15-453
FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY
A non-deterministic finite automaton (NFA) is a 5-tuple \( N = (Q, \Sigma, \delta, Q_0, F) \)

- \( Q \) is the set of states (finite)
- \( \Sigma \) is the alphabet (finite)
- \( \delta : Q \times \Sigma \varepsilon \to 2^Q \) is the transition function
- \( Q_0 \subseteq Q \) is the set of start states
- \( F \subseteq Q \) is the set of accept states

* \( 2^Q \) is the set of subsets of \( Q \) and \( \Sigma \varepsilon = \Sigma \cup \{\varepsilon\} \)
Let \( w \in \Sigma^* \) and suppose \( w \) can be written as \( w_1 \ldots w_n \) where \( w_i \in \Sigma \varepsilon \) (\( \varepsilon \) is viewed as representing the empty string)

Then \( N \) accepts \( w \) if there are \( r_0, r_1, \ldots, r_n \in Q \) such that

1. \( r_0 \in Q_0 \)
2. \( r_{i+1} \in \delta(r_i, w_{i+1}) \) for \( i = 0, \ldots, n-1 \), and
3. \( r_n \in F \)

\[ L(N) = \text{the language of machine } N = \text{set of all strings machine } N \text{ accepts} \]

A language \( L \) is recognized by an NFA \( N \) if \( L = L(N) \).
FROM NFA TO DFA

Input: NFA \( N = (Q, \Sigma, \delta, Q_0, F) \)

Output: DFA \( M = (Q', \Sigma, \delta', q_0', F') \)

\[
Q' = 2^Q
\]

\[
\delta' : Q' \times \Sigma \rightarrow Q'
\]

\[
\delta'(R, \sigma) = \bigcup_{r \in R} \varepsilon(\delta(r, \sigma))
\]

\[
q_0' = \varepsilon(Q_0)
\]

\[
F' = \{ R \in Q' \mid f \in R \text{ for some } f \in F \}
\]

For \( R \subseteq Q \), the \( \varepsilon \)-closure of \( R \), \( \varepsilon(R) = \{ q \text{ that can be reached from some } r \in R \text{ by traveling along zero or more } \varepsilon \text{ arrows} \} \)
RLs ARE CLOSED UNDER STAR

Star: $A^* = \{ s_1 \ldots s_k \mid k \geq 0 \text{ and each } s_i \in A \}$

Let $M$ be a DFA, and let $L = L(M)$

Can construct an NFA $N$ that recognizes $L^*$
REGULAR LANGUAGES ARE CLOSED UNDER THE REGULAR OPERATIONS

- **Union:** $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

- **Intersection:** $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

- **Negation:** $\neg A = \{ w \in \Sigma^* \mid w \notin A \}$

- **Reverse:** $A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \}$

- **Concatenation:** $A \cdot B = \{ vw \mid v \in A \text{ and } w \in B \}$

- **Star:** $A^* = \{ w_1 \ldots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$
THE PUMPING LEMMA FOR REGULAR LANGUAGES and REGULAR EXPRESSIONS
WHICH OF THESE ARE REGULAR?

\[ B = \{0^n1^n \mid n \geq 0\} \]

\[ C = \{ w \mid w \text{ has equal number of occurrences of } 01 \text{ and } 10 \} \]

\[ D = \{ w \mid w \text{ has equal number of } 1s \text{ and } 0s\} \]
THE PUMPING LEMMA

Let $L$ be a regular language with $|L| = \infty$

Then there is a positive integer $P$ s.t.

if $w \in L$ and $|w| \geq P$

then can write $w = xyz$, where:

1. $|y| > 0$ ($y$ isn’t $\varepsilon$)
2. $|xy| \leq P$
3. For every $i \geq 0$, $xy^i z \in L$

Why is it called the pumping lemma? The word $w$ gets PUMPED into something longer…
Proof: Let M be a DFA that recognizes L

Let P be the number of states in M

Assume \( w \in L \) is such that \( |w| \geq P \)

We show: \( w = xyz \)

1. \( |y| > 0 \)
2. \( |xy| \leq P \)
3. \( xy^iz \in L \) for all \( i \geq 0 \)

There must be \( j \) and \( k \) such that \( j < k \leq P \), and \( r_j = r_k \) (why?) (Note: \( k - j > 0 \))
USING THE **PUMPING LEMMA**

Let’s prove that \( B = \{0^n1^n \mid n \geq 0\} \) is not regular

Assume \( B \) is regular. Let \( w = 0^P1^P \)

If \( B \) is regular, can write \( w = xyz, \ |y| > 0, \ |xy| \leq P, \) and for any \( i \geq 0, \ xy^iz \) is also in \( B \)

- \( y \) must be all 0s: Why? \( |xy| \leq P \)

- \( xyyz \) has more 0s than 1s

**Contradiction!**
USING THE **PUMPING LEMMA**

\[ D = \{ w \mid w \text{ has equal number of 1s and 0s}\} \]

is not regular

Assume \( D \) is regular. Let \( w = 0^P1^P \) (\( w \) is in \( D \)!

