Applications of Grammars

- Specifying syntax of programming languages

- Representing syntactic structures in natural languages

- Models of computation

Context-free grammars are the most commonly used kind of grammar in computer science.
Progression of Concepts

- **Context-free grammar**: Variables, terminals, rules, start symbol

- **Derivation**: How one string derives another

- **Context-free language**: Generated by a context-free grammar

- **Derivation tree**: Graphically representing a derivation
Context-Free Grammar

A context-free grammar (CFG) is a 4-tuple \( G = (V, \Sigma, P, S) \) where

- \( V \) is an alphabet of variables (or nonterminals);

- \( \Sigma \) is an alphabet of terminals, disjoint from \( V \);

- \( P \) is a finite subset of \( V \times (V \cup \Sigma)^* \), called the set of rules (or productions); and

- \( S \in V \) is the start symbol.
Example CFG

\[ G_1 = (\{S, A, B\}, \{a, b\}, \]
\[ \{(S, aAbBa), (A, aS), (B, Ab), \]
\[ (B, SbB), (B, \lambda)\}, S) \]

Variables are capitalized. The start symbol is (almost) always \( S \).

Rules are usually written

\[ S \rightarrow aAbBa \]

or

\[ S \xrightarrow{G_1} aAbBa \]

to emphasize the grammar \( G_1 \).
Example CFG Continued

$G_1$ is compactly written:

\[
\begin{align*}
S & \rightarrow aAbBa \\
A & \rightarrow aS \\
B & \rightarrow Ab \\
B & \rightarrow SbB \\
B & \rightarrow \lambda
\end{align*}
\]
One-Step Derivation

Want to define $\Rightarrow$, a binary relation on $(V \cup \Sigma)^*$. 

Let $G = (V, \Sigma, P, S)$ be a CFG. Suppose that $u, v \in (V \cup \Sigma)^*$, that $A \in V$, and that 

$$A \xrightarrow{G} w$$

is a rule. Then the string $uAv$ derives (in one step) the string $uvw$, written 

$$uAv \xrightarrow{G} uvw.$$

**EXAMPLE.**

$$BaAb \xrightarrow{G_1} SbBaAb$$

$$BaAb \xrightarrow{G_1} BaaSb$$
Derivation in Zero or More Steps

We give a recursive definition for $\Rightarrow^*$, a binary relation on $(V \cup \Sigma)^*$.

**Basis:** If $v \in (V \cup \Sigma)^*$, then $v \Rightarrow^* v$. Also, $v \Rightarrow^0 v$, read $v$ derives $v$ in zero steps.

**Recursive step:** If $u, v, w \in (V \cup \Sigma)^*$
- $u \Rightarrow^* v$, and $v \Rightarrow w$, then $u \Rightarrow^* w$. Also, if $u \Rightarrow^n v$ and $v \Rightarrow w$, then $u \Rightarrow^{n+1} w$. 
Derivation Example

Here is a derivation in $G_1$:

$BaAb \Rightarrow SbBaAb$

$\Rightarrow SbBaaSb$

$\Rightarrow SbBaaaAbBab$

$\Rightarrow SbaaaAbBab$

$\Rightarrow SbaaaAbAbab$

$\Rightarrow SbaaaaSbAbab.$

We can conclude, for example, that

$BaAb \xRightarrow{4}{G_1} SbaaaAbBab$

and that

$BaAb \xRightarrow{*}{G_1} SbaaaaSbAbab.$
Context-Free Language

Let $G = (V, \Sigma, P, S)$ be a context-free grammar.

- If $S \xrightarrow{G}^* w$, then $w$ is a sentential form of $G$.

- If $S \xrightarrow{G}^* w$ and $w \in \Sigma^*$, then $w$ is a sentence of $G$.

