Overview of CFGs

• Why? -- overcoming FA/RE limitations

• CFG Basics

• Programming Language examples
  o Algebraic expressions
  o If Statements

• Ambiguous structure
  o Expressing it
  o Eliminating it
Regular Languages

• Regular Languages are Useful
  o Numbers
  o Variable Names

• And Have Several Forms
  o RE, FA, NFA(-L)
  o Lex, a Genuine Software Tool

• But …
Limitations of Regular Languages

• No Balanced Delimiters

• No (other) Nested Promises, like…
  
  ➢ C: if ( ) { };
  ➢ Unix: if … fi

• “Limited Representational Power”
Lower Bound on How Many States

- (ab)* requires at least 2 states
  - some strings are accepted, …
  - some not.

- a*b* requires at least 2 states
  - For some strings, adding a is accepted, …
  - some not.

- Some languages require at least 3 (or k) states.

- Some languages require more and more states.
“Distinguishable” Strings

• Motivation: show the weakness of regular languages

• The idea of “Distinguishing”:

   It’s a relationship among 4 things:
   strings \( x, y \) and \( z \) and language \( L \).

   \( z \) distinguishes \( x \) and \( y \) with respect to \( L \)

   \( xz \) and \( yz \) differ in \( L \)-membership;

   (i.e., one is \( \text{in} \) and the other is \( \text{out} \).)
Example of Distinguishability

• Let $L = \{ab\}^* = \emptyset, ab, abab, ...$

• Let $x = aba$ and $y = abab$

• Let $z = ba$

• Then $z$ distinguishes $x$ and $y$ wrt $L$.

• Why?

• Also $\emptyset$ distinguishes $x$ and $y$ wrt $L$. 
“Distinguishable” Strings

- Distinguishable:

  \[ x: y: L: \]
  \[ x \text{ and } y \text{ are distinguishable with respect to } L \]
  \[ z: z \text{ distinguishes them} \]

- Distinguished by \[ \curvearrowright \]

  \[ x \curvearrowright = x \curvearrowright L \text{ but } y \curvearrowright = y \curvearrowright L \text{ (or vice versa)} \]

- Now \[ \curvearrowleft \text{ or in the future (non-\curvearrowleft)} \]
$L = \{a^n b^n\}$

- Distinguishable strings need different states of a DFA.

- $aaa$ and $aaaaa$ are distinguishable with respect to $L$ (use $bbb$)

- $a^i$ and $a^j$ are distinguishable whenever $i \neq j$.

- $\{a^n b^n\}$ needs a state for each integer.

- so $L$ has no Finite Automaton
  (suppose it did; how many states?)

- No DFA \quad No FA \quad No RE
CFGs

• Context-free grammar is a 4-tuple,

\[ G = (V, \Sigma, S, P) \]

• Example:

\[ G_1 = (\{S\}, \{a, b\}, S, P) \]

where \[ P = \{S \rightarrow aSb, S \rightarrow \epsilon\} \]
Deriving Strings

• $\epsilon \in \Sigma^*$

• $S \in aSb \in aaSbb \in aaaSbbb \in aaabbb$

• $S \in \epsilon \in aaabbb$

• $n : S \in \epsilon \in a^n b^n$
Language Formalities

\[ G = (V, \epsilon, S, P) \]

\[ \mathcal{L}(G) = \{ x \mid S \rightsquigarrow^* x \} \]

\[ G_1 = (\{S\}, \{a, b\}, S, P) \]

where \( P = \{ S \rightsquigarrow aSb, S \rightsquigarrow \epsilon \} \)

\[ \mathcal{L}(G_1) = \{ a^n b^n \} \]
$$\mathcal{L}(FA) = \mathcal{L}(RE) \cap \mathcal{L}(CFG)$$

- $a^n b^n$ is representable by CFG, not FA, RE

- and CFGs can do the 3 ops of RE
  
  - concatenation: $A \cdot B \cdot C$
  - alternation: $A \cup B \cup C$
  - closure: $A \cdot A \cdot B \cdot C$

- Therefore $\mathcal{L}(RE) \cap \mathcal{L}(CFG)$.
Example

- Write a CFG for \((a+b*c)d)*\n
\[
S \rightarrow TS | \epsilon \\
T \rightarrow Ud \\
U \rightarrow a | V \\
V \rightarrow bV | c
\]
Balanced Parentheses

- $G_3 = \{\{S\}, \{a, b\}, S, P\}$ where $P$ contains

  $S \rightarrow \varepsilon$
  $S \rightarrow S \ S$
  $S \rightarrow ( \ S \ )$

- A Derivation in $G_3$

  $S \rightarrow ( S ) \rightarrow ( S S ) \rightarrow (( S ) S )$
  $\rightarrow (( S)( S )) \rightarrow ((( ) ( S )))$
  $\rightarrow ((( )))$
Algebraic Expressions

• $E \rightarrow E + E \mid E \mid a \mid b \mid c$

• Generates the right strings, but...

• $E + E \rightarrow E$ (e.g.) is ambiguous
Fixing the Grammar

• Remove ambiguity by one-way branching

\[ E \rightarrow E + T \mid E \rightarrow T \mid T \]
\[ T \rightarrow a \mid b \mid c \]
Fixing the Grammar, part II

- Fix precedence with an extra level

\[
E \rightarrow E + T \mid T \\
T \rightarrow T \rightarrow F \mid F \\
F \rightarrow a \mid b \mid c
\]
Fixing the Grammar, part III

• Parentheses overcome precedence

\[
\begin{align*}
E &\to E + T | T \\
T &\to T \cdot F | F \\
F &\to (E) | a | b | c
\end{align*}
\]