4CS4-6: Theory of Computation (Context-free languages and Grammars)

Lecture 12: Feb. 19, 2019

Prof. K.R. Chowdhary *: Professor of CS*

12.1 Introduction

Context-free grammars have been extensively used for describing the syntax of programming languages and natural languages. The parsing algorithms for context-free grammars play a large role in the implementation of compilers and interpreters for programming languages, and in programs that understand or translate natural language.

In the case of regular languages studied earlier, the regular expressions are generators of these languages, and the finite automata are recognizer of these languages. Consider a regular expression $a^*(b^* + c^*)b$. The algorithm for construction of all the possible strings for this regular expression can be specified in following steps:

1. Write character a, zero or any arbitrary number of times,
2. Write character b or c zero or arbitrary number of times,
3. Write b at the end.

If this algorithm executes infinitely large number of times and the string produced every time is compared with all the previously constructed strings, if it matches any of them, the current string is dropped otherwise it is retained. The set of all these retained strings is language generated by this regular expression. To recognize any of such string, we feed this string into the corresponding finite automaton. If it leads to a final state then the string it recognized by the finite automaton, else not.

Definition 12.1 *(Language recognizer.) A recognizer is an algorithm, which takes an input string and either accepts or rejects it depending on whether or not the string is a sentence of the language.*

Definition 12.2 *(Parser.) A parser is a recognizer, which also outputs the set of all legal derivation trees of the string sentence.*

The CFGs and CFLs are fundamental to computer science, because they help describe structure underlying programming languages. The basic "signature" of a context-free language

is "nested parentheses". For example, nesting occurs in expressions and in many statically scoped structures in computer programs. In contrast, the basic structure of regular languages is "iteration (looping)" according to some ultimate periodicity.

Example 12.3 *Represent the C language productions for balanced braces.*

The productions are:

$$\text{Lang-C} \rightarrow main \;\; parenthesis \;\; braces$$
$$parenthesis \rightarrow ()$$
$$braces \rightarrow \varepsilon \mid \{braces\}$$

12.2 Context-free Grammars

The context-free languages are powerful and more complex than the regular languages discussed earlier. The generators of context-free languages are called context-free grammars, and they are based on the more complete understanding of the structure of the strings accepted by the context free languages.

To understand the above process, let us consider the regular expression $a^*(b^* + c^*)b$ again, and divide it into three parts : A for the start, which corresponds to a^*,M for middle corresponding to b^*+c^*, and b is last part. Let the symbol S be start symbol for construction of any string in the language corresponding to $a^*(b^* + c^*)b$, the symbol "$\rightarrow$" stands for "can be substituted by". Then we write,

(1) $S \rightarrow AMb$

which means that S can be replaced by the sequence of symbols A, M and b. Symbol A consists alphabet a null or more number of times. Therefore,

(2) $A \rightarrow \varepsilon$

(3) $A \rightarrow aA$

In the rule (2) above, the symbol ε stands for empty string, which shows that non-terminal symbol A can be substituted by empty string (or call it null string).

The symbol M consists of an alphabet a any number of times, including null, or b any number of times including null. This can be specified in two stages, first for M, then its constituent parts.

(4) $M \rightarrow B$

(5) $M \rightarrow C$

Next, B and C can be specified in the similar way A was specified above.

(6) $B \to \varepsilon$

(7) $B \to bB$

(8) $C \to \varepsilon$

(9) $C \to cC$

The equations (1)-(9) in the above are called the *rules* of the context-free grammar , which is generator of context-free language. These are also called substitution rules or production rules, because they guide a substitution process to arrive at the string, which belong to the context-free language. Making use of these rules, the language corresponding to regular expression $a^*(b^* + c^*)b$ can be defined by an alternate language generator specified in the form of following algorithm.

Algorithm 1 Generate regular language for $a^*(b^* + c^*)b$.

1: Start with symbol S (the current sentence)
2: **while** there exists a symbol in the current string matching with any of the left hand side symbols of production rules **do**
3: Search a variable symbol in the current sentence (left to right) matching with some left hand side symbol of a production (say A).
4: Replace A by sequence of symbols in the right hand side of the production rule.
5: **end while**

In the above, the current string is progressively constructed starting with S and it is modified through the process of substitutions.

