Lecture 6 Outline
- Big Picture
- Context-free grammars (CFGs)
  - examples
  - definition
- Parse trees, ambiguity
- Chomsky Normal Form

Big Picture: What is Computation?
- DFAs are a nice model, but too limited.
  Some simple languages are not regular.
- Context-Free languages are in between regular
  and turing-recognizable languages, and are the
  subject of today.
- Eventually, we will define the Turing machine,
  which recognizes Turing-recognizable languages.
  The Turing machine will be the gold standard
  of a model of computation.

While Turing machines are our eventual goal,
context-free grammars are important because they
often come up, e.g. in compilers, and in modeling
natural languages like English.
Context-Free Grammars

What is a Context-Free Grammar?

Example: Simple model of English.

\[
\begin{align*}
\langle \text{sentence} \rangle & \to \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle \\
\langle \text{noun phrase} \rangle & \to \langle \text{article} \rangle \langle \text{bunch of adjectives} \rangle \langle \text{noun} \rangle \\
\langle \text{article} \rangle & \to \text{a} \mid \text{the} \\
\langle \text{bunch of adjectives} \rangle & \to \varepsilon \mid \langle \text{adjective} \rangle \langle \text{bunch of adjectives} \rangle \\
\langle \text{noun} \rangle & \to \text{girl} \mid \text{boy} \mid \text{flower} \mid \text{apple} \\
\langle \text{verb phrase} \rangle & \to \langle \text{verb} \rangle \langle \text{noun phrase} \rangle \\
\langle \text{verb} \rangle & \to \text{sees} \mid \text{eats} \mid \text{likes} \\
\langle \text{adjective} \rangle & \to \text{red} \mid \text{small} \mid \text{happy} \\
\end{align*}
\]

Formal Definition

A context-free grammar (CFG) is a 4-tuple \((V, \Sigma, R, S)\), such that:

1. \(V\) is a set of variables
2. \(\Sigma\) is a set of terminals (like letters)
3. \(R\) is a finite set of rules:
   \(R \subseteq V \times (V \cup \Sigma)^*\), where
   - \(A \to \omega\) is a rule, \(A\) is a variable and \(\omega\) is a string of variables and terminals, i.e., \(\omega \in (V \cup \Sigma)^*\).
   - \(S \in V\) is the start variable.
In the example, \( \langle \text{sentence} \rangle, \langle \text{noun phrase} \rangle, \langle \text{verb} \rangle \), etc. were variables, and girl, apple, red, happy, etc. were terminals. The start state \( S \) was \( \langle \text{sentence} \rangle \). The rules were abbreviated a bit:

\[
\langle \text{noun} \rangle \rightarrow \text{girl} \mid \text{boy} \mid \text{flour} \mid \text{apple}
\]

is an abbreviation for four different rules:

\[
\begin{align*}
\langle \text{noun} \rangle & \rightarrow \text{girl} \\
\langle \text{noun} \rangle & \rightarrow \text{boy} \\
\langle \text{noun} \rangle & \rightarrow \text{flour} \\
\langle \text{noun} \rangle & \rightarrow \text{apple}.
\end{align*}
\]

Definition. If \( u, v, w \) are strings over \( \{V, U, S, \} \), i.e. \( u, v, w \in \{V, U, S, \}^* \), and \( A \rightarrow w \) is a rule in \( R \), then we say

\[
\Delta uAv \Rightarrow uwv
\]

(\( uAv \) yields \( uwv \))

For example:

\[
\begin{align*}
\langle \text{sentence} \rangle & \Rightarrow \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle \\
& \quad \Rightarrow \langle \text{article} \rangle \langle \text{bunch of adj} \rangle \langle \text{verb phrase} \rangle \\
& \quad \Rightarrow \langle \text{article} \rangle \langle \text{adjective} \rangle \langle \text{bunch of adjectives} \rangle \langle \text{verb phrase} \rangle \\
& \quad \Rightarrow \langle \text{article} \rangle \langle \text{adjective} \rangle \langle \text{bunch of adj} \rangle \langle \text{verb} \rangle \langle \text{noun phrase} \rangle \\
& \Rightarrow \text{ } \cdots \\
& \Rightarrow \text{A happy girl sees a red flower.}
\end{align*}
\]

