15-453
FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY
THE PUMPING LEMMA FOR
REGULAR LANGUAGES
and
REGULAR EXPRESSIONS

TUESDAY Jan 21
WHICH OF THESE ARE REGULAR?

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

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

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

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

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

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

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 $i \geq 0$, $xy^iz \in L$

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. $xy^i z \in L$ for all $i \geq 0$

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

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

is not regular

Assume \( D \) is regular. Let \( w = 0P1P \) (\( w \) 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? \( |xy| \leq P \)

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

Contradiction!
WHAT DOES D LOOK LIKE?

\[ D = \{ w \mid w \text{ has equal number of occurrences of 01 and 10} \} \]
WHAT DOES C LOOK LIKE?

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

\[ = \{ w \mid w = 1, w = 0, w = \varepsilon \text{ or } w \text{ starts with a 0 and ends with a 0 or } w \text{ starts with a 1 and ends with a 1} \} \]

\[ 1 \cup 0 \cup \varepsilon \cup 0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1 \]
REGULAR EXPRESSIONS
(expressions representing languages)

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

\( \varepsilon \) 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
EXAMPLE

$R_1 \ast R_2 \cup R_3 = (((R_1 \ast) R_2) \cup R_3)$
{ \( w \mid \text{w has exactly a single 1} \} \\
0^*10^*
What language does $\emptyset^*$ represent?
What language does $\emptyset^* \text{ represent?}$
\{ w \mid \text{w has length } \geq 3 \text{ and its 3rd symbol is 0} \}
\{ w \mid w \text{ has length } \geq 3 \text{ and its 3rd symbol is 0} \} \\
(0 \cup 1)(0 \cup 1)0(0 \cup 1)^*
\{ w \mid \text{every odd position of } w \text{ is a 1} \}
\{ w \mid \text{every odd position of } w \text{ is a } 1 \}\}

\((1(0 \cup 1))^*(1 \cup \varepsilon)\)
EQUVALENCE

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):**

- $R = \sigma$
  
  \[ \xymatrix{ O \ar[r]^\sigma & O } \]

- $R = \varepsilon$
  
  \[ \xymatrix{ O \ar[r] & O } \]

- $R = \emptyset$
  
  \[ \xymatrix{ O \ar[r] & O } \]
Inductive Step:

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

Three possibilities for $R$:

- $R = R_1 \cup R_2$  
  (Union Theorem!)
- $R = R_1 R_2$  
  (Concatenation)
- $R = (R_1)^*$  
  (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 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 unique and distinct start and accept states

While machine has more than 2 states:

Pick an internal state, rip it out and re-label the arrows with regexps, to account for the missing state
Add unique and distinct start and accept states
While machine has more than 2 states:
Pick an internal state, rip it out and re-label the arrows with regexps, to account for the missing state

01*0
While machine has more than 2 states:

More generally:
While machine has more than 2 states:

More generally:

\[ R(q_1, q_2)R(q_2, q_2)^*R(q_2, q_3) \cup R(q_1, q_3) \]
$R(q_0, q_3) =$
represents $L(N)$
R(q₀, q₃) = (a*b)(a∪b)*

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

Run CONVERT(G): (Outputs a regexp)

If #states = 2

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

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

If $\#\text{states} > 2$

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 GNFA

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
$bb \cup (a \cup ba)b^*a$

$(bb \cup (a \cup ba)b^*a)^* (b \cup (a \cup ba)b^*)$
Convert the NFA to a regular expression

- \( q_1 \) to \( q_2 \) via \( a, b \)
- \( q_1 \) to \( q_3 \) via \( a \)
- \( q_3 \) to \( q_2 \) via \( b \)
- \( q_2 \) to itself via \( b \)
DEFINITION

DFA ↔ NFA

Regular Language ↔ Regular Expression
WWW.FLAC.WS

Finish Chapter 1 of the book for next time