Example 12.4 *Find out whether the string aacccb is recognized as an element of the language corresponding to the regular expression* $a^*(b^* + c^*)b$.

We generate *aacccb* using following steps:

$$\begin{aligned}
S &\Rightarrow AMb, && \text{by rule (1)}\\
&\Rightarrow aAMb, && \text{by rule (3)}\\
&\Rightarrow aaAMb, && \text{by rule (3)}\\
&\Rightarrow aaMb, && \text{by rule (2)}\\
&\Rightarrow aaCb, && \text{by rule (5)}\\
&\Rightarrow aacCb, && \text{by rule (9)}\\
&\Rightarrow aaccCb, && \text{by rule (9)}\\
&\Rightarrow aacccCb, && \text{by rule (9)}\\
&\Rightarrow aacccb, && \text{by rule (8)}
\end{aligned}$$

Since, we are able to generate the string of the language of regular expression $a^*(b^* + c^*)b$, it confirms our claim of using context-free grammar as a language generator for a language called context free language. □

It must be noted that, first we designated the language corresponding to $a^*(b^* + c^*)b$ as regular language, and when it was shown to be generated by the CFG, it has been designated as CFL. This is true. In fact, all the regular languages are subset of context-free languages also.

During the process of generating string aacccb using CFG, the symbols S, M, A, B, C, a, b, and c have been used. The symbols a, b, c the final symbols in the generated string, are called *terminal symbols* or *terminals* of the CFG. The symbols S, M, A, B, C are encountered in the process of generation but they are not the final symbols, are called *Non-terminal* symbols. Since the generation process started with the symbol S, it is called *start symbol* of the CFG, and rules (1) to (9) are called *production rules* of the CFG.

Let us consider an intermediate stage in the substitution process, and look at the current strings, say $aaCb$ and $aaccCb$. The letter C present in these strings is in two different contexts. In the first case the substrings aa and b present before and after C respectively, represent context of C. In the second case the substrings $aacc$ and b represent a different context of C. In both of these cases, C has been substituted by cC as per substitution rule (9) CcC without any consideration for the contexts, i.e., substrings before and after C. Therefore, the substitution process in CFG is independent of the context. Hence the term *context-free grammar* and corresponding language as *context free language*. It should be noted that this feature is due to allowing only a single variable and no other symbol on the left side of the productions.

Definition 12.5 *Context-free Grammar. A context-free grammar G is a quadruple,*

$$G = (V, \Sigma, S, P) \tag{12.1}$$

where,

- V *is a finite set of variables called Non-terminal symbols,*
- Σ *is finite set of terminal symbols; V and Σ are disjoint, i.e., $V \cap \Sigma = \phi$,*
- S *is a special variable symbol, called the start symbol,*
- P *is finite set of production rules, where each production is of the form $A \to \alpha$, where $A \in V$ and α is a string of symbols defined as $\alpha \in (V \cup \Sigma)^*$.* □

The language $L(G)$, generated by G, is called CFL if G can generate every string in $L(G)$.

Every regular grammar is context-free and hence all the regular languages are context-free languages. It is already shown that regular language corresponding to the regular expression $a^*(b^* + c^*)b$ can be generated by context-free grammar. Later on it will be shown that language $\{a^n b^n \mid n \geq 0\}$ is context-free language. However, we know that the same is not regular. Therefore, we see that family of regular languages is proper subset of a family of context-free languages (Fig. 12.1).

It can be easily verified from the above definition that following grammars are context free:

$$\begin{aligned} G_1 &= (\{S\}, \{a, b\}, S, \{S \to aSa, S \to bSb, S \to \varepsilon\}) \\ L(G_1) &= \{a^n b^n \mid n \geq 0\} \\ G_2 &= aSa \mid bSb \mid a \mid b \\ L(G_2) &= \{ww^R \mid w \in \{a, b\}^*\} \end{aligned}$$

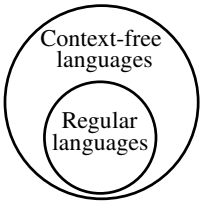

Figure 12.1: Relation between Regular and CF Languages.