- The language generated by $G$, written $L(G)$, is the set of all sentences of $G$:

$$L(G) = \left\{ w \in \Sigma^* \mid S \xrightarrow{G}^* w \right\}.$$ 

Any language generated by a CFG is a context-free language (CFL).
Example of a CFL

Let $G_2 = (\{S, A, B, C\}, \{0, 1\}, P_2, S)$ be a CFG, where $P_2$ is given by

$$
\begin{align*}
S & \rightarrow A \mid B \mid C \\
A & \rightarrow 0C \mid \lambda \\
B & \rightarrow 1B \mid 1 \\
C & \rightarrow BA.
\end{align*}
$$

What is $L(G_2)$?

A typical derivation might be

$$
\begin{align*}
S & \Rightarrow C \\
& \Rightarrow BA \\
& \Rightarrow B0C \\
& \Rightarrow^* (B0)^nC \\
& \Rightarrow^* (B0)^nBA \\
& \Rightarrow (B0)^nB,
\end{align*}
$$

for some $n \geq 0$. 
Continue Example of a CFL

So every

$$(B0)^nB$$

is a sentential form.

We note that $B$ generates $1^+$ and that $\lambda \in L(G_2)$.

We also have sentential forms

$$0(B0)^nB = (0B)^{n+1}.$$ 

Putting it all together, we find that

$$L(G_2) = \{w \in \{0, 1\}^* \mid \text{each 0 in } w \text{ is followed immediately by a 1}\}.$$
Derivation Tree

For every derivation of a sentential form \( S \rightarrow^* w \), there is a corresponding derivation tree (DT), which is an oriented tree with these characteristics:

- The root of the tree is labeled \( S \).
- Every internal node is labeled with a variable.
- Every leaf is labeled with an element of \( V \cup \Sigma \cup \{\lambda\} \).
- The string gotten by reading the leaves from left to right is \( w \). It is called the yield or frontier of the tree.
Example DT

For this derivation in $G_1$

\[
S \implies aA\text{bB}a \\
\implies aA\text{bSbB}a \\
\implies aaSbSbBa \\
\implies aaSbSba, \\
\]

we obtain this derivation tree

```
   S
  / \  /  \
 a   A b   B
/ \ / \ /   / \
 a S S b B
   \   \   \
    \   \   
     \   \   
      \   \   
       \   \   
        \   \   
         \   \   
          \   \   
           \   \   
            \   \   
             \   \   
              \   \   
               \   \   
                \   \   
                 \   \   
                  \   \   
                   \   \   
                    \   \   
                     \   \   
                      \   \   
                       \   \   
                        \   \   
                         \   \   
                          \   \   
                           \   \   
                            \   \   
                             \   \   
                              \   \   
                               \   \   
                                \   \   
                                 \   \   
                                  \   \   
                                   \   \   
                                    \   \   
                                     \   \   
                                      \   \   
                                       \   \   
                                        \   \   
                                         \   \   
                                          \   \   
                                           \   \   
                                            \   \   
                                             \   \   
                                              \   \   
                                               \   \   
                                                \   \   
                                                 \   \   
                                                  \   \   
                                                   \   \   
                                                    \   \   
                                                     \   \   
                                                      \   \   
                                                       \   \   
                                                        \   \   
                                                         \   \   
                                                          \   \   
                                                           \   \   
                                                            \   \   
                                                             \   \   
                                                              \   \   
                                                               \   \   
                                                                \   \   
                                                                 \   \   
                                                                  \   \   
                                                                   \   \   
                                                                    \   \   
                                                                                       
```
DT Exercise

This is a sample derivation in $G_2$

\[
S \implies A \\
\implies 0C \\
\implies 0BA \\
\implies 0B0C \\
\implies 01B0C \\
\implies 01B0BA \\
\implies 0110BA \\
\implies 0110B \\
\implies 01101.
\]

What is the derivation tree obtained from this derivation? What other derivations have the same DT? Which is leftmost? Which is rightmost?
Languages from Grammars

Consider this grammar

\[ G_1 : \quad S \rightarrow abSb \mid A \]
\[ A \rightarrow dAcb \mid \lambda \]

and the derivation

\[ S \quad \Rightarrow \quad abSb \]
\[ \quad \Rightarrow \quad ababSbb \]
\[ \quad \Rightarrow \quad ababAbb \]
\[ \quad \Rightarrow \quad ababdAcbbb \]
\[ \quad \Rightarrow \quad ababddAcbcbbb \]
\[ \quad \Rightarrow \quad ababddcbcbbb. \]

Give the derivation tree?

