Context-Free Grammars (Part 2)

COMS3003A: Lecture Note 11

May 13, 2021

Definition: Context-free Grammar

Definition:
A *context-free grammar* (CFG) is a 4-tuple

$$G = (V, \Sigma, S, P) \quad ,$$

where

- V and Σ are finite sets, with $V \cap \Sigma = \emptyset$,
- $S \in V$ and
- P is a finite set of rules of the form $A \to \alpha$, where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$.

Note:
Elements of V are called variables/non-terminal symbols
Elements of Σ: terminal symbols
S: Start symbol
Elements of P: grammar rules/productions

Suppose $G = (V, \Sigma, S, P)$ is a CFG. From now on we'll use $\Longrightarrow_{\mathrm{G}}$ to denote one step in a derivation.

$$\alpha \Longrightarrow_{\mathrm{G}} \beta$$

denotes that the string β can be obtained from the string α by replacing some variable on the left-hand side of a production in P by the corresponding right-hand side.
In other words,

$$\alpha = \alpha_1 A \alpha_2$$

$$\beta = \alpha_1 \gamma \alpha_2$$

and one of the productions in P is $A \to \gamma$.
If it is clear which grammar is being used, $\Longrightarrow$ is used instead of $\Longrightarrow_{\mathrm{G}}$.

We use $\Longrightarrow_{\mathrm{G}}^{*}$ to denote the reflexive, transitive closure of $\Longrightarrow_{\mathrm{G}}$.

$$\alpha \Longrightarrow_{\mathrm{G}}^{*} \beta$$

denotes that α derives β in zero or more steps, i.e.,

- $\alpha = \beta$, or
- there exist an integer $k \geq 1$ and strings $\alpha_0, \alpha_1, \ldots, \alpha_k$, with $\alpha_0 = \alpha$ and $\alpha_k = \beta$, so that $\alpha_i \Longrightarrow_{\mathrm{G}} \alpha_{i+1}$ for every i with $0 \leq i \leq k-1$.

If it is clear which grammar is being used, $\Longrightarrow^{*}$ is used instead of $\Longrightarrow_{\mathrm{G}}^{*}$.

The language generated by a CFG

Definition
Let $G = (V, \Sigma, S, P)$ be a CFG.
The language generated by G is

$$L(G) = \{x \in \Sigma^* \mid S \Longrightarrow_{\mathrm{G}}^* x\}$$

A language L is a *context-free language* (CFL) if there is a CFG G so that $L = L(G)$.

The origin of the term *context-free*

Consider

$$\alpha = \alpha_1 A \alpha_2$$

$$\beta = \alpha_1 \gamma \alpha_2$$

where β is obtained by using the production $A \to \gamma$ in P.

If the string at some point of a derivation contains A, then we may use the production $A \to \gamma$ to obtain the next string in the derivation. We do not need to take the context, namely α_1 and α_2, into account.

How to show that a CFG generates a language

To demonstrate that a CFG generates a language, we need to show two things:

- The grammar can derive every string in the language.
- The grammar cannot derive any string not in the language.

Either one or both steps can be difficult.

Sometimes we can use the facts that

- the union of two CFLs is a CFL,
- the concatenation of two CFLs is a CFL,
- the Kleene* of a CFL is a CFL.

Unfortunately the complement of a CFL isn't necessarily a CFL.

The union of two CFLs is a CFL

Theorem:If L_1 and L_2 are CFLs, then $L_1 \cup L_2$ is also a CFL.
Sketch of proof : Let

$$G_1 = (V_1, \Sigma, S_1, P_1)$$

generate L_1 and

$$G_2 = (V_2, \Sigma, S_2, P_2)$$

generate L_2.

Construct the grammar

$$G_u = (V_u, \Sigma, S_u, P_u) \quad :$$

- Rename the elements of V_1 and V_2 (if necessary) so that $V_1 \cap V_2 = \emptyset$. Let S_u be any symbol not in $V_1 \cup V_2$. Define $V_u = V_1 \cup V_2 \cup \{S_u\}$.
- Let $P_u = P_1 \cup P_2 \cup \{S_u \to S_1 \mid S_2\}$.

G_u generates $L_1 \cup L_2$.

The concatenation of two CFLs is a CFL

Theorem If L_1 and L_2 are CFLs, then L_1L_2 is also a CFL.
Sketch of proof Let

$$G_1 = (V_1, \Sigma, S_1, P_1)$$

generate L_1 and

$$G_2 = (V_2, \Sigma, S_2, P_2)$$

generate L_2.
Construct the grammar

$$G_c = (V_c, \Sigma, S_c, P_c) \quad :$$

- Rename the elements of V_1 and V_2 so that $V_1 \cap V_2 = \emptyset$. Let S_c be any symbol not in $V_1 \cup V_2$. Define $V_c = V_1 \cup V_2 \cup \{S_c\}$.
- Let $P_c = P_1 \cup P_2 \cup \{S_c \to S_1S_2\}$.

G_c generates L_1L_2.

The Kleene* of a CFL is a CFL

Theorem If L_1 is a CFL, then L_1^* is also a CFL.
Sketch of proof Let

$$G_1 = (V_1, \Sigma, S_1, P_1)$$

generate L_1.
Construct the grammar

$$G_* = (V, \Sigma, S, P) \quad :$$

- Let S be any symbol not in V_1. Define $V = V_1 \cup \{S\}$.
- Let $P = P_1 \cup \{S \to S_1 S \mid \lambda\}$.

G_* generates L_1^*.