Then we write \( \langle \text{sentence} \rangle \Rightarrow^* \text{A happy girl sees a red flower.} \)
Definition. If \( u, v \in (V \cup US)^* \), we say
\( u \) derives \( v \), and write
\( u \Rightarrow v \)
if either \( u = v \), or there exist strings \( u_1, u_2, \ldots, u_n \)
such that
\( u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_n \Rightarrow v \).

Definition. Given a context-free grammar
\[ G = (V, \Sigma, R, S) \]
The **language** of \( G \) is defined as
\[ L(G) = \{ w \mid w \in \Sigma^* \text{ and } S \Rightarrow w \} \]
I.e., \( L(G) \) is the set of strings of terminals which can be derived from \( S \).

**Example 1**
\[ B = \{ 0^n 1^n \mid n \geq 0 \} \]
Grammar \( G \) such that \( L(G) = B \):
\[ S \rightarrow \varepsilon \mid 0S1 \]
For example, \( 000111 \in L(G) \), because
\[ S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 000111 \].

**Example 2**
If \( G \) is just
\[ S \rightarrow S \]
Then \( L(G) = \emptyset \), because the rule doesn't get anywhere—you can't get \( \varepsilon \) from \( S \) to derive a string of terminals.
Example 3

Consider the regular expression

\[00 (1U10)^*1\]

Then we can make a CFG \(G\) to recognize this regular expression:

\[S \rightarrow 00A1\]
\[A \rightarrow \varepsilon | AA | 1 | 10.\]

Definition. A set of strings \(L\) is a context-free language if \(L = L(G)\) for some context-free grammar \(G\).

Lemma. If \(L\) is regular, then it is also context-free.

Example 4

Let \(C = \{w \mid w\ has\ the\ same\ \#\ of\ Os\ and\ 1s\}\). This is recognized by the CFG

\[S \rightarrow 0S1 | 1S0 | SS | \varepsilon.\]

To derive \(001101,\)

\[S \Rightarrow 0S1 \Rightarrow 0S51 \Rightarrow 00S1S1 \Rightarrow 001S1 \Rightarrow 001101.\]

But there is another way to derive the same string!

\[S \Rightarrow SS \Rightarrow 0S1S \Rightarrow 00S1S1 \Rightarrow 001S1S \Rightarrow 001101.\]
We can represent the two different derivations as Parse trees.

**Derivation #1**

**Derivation #2**

Depth-first traversal of leaves:

**Derivation #1:**

```
S 1
O S 1
O S 1
O S 1
S 1
E
```

```
Depth-first traversal of leaves:
O O E 1 1 0 1
= 0 0 1 1 0 1.
```

These derivations are considered to be different because they have different parse trees. Note that there can be multiple derivations with the same parse tree—left-most derivation vs. right-most derivation, for example.

**Definition.** A CFG $G$ is ambiguous if $\exists w \in L(G)$ such that $w$ can be derived by two different parse trees. Equivalently, $w$ has two distinct left-most derivations.

(A left-most derivation means you are only allowed to apply a rule at the left-most variable in a string.)
Chomsky Normal Form

Definition.
A CFG G is in Chomsky normal form if every rule is of one of the following kinds:

\[ A \rightarrow BC \quad \text{where } B, C \in V \setminus \{S\} \quad (A \in V) \]
or
\[ A \rightarrow a \quad \text{where } A \in V, a \in \Sigma \]
or
\[ S \rightarrow \epsilon. \]

Next Class:

→ Pumping Lemma for CFGs, using Chomsky normal form

→ In order to prove the Pumping Lemma, we'll first show that every grammar can be converted to Chomsky Normal Form.