COMS W3261
Computer Science Theory
Lecture 11
Closure and Decision Properties of CFL's

Alexander Roth

2014 – 10 – 08

Outline

1. Closure properties of CFL's
2. Nonclosure properties of CFL's
3. Cocke-Younger-Kasami algorithm
4. Testing emptiness of a CFG
5. Undecidable CFL problems

1 Closure Properties of CFL's

- The context-free languages are closed under the following operations:
  - Substitution
    - Let Σ be an alphabet and let L_a be a language for each symbol a in Σ. These languages define a substitution s on Σ.
    - If $w = a_1 a_2 \ldots a_n$ is a string in Σ^*, then $s(w) = \{x_1 x_2 \ldots x_n \mid x_i$ is a string in $s(a_i)$ for $1 \leq i \leq n\}$.
    - If L is a language, $s(L) = \{s(w) \mid w$ is in $L\}$.
    - If L is a CFL over Σ and $s(a)$ is a CFL for each a in Σ, then $s(L)$ is a CFL.
  - Union
  - Concatenation
  - Kleene star
  - Homomorphism

  - Reversal
  - Intersection with a regular set
  - Inverse Homomorphism

2 Nonclosure Properties of CFL's

- The context-free languages are not closed under the following operations:
  - Intersection
    * $L_1 = \{a^n b^n c^i \mid n, i \leq 0\}$ and $L_2 = \{a^i b^n c^n \mid n, i \leq 0\}$ are CFL's. But $L = L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ is not a CFL.
  - Complement
    * Suppose $comp(L)$ is context free if L is context free. Since $L_1 \cap L_2 = comp(comp(L_1) \cup comp(L_2))$, this would imply the CFL's are closed under intersection.
  - Difference
    * Suppose $L_1 - L_2$ is context free if L_1 and L_2 are context free. If L is a CFL over Σ, then $comp(L) = \Sigma^* - L$ would be context free.

Class Notes

$$L = \{a^n b^n c^n \mid n \geq 0\}$$

Prove L is Context Free!
half(L) $= \{x \mid xy$ is in L and $|x| = |y|\}$.

3 Cocke – Younger – Kasami Algorithm for Testing Membership in a CFL

- Input: a Chomsky normal form CFG $G = (V, T, P, S)$ and a string $w = a_1 a_2 \dots a_n$ in T^*.
- Output: "yes" if w is in $L(G)$, "no" otherwise.
- Method: The CYK algorithm is a dynamic programming algorithm that fils in a triangular table x_{ij} with nonterminals A such that $A \overset{*}{\Rightarrow} a_i a_i + j \dots a_j$.
- The algorithm adds nonterminal A to x_{ij} iff there is a production $A \rightarrow BC$ in P where $B \overset{*}{\Rightarrow} a_i a_{i+1} \dots a_k$ and $C \overset{*}{\Rightarrow} a_{k+1} a_{k+2} \dots a_j$.
- To compute entry x_{ij}, we examine at most n pairs of entries: $(x_{ii}, x_{i+1,j})$,$(x_{i,i+1}, x_{i+2,j})$, and so on until $(x_{i,j-1}, x_{j,j})$.
- The running time of the CYK algorithm is $O(n^3)$.

4 Testing Emptiness of a CFG

- Problem: Given a CFG G, is $L(G)$ empty?
  - A problem is decidable if there is an algorithm to solve it.
- Emptiness problem is decidable: determine whether the start symbol of G is generating.
  - Naive algorithm has $O(n^2)$ time complexity where n is the size of G (sum of the lengths of the productions).
  - With a more sophisticated list-processing algorithm, emptiness problems can be solved in linear time. See HMU, p. 302.

5 Undecidable CFL Problems

- We say a problem that cannot be solved by any Turing machine is *undecidable*. There is no algorithm that can solve an undecidable problem.
- We shall see that several fundamental questions about context-free grammars and languages are undecidable, such as:
  1. Is a given CFG ambiguous?
  2. Given a CFG, is there another equivalent CFG that is unambiguous?
  3. Do two given CFG's generate the same language?
  4. Is the intersection of the languages generated by two CFG's empty?
  5. Given a CFG $G = (V, T, P, s)$, is $L(G) = T^*$?

6 Class Notes

1. Given a CNF CFG G and an input string w is w in $L(G)$? This is decidable. Runs in $O(|w|^3)$ time.

```
for i = 1 to n do
  if A → a_i is in P then
    add A to X_ii
fill in the table, row-by-row, from row 2 to row n
  fill in the cells in each row from left-to-right
    if (A → BC is in P) and for some i ≤ k < j
      (B is in X_ik) and (C is in X_{k+1,j}) then
        add A to X_ij
if S is in X_1n then
  output "yes"
else
  output "no"
```

Figure 1: The CYK Algorithm in Pseudocode