CSE211 – Formal Languages and Automata Theory

U2L10_Simplification of CFG

Dr. P. Saravanan
School of Computing
SASTRA Deemed University

Outline

- Recap of previous class
- Properties of CGL
- Substitution rule
- Simplification of CFG
  - Eliminating useless production
  - Eliminating e-production
  - Eliminating unit production
- Reason for Simplication

Properties of CFL

- CFG's may be simplified to fit certain special forms, like
  - *Chomsky Normal Form (CNF)* and
  - *Greiback Normal Form (GNF)*.
- Some, but not all, properties of RL's are also possessed by the CFL's.
- Unlike the RL, many computational problems about the CFL *cannot* be answered.
- That is, there are many undecidable problems about CFL's.

A Substitution Rule

$$S \to aB$$
$$A \to aaA$$
$$A \to abBc$$
$$B \to aA$$
$$B \to b$$

Substitute
$B \to b$

$$S \to aB \mid ab$$
$$A \to aaA$$
$$A \to abBc \mid abbc$$
$$B \to aA$$

In general

$$A \to xBz$$

$$B \to y_1$$

Substitute

$$B \to y_1$$

$$A \to xBz \mid xy_1z$$

Simplification of CFG

- *Every CFG can be transformed into an equivalent grammar in Chomsky Normal Form,* after simplifying the CFG in the following ways:
-
  - eliminating *useless symbols* (which do not appear in any derivation from the start symbol);
  - eliminating *ε-productions* (of the form $A \to \varepsilon$);
  - eliminating *unit productions* (of the form $A \to B$);

Eliminating Useless Symbols

- We say symbol X is *useful* for a grammar $G = (V, T, P, S)$ if there is some derivation of the form
  - $S \overset{*}{\Rightarrow} aXb \overset{*}{\Rightarrow} w$ *with* $w \in T^*$.
- A symbol is said to be *useless* if not useful.
- Omitting useless symbols obviously will not change the language generated by the grammar.
- There are two types of *usefulness* ---
  - X is *generating* if $X \overset{*}{\Rightarrow} w$;
  - X is *reachable* if $S \overset{*}{\Rightarrow} aXb$.

Eliminating Useless Symbols Example 1

- Eliminate useless symbols in a grammar with the following productions:
  - $S \rightarrow AB \mid a$
  - $A \rightarrow b$.
- *B* is *not generating*, and is so eliminated at first, resulting in $S \rightarrow a$, $A \rightarrow b$, in which A is *not reachable*
- and so eliminated too, with $S \rightarrow a$ as the only production left.
- The order of eliminations is *essential*: *eliminate non-generating symbols at first.*

Eliminating Useless Symbols Thorem

- Let $G = (V, T, P, S)$ be a CFG, and assume that $L(G) \neq \phi$, i.e., assume that G generates at least one string. Let $G_1 = (V_1, T_1, P_1, S)$ be the grammar obtained by the following steps *in order*:
  - eliminate non-generating symbols and all related productions, resulting in grammar G_2;
  - eliminate all symbols not reachable in $G2$.
- Then, G_1 has no useless symbol and $L(G_1) = L(G)$.

Computing Generating and Reachable Symbols

- **How to compute generating symbols?**
- *Basis*: Every terminal symbol is generating.
- *Induction*: if every symbol in α in $A \to \alpha$ is generating, then A is generating.
-
- **How to compute reachable symbols?**
- *Basis*: the start symbol S is reachable.
- *Induction*: if nonterminal A is reachable, then all the symbols in $A \to \alpha$ are reachable.

Eliminating ε -Productions

- **A definition** --- a nonterminal A is said to be *nullable* if
  - $A \overset{*}{\Rightarrow} \varepsilon$.
- **A Theorem** --- We want to prove that
  - if a language *L* has a CFG, then the language *L* – {ε } can be generated by a CFG without ε -production.
- Two steps for the above proof:
  - find "nullable" symbols;
  - transform productions into ones which generate no empty string using the nullable symbols.

Eliminating ε -Productions

- Given a grammar with productions as follows:

$$S \to AB$$

$$A \to aAA \mid \varepsilon$$

$$B \to bBB \mid \varepsilon$$

- then, we can see the following facts:
  - *A* and *B* are *nullable* because they derive empty strings;
  - *S* is also *nullable* because *A* and *B* are nullable.

Eliminating ε -Productions

- How to find nullable symbols systematically?
- ***Algorithm 1 ---***
- *Basis*: if $A \rightarrow \varepsilon$ is a production, then A is nullable
- *Induction*: if all C_i in $B \rightarrow C_1C_2\ldots C_k$ are nullable, then B is nullable, too.
-

Eliminating ε -Productions

- How to transform productions into ones which generate no empty string?
- ***Algorithm 2 ---***
- For each production $A \rightarrow X_1X_2 \ldots X_k$, in which m of the k X_i's are nullable, then generate accordingly $2m$ versions of this production where
  - (1) the nullable X_i's in all possible combinations are present or absent; and
  - (2) if $A \rightarrow \varepsilon$ is in the $2m$ ones, eliminate it.

Eliminating ε -Productions

- For $S \to AB$, $A \to aAA \mid \varepsilon$, $B \to bBB \mid \varepsilon$:
  - We know S, A, B are *nullable.*
  - From $S \to AB$, we get $S \to AB \mid A \mid B \mid \varepsilon$ where $S \to \varepsilon$ should be eliminated.
  - From $A \to aAA$, we get $A \to aAA \mid aA \mid aA \mid a$ where the repeated $A \to aA$ should be removed.
  - And from $B \to bBB$, similarly we get $B \to bBB \mid bB \mid b$.
  - Overall result:

$$
\begin{aligned}
S &\to AB \mid A \mid B \\
A &\to aAA \mid aA \mid a \\
B &\to bBB \mid bB \mid b
\end{aligned}
$$

Why do we simplify & Example?

Summary

- Recap of previous class
- Properties of CGL
- Substitution rule
- Simplification of CFG
  - Eliminating useless production
  - Eliminating e-production
  - Eliminating unit production
- Reason for Simplication

References

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory,* Languages, and Computation, Pearson, 3rd Edition, 2011.
- Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.

Next Class:

Chomsky Normal Form (CNF)

THANK YOU