Today's learning goals

- Use NFAs to prove closure properties of regular languages
- Describe regular languages using Regular Expressions
Differences between NFA and DFA

- **DFA**: unique computation path for each input
- **NFA**: allow several (or zero) alternative computations on same input
  - $\delta(q,x)$ may specify *more than one* possible next states
  - $\epsilon$ transitions allow the machine to *transition between states spontaneously*, without consuming any input symbols
  - computation can *get stuck* at some state, if there's a missing arrow
Formal definition of NFA

A **nondeterministic finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\) where

1. \(Q\) is a finite set called the states
2. \(\Sigma\) is a finite set called the alphabet
3. \(\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)\) is the transition function
4. \(q_0 \in Q\) is the start state
5. \(F \subseteq Q\) is the set of accept states.
Simulating NFA with DFA

For any language recognized by an NFA, there is a DFA that recognizes this language.

**Proof:**

**Given** \( A \), a language recognized by \( N = (Q, \Sigma, \delta, q_0, F) \) a NFA

**WTS** there is some DFA \( M \) with \( L(M) = A \)

**Construction**

**Correctness**

**Conclusion**
Application

A language $A$ over $\Sigma$ is **regular** if and only if

- it is recognized by a DFA
- it is recognized by a NFA

To prove that the class of regular languages is closed under operation $\ldots$:

Let $A$ be a regular language, so recognized by DFA $M$. Build a **NFA** that recognizes the result of $\ldots$ on $A$. Conclude this result is also a regular language.
The regular operations  Sipser Def 1.23 p. 44

For A, B languages over same alphabet, define:

\[ A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \]
\[ A \circ B = \{ xy \mid x \in A \text{ and } y \in B \} \]
\[ A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A \} \]

How can we prove that the concatenation of two regular languages is a regular language?
Concatenation

• "Guess" some stage of input at which switch modes

Given $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ build

$N = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$ with $\delta$...
Concatenation

\[ \delta(q, x) = \begin{cases} ? & \text{if } q \text{ is in } Q_1, x \text{ is in } \Sigma \\ ? & \text{if } q \text{ is in } Q_2, x \text{ is in } \Sigma \end{cases} \]
Concatenation

\[ \delta(q, x) = \begin{cases} \delta_1(q, x) & \text{if } q \text{ is in } Q_1, x \text{ is in } \Sigma \\ \delta_2(q, x) & \text{if } q \text{ is in } Q_2, x \text{ is in } \Sigma \end{cases} \]
Concatenation

\[ \delta( q, x ) = \begin{cases} 
\{ \delta_1(q,x) \} & \text{if } q \text{ is in } Q_1, x \text{ is in } \Sigma \\
\{ \delta_2(q,x) \} & \text{if } q \text{ is in } Q_2, x \text{ is in } \Sigma \\
\{ q_2 \} & \text{if } q \text{ is in } F_1, x = \varepsilon \\
\emptyset & \text{otherwise}
\end{cases} \]

Correctness proof in the book (page 61)
Star operation

Given $M = (Q, \Sigma, \delta, q, F)$, build

$$N = (Q \cup \{s\}, \Sigma, \delta, s, F \cup \{s\})$$

and $\delta(q, x) = \ldots$

*Construction in the book (page 63)*
Regular languages

To prove that a set of strings over the alphabet $\Sigma$ is regular,

- Build a **DFA** whose language is this set.
- Build an **NFA** whose language is this set.
- Use the **closure properties** of the class of regular languages to construct this set from others known to be regular.
  - Union
  - Intersection
  - Complementation
  - Concatenation
  - Flip bits
  - Kleene star
Inductive application of closure

R is a **regular expression** over Σ 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.

Watch out for overloaded symbols!
Regular expressions

Conventions:
• \( \Sigma \) is shorthand for \((0 \cup 1)\) if \( \Sigma = \{0,1\} \)
• Parentheses may be omitted
  • Precedence: star, then concatenation, then union
• \( R^+ \) is shorthand for \( RR^* \), \( R^k \) is shorthand for \( R \) concatenated with itself \( k \) times
• Circle indicated concatenation may be omitted

Which of the following is **not** a regular expression over \( \{0,1\} \)?

A. \( (\Sigma\Sigma\Sigma)^* \)  
B. \( (\Sigma \cap 1) \)  
C. \( (1^*\emptyset 0) \)  
D. \( \varepsilon\varepsilon \)  
E. I don't know
From RegEx to Languages

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

- \( L(a) = \{a\} \) (for all \( a \) in \( \Sigma \))
- \( L(\varepsilon) = \{\varepsilon\} \)
- \( L(\emptyset) = \emptyset \)
- \( L(R_1 \cup R_2) = \{w \mid w \text{ in } L(R_1) \text{ or } w \text{ in } L(R_2)\} \)
- \( L(R_1 \circ R_2) = \{w_1w_2 \mid w_1 \text{ in } L(R_1) \text{ and } w_2 \text{ in } L(R_2)\} \)
- \( L(R^*) = L(R)^* \)
Which of the following strings is **not** in the language described by
\[
( (00)^*(11) ) \cup 01
\]^* 

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

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 U 11)*
DFA to regular expression

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

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

Lemma 1.60, page 69
Formal definition of GNFA

- A GNFA is a 5-tuple \((Q, \Sigma, \delta, s, f)\) where
  - \(Q\) is a finite set of states
  - \(\Sigma\) is a finite alphabet
  - \(s\) and \(f\) are distinct elements of \(Q\)
  - \(\delta: (Q \setminus \{f\}) \times (Q \setminus \{s\}) \to \text{RegEx}(\Sigma)\)

- Idea:
  - GNFA can go from \(q_1\) to \(q_2\) reading any string from \(L(\delta(q_1, q_2))\)
  - No transitions pointing to \(s\)
  - No transitions leaving \(f\)
Formal definition of GNFA

- A GNFA is a 5-tuple \((Q,\Sigma,\delta,s,f)\) where
  - \(Q\) is a finite set of states
  - \(\Sigma\) is a finite alphabet
  - \(s\) and \(f\) are distinct elements of \(Q\)
  - \(\delta: (Q \setminus \{f\}) \times (Q \setminus \{s\}) \rightarrow \text{RegEx}(\Sigma)\)

- Idea:
  - GNFA can go from \(q_1\) to \(q_2\) reading any string from \(L(\delta(q_1,q_2))\)
  - No transitions pointing to \(s\)
  - No transitions leaving \(f\)
"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
Next time

Question: Are there languages that are not regular?

Can you come up with a language that cannot be described by any of:
- DFA
- NFA
- GNFA
- Regular Expression