[Audience looks around]
“What is going on? There must be some context we’re missing”
• Subsets of strings are called **languages**

• Examples:
  - $\Sigma^* = \text{All strings over alphabet } \Sigma$
  - Palindromes over $\Sigma$
  - Binary strings that don’t have a 0 after a 1
  - Binary strings with an equal # of 0’s and 1’s
  - Legal variable names in Java/C/C++
  - Syntactically correct Java/C/C++ programs
  - Valid English sentences
Regular expressions over $\Sigma$

• **Basis:**
  - $\emptyset$ is a regular expression (could also include $\emptyset$)
  - $a$ is a regular expression for any $a \in \Sigma$

• **Recursive step:**
  If $A$ and $B$ are regular expressions then so are:
  - $A \cup B$
  - $AB$
  - $A^*$
Last class: Regular Expression is a “pattern”

ε matches the empty string

a matches the one character string a

A ∪ B matches all strings that either A matches or B matches (or both)

AB matches all strings that have a first part that A matches followed by a second part that B matches

A* matches all strings that have any number of strings (even 0) that A matches, one after another

Yields a language = the set of strings matched by the regular expression
## Last class: Examples

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>001*</td>
<td>{00, 001, 0011, 00111, ...}</td>
</tr>
<tr>
<td>0<em>1</em></td>
<td>{Binary strings with any number of 0s followed by any number of 1s}</td>
</tr>
<tr>
<td>((0 \cup 1)\ 0\ (0 \cup 1)\ 0)</td>
<td>{0000, 1000, 0010, 1010}</td>
</tr>
<tr>
<td>(0<em>1</em>)*</td>
<td>{All binary strings}={0,1}*</td>
</tr>
<tr>
<td>(0 \cup 1)*</td>
<td>{All binary strings}={0,1}*</td>
</tr>
<tr>
<td>(0 \cup 1)* 0110 (0 \cup 1)*</td>
<td>{All binary strings containing substring 0110}</td>
</tr>
</tbody>
</table>
Regular Expressions in Practice

• Used to define the *tokens* of a programming language
  – legal variable names, keywords, etc.

• Used in `grep`, a program that does pattern matching searches in UNIX/LINUX

• We can use regular expressions in programs to process strings!
Regular Expressions in Java

Pattern \( p = \text{Pattern.compile}("a*b") \);
Matcher \( m = p.\text{matcher}("aaaaab") \);
boolean \( b = m.\text{matches}() \);

\[01\] a 0 or a 1  \^ start of string  \$ end of string
\[0-9\] any single digit  \. period  \, comma  \- minus
\. any single character
ab a followed by b \( (AB) \)
\(a|b\) a or b \( (A \cup B) \)
a? zero or one of a \( (A \cup \varepsilon) \)
a* zero or more of a \( A^* \)
a+ one or more of a \( AA^* \)

• e.g. \( ^[\-+]?[0-9]*(\.\|\||,)?[0-9]+$ \)
  General form of decimal number e.g. 9.12 or -9,8 (Europe)
Examples

- All binary strings that have an even # of 1’s

\[ x (0^*10^*10^*)^* \]
\[ x (011)^* \]
\[ x (0^*10^*1)^* \]
\[ x (011)^* \]
\[ x (0^*10^*1)^* \]
Examples

- All binary strings that have an even # of 1's

  e.g., 0* (10*10*)*
Examples

• All binary strings that have an even # of 1’s
  
  e.g., \(0^* (10^*10^*)^*\)

• All binary strings that don’t contain 101

\[101 \times 0^* (00^*1)^*\]

\[110 \times 0^* (00^*1)^*\]

\[01 \times 0^* (00^*1)^*\]

\[10001 \times 0^* (00^*1)^*\]
Examples

- All binary strings that have an even # of 1’s
  
  e.g., \(0^* (10^*10^*)^*\)

- All binary strings that *don’t* contain 101
  
  e.g., \(0^* (1 \cup 000^*)^* 0^*\)
  at least two 0s between 1s
Limitations of Regular Expressions

- Not all languages can be specified by regular expressions
- Even some easy things like
  - Palindromes
  - Strings with equal number of 0’s and 1’s
- But also more complicated structures in programming languages
  - Matched parentheses
  - Properly formed arithmetic expressions
  - etc.
Context-Free Grammars

• A Context-Free Grammar (CFG) is given by a finite set of substitution rules involving
  – Alphabet $\Sigma$ of *terminal symbols* that can’t be replaced
  – A finite set $V$ of *variables* that can be replaced
  – One variable, usually $S$, is called the *start symbol*

• The substitution rules involving a variable $A$, written as
  $$A \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$$
  where each $w_i$ is a string of variables and terminals
  – that is $w_i \in (V \cup \Sigma)^*$
How CFGs generate strings

