CS381 Fall 2000	Handout 25
October 27, 2000	

1. Reading: D. Kozen *Automata and Computability*, Lecture 27
J. Hopcroft and J. Ullman *Introduction to Automata Theory, etc.*, section 6.3.

2. The main message of this lecture:

> **There are algorithms that given CFG determine whether the generated CFL is a) empty, b) finite, c) contains a given string. The latter is called the membership problem.**

Theorem 25.1. *There is an algorithm to determine if a CFL is empty.*

Proof. Let $G = (\Gamma, \Sigma, P, S)$ be a CFG. Consider a set of nonterminals $V \subseteq \Gamma$ consisting of all A's such that $A \xrightarrow{*} w$ for some terminal string w. Such V can be obtained by the following iterative procedure. Place to V all A's such that $A \longrightarrow w$ is a production from P. If $A \longrightarrow X_1X_2 \ldots X_n$ is a production, where each X_i is either a terminal or a nonterminal already placed in V, add A to V. If a scanning of the set of productions does not produce new members of V, halt. A proof that this algorithm produces V is straightforward (Hopcroft & Ullman, pp. 88-89). Now, $L(G) = \emptyset \Leftrightarrow S \notin V$. Q.E.D.

Definition 25.2. A symbol $A \in \Gamma \cup \Sigma$ is *useful* in a CFG $G = (\Gamma, \Sigma, P, S)$ if there is a derivation of some terminal string w from S containing A. Otherwise A is *useless*.

Theorem 25.3. *Given a CFG $G = (\Gamma, \Sigma, P, S)$ with nonempty $L(G)$ we can effectively find an equivalent CFG $G' = (\Gamma', \Sigma', P', S)$ without useless symbols.*

Proof. Since $L(G) \neq \emptyset$, for V from 25.1 $S \in V$. Obviously, we can get rid of all nonterminals not occurring in V, and all productions involving such nonterminals to get $G_1 = (V, \Sigma, P_1, S)$. This alone does not guarantee, however, that all nonterminals in V are useful. For example, in $\{S \longrightarrow a, \quad X \longrightarrow b\}$ $V = \{S, X\}$, but X is useless. Now, place S in Γ', and proceed with the following iterative algorithm. If $A \in \Gamma'$ and $A \longrightarrow \alpha$ is a production from P_1, place all nonterminals from α to Γ' and all terminals from α to Σ'. Let P' to be the set of productions containing only symbols from $\Gamma' \cup \Sigma'$. Note, that if the original grammar G was in CNF, then the resulting G' is again in CNF. Q.E.D.

Theorem 25.4. *There is an algorithm to determine if a CFL is finite.*

Proof. By 25.3, without loss of generality we may assume that $G = (\Gamma, \Sigma, P, S)$ is a CNF without useless symbols such that $L(G) \neq \emptyset$. Draw a graph E with edges (A, B) such that $A \longrightarrow BC$ or $A \longrightarrow CB$ is a production. Then $L(G)$ is finite if and only if this graph has no cycles. Indeed,

1^o. The graph E has a cycle $B, B_1, B_2, \ldots, B_{n-1}, B$. Therefore

$$B \longrightarrow \alpha_1 B_1 \beta_1 \longrightarrow \alpha_2 B_2 \beta_2 \longrightarrow \ldots \longrightarrow \alpha_{n-1} B_{n-1} \beta_{n-1} \longrightarrow \alpha_n B \beta_n,$$

where α_i, β_i are strings of nonterminals of total length i (in a CNF the length of a string after a nonterminal substitution becomes exactly one unit longer). Since there are no useless symbols in G, there are terminal strings u and v of total length at least n such that $\alpha_n \xrightarrow{*} u$

and $\beta_n \xrightarrow{*} v$, therefore, $B \xrightarrow{*} uBv \xrightarrow{*} u^2Bv^2 \xrightarrow{*} \dots \xrightarrow{*} u^mBv^m$ for any $m = 0, 1, 2, \dots$. (Note that the argument more and more resembles the one from the Pumping Lemma for CFGs). Since B is useful, there is a derivation of some terminal string from S containing B, say $S \xrightarrow{*} \alpha B\beta \xrightarrow{*} xyz$, where $\alpha \xrightarrow{*} x$, $\beta \xrightarrow{*} z$, and $B \xrightarrow{*} y$. A substitution of u^mBv^m for B gives $S \xrightarrow{*} \alpha u^mBv^m\beta \xrightarrow{*} xu^myv^mz$ for each $m = 0, 1, 2, \dots$. Thus $L(G)$ is infinite!

2^o. The graph E does not have cycles. Then the longest path in E has length $\leq k = |V|$ and every parsing tree in G has height $\leq k + 1$, and every derivable string has length $\leq 2^k$. There is only finite number of such strings over Σ. Q.E.D.

Example 25.5. Let G have productions $S \to AB,\ A \to BC,\ A \to a,\ B \to b,\ C \to c$. There are no cycles (note that S, A, B, S is not a cycle!), therefore $L(G)$ should be finite. A direct computation shows that $L(G) = \{ab, bcb\}$.

Example 25.6. Let G have productions $S \to AB,\ A \to BS,\ A \to a,\ B \to b,\ C \to c$. First of all, we prune out useless symbols c, C. Remaining productions contain a cycle S, A, S, therefore $L(G)$ is infinite. Indeed, it is easy to see that G generates strings b^mabb^m for all $m = 0, 1, 2, \dots$.

A membership problem: given $G = (\Gamma, \Sigma, P, S)$ and string x, is $x \in L(G)$? There is an obvious inefficient algorithms solving membership: Transform G into a CNF G', and try all derivations in G' shorter then $2|x|$ (there is only finite number of such derivations). Since any possible derivation of x in a CNF is shorter than $2|x|$, we will get a definite answer. Unfortunately, there might be exponentially many (of $|x|$) such derivations, and this algorithm is inefficient.

Theorem 25.7. *There is a cubic algorithm of solving the membership problem in CFG.*

Proof (Cocke-Kasami-Younger Algorithm). Again we assume that $G = (\Gamma, \Sigma, P, S)$ is a CNF such that $L(G) \neq \emptyset$. Using *dynamic programming* approach, determine for each i and j and for each nonterminal A whether $A \xrightarrow{*} x_{ij}$, where x_{ij} is the substring of x of length j beginning at position i.

begin
  1) **for** $i := 1$ **to** n **do**
      $V_{i1} := \{A \mid A \to a$ is a production and the ith symbol of x is $a\}$;
  2) **for** $j := 2$ **to** n **do**
  3)   **for** $i := 1$ **to** $n - j + 1$ **do**
  4)   **begin**
  5) $V_{ij} := \emptyset$;
  6)   **for** $k := 1$ **to** $j - 1$ **do**
  7) $V_{ij} := V_{ij} \cup \{A \mid A \to BC$ is a production, $B \in V_{ik}$ and $C \in V_{i+k,j-k}\}$
          end
end

Step (1) takes $O(n)$ time. Step (7) alone takes a constant time depending upon $|P|$. The innermost loop (6) and (7) together takes $O(n)$ time, (5) is again of a constant time, therefore (4) takes $O(n)$ time. The loop (3) then takes $O(n^2)$ time, the loop (2) takes $O(n^3)$. The aggregate time for the entire algorithm is the total time spent (1) and (2), which is $O(n^3)$.

Eventually, $x \in L(G)$ if and only if $S \in V_{1n}$.

Homework problems. Kozen, p.338 ♯ 91.