

Email fstephan@comp.nus.edu.sg
Webpage `http://www.comp.nus.edu.sg/~fstephan/computability.html`
Telephone office +65-6874-7354
Office room number S14#06-06
Office hours Thursday 15.00-17.00h

**Assignment for 16.02.2005.** Can be corrected on request, it is not obligatory to hand the homework in.

**1. Analyzing a Formal Grammar.** Consider the following grammar:

- Start symbol $S$;
- Terminal alphabet $\{0,1,2,3,4,5,6,7,8,9\}$;
- Nonterminal alphabet $\{S,A,B,C\}$;
- Rules: $S \to SABC$, $S \to 1$, $AB \to BA$, $BA \to AB$, $AC \to CA$, $CA \to AC$, $BC \to CB$, $CB \to BC$, $A \to 2$, $B \to 3$, $C \to 4$.

Apply the rules $S \to SABC$, $S \to SABC$, $S \to 1$, $CA \to AC$, $CB \to BC$, $BA \to AB$, $A \to 2$, $B \to 3$, $C \to 4$. $A \to 2$, $B \to 3$, $C \to 4$ in this order. What is the result of that derivation?

Which of the following numbers are in this set: 1, 10, 15, 1234, 1432, 1223344, 1342342, 1444433, 1444433332222, 14444333322220? Characterize the set of decimal numbers generated by the grammar.

**2. Designing a formal grammar.** Write grammars $G_1$ and $G_2$ which generate all binary numbers $(0,1,10,11,100,101,110,111,1000,\ldots)$ which are multiple of 3 and 5, respectively. So $G_1$ generates the set $\{0,11,110,1001,1100,1111,\ldots\}$ and $G_2$ generates $\{0,101,1010,1111,\ldots\}$.

**3. Recursively enumerable sets.** Which two of the following statements are equivalent to $A$ being a recursively enumerable set of words over $\Sigma$?
(a) There is a grammar which generates $A$.
(b) Every Turing machine halts on an input $x$ iff $x \in A$.
(c) There is a Turing machine which halts on input $x$ if and only if $x \in A$.
(d) There is a computable function from words to $\{0,1\}$ which outputs 0 if $x \notin A$ and 1 if $x \in A$.

**4. Computable functions.** Let $1^n$ denote a word consisting of $n$ ones, so $1^2 = 11$ and $1^3 = 111$. Which 3 of the following statements is equivalent to saying that $f$ is a computable function from the natural numbers to the natural numbers?
(a) There is a Turing machine which halts on input $1^n$ with output $1^{f(n)}$ if $f(n)$ is defined and does not halt otherwise.
(b) The set $\{1^n01^m : f(n)\downarrow \wedge m < f(n)\}$ is recursively enumerable.

(c) The set $\{1^n01^m : f(n) \downarrow \wedge m = f(n)\}$ is recursively enumerable.
(d) The set $\{1^m01^n : f(n) \downarrow \wedge m < f(n)\}$ is recursively enumerable.
(e) The set $\{1^m01^n : f(n) \downarrow \wedge m = f(n)\}$ is recursively enumerable.

**5. Turing machines.** Construct a Turing machine which works on the alphabet consisting of $0, 1, 2$ plus the blanc and which does the following: It changes in an input word $w$ every 0 to 1 and every 1 to 0. Every 2 remains unchanged.

**Some Information on formal grammars.** This information is based on

`http://en.wikipedia.org/wiki/Formal_grammar`

but contains only those parts which are relevant to this lecture.

In computer science a formal grammar is an abstract structure that describes a formal language precisely, that is, a set of rules that mathematically delineates a (usually infinite) set of finite-length strings (= words) over a finite alphabet. Formal grammars are so named by analogy to grammar in human languages.

A grammar consists of a set of rules by which all possible strings in the language to be described can be generated by successively rewriting strings starting from a designated start symbol. It in effect formalizes an algorithm that generates strings in the language.

This set of rules for is used for transforming strings. To generate a string in the language, one begins with a string consisting of only a single start symbol and then successively applies the rules (any number of times, in any order) to rewrite this string. The language consists of all the strings that can be generated in this manner. Any particular sequence of legal choices taken during this rewriting process yields one particular string in the language, but there might be multiple different ways of generating a single string.

For example, assume the alphabet consists of 'a' and 'b', the start symbol is 'S' and that the following rules are given:

```
1. S -> aSb
2. S -> ba
```

Then a derivation starts with 'S' and one can choose a rule to apply to it. If one chooses rule 1, then 'S' is replaced with 'aSb' and one obtains 'aSb'. Choosing rule 1 again and replacing 'S' with 'aSb' gives 'aaSbb'. This process is repeated until only symbols from the alphabet remain, that is, the symbols 'a' and 'b'. Finishing off the example, if now rule 2 is chosen, the 'S' is replaced by 'ba' giving 'aababb'. One can write this series of choices more briefly, using symbols:

```
S -> aSb -> aaSbb -> aababb.
```

The language of the grammar is the set of all the strings that can be generated using this process: $\{ba, abab, aababb, aaababbb, \ldots\} = \{a^nbab^n \mid n \geq 0\}$.

**Formal definition.** Noam Chomsky introduced the concept of formal grammars. A grammar $G$ consists of the following components:

- A finite set $N$ of nonterminal symbols.

- A finite set $\Sigma$ of terminal symbols that is disjoint from $N$.
- A finite set $P$ of production rules where a rule $\alpha \to \beta$ where $\alpha$ contains at least one symbol from $N$ and $\alpha, \beta$ are finite words over the alphabet $N \cup \Sigma$.
- A symbol $S$ in $N$ that is indicated as the start symbol.

Usually such a formal grammar $G$ is simply summarized as $(N, \Sigma, P, S)$.

**Examples.** Consider the grammar $G$ with $N = \{S, B\}$, $\Sigma = \{a, b, c\}$ and $P$ consisting of the following production rules

```
1. S -> aBSc
2. S -> abc
3. Ba -> aB
4. Bb -> bb
```

and the nonterminal symbol $S$ as the start symbol. Some examples of the derivation of strings in L(G) are

```
* S -> (2) abc
* S -> (1) aBSc -> (2) aBabcc -> (3) aaBbcc -> (4) aabbcc
* S -> (1) aBSc -> (1) aBaBScc -> (2) aBaBabccc -> (3) aaBBabccc
    -> (3) aaBaBbccc -> (3) aaaBBbccc -> (4) aaaBbbccc
    -> (4) aaabbbccc
```

where the used production rules are indicated in brackets. This grammar defines the language $\{a^n b^n c^n \mid n > 0\}$.

The language $\{a^n b^n \mid n > 0\}$ has a much easier grammar where $\Sigma = \{a, b\}$, $N = \{S\}$, the unique element $S$ of $N$ is the start symbol and the following two rules exist:

```
1. S -> aSb
2. S -> ab
```

The last example gives the language $\{a^n b^m \mid n, m > 0\}$ where one does not enforce that the number of 'a' and 'b' is equal. A grammar for this language uses again $\Sigma = \{a, b\}$ and $N = \{A, B, S\}$ where $S$ is the start symbol. The productions are

```
1. S -> aA
2. A -> aA
3. A -> bB
4. A -> b
5. B -> bB
6. B -> b
```

Note that these productions are of a special form: on the right side, the nonterminals are always at the end of the generated word and there is always only one of them.