• Begin with “S”

• If there is some variable A in the current string, you can replace it by one of the w’s in the rules for A
  – $A \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$
  – Write this as $xAy \Rightarrow xwy$
  – Repeat until no variables left

• The set of strings the CFG describes are all strings, containing no variables, that can be *generated* in this manner after a finite number of steps
Example Context-Free Grammars

Example:  \[ S \rightarrow 0S \mid S1 \mid \varepsilon \]

\[
S \Rightarrow 0S \Rightarrow 0S1 \Rightarrow 010 = 100\\
S \Rightarrow 0S \Rightarrow 0S0 \Rightarrow 000
\]
Example Context-Free Grammars

Example: \[ S \rightarrow 0S \mid S1 \mid \varepsilon \]

\[ 0^*1^* \]

any # of 0s followed by any # of 1s
Example Context-Free Grammars

Example: \[ S \rightarrow 0S \mid S1 \mid \varepsilon \]

\[ 0^*1^* \]

Example: \[ S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon \]

\[ S \Rightarrow 0S0 \Rightarrow 000 \Rightarrow 001 \varepsilon 100 \Rightarrow 0011100 \]
Example Context-Free Grammars

Example: \[ S \rightarrow 0S | S1 | \varepsilon \]

0*1*

Example: \[ S \rightarrow 0S0 | 1S1 | 0 | 1 | \varepsilon \]

The set of all binary palindromes
Example Context-Free Grammars

Grammar for \( \{0^n1^n : n \geq 0\} \)
(i.e., matching \(0^*1^*\) but with same number of 0's and 1's)

\[ S \rightarrow 0S1 | \epsilon \]
Example Context-Free Grammars

Grammar for \( \{0^n1^n: n \geq 0\} \)
(i.e., matching \( 0^*1^* \) but with same number of 0’s and 1’s)

\[
S \rightarrow 0S1 \mid \varepsilon
\]
Example Context-Free Grammars

Grammar for \( \{0^n1^n : n \geq 0\} \)
(i.e., matching \( 0^*1^* \) but with same number of 0’s and 1’s)

\[
S \rightarrow 0S1 \mid \varepsilon
\]

Grammar for \( \{0^n1^{2n} : n \geq 0\} \)
Example Context-Free Grammars

Grammar for \( \{0^n1^n: n \geq 0\} \)
(i.e., matching \( 0^*1^* \) but with same number of 0’s and 1’s)

\[
S \rightarrow 0S1 \mid \varepsilon
\]

Grammar for \( \{0^n1^{2n}: n \geq 0\} \)

\[
S \rightarrow 0S11 \mid \varepsilon
\]
Example Context-Free Grammars

Grammar for \( \{0^n1^n : n \geq 0\} \)
(i.e., matching \(0^*1^*\) but with same number of 0’s and 1’s)

\[
S \rightarrow 0S1 | \varepsilon
\]

Grammar for \( \{0^n1^{n+1}0 : n \geq 0\} \)

\[
S \rightarrow AB
A \rightarrow 0A1 | \varepsilon
B \rightarrow 10
\]

\[
S \rightarrow A10
A \rightarrow 0A1 | \varepsilon
\]
Example Context-Free Grammars

Grammar for \( \{0^n1^n: n \geq 0\} \)
(i.e., matching \(0^*1^*\) but with same number of 0’s and 1’s)

\[ S \rightarrow 0S1 | \varepsilon \]

Grammar for \( \{0^n1^{n+1}0: n \geq 0\} \)

\[ S \rightarrow A \ 10 \]
\[ A \rightarrow 0A1 | \varepsilon \]
Example Context-Free Grammars

Example: \[ S \rightarrow (S) \mid SS \mid \varepsilon \]

\[ S \Rightarrow (S) \Rightarrow SS \Rightarrow (S)S \]
\[ \Rightarrow ((S)(S)) \Rightarrow ((1)(S)) \Rightarrow ((U)(U)) \]
Example Context-Free Grammars

Example: \[ S \rightarrow (S) \mid SS \mid \varepsilon \]

The set of all strings of matched parentheses
Example Context-Free Grammars

Binary strings with equal numbers of 0s and 1s
(not just $0^n1^n$, also 0101, 0110, etc.)

$$S \rightarrow 0S1 \mid 1S0 \mid \varepsilon \mid SS$$

0110
Example Context-Free Grammars

Binary strings with equal numbers of 0s and 1s (not just $0^n1^n$, also 0101, 0110, etc.)

$$S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$$

An easy structural induction can show that everything generated by $S$ has an equal # of 0s and 1s

Why does this generate all such strings?