What language is generated by \( G_1 \)?
More Languages from Grammars

Consider this grammar

\[ G_2 : \quad S \rightarrow aAb \mid SS \]
\[ \quad A \rightarrow cBd \]
\[ \quad B \rightarrow bBa \mid \lambda. \]

What language is generated by \( G_2 \)?
Grammars from Languages

For each of the following languages, find (if possible) a grammar that generates it.

\[ L_1 = \{ww^R \mid w \in \{0, 1\}^*\} \]

\[ L_2 = \{w \mid w \in \{0, 1\}^*, w = w^R\} \]

\[ L_3 = \{0^n1^m0^m1^n \mid m, n \geq 0\} \]

\[ L_4 = \{(010)^m(11)^n \mid 0 \leq m \leq n\} \]

\[ L_5 = \{0^m1^m0^n \mid m, n \geq 0\} \cup \{0^m1^n0^n \mid m, n \geq 0\} \]
More Grammars from Languages

For each of the following languages, find (if possible) a grammar that generates it.

\[ L_6 = \{0^m1^n \mid m, n \geq 0 \text{ and } m \neq n\} \]

\[ L_7 = \{w \in \{(,\}) \mid w \text{ is a balanced string of parentheses}\} \]

\[ L_8 = \{0, 1\}^+ \]

For \( L_8 \), give a grammar that has every oriented tree with outdegree at most 2 as a derivation tree.
Regular Grammar

A regular grammar is a CFG where every rule has one of these forms:

1. \( A \rightarrow a \)

2. \( A \rightarrow aB \)

3. \( A \rightarrow \lambda \)

EXAMPLE. A regular grammar:

\[
G_3 : \quad S \rightarrow aA \mid \lambda \\
A \rightarrow bB \mid cC \\
B \rightarrow bB \mid bS \\
C \rightarrow cC \mid cS
\]

Give a regular expression for \( L(G_3) \)?
Regular Language

A regular language is one generated by a regular grammar.

We prove much later that the regular sets and the regular languages are identical.

In the meantime, we can practice using both regular expressions and regular grammars to represent regular languages.
Regular Grammars from Regular Expressions

For each of the following regular expressions, find a regular grammar that generates the language it represents.

\[ r_1 = (a \cup b)^* \]

\[ r_2 = ((ab)^*(bc)^*)^* \]

\[ r_3 = (a^*b) \cup (b^+ab) \]
Regular Expressions from Regular Grammars

Consider this regular grammar

\[ G_4 : \quad \begin{align*}
S & \rightarrow aA \mid bS \\
A & \rightarrow bB \mid a \\
B & \rightarrow bA \mid \lambda
\end{align*} \]

Give a regular expression for \( L(G_4) \)?
Proving Properties

Given a CFG $G = (V, \Sigma, P, S)$, we need to prove properties about $L(G)$.

Often we want to show that $L(G)$ is equivalently given by some alternate description $L'$.

**Strategy:** Show $L' \subseteq L(G)$ and $L(G) \subseteq L'$.

- To show $L' \subseteq L(G)$, give a derivation schema for each string in $L'$.

- To show $L(G) \subseteq L'$, typically use induction on the number of steps in a derivation.

Useful to define $n_\sigma(w)$, where $\sigma \in (V \cup \Sigma)$ and $w \in (V \cup \Sigma)^*$, to be the number of $\sigma$’s in $w$. 
Equality of Languages

EXAMPLE.

