

Section 14.2 A Hierarchy of Languages

Context-Sensitive Languages

A *context-sensitive grammar* has productions of the form $xAz \rightarrow xyz$, where A is a nonterminal and x, y, z are strings of grammar symbols with $y \neq \Lambda$. The production $S \rightarrow \Lambda$ 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.

$$\begin{aligned} S &\rightarrow aTb \mid ab \\ aT &\rightarrow aaTb \mid ac. \end{aligned}$$

Quiz. What is the language of the grammar?

Answer: $\{ab\} \cup \{a^{n+1}cb^{n+1} \mid n \in \mathbf{N}\}$. This language is context-free. For example, it has the grammar $S \rightarrow aTb \mid ab$, and $T \rightarrow aTb \mid c$.

Any context-free language is context sensitive.

Example. $\{a^n b^n c^n \mid n \geq 0\}$ is context-sensitive but not context-free. Here is a csg.

$$\begin{aligned} S &\rightarrow \Lambda \mid abc \mid aTBC \\ T &\rightarrow abC \mid aTBC \\ CB &\rightarrow CX \rightarrow BX \rightarrow BC \\ bB &\rightarrow bb. \\ Cc &\rightarrow cc. \end{aligned}$$

Quiz. Derive $aaabbbccc$.

Answer: $S \Rightarrow aTBC \Rightarrow aaTBCBc \Rightarrow aaabCBCBc \Rightarrow aaabBCCBc \Rightarrow aaabBCBCCc$
 $\Rightarrow aaabBBCCc \Rightarrow aaabbBCCc \Rightarrow aaabbbCCc \Rightarrow aaabbbCcc \Rightarrow aaabbbccc.$

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. An LBA to check whether a natural number n is divisible by $k \neq 0$.

Idea for a Solution: Use a 2-tape (or 2-head single 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 a a a a a a a a a

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 1 's 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 .

Theorem: The context-sensitive languages are exactly the languages that are accepted by linear bounded automata (LBA).

Example/Quiz. Find an LBA to accept $\{a^p \mid p \text{ is prime}\}$.

Idea for a Solution: Use the the divisibility algorithm in the previous example to implement the following algorithm, where $p \geq 2$.

$k := 2$;

while $k < p$ **do**

if k divides p **then** stop (but don't enter the halt state) **else** $k := k + 1$ **fi**

od;

Enter the halt state (accept).

Recursively Enumerable Languages

An *unrestricted grammar* has productions of the form $s \rightarrow t$, 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.

$$\begin{aligned} S &\rightarrow TbC \\ Tb &\rightarrow c \\ cC &\rightarrow Sc \mid \Lambda. \end{aligned}$$

This grammar is not context-sensitive, not context-free, and not regular.

But we can transform it into $S \rightarrow Sc \mid \Lambda$. So the language of the grammar is regular.

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: $\{a^n \mid f_n(n) \text{ halts}\}$ 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 go to (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 any finite alphabet. So there are an uncountable number of languages don't have a grammar.

Theorem: $\{a^n \mid f_n \text{ is total}\}$ 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. 3