CS 154 – Notes for Lecture 9 (4/26/00)

Cleaning Up Grammars

We can "simplify" grammars to a great extent. Some of the things we can do are:

1. Get rid of *useless* symbols — those that do not participate in any derivation of a terminal string.
2. Get rid of *ϵ-productions* — those of the form variable $\to \epsilon$.
   - Well sort of; you lose the ability to generate ϵ as a string in the language.
3. Get rid of *unit productions* — those of the form variable $\to$ variable.
4. *Chomsky normal form* — only production forms are variable $\to$ two variables and variable $\to$ terminal.

Useless Symbols

In order for a symbol X to be useful, it must:

1. Derive some terminal string (possibly X *is* a terminal).
2. Be *reachable* from the start symbol; i.e., $S \overset{*}{\Rightarrow} \alpha X \beta$.

- Note that X wouldn't really be useful if α or β included a symbol that didn't satisfy (1), so it is important that (1) be tested first, and symbols that don't derive terminal strings be eliminated *before* testing (2).

Finding Symbols That Don't Derive Any Terminal String

Recursive construction:

Basis: A terminal surely derives a terminal string.

Induction: If A is the head of a production whose body is $X_1X_2 \cdots X_k$, and each X_i is known to derive a terminal string, then surely A derives a terminal string.

- Keep going until no more symbols that derive terminal strings are discovered.

Example

$S \to AB \mid C$; $A \to 0B \mid C$; $B \to 1 \mid A0$; $C \to AC \mid C1$.

- Round 1: 0 and 1 are "in."

- Round 2: $B \rightarrow 1$ says B is in.
- Round 3: $A \rightarrow 0B$ says A is in.
- Round 4: $S \rightarrow AB$ says S is in.
- Round 5: Nothing more can be added.
- Thus, C can be eliminated, along with any production that mentions it, leaving $S \rightarrow AB$; $A \rightarrow 0B$; $B \rightarrow 1 \mid A0$.

Finding Symbols That Cannot Be Derived From the Start Symbol

Another recursive algorithm:

Basis: S is "in."

Induction: If variable A is in, then so is every symbol in the production bodies for A.

- Keep going until no more symbols derivable from S can be found.

Example

$S \rightarrow AB$; $A \rightarrow 0B$; $B \rightarrow 1 \mid A0$.

- Round 1: S is in.
- Round 2: A and B are in.
- Round 3: 0 and 1 are in.
- Round 4: Nothing can be added.
- In this case, all symbols are derivable from S, so no change to grammar.
- Reader has an example where not only are there symbols not derivable from S, but you *must* eliminate first the symbols that don't derive terminal strings, or you get the wrong grammar.

Eliminating ϵ-Productions

A variable A is *nullable* if $A \stackrel{*}{\Rightarrow} \epsilon$. Find them by a recursive algorithm:

Basis: If $A \rightarrow \epsilon$ is a production, then A is nullable.

Induction: If A is the head of a production whose body consists of only nullable symbols, then A is nullable.

- Once we have the nullable symbols, we can add additional productions and then throw away the productions of the form $A \rightarrow \epsilon$ for any A.

- If $A \rightarrow X_1X_2 \cdots X_k$ is a production, add all productions that can be formed by eliminating some or all of those X_i's that are nullable.
  - ❖ But, don't eliminate all k if they are all nullable.

Example

If $A \rightarrow BC$ is a production, and both B and C are nullable, add $A \rightarrow B \mid C$.

Eliminating Unit Productions

1. Eliminate useless symbols and ϵ-productions.
2. Discover those pairs of variables (A, B) such that $A \stackrel{*}{\Rightarrow} B$.
   - ❖ Because there are no ϵ-productions, this derivation can only use unit productions.
   - ❖ Thus, we can find the pairs by computing reachablity in a graph where nodes = variables, and arcs = unit productions.
3. Replace each combination where $A \stackrel{*}{\Rightarrow} B \Rightarrow \alpha$ and α is other than a single variable by $A \rightarrow \alpha$.
   - ❖ I.e., "short circuit" sequences of unit productions, which must eventually be followed by some other kind of production.

   Remove all unit productions.

Chomsky Normal Form

0. Get rid of useless symbols, ϵ-productions, and unit productions (already done).
1. Get rid of productions whose bodies are mixes of terminals and variables, or consist of more than one terminal.
2. Break up production bodies longer than 2.

- Result: All productions are of the form $A \rightarrow BC$ or $A \rightarrow a$.

No Mixed Bodies

1. For each terminal a, introduce a new variable A_a, with one production $A_a \rightarrow a$.
2. Replace a in any body where it is not the entire body by A_a.
   - ❖ Now, every body is either a single terminal or it consists only of variables.

Example

$A \rightarrow 0B1$ becomes $A_0 \rightarrow 0$; $A_1 \rightarrow 1$; $A \rightarrow A_0BA_1$.

Making Bodies Short

If we have a production like $A \rightarrow BCDE$, we can introduce some new variables that allow the variables of the body to be introduced one at a time.

- A body of length k requires $k - 2$ new variables.
- Example: Introduce F and G; replace $A \rightarrow BCDE$ by $A \rightarrow BF$; $F \rightarrow CG$; $G \rightarrow DE$.

Summary Theorem

If L is any CFL, there is a grammar G that generates $L - \{\epsilon\}$, for which each production is of the form $A \rightarrow BC$ or $A \rightarrow a$, and there are no useless symbols.

CFL Pumping Lemma

Similar to regular-language PL, but you have to pump two strings in the middle of the string, in tandem (i.e., the same number of copies of each). Formally:

- $\forall$ CFL L
- $\exists$ integer n
- $\forall$ z in L, with $|z| \geq n$
- $\exists$ $uvwxy = z$ such that $|vwx| \leq n$ and $|vx| > 0$
- $\forall$ $i \geq 0$, uv^iwx^iy is in L.

Outline of Proof of PL

- Let there be a Chomsky-normal-form CFG for L with m variables. Pick $n = 2^m$.
- Because CNF grammars have bodies of no more than 2 symbols, a string z of length $\geq n$ must have some path with at least $m + 1$ variables.
- Thus, some variable must appear twice on the path.
  - ❖ Compare with the DFA argument about a path longer than the number of states.

- Focus on some path that is as long as any path in the tree. In this path, we can find a duplication of some variable A among the bottom $m+1$ variables on the path.
    - ✥ Let the lower A derive w and the upper A derive vwx.
- CNF guarantees us that $|vwx| \leq n$ and $vx \neq \epsilon$.
- By repeatedly replacing the lower A's tree by the upper A's tree, we see uv^iwx^iy has a parse tree for all $i > 1$.
    - ✥ And replacing the upper by the lower shows the case $i = 0$; i.e., uwy is in L.

Example

$L = \{0^{k^2} \mid k \text{ is any integer}\}$ is not a CFL.

- Suppose it were. Then let n be the PL constant for L.
- Consider $z = 0^{n^2}$. We can write $z = uvwxy$, with $|vwx| \leq n$ and $|vx| > 0$.
- Then $uvvwxxy$ is in L. But $n^2 < |uvvwxxy| \leq n^2 + n < (n+1)^2$, so there is no perfect square that $|uvvwxxy|$ could be.
- By "proof by contradiction," L is not a CFL.