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
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, $L(R)$:

- $L((0 \cup 1) \cup 1) = \{0, 1\}$

- $L((0 \cup 1) \cup 0) = L(0) \cup L(1) = \{0\} \cup \{1\} = \{0, 1\}$

- $L((\Sigma \Sigma \Sigma \Sigma)^*) = \{w \in \Sigma^* | 1w1w4\}^* = \{w \in \Sigma^* | 1w1w4\}^{*k, o}$

- $L(1^*00) = \{xyz | x \in L(1^*), y \in L(\emptyset), z \in L(0)\}$

**Shorthand: $\emptyset$ may be dropped**

1. $R = a$, where $a \in \Sigma$
2. $R = \varepsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$
5. $R = (R_1 \circ R_2)$
6. $(R_1^*)$
L(R)

Which of the following strings is **not** in the language described by

\[
L = \{(00)^{\ast}(11) \cup 01\}^{\ast}
\]

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

\[
A = \{01, (00)^{n}11 / n \geq 0 \}
\]

every run \(00\)s is followed by 11
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. (if build NFA, then use subset construction)

**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

Lemma 1.60, page 69

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

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

\[ 1 (0 \cup 1)^* \]