Start with the language

\[ L' = \{0^m1^n \mid 1 \leq m \leq n\} \cup \{\lambda\} \]

A candidate grammar to generate \( L' \) is

\[
G : \quad S &\rightarrow 0SA | \lambda \\
A &\rightarrow A1 | 1.
\]

To show \( L' \subseteq L(G) \), since \( \lambda \in L(G) \) by \( S \rightarrow \lambda \), we only need to give a derivation schema for each \( 0^m1^n \in L' \), where \( 1 \leq m \leq n \).

<table>
<thead>
<tr>
<th>Derivation</th>
<th>Production Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \xrightarrow{m} 0^mSA^m )</td>
<td>( S \rightarrow 0SA )</td>
</tr>
<tr>
<td>( \xrightarrow{} 0^mA^m )</td>
<td>( S \rightarrow \lambda )</td>
</tr>
<tr>
<td>( \xrightarrow{n-m} 0^mA^m1^{n-m} )</td>
<td>( A \rightarrow A1 )</td>
</tr>
<tr>
<td>( \xrightarrow{m} 0^m1^n )</td>
<td>( A \rightarrow 1 )</td>
</tr>
</tbody>
</table>
Induction on Derivation Steps

Prove \( L(G) \subset L' \) by induction on \( k \), the number of derivation steps to reach a sentential form \( w \).

**Inductive Hypothesis:** Suppose \( S \xrightarrow{G^k} w \). Then \( w \) satisfies:

1. \( n_0(w) \leq n_1(w) + n_A(w) \); and

2. \( w \in 0^*(S \cup \lambda)(A \cup 1)^* \).
Induction on Derivation Steps (Concluded)

- **Basis:** For 0 steps, \( w = S \) is the only sentential form. Check that the inductive hypothesis holds.

- **Inductive Hypothesis:** Previously given.

- **Inductive Step:** For some fixed \( k \geq 0 \), assume that the inductive hypothesis holds for derivations of \( \leq k \) steps. Suppose that \( S \xrightarrow{G}^{k+1} w \). Show that \( w \) satisfies the inductive hypothesis.

By the Principle of Mathematical Induction, the inductive hypothesis holds for all sentential forms \( w \). Since any string of terminals satisfying the inductive hypothesis is in \( L' \), we conclude that \( L(G) \subseteq L' \).
Properties of Languages

Example 3.4.3

\[ G : \quad S \rightarrow aASB \mid \lambda \]
\[ A \rightarrow ad \mid d \]
\[ B \rightarrow bb. \]

Want to show that every \( w \in L(G) \) satisfies:
\[ n_a(w) \leq n_b(w). \]

Rather we show something stronger:

For every sentential form \( z \),
\[ n_a(z) + n_A(z) \leq n_b(z) + 2n_B(z). \]

This stronger statement implies what we want to show.
Induction Again

- **Basis:** The only sentential form derivable in 0 steps is \( z = S \), which satisfies
  \[
  n_a(S) + n_A(S) \leq n_b(S) + 2n_B(S).
  \]

- **Inductive Hypothesis:** Every sentential form \( z \) derivable in \( k \) or fewer steps satisfies
  \[
  n_a(z) + n_A(z) \leq n_b(z) + 2n_B(z).
  \]

- **Inductive Step:** Suppose \( k \geq 0 \) and the inductive hypothesis holds for \( k \). Further suppose that
  \[
  S \xrightarrow{k+1} z.
  \]
  Show that the sentential form \( z \) satisfies
  \[
  n_a(z) + n_A(z) \leq n_b(z) + 2n_B(z).
  \]
Induction Again (Continued)

There must be a sentential form $z'$ such that

$$S \xrightarrow{k} z'$$

$$\Longrightarrow z.$$

By the inductive hypothesis, we know that

$$n_a(z') + n_A(z') \leq n_b(z') + 2n_B(z'). \quad (1)$$

There are five cases to consider for the production that is used to derive $z$ from $z'$. Consider each in turn.
Induction Again (Continued)

