Today's learning goals

- Decide whether or not a string is described by a given regular expression
- Design a regular expression to describe a given language
- Convert between regular expressions and automata
- Explore the limits of regular sets

Reminder: Exam 1 is Tuesday April 25 see seating map!
Inductive application of closure

R is a regular expression over $\Sigma$ if

1. $R = a$, where $a \in \Sigma$
2. $R = \varepsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$, where $R_1, R_2$ are themselves regular expressions
5. $R = (R_1 \circ R_2)$, where $R_1, R_2$ are themselves regular expressions
   \[ L(R_1 \circ R_2) = L(R_1) \circ L(R_2) = \{xy | x \in L(R_1), j \in L(R_2) \} \]
6. $(R_1^*)$, where $R_1$ is a regular expression.

$\Sigma$ is shorthand for $(0 \cup 1)$ if $\Sigma = \{0,1\}$, Parentheses may be omitted, $R^+$ means $RR^*$, $R^k$ means $R$ concatenated with itself $k$ times.
Syntax → Languages

The language described by a regular expression, $L(R)$:

1. $L(0 \cup 1) = \{0, 1\}$
2. $L(\varepsilon) = \{\varepsilon\}$
3. $L(\emptyset) = \{\}$
4. $L((R_1 \cup R_2))$
5. $L((R_1 \circ R_2))$
6. $L(R^*)$

Abbreviations: $\Sigma$ use $\cup$ if $\Sigma = \{0, 1\}$
Which of the following strings is **not** in the language described by

\[ L = \left( ((00)^* (11)) \cup 01 \right)^* \]

A. 00  
B. 01  
C. 1101  
D. \(\varepsilon\)  
E. I don't know

\[
\begin{align*}
L((00)^*(11) \cup 01)^* &= \left( \bigcup_{n \geq 0} \{00\}^{n(11)} \cup \{01\} \right)^* \\
&= \{0, 11, 0011, 000011, \ldots\}^* \\
\text{for all values of } D,
\end{align*}
\]

String in lang: 00110011, 00110000011
Let $L$ be the language over $\{a,b\}$ described by the regular expression

$$((a \cup \emptyset) \ b^*)^*$$

Which of the following is not true about $L$?

A. Some strings in $L$ have equal numbers of $a$'s and $b$'s
B. $L$ contains the string aaaaaaa
C. $a$'s never follow $b$'s in any string in $L$
D. $L$ can also be represented by the regular expression $(ab^*)^*$
E. More than one of the above.
Regular expressions in practice

- **Compilers**: first phase of compiling transforms Strings to Tokens **keywords, operators, identifiers, literals**
  - One regular expression for each token type

- **Other software tools**: grep, Perl, Python, Java, Ruby, …
"Regular = regular"

Theorem: A language is regular if and only if some regular expression describes it.

Lemma 1.55: If a language is described by a regular expression, then it is regular.

Lemma 1.60: If a language is regular, then it is described by some regular expression.
L(R) to NFA (to DFA)

• Idea: basic regular expressions are easy to implement as DFA, for inductive step of definition, use closure under regular operations.

• E.g.: build NFA recognizing the language described by

\[ (00 \cup 11)^* \]
DFA to regular expression

- Idea: use intermediate model **GNFA** whose labels are regular expressions

E.g.: build regular expression describing language recognized by

\[ (0 + 1)^* \]