Example: A CFG for $\{x \mid n_0(x) = n_1(x)\}$

Consider the language

$$L = \{x \in \{0,1\}^* \mid n_0(x) = n_1(x)\}$$

where $n_i(x)$ is the number of i's in x.

- $\lambda \in L$
- For $x \in L$, $0x1 \in L$ and $1x0 \in L$. This generates all strings where the start symbol is different from the end symbol, e.g., 001011 and 1100.
- For $x, y \in L$, $xy \in L$. This generates, amongst others, all strings that start and end with the same symbol, e.g., 0110 or 10001101.

Thus we have the productions

$$S \to \lambda \mid 0S1 \mid 1S0 \mid SS$$

Example: A CFG for $\{x \mid n_0(x) \neq n_1(x)\}$

Consider the language

$$L = \{x \in \{0,1\}^* \mid n_0(x) \neq n_1(x)\}$$

where $n_i(x)$ is the number of i's in x.
This is the complement of the language in the previous example, but in the case of CFLs that observation doesn't help.
However, note that every string has either more 0's than 1's or vice versa.
Let

$$L_0 = \{x \in \{0,1\}^* \mid n_0(x) > n_1(x)\} \text{ and}$$

$$L_1 = \{x \in \{0,1\}^* \mid n_1(x) > n_0(x)\} \ .$$

Then

$$L = L_0 \cup L_1 \ .$$

Consider L_0:

- Adding 0's:
  - $0 \in L_0$
  - For $x \in L_0$, $x0 \in L_0$ and $0x \in L_0$.

  Thus we need the productions

$$S \to 0 \mid S0 \mid 0S$$

- Adding 1's:
  For $x \in L_0$, $x1$ or $1x$ may have the same number of 0's and 1's, therefore in general $x1 \notin L_0$ and $1x \notin L_0$.
  For $x, y \in L_0$, xy has at least two more 0 than 1's. Therefore, $1xy$, $x1y$, $xy1$ will have at least one more 0 than 1's. The productions are

$$S \to 1SS \mid SS1 \mid S1S$$

For L_0 we need

$$\begin{aligned} S &\rightarrow 0 \mid S0 \mid 0S \\ S &\rightarrow 1SS \mid SS1 \mid S1S \end{aligned}$$

For L_1 we need

$$\begin{aligned} S &\rightarrow 1 \mid S1 \mid 1S \\ S &\rightarrow 0SS \mid SS0 \mid S0S \end{aligned}$$

To form the grammar for L we must first rename the variables in at least one of L_0 and L_1.
For convenience sake we use A and B as start symbols of the grammars for L_0 and L_1, i.e.,

$$A \rightarrow 0 \mid A0 \mid 0A \mid 1AA \mid AA1 \mid A1A$$

and

$$B \rightarrow 1 \mid B1 \mid 1B \mid 0BB \mid BB0 \mid B0B$$

Finally,

$$S \rightarrow A \mid B$$

Example: $L = \{0^i 1^j 0^k \mid j > i + k\}$

Consider the language

$$L = \{0^i 1^j 0^k \mid j > i + k\} \quad .$$

We would like to write L as the concatenation of languages, but the naive approach:

$$L = L_1 L_2 L_3 \quad ,$$

where

$$\begin{aligned} L_1 &= \{0^i \mid i \geq 0\} \\ L_2 &= \{1^j \mid j \geq 1\} \\ L_3 &= \{0^k \mid k \geq 0\} \end{aligned}$$

won't work, since j is dependent on i and k.

Consider $x \in L$. Then

$$x = 0^i 1^{i+k+m} 0^k, \text{ with } m > 0 \ .$$

Then

$$x = 0^i 1^i 1^m 1^k 0^k, \text{ with } m > 0 \ .$$

We can therefore write

$$L = L_1 L_2 L_3 \ ,$$

where

$$\begin{aligned} L_1 &= \left\{ 0^i 1^i \mid i \geq 0 \right\} \\ L_2 &= \left\{ 1^m \mid m > 0 \right\} \\ L_3 &= \left\{ 1^k 0^k \mid k \geq 0 \right\} \end{aligned}$$

Consider $L_1 = \{0^i 1^i \mid i \geq 0\}$.

- $\lambda \in L_1$,
- for any $x \in L_1$, $0x1 \in L_1$,
- nothing else is in L_1.

We therefore need the productions

$$A \to 0A1 \mid \lambda$$

For $L_3 = \{1^k 0^k \mid k \geq 0\}$, we need the productions

$$C \to 1C0 \mid \lambda$$

For $L_2 = \{1^m \mid m > 0\}$, we need the productions

$$B \to 1B \mid 1$$

We generate $L = L_1L_2L_3$ with the CFG $G = (V, \Sigma, S, P)$, where

- $V = \{S, A, B, C\}$,
- $\Sigma = \{0, 1\}$,
- P has the four elements

$$\begin{aligned} S &\rightarrow ABC \\ A &\rightarrow 0A1 \mid \lambda \\ B &\rightarrow 1B \mid 1 \\ C &\rightarrow 1C0 \mid \lambda \end{aligned}$$

$$
\begin{aligned}
S &\Longrightarrow ABC \\
&\Longrightarrow 0A1BC \\
&\Longrightarrow 0\lambda 1BC \\
&\Longrightarrow 011C \\
&\Longrightarrow 0111C0 \\
&\Longrightarrow 01111C00 \\
&\Longrightarrow 01111\lambda 00 \\
&= 0111100
\end{aligned}
$$