Case 1: $S \rightarrow aASB$

\[
\begin{align*}
 n_a(z) &= n_a(z') + 1 \\
 n_A(z) &= n_A(z') + 1 \\
 n_b(z) &= n_b(z') \\
 n_B(z) &= n_B(z') + 1
\end{align*}
\]

Combining these with Inequality 1, we obtain

\[
(n_a(z') + 1) + (n_A(z') + 1) \leq n_b(z') + 2(n_B(z') + 1)
\]

\[
n_a(z) + n_A(z) \leq n_b(z) + 2n_B(z).
\]
Induction Again (Continued)

Case 2: $S \rightarrow \lambda$

\[
\begin{align*}
  n_a(z) &= n_a(z') \\
  n_A(z) &= n_A(z') \\
  n_b(z) &= n_b(z') \\
  n_B(z) &= n_B(z')
\end{align*}
\]

Combining these with Inequality 1, we obtain

\[
\begin{align*}
  n_a(z') + n_A(z') &\leq n_b(z') + 2n_B(z') \\
  n_a(z) + n_A(z) &\leq n_b(z) + 2n_B(z).
\end{align*}
\]
Induction Again (Continued)

Case 3: \( A \rightarrow ad \)

\[
\begin{align*}
n_a(z) &= n_a(z') + 1 \\
n_A(z) &= n_A(z') - 1 \\
n_b(z) &= n_b(z') \\
n_B(z) &= n_B(z')
\end{align*}
\]

Combining these with Inequality 1, we obtain

\[
(n_a(z') + 1) + (n_A(z') - 1) \leq n_b(z') + 2n_B(z')
\]

\[
n_a(z) + n_A(z) \leq n_b(z) + 2n_B(z).
\]
Induction Again (Continued)

Case 4: \( A \rightarrow d \)

\[
\begin{align*}
n_a(z) & = n_a(z') \\
n_A(z) & = n_A(z') - 1 \\
n_b(z) & = n_b(z') \\
n_B(z) & = n_B(z')
\end{align*}
\]

Combining these with Inequality 1, we obtain

\[
\begin{align*}
n_a(z') + (n_A(z') - 1) & \leq n_b(z') + 2n_B(z') \\
n_a(z) + n_A(z) & \leq n_b(z) + 2n_B(z).
\end{align*}
\]
Induction Again (Continued)

Case 5: $B \rightarrow bb$

\[
\begin{align*}
  n_a(z) & = n_a(z') \\
  n_A(z) & = n_A(z') \\
  n_b(z) & = n_b(z') + 2 \\
  n_B(z) & = n_B(z') - 1
\end{align*}
\]

Combining these with Inequality 1, we obtain

\[
\begin{align*}
  n_a(z') + n_A(z') & \leq (n_b(z') + 2) + 2(n_B(z') - 1) \\
  n_a(z) + n_A(z) & \leq n_b(z) + 2n_B(z).
\end{align*}
\]
Induction Again (Concluded)

In all 5 cases, we obtain

\[ n_d(z) + n_A(z) \leq n_b(z) + 2n_B(z). \quad (2) \]

By the Principle of Mathematical Induction, Inequality 2 holds for every sentential form \( z \). This is what we wanted to prove.
Exercise on CFG Proof

Start with the CFG

\[ G : \quad S \rightarrow (S)S | \lambda. \]

Let \( L' \) be the following language:

\[ L' = \{ w \in \{(,)\}^* \mid w \text{ is a balanced string of parentheses}\}. \]

Show that \( L' = L(G) \).
Exercise on CFG Proof (Hint)

The problem with the exercise is that the concept of a “balanced string of parentheses” is not precisely defined. At some point we have to develop a precise definition. Here is one.

