# CS3133

HW#6

DUE: Tuesday, October 14

1. (8 points) ***a*** Describe the language generated by the CFG $G$

$$S \rightarrow 1S \mid S1 \mid 0$$

***b*** Is $G$ ambiguous? Justify your response.
***c*** If your answer to ***b*** is "no", then you are done with this question. Otherwise either show an unambiguous grammar to generate $L(G)$ or prove that such a grammar doesn't exist.

2. (12 points) For each of the following questions, construct an algorithm to accept as input an arbitrary CFG $G$ and tell whether or not $G$ satisfies the property in the question.

***a*** Is $L(G)$ empty?

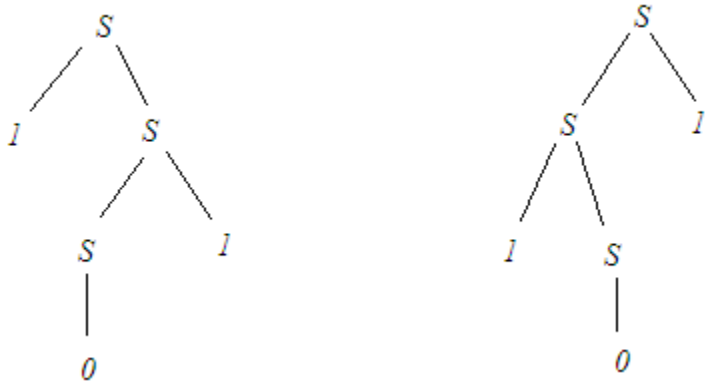
***b*** Is $L(G)$ finite?

3. (24 points) Answer each of the follwing questions, and justify your answers.
***a*** Is the class of recursive languages closed under union?
***b*** Is the class of recursive languages closed under intersection?
***c*** Is the class of recursive languages closed under complement?
***d*** Is the class of recursively enumerable languages closed under union?
***e*** Is the class of recursively enumerable languages closed under intersection?
***f*** Is the class of recursively enumerable languages closed under complement?

# CS3133
# Solutions for HW#6

1. ***a*** $1^*01^*$

***b*** $G$ is ambiguous. String $101 \in L(G)$ admits the two parse trees




***c***

$$S \to A0A$$
$$A \to 1A \mid \varepsilon$$

2. ***a*** Given CFG $G$, we convert it to an essentially equivalent CFG $G^*$ in Chomsky Normal Form.

We define $\text{GOOD} = \left\{ A \in N \mid \left(\exists w \in \Sigma^*\right)\left( A \overset{*}{\to} w \right)\right\}$. That is, GOOD is the set of all nonterminals which derive strings of terminals. We note that $L(G) = \varnothing$ if and only if $S \in \text{GOOD}$.

$\text{GOOD} \leftarrow \left\{ A \in N \mid \left(\exists a \in \Sigma\right)\left(A \to a\right)\right\}$

**repeat**

$\text{GOOD} \leftarrow \text{GOOD} \cup \left\{ A \in N \mid \left(\exists B, C \in N\right)\left(A \to BC\right)\right\}$

**until** GOOD doesn't change

**if** $S \in \text{GOOD}$ **then return** "$L(G) \neq \varnothing$"

**else return** "$L(G) = \varnothing$"

3. ***a*** Yes. For any recursive $L_0$ and $L_1$, there are total machines $M_0$ and $M_1$ to accept $L_0$ and $L_1$. We construct $M^*$ which runs $M_0$ and $M_1$ on any input $z$, and accepts $z$ if and only if at least one of $M_0$ and $M_1$ accepts $z$. Since $M_0$ and $M_1$ are total, so is $M^*$, and it accepts $L_0 \cup L_1$.

***b*** Yes. For any recursive $L_0$ and $L_1$, there are total machines $M_0$ and $M_1$ to accept $L_0$ and $L_1$. We construct $M^*$ which runs $M_0$ and $M_1$ on any input $z$, and accepts $z$ if and only if each of $M_0$ and $M_1$ accepts $z$. Since $M_0$ and $M_1$ are total, so is $M^*$, and it accepts $L_0 \cap L_1$.

***c*** Yes. For any $L$, there is total machine $M$ to accept $L$. We construct $M^*$ which runs $M$ on any input $z$, and accepts $z$ if and only if at least one of $M$ rejects $z$. Since $M$ is total, so is $M^*$, and it accepts $\sim L$.

***d*** Yes. For any recursively enumerable $L_0$ and $L_1$, there are machines $M_0$ and $M_1$ to accept $L_0$ and $L_1$. We construct $M^*$ which runs $M_0$ and $M_1$ on any input $z$, and accepts $z$ if and only if at least one of $M_0$ and $M_1$ accepts $z$. If $z \in L_0 \cup L_1$, then one of $M_0$ and $M_1$ must accept it. So $M^*$ accepts it also.

***e*** Yes. For any recursively enumerable $L_0$ and $L_1$, there are machines $M_0$ and $M_1$ to accept $L_0$ and $L_1$. We construct $M^*$ which runs $M_0$ and $M_1$ on any input $z$, and accepts $z$ if and only if both of $M_0$ and $M_1$ accepts $z$. If $z \in L_0 \cap L_1$, then each of $M_0$ and $M_1$ must accept it. So $M^*$ accepts it also.

***f*** No. The set of Turing Machines which halt when reading themselves as input is r.e., but the set of Turing Machines which do not halt when reading themselves as input is not r.e.