Apart from the symbols used in the definition of CFG, following other symbols are also used during the derivation process of language strings, discussed in the remaining part of this chapter.

- The uppercase English alphabets A, B, C, D, E, S represent variable symbols; these are members of V. The symbol S implicitly represents start symbol.
- The lowercase letters a, b, c, d, e and digits (i.e., $0 \ldots 9$) represent terminals.
- The uppercase letters X, Y, Z represent variables as well as terminals.
- The lowercase letters u, v, w, x, y, z represent the strings of terminals.
- The Greek alphabet α, β, γ represent strings as members of $(V \cup \Sigma)^*$.

12.3 Context-free v/s Regular Grammars

It was shown in the previous section that regular expressions can be represented by context-free grammars. Also, it should be noted that the approach of representation of regular expression or regular language by CFG is a far simplistic approach, compared to FA.

Let us consider the regular expression $a^*ba^*b(a+b)^*$ and its FA shown in Fig. 12.2(a), $M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_2\})$.

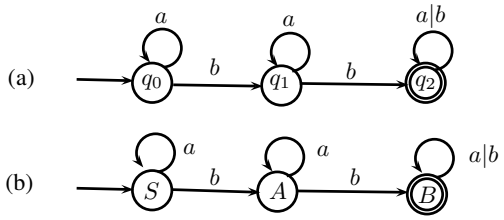

Figure 12.2: (a) FA for regular expression $a^*ba^*b(a+b)^*$, (b) States in FA expressed by non-terminals of CFG.

It can be easily verified that $w = aababab \in L(M)$.

Let us represent the states in the above FA by variables of CFG: S, A, B. The modified FA is shown in Fig. 12.2(b). To generate the string $aababab$ using this FA, we traverse through the states S, A, B and list the string progressively, as it gets constructed.

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aabA \Rightarrow aabaA$$
$$\Rightarrow aababB \Rightarrow aababaB \Rightarrow aababababB.$$

Since S generates the final string $aababababB$, we can write $S \Rightarrow^* aababababB$. We terminate this process using the production $B\varepsilon$. Hence, $S \Rightarrow^* aababab$. In fact, this has been achieved through use of productions rules. For example, to derive from S to aS, production rule $S \to aS$ has been used, and, to derive from aaS to $aabA$, the production rule $S \to bA$ has been used. All these productions are listed below.

$$S \to aS, S \to bA, A \to aA,$$
$$A \to bB, B \to aB, B \to bB, B \to \varepsilon$$

Each *variable* in the production rules of CFG correspond to a state in the FA. These productions correspond to regular grammars (definition 12.7). Hence, correspondence exists between various transitions in the states and production rules as follows:

$(S, a) = S$ corresponds to production rule $S \to aS$

$(S, b) = A$ corresponds to production rule $S \to bA$, and

$(B, a) = B$ corresponds to production rule $B \to aB$

In general, we say, $\delta(p, a) = q$, corresponds to $p \to aq$. Also, for each final state $f \in F$, there is a production $f \to \varepsilon$. Every FA leads to CFG exactly in a similar way, and therefore a language generated by CFG for a regular expression is exactly the language recognized by the FA.

Definition 12.6 *A context-free grammar specified as $G = (V, \Sigma, S, P)$ is regular grammar if and only if every production is out of the following forms only.*

$$A \to a$$
$$A \to aA$$
$$A \to \varepsilon.$$

where $A, B \in V$ and $a \in \Sigma$.

Definition 12.7 Right and Left linear Grammars. *A grammar is called right linear if all its rules are right linear (of the form $A \to aB$, $a \in \Sigma$) or terminating (of the form $A \to a$) and, is called left linear if every rule is left linear (of the form $A \to Ba$) or terminating.* □

Every left or right linear grammar generates a finite state language, i.e., one corresponding to a regular expression. Also, every finite state language is generated by a left linear and by right linear grammar.