بسم اللّه الرّحمن الرّحیم

دانشگاه صنعتی اصفهان ـ دانشکدهٔ مهندسی برق و کامپیوتر
(نیم‌سال تحصیلی ۴۰۲۱)

نظریهٔ زبان‌ها و ماشین‌ها

حسین فلسفین

☞ *In Chapter 1 we introduced two different, though equivalent, methods of describing languages: finite automata and regular expressions. We showed that many languages can be described in this way but that some simple languages, such as* $\{0^n 1^n | n \geq 0\}$, *cannot.*

☞ *In this chapter we present context-free grammars (CFGs), a more powerful method of describing languages.*

☞ *An important application of context-free grammars occurs in the specification and compilation of programming languages.*

☞ *The collection of languages associated with context-free grammars are called the context-free languages (CFLs). They include all the regular languages and many additional languages. In this chapter, we give a formal definition of context-free grammars and study the properties of context-free languages. We also introduce pushdown automata, a class of machines recognizing the context-free languages.*

☞ *Regular languages and finite automata are too simple and too restrictive to be able to handle languages that are at all complex. Using context-free grammars allows us to generate more interesting languages; much of the syntax of a high-level programming language, for example, can be described this way.*

☞ *This class is important. For most programming languages, the set of syntactically legal statements is (except possibly for type checking) a context-free language.*

☞ *It will be shown that a pushdown automaton and a context-free grammar are equivalent in power to specify languages. This equivalence is useful because it gives us two options for proving that a language is context free. We can give either a context-free grammar generating it or a pushdown automaton recognizing it. Certain languages are more easily described in terms of generators, whereas others are more easily described by recognizers.*

$$
\begin{aligned}
A &\rightarrow 0A1 \\
A &\rightarrow B \\
B &\rightarrow \#
\end{aligned}
$$

The above is an example of a CFG, which we call G_1. A grammar consists of a collection of substitution rules, also called productions.

☞ *Each rule appears as a line in the grammar, comprising a symbol and a string separated by an arrow. The symbol is called a variable. The string consists of variables and other symbols called terminals.*

☞ *The variable symbols often are represented by capital letters. The terminals are analogous to the input alphabet and often are represented by lowercase letters, numbers, or special symbols.*

☞ *One variable is designated as the start variable. It usually occurs on the left-hand side of the topmost rule. For example, grammar G_1 contains three rules. G_1's variables are A and B, where A is the start variable. Its terminals are 0, 1, and $\#$.*

You use a grammar to describe a language by generating each string of that language in the following manner.

1. *Write down the start variable. It is the variable on the left-hand side of the top rule, unless specified otherwise.*

2. *Find a variable that is written down and a rule that starts with that variable. Replace the written down variable with the right-hand side of that rule.*

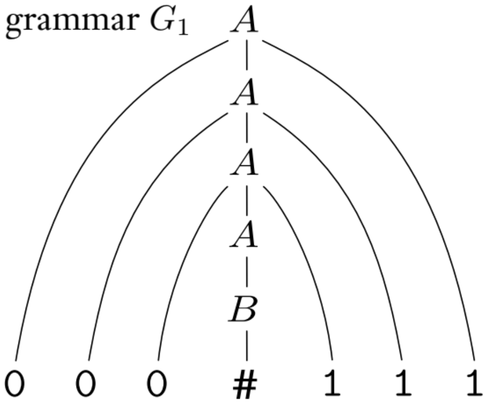
3. *Repeat step 2 until no variables remain.*

For example, grammar G_1 generates the string $000\#111$. The sequence of substitutions to obtain a string is called a derivation. A derivation of string $000\#111$ in grammar G_1 is

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111.$$

You may also represent the same information pictorially with a parse tree.