If \( D \) is regular, can write \( w = xyz, |y| > 0, |xy| \leq P \), where for any \( i \geq 0 \), \( xy^iz \) is also in \( D \)

\( y \) must be all 0s: Why? \( |xy| \leq P \)

\( xyxyz \) has more 0s than 1s

**Contradiction!**
WHAT DOES C LOOK LIKE?

\[
C = \{ w \mid \text{w has equal number of occurrences of 01 and 10} \} \\
= \{ w \mid w = 1, w = 0, w = \varepsilon \text{ or w starts with a 0 and ends with a 0 or w starts with a 1 and ends with a 1} \} \\
1 \cup 0 \cup \varepsilon \cup 0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1
\]
REGULAR EXPRESSIONS
(expressions representing languages)

\(\emptyset\) is a regexp representing \(\{\emptyset\}\)

\(\varepsilon\) is a regexp representing \(\{\varepsilon\}\)

\(\emptyset\) is a regexp representing \(\emptyset\)

If \(R_1\) and \(R_2\) are regular expressions representing \(L_1\) and \(L_2\) then:

\((R_1R_2)\) represents \(L_1 \cdot L_2\)

\((R_1 \cup R_2)\) represents \(L_1 \cup L_2\)

\((R_1)^*\) represents \(L_1^*\)
PRECEDENCE

* · U
EXAMPLE

\[ R_1^* R_2 \cup R_3 = ((R_1^*) R_2) \cup R_3 \]
\{ \text{w} \mid \text{w has exactly a single 1} \}
What language does $\emptyset^*$ represent?

$\{\varepsilon\}$
\{ w \mid w \text{ has length } \geq 3 \text{ and its 3rd symbol is } 0 \}\}

(0U1)(0U1)0(0U1)^*
\{ w \mid \text{every odd position of } w \text{ is a } 1 \} \\
(1(0 \cup 1))^*(1 \cup \varepsilon)
EQUIVALENCE

L can be represented by a regexp
⇔ L is regular

1. L can be represented by a regexp
   ⇒ L is regular

2. L can be represented by a regexp
   ⇐ L is a regular language
1. Given regular expression $R$, we show there exists NFA $N$ such that $R$ represents $L(N)$

**Induction on the *length* of $R$:**

**Base Cases ($R$ has length 1):**

- $R = \sigma$
  
  ![Diagram](image1)

- $R = \varepsilon$
  
  ![Diagram](image2)

- $R = \emptyset$
  
  ![Diagram](image3)
Inductive Step:

Assume $R$ has length $k > 1$, and that every regular expression of length $< k$ represents a regular language.

Three possibilities for $R$:

- $R = R_1 \cup R_2$ \hspace{1cm} (Union Theorem!)
- $R = R_1 R_2$ \hspace{1cm} (Concatenation)
- $R = (R_1)^*$ \hspace{1cm} (Star)

Therefore: $L$ can be represented by a regexp $\Rightarrow L$ is regular.
Give an NFA that accepts the language represented by \((1(0 \cup 1))^*\)
2. L can be represented by a regexp

\[ L \text{ is a regular language} \]

**Proof idea**: Transform an NFA for L into a regular expression by **removing states** and re-labeling arrows with regular expressions.
Add \textit{while} each internal state has more than two states:

Pick an internal state, \textit{rip it out} and \textit{re-label the arrows with regexps}, to account for the missing state.
While machine has more than 2 states:

More generally:

\[ R(q_1, q_3) \cup R(q_2, q_2) \cup R(q_1(q_2), q_2) R(q_2, q_2)^* R(q_2, q_3) \]
$R(q_0, q_3) = (a^*b)(a \cup b)^*$

represents $L(N)$
Formally: Add $q_{\text{start}}$ and $q_{\text{accept}}$ to create G (GNFA)

Run CONVERT(G):  
(Outputs a regexp)

If $\#\text{states} = 2$
return

If $\#\text{states} \neq 2$
the expression on the arrow going from $q_{\text{start}}$ to $q_{\text{accept}}$
Formally: Add $q_{\text{start}}$ and $q_{\text{accept}}$ to create $G$ (GNFA)

Run CONVERT($G$): (Outputs a regexp)

If \#states > 2

select $q_{\text{rip}} \in Q$ different from $q_{\text{start}}$ and $q_{\text{accept}}$

define $Q' = Q - \{q_{\text{rip}}\}$

define $R'$ as:

$$R'(q_i, q_j) = R(q_i, q_{\text{rip}})R(q_{\text{rip}}, q_{\text{rip}})^*R(q_{\text{rip}}, q_j) \cup R(q_i, q_j)$$

($R'$ = the regexps for edges in $G'$)

We note that $G$ and $G'$ are equivalent

return CONVERT($G'$)
Claim: CONVERT(G) is equivalent to G

Proof by induction on k (number of states in G)

Base Case:
  \[ k = 2 \]

Inductive Step:
Assume claim is true for k-1 state GNFAs

Recall that G and G′ are equivalent

But, by the induction hypothesis, G′ is equivalent to CONVERT(G′)

Thus: CONVERT(G′) equivalent to CONVERT(G)

QED
(bb ∪ (a ∪ ba)b*a)* (b ∪ (a ∪ ba)b*)
Convert the NFA to a regular expression

\[ (a \cup b) b^* b (b b^* b)^* a \]
\[(a \cup b)b*b(bb*b)*a)* \cup
(a \cup b)b*b(bb*b)*a)*(a \cup b)b*b(bb*b)*\]
DEFINITION

DFA  <->  NFA

Regular Language  <->  Regular Expression
WWW.FLAC.WS

Finish Chapter 1 for next time.