CFL's in P

1 Introduction

We sketch the proof that all CFG's are in Poly time. We will first need to get a CFG into a certain form.

2 Definitions

Some productions are never used so we want to get rid of them. We now define *useful* rigorously. Its negation will be *useless*.

Def 2.1 Let $G = (N, \Sigma, P, S)$ be a CFG. Let $A \in N$ and $\alpha \in (N \cup \sigma)^*$. $A \implies \alpha$ means that there is a sequence of applications of productions that take you from A to α. (This is often written with a G under the $\implies$ and a * over it.)

Def 2.2 Let $G = (N, \Sigma, P, S)$ be a CFG such that $L(G) \neq \emptyset$. A Nonterminal A is *useful* if the following two hold.

- There exists $w \in \Sigma^*$ such that $A \implies w$.
- There exists $\alpha, \beta \in (N \cup \Sigma)^*$ such that $S \implies \alpha A \beta$.

Note 2.3 If $L(G) = \emptyset$ then it's not clear how you can define useful nonterminals since S would be useless. To avoid this problem we only deal with G such that $L(G) \neq \emptyset$.

We can get by WITHOUT useless productions. We state this formally but do not prove it.

Theorem 2.4 *There is an algorithm that will, given a CFG G such that $L(G) \neq \emptyset$, produce a CFG G' with no useless productions such that $L(G') = L(G)$.*

Def 2.5 Let $G = (N, \Sigma, P, S)$ be a CFG. A production is a *Unit Production* if it is of the form $A \rightarrow B$ where A and B are nonterminals.

We can get by WITHOUT unit productions. We state this formally but do not prove it.

Theorem 2.6 *There is an algorithm that will, given a CFG G, produce a CFG G' with no unit productions such that $L(G) = L(G')$. (This procedure does not introduce useless productions.)*

Def 2.7 Let $G = (N, \Sigma, P, S)$ be a CFG. A production is an ϵ-*Production* if it is of the form $A \rightarrow \epsilon$.

Can we get by without ϵ-productions? If $e \in L$ then we need them. However, otherwise we do not. We state this formally but do not prove it.

Theorem 2.8 *There is an algorithm that will, given a CFG G produce a CFG G' with no e-productions such that $L(G') = L(G) - \{e\}$. (This procedure does not introduce useless or unit productions.)*

Putting together the above three theorems we have the following:

Theorem 2.9 *There is an algorithm that will, given a CFG G such that $L(G) \neq \emptyset$ produce a CFG G' with no useless productions, no unit productions, and no e-productions such that $L(G') = L(G) - \{e\}$.*

3 Chomsky Normal Form

Def 3.1 A grammar Let $G = (N, \Sigma, P, S)$ is in *Chomsky Normal Form* if every production is either of the form $A \rightarrow BC$ or $A \rightarrow \sigma$ where $\sigma \in \Sigma$.

Theorem 3.2 *There exists an algorithm that will, given a CFG $G = (N, \Sigma, P, S)$ such that $L(G) = \emptyset$ and $e \notin L(G)$ will output a grammar $G' = (N', \Sigma, P', S')$ in Chomsky Normal Form such that such that $L(G') = L(G) - \{\epsilon\}$.*

Proof:

By Theorem **??** there is a CFG for L with not useless productions, unit productions, or e-productions.

Look at each rule of the form $A \rightarrow \alpha_1\alpha_2 \cdots \alpha_m$. Note that $m \neq 1$ since that would be a unit production. If $m = 2$ then we do nothing since the production is already of the right form. So we assume $m \geq 3$. We do the following.

1. Replace every terminal α_i with nonterminals $[\alpha_i]$ and add the rule $[\alpha_i] \rightarrow \alpha_i$.
2. Note that the rule is now of the form

   $A \rightarrow \beta_1 \cdots \beta_m$

   where each β_i is a nonterminal.

   Replace this with the following:

   $A \rightarrow [\beta_1 \cdots \beta_{m-1}]\beta_m$

   $[\beta_1 \cdots \beta_{m-1}] \rightarrow [\beta_1 \cdots \beta_{m-2}]\beta_{m-1}$

   $[\beta_1 \cdots \beta_{m-2}] \rightarrow [\beta_1 \cdots \beta_{m-3}]\beta_{m-2}$

   etc until

   $[\beta_1\beta_2\beta_3]] \rightarrow [\beta_1\beta_2]\beta_3$

   $[\beta_1\beta_2] \rightarrow \beta_1\beta_2$.

▮

```
for i=1 to n
        A[i,i] = {B | B → w_i}
for d=1 to n−1
        for i=1 to n−d
                j=i+d
                A[i,j] = ⋃_{i≤k<j} {D | B ∈ A[i,k] ∧ C ∈ A[k+1,j] ∧ D → BC}
If S ∈ A[1,n] then output YES, else output NO.
```

4 CFL's in P

Theorem 4.1 *If L is a CFL then L is in $O(n^3)$.*

Proof: If $L = \emptyset$ then L is in $O(n^3)$ time. Apply the procedure in Theorem **??** to G to obtain a G' such that $L(G') = L(G) - \{\epsilon\}$. We show that $L(G')$ is in $O(n^3)$. This time does not count for the algorithm. This time is preprocessing.

We use DYNAMIC PROGRAMMING! Intuitively: Given a string $w = w_1 w_2 \ldots w_n$ we want to look which nonterminals A can produce $w_i ... w_j$. We do this, first for $i = j$ (that is $j - i = 0$) then for $j - i = 1$, $j - i = 2$, etc. The KEY is that D generates $w_i w_{i+1} \ldots w_j$ iff $D \rightarrow BC$ and B generates a prefix, say $w_i \cdots w_k$, and C generates the remaining suffice, say $w_{k+1} \cdots w_n$.

The formal algorithm is above.

There are $O(n^2)$ spaces in the array to fill out. Each one takes at most $O(n)$ to fill out.

∎