Parse tree for 000#111 in grammar G_1



☞ *Grammar G generates a string w when w consists of terminal symbols and is derived from the start symbol.*

☞ *All strings generated in this way constitute the language of the grammar. We write $L(G_1)$ for the language of grammar G_1. Some experimentation with the grammar G_1 shows us that*

$$L(G_1) = \{0^n\#1^n | n \geq 0\}.$$

☞ *Grammar G generates language L when L consists of all the strings that are generated by G. The language that G generates is denoted by $L(G)$. Any language that can be generated by some context-free grammar is called a context-free language (CFL).*

☞ *For convenience when presenting a context-free grammar, we abbreviate several rules with the same left-hand variable, such as $A \to 0A1$ and $A \to B$, into a single line $A \to 0A1|B$, using the symbol "$|$" as an "or".*

چند مــثــال

مثال ۱ : خود گرامر که آنرا G_1 می‌نامیم:

$$S \rightarrow 0|0S0|0S1|1S0|1S1$$

مثالی از تولید رشته توسط گرامر:

$$S \Rightarrow 0S1 \Rightarrow 01S11 \Rightarrow 010S011 \Rightarrow 0100011$$

زبان نظیر گرامر:

$L(G_1) = \{w \mid$ ***the length of*** w ***is odd and its middle symbol is a*** $0\}$

Example 2: Find a context-free grammar that generates the language $L = \{w|$***the length of*** w ***is odd***$\}$.
Solution:

$$S \rightarrow 1|0|0S0|0S1|1S0|1S1$$

Example 3:** The grammar $G = (\{S\}, \{a, b\}, R, S)$, **with productions

$$\begin{aligned} S &\rightarrow aSa \\ S &\rightarrow bSb \\ S &\rightarrow \varepsilon \end{aligned}$$

is context-free. A typical derivation in this grammar is

$$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabbaa.$$

This, and similar derivations, make it clear that

$$L(G) = \{ww^R : w \in \{a, b\}^*\}.$$

The language is context-free, but as shown in previous session, it is not regular.

***Example 4:** Find a context-free grammar that generates the language*

$$L = \{w \in \{0,1\}^* \mid w = w^R, \text{ that is, } w \text{ is a palindrome}\}.$$

Solution: $S \to \varepsilon|0|1|0S0|1S1$.

***Example 5:** Find a context-free grammar that generates the language* $\{w|w$ *starts and ends with the same symbol*$\}$.

Solution:

$$\begin{aligned} S &\to 0|1|0T0|1T1 \\ T &\to 0T|1T|\varepsilon \end{aligned}$$

***Example 6:** Find a context-free grammar that generates the language* $L = \{0^n1^{2n}|n \geq 0\}$.

Solution: $S \to 0S11|\varepsilon$.

مثال ۷: خود گرامر که آن‌را G می‌نامیم:

$$\begin{aligned} S &\rightarrow R1R1R1R \\ R &\rightarrow 0R|1R|\varepsilon \end{aligned}$$

مثالی از تولید رشته توسط گرامر:

$$S \Rightarrow R1R1R1R \Rightarrow 1R1R1R \Rightarrow 11R1R \Rightarrow$$
$$110R1R \Rightarrow 1101R \Rightarrow 1101$$

زبان نظیر گرامر:

$$L(G) = \{w \mid w \textbf{\textit{ contains at least three }} 1\textbf{\textit{s}}\}$$

☞**Remark:** *Many CFLs are the union of simpler CFLs. If you must construct a CFG for a CFL that you can break into simpler pieces, do so and then construct individual grammars for each piece. These individual grammars can be easily merged into a grammar for the original language by combining their rules and then adding the new rule $S \to S_1|S_2|\cdots|S_k$, where the variables S_i are the start variables for the individual grammars. Solving several simpler problems is often easier than solving one complicated problem.*

☞*For example, to get a grammar for the language $\{0^n1^n|n \geq 0\} \cup \{1^n0^n|n \geq 0\}$, first construct the grammar $S_1 \to 0S_11|\varepsilon$ for the language $\{0^n1^n|n \geq 0\}$ and the grammar $S_2 \to 1S_20|\varepsilon$ for the language $\{1^n0^n|n \geq 0\}$ and then add the rule $S \to S_1|S_2$ to give the grammar*

$$\begin{array}{lcl} S & \to & S_1|S_2 \\ S_1 & \to & 0S_11|\varepsilon \\ S_2 & \to & 1S_20|\varepsilon \end{array}$$

Example 8: ***Find a context-free grammar that generates the language*** $L = \{a^n b^m : n \neq m\}$.