A string \( w \in \{(,\}\}^* \) is left dominant if, for every prefix \( v \) of \( w \), we have

\[
n\left( v \right) \geq n\left( v \right).
\]

The string \( w \) is balanced if it is left dominant and if

\[
n\left( w \right) = n\left( w \right).
\]

With this definition, we can complete the proof.
Programming Language Syntax

The syntax of programming languages is typically given by a context-free grammar written in Backus-Naur Form (BNF).

EXAMPLE.

\[
\begin{align*}
\langle unsigned \ real \rangle & \rightarrow \langle unsigned \ integer \rangle \\
& \quad \langle unsigned \ integer \rangle | \langle unsigned \ integer \rangle \ E \langle scale \ factor \rangle \\
\langle unsigned \ integer \rangle & \rightarrow \langle digit \rangle \ \{\langle digit \rangle\} \\
\langle digit \rangle & \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \\
\langle scale \ factor \rangle & \rightarrow \langle unsigned \ integer \rangle | \langle sign \rangle \langle unsigned \ integer \rangle \\
\langle sign \rangle & \rightarrow + | -
\end{align*}
\]
Pascal Assignment Statement

\[
\langle \text{assignment statement} \rangle \\
\quad \rightarrow \quad \langle \text{variable} \rangle := \langle \text{expression} \rangle \\
\quad \rightarrow \quad \langle \text{entire variable} \rangle := \langle \text{expression} \rangle \\
\quad \rightarrow \quad \langle \text{variable identifier} \rangle := \langle \text{expression} \rangle \\
\quad \rightarrow \quad \langle \text{identifier} \rangle := \langle \text{expression} \rangle \\
\quad \rightarrow \quad \text{time2go8} := \langle \text{expression} \rangle
\]

The last steps use the productions:

\[
\langle \text{identifier} \rangle \rightarrow \langle \text{letter} \rangle \{\langle \text{letter or digit} \rangle\}
\]

\[
\langle \text{letter or digit} \rangle \rightarrow \langle \text{letter} \rangle \mid \langle \text{digit} \rangle
\]

\[
\langle \text{letter} \rangle \rightarrow a \mid b \mid \ldots \mid z
\]

\[
\langle \text{digit} \rangle \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\]
Pascal Expression

\[ \langle \text{expression} \rangle \rightarrow \langle \text{simple expression} \rangle | \langle \text{relational operator} \rangle \langle \text{simple expression} \rangle \]

\[ \langle \text{relational operator} \rangle \rightarrow = | <> | < | <= | > = | > | \text{in} \]

\[ \langle \text{simple expression} \rangle \rightarrow \langle \text{term} \rangle | \langle \text{sign} \rangle \langle \text{term} \rangle | \langle \text{simple expression} \rangle \langle \text{adding operator} \rangle \langle \text{term} \rangle \]

\[ \langle \text{adding operator} \rangle \rightarrow + | - | \text{or} \]

\[ \langle \text{term} \rangle \rightarrow \langle \text{factor} \rangle | \langle \text{term} \rangle \langle \text{multiplying operator} \rangle \langle \text{factor} \rangle \]

\[ \langle \text{multiplying operator} \rangle \rightarrow * | / | \text{div} | \text{mod} | \text{and} \]

\[ \langle \text{factor} \rangle \rightarrow \langle \text{variable} \rangle | \langle \text{unsigned constant} \rangle | \langle \text{expression} \rangle | \langle \text{function designator} \rangle | \langle \text{set} \rangle | \text{not} \langle \text{factor} \rangle \]
Simplified CFG For Expressions

The terminal alphabet $\Sigma = \{ (, ), x, 9, +, \times, < \}$.

The nonterminal alphabet $V = \{ E, S, T, F \}$.

Here are the rules:

$$
G_E : \quad E \rightarrow S \mid S < S \\
S \rightarrow T \mid + T \mid S + T \\
T \rightarrow F \mid T \times F \\
F \rightarrow x \mid 9 \mid (E)
$$

Give a derivation for $x + 9 + (9 \times x) \times (x + 9)$

?  

Give the corresponding derivation tree

?  

Give the expression tree

?