A Hierarchy of Languages (Section 12.4)

Context-Sensitive Languages

- A context-sensitive grammar has productions of the form $xAz \to xyz$, where A is a nonterminal and x, y, z are strings of grammar symbols with $y \neq \Lambda$. The production $S \to \Lambda$ is also allowed if S is the start symbol and it does not appear on the right side of any production. A context-sensitive language has a context-sensitive grammar.
- *Example:* The following grammar is context-sensitive:
  - $S \to aTb|ab$
  - $aT \to aaTb|ac$
- The language of this grammar is $\{ab\} \cup \{a^{n+1}cb^{n+1}|n \in \mathbb{N}\}$. This language is context-free. It has the context-free grammar:
  - $S \to aTb|ab$
  - $T \to aTb|c$
- Any context-free language is context-sensitive.

Another context-sensitive grammar

- $\{a^n b^n c^n\}$ is context-senstive, but not context-free. Here is a grammar:
  - $S \rightarrow \Lambda | abc | aTBc$
  - $T \rightarrow abC | aTBC$
  - $CB \rightarrow CX \rightarrow BX \rightarrow BC$ (a monotonic rule)
  - $bB \rightarrow bb$
  - $Cc \rightarrow cc$
- *Quiz:* Derive $aaabbbccc$
- *Answer:* $S \Rightarrow aTBc \Rightarrow aaTBCBc \Rightarrow aaabCBCBc \Rightarrow aaabBCCBc \Rightarrow aaabBCBCc \Rightarrow aaabBBCCc \Rightarrow aaabbbCCc \Rightarrow aaabbbCcc \Rightarrow aaabbbccc$

Linear Bounded Automata (LBA)

- A linear bounded automaton (LBA) is a Turing machine that may be nondeterministic and that restricts the tape to the length of the input with two boundary cells that may not change.
- *Example:* Describe an LBA to check whether a natural number n is divisible by $k \neq 0$.
- *Idea for a solution:* Use a 2-tape machine. For ease of explanation, represent k by the nonempty string 1^k and represent n by the string a^n. For example, if $k = 3$ and $n = 9$, the input is represented by:
- 111 aaaaaaaaa
- If $n = 0$, which is represented by Λ then halt. Otherwise, move both tape heads to the right k places while there are a's to read. THen leave the tape head for a's in place and move the tape head for k back to the left end and start the process over. Continue in this manner and enter the halt state if both tape heads point to Λ.

Recursively Enumerable Languages

- An unrestricted grammar has productions of the form $s \to t$ where $s \neq \Lambda$. So, any grammar is an unrestricted grammar.
- An unrestricted or recursively enumerable language has an unrestricted grammar.
- *Example:* THe following grammar is unrestricted:
  - $S \to TbC$
  - $Tb \to c$
  - $cC \to Sc|\Lambda$
- This grammar is not context-sensitive, not context-free and not regular.
- But we can transform it into $S \to Sc|\Lambda$ so the language of the grammar is regular.

Theorems

- *Theorem:* The recursively enumerable languages are exactly the languages that can be accepted by Turing machines. These languages can also be enumerated by Turing machines. (That's where "enumerable" comes from.)
- *Theorem:* $\{a^n | f_n(n) \text{ halts}\}$ is recursively enumerable and not context-sensitive.
- *Proof:* (1) Set $k := 0$. (2) For each pair (n, m), where $n + m = k$, execute $f_n(n)$ for m steps to see if it halts. If it halts, output a^n. (3) Increment k and goto (2).

Languages with no grammars

- Since there are a countable number of Turing machines, there are a countable number of languages with grammars. But there are an uncountable number of languages over an finite alphabet. So there are an uncountable number of languages that don't have a grammar.
- *Theorem:* $\{a^n | f_n \text{ is total}\}$ is not recursively enumerable. So it has no grammar.
- *Proof:* If, BWOC, the language is recursively enumerable, then we can enumerate it by a TM. So we can enumerate the total computable functions, which we can't do. QED.