Solution:

$$
\begin{array}{lcl}
S & \rightarrow & AS_1 | S_1 B \\
S_1 & \rightarrow & aS_1 b | \varepsilon \\
A & \rightarrow & aA | a \\
B & \rightarrow & bB | b
\end{array}
$$

Example 9: Give a context-free grammar that generates the language

$$L_1 = \{a^i b^j c^k \mid i = j \textbf{ or } j = k \textbf{ where } i, j, k \geq 0.\}$$

For example, the strings $aabbc$, abc, aaa are in L_1, while bcc, $abbc$, cab are not in L_1. The idea behind this solution is to cover either of the two cases $i = j$ or $j = k$ in separate grammar rules. The variable X generates all strings which have the same number of a's as b's, and the variable Y generates strings which have the same number of b's as c's. The rules for the start variable S specifies that any number of c's can follow X, or any number of a's can precede Y.

$$\begin{aligned} S &\to XC|AY \\ X &\to aXb|\varepsilon \\ Y &\to bYc|\varepsilon \\ A &\to Aa|\varepsilon \\ C &\to Cc|\varepsilon \end{aligned}$$

Example 10: Find a context-free grammar that generates the language $L = \{a^m b^n c^p d^q \ : \ m, n, p, q \geq 0$ ***and*** $m + n = p + q\}$.

Solution:

$$\begin{array}{rcl} S & \to & aSd \,|\, A \,|\, B \\ A & \to & bAd \,|\, D \\ B & \to & aBc \,|\, D \\ D & \to & bDc \,|\, \varepsilon \end{array}$$

Formal definition of a CFG

A ***context-free grammar*** is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the ***variables***,
2. Σ is a finite set, disjoint from V, called the ***terminals***,
3. R is a finite set of ***rules***, with each rule being a variable and a string of variables and terminals, and
4. $S \in V$ is the start variable.

In other words, a CFG is a 4-tuple $G = (V, \Sigma, R, S)$, where V and Σ are disjoint finite sets, $S \in V$, and R is a finite set of formulas of the form $A \to \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$. Elements of Σ are called terminal symbols, or terminals, and elements of V are variables, or nonterminals. S is the start variable, and elements of R are grammar rules, or productions.

دربارهٔ نمادهای $\to$، $\Rightarrow$، $\Rightarrow^n$، و $\Rightarrow^*$

We will reserve the symbol $\to$ for productions in a grammar, and we will use $\Rightarrow$ for a step in a derivation. The notations $\alpha \Rightarrow^n \beta$ and $\alpha \Rightarrow^* \beta$ refer to a sequence of n steps and a sequence of zero or more steps, respectively, and we sometimes write $\alpha \Rightarrow_G \beta$ or $\alpha \Rightarrow_G^n \beta$ or $\alpha \Rightarrow_G^* \beta$ to indicate explicitly that the steps involve productions in the grammar G.

If $G = (V, \Sigma, R, S)$ is a CFG, the language generated by G is $L(G) = \{x \in \Sigma^* \mid S \Rightarrow_G^* x\}$. A language L is a context-free language (CFL) if there is a CFG G with $L = L(G)$.

☞ *If u, v, and w are strings of variables and terminals, and $A \rightarrow w$ is a rule of the grammar, we say that uAv **yields** uwv, written $uAv \Rightarrow uwv$. Say that u derives v, written $u \Rightarrow^* v$, if $u = v$ or if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and*

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v.$$

☞ *For strings x and y, $x \Rightarrow y$ means that string x can be replaced by string y: more precisely, there exists a rule $A \rightarrow w$ and $u, v \in (V \cup \Sigma)^*$ such that $x = uAv$, $y = uwv$.*

☞ *If $x \Rightarrow^* y$, it is said that y is derived from x. And*

$$x_0 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_k$$

is called a derivation of x_k from x_0.

☞ *If two grammars G_1 and G_2 generate the same language, i.e., $L(G_1) = L(G_2)$, then G_1 and G_2 are equivalent.*

دربارهٔ وجه تسمیهٔ گرامرهای مستقل از متن

Context-free grammars derive their name from the fact that the substitution of the variable on the left of a production can be made any time such a variable appears in a sentential form. It does not depend on the symbols in the rest of the sentential form (the context). This feature is the consequence of allowing only a single variable on the left side of the production.