Nonregular Languages
Nonregular Languages: Intuition

- Regular languages correspond to problems that can be solved with finite memory
  - Only need to remember one of finitely many things
- Nonregular languages correspond to problems that cannot be solved with finite memory
  - May need to remember one of infinitely many different things
Picking the Right Strings

**Theorem.** The language $C = \{ uu \mid u \in \{0,1\}^* \}$ is not regular.

**Proof.**
Using Closure and Known Non-Regular Languages

**Theorem.** The language $L = \{ w \mid w \text{ has an equal number of 0s and 1s} \}$ is not regular.

**Proof.**
Questions

- Do all regular languages satisfy the pumping lemma?
- If a language satisfies the pumping lemma does that mean it is regular?
Context-Free Languages
Extending Our Reach

- Finite automata are *language recognition devices* and regular expressions are *language generating devices*

- Finite automata *recognize* and regular expressions *generate* an important but limited class of languages

- We are now ready to move beyond regular languages
Regular Languages
- Finite Automaton
  \[ 1^*0^*, (0 \cup 1)^*0 \]

Context-free Languages
- Push-down Automaton
  \[ 0^n1^n2^n, \quad 0^n1^n, \quad ww^R \]

Decidable Languages
- Decidable by Turing Machine
  \[ 0^n1^n2^n \]

Recursively-Enumerable Languages
- Recognized by Turing Machines

All Languages

We are here
Context-Free Grammar

- New formalism to specify languages, called *context-free languages*
- First used in the study of human languages
- Applications in specification and compilation of programming languages
- Can be used to generate some nonregular languages like $0^n1^n$, $ww^R$
A Sample Context-Free Grammar

- Consists of a collection of substitution rules, also called productions
- Each rule has a left-hand side consisting of a single variable and a right-hand side consisting of variables and terminals
- Start variable: by convention on the left-hand side of the topmost rule

\[
S \rightarrow 0S1 \\
S \rightarrow \# 
\]
Derivations to Generate Strings

A sequence of substitutions starting with the start variable and ending in a string of terminals is a derivation. For example, a derivation of $000#111$ using this grammar.

\[
S \rightarrow 0S1 \\
S \rightarrow \#
\]

Derivation: \( S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 000#111 \)
Parse Trees

You can also represent derivations of a context-free grammar using parse trees

$S \rightarrow 0S1$

$S \rightarrow \#$
Language of the Grammar

- All strings generated by a grammar constitute the **language of the grammar**
- Any language that can be generated by some context-free grammar is called a **context-free language**

\[
S \rightarrow 0S1 \\
S \rightarrow \# 
\]

**Question.** What is the language \(L(G)\) of the grammar \(G\) above?
Fragment of the English language

A grammar for the English language tells us whether a particular sentence is well formed or not. Here is such an example.

\[
\text{<Sentence>} \rightarrow \text{<NounPhrase> <VerbPhrase>}
\]
\[
\text{<NounPhrase>} \rightarrow \text{<Article> <NounUnit>}
\]
\[
\text{<NounUnit>} \rightarrow \text{<Noun> | <Adjective> <NounUnit>}
\]
\[
\text{<VerbPhrase>} \rightarrow \text{<Verb> <NounPhrase>}
\]
\[
\text{<Article>} \rightarrow \text{a | the}
\]
\[
\text{<Adjective>} \rightarrow \text{big | small | black | green | colorless}
\]
\[
\text{<Noun>} \rightarrow \text{dog | cat | mouse | bug | ideas}
\]
\[
\text{<Verb>} \rightarrow \text{loves | chases | eats | sleeps}
\]

Some generated sentences:

*The black dog loves the small cat*
*A cat chases a mouse*
*The colorless bug chases the green ideas*
Syntax of a programming language

<program> → <block>
<brack> → { <command-list> }
<command-list> → ε
<command-list> → <command> <command-list>
<command> → <block>
<command> → <assignment>
<command> → <one-armed-conditional>
<command> → <two-armed-conditional>
<command> → <while-loop>
<assignment> → <var> := <expr>
<one-armed-conditional> → if <expr> <command>
<two-armed-conditional> → if <expr> <command> else <command>
<while-loop> → while <expr> <command>

Possible generated program

{ x := 4
  while x > 1
    x := x - 1 }


Context-Free Grammars

A context-free grammar $G$ is a quadruple $(V, \Sigma, R, S)$, where

- $V$ is a finite set called **variables**, 
- $\Sigma$ is a finite set, disjoint from $V$, called the **terminals**  
- $R$ is a finite subset of $V \times (V \cup \Sigma)^*$ called **rules**, and
- $S$ (the **start symbol**) is an element of $V$
- For any $A \in V$ and $u \in (V \cup \Sigma)^*$, we write $A \to u$ whenever $(A, u) \in R$. 

$S \to 0S1$

$S \to \#$
The Language of a Grammar

If \( u, v, w \in (V \cup \Sigma)^* \) and \( A \rightarrow w \) is a rule, then we say \( uAv \) yields \( uwv \) and write

\[
uAv \Rightarrow uwv
\]

We say \( u \) derives \( v \), written, \( u \Rightarrow^* v \) if

\[
u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v
\]

The language of the grammar \( G \) is

\[
L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}
\]
Exercise with CFG

Let’s design context-free grammar for the following languages

1. \( L = \{0^n 1^n \mid n \in \mathbb{N}\} \)

2. \( L = \{w \in \{a, b\}^* \mid |w| \text{ is even}\} \)

3. \( L = \{w \in \{0, 1\}^* \mid w = w^R\} \)
Arithmetic Expressions & Parse Trees

Consider $G = (V, \Sigma, R, S)$, where

$V = \{<EXPR>\}$,

$\Sigma = \{a, +, \times, (, )\}$ and the rules are

$$<EXPR> \rightarrow <EXPR>+<EXPR> | <EXPR>\times<EXPR> | (<EXPR>) | a$$

**Question.** What is the parse tree for $a+a \times a$?