Lecture 6:

- Regexes = NFAs
- Non-regular languages

Reading:

- Sipser Ch 1.3
- “Myhill-Nerode” note

Mark Bun
September 22, 2022
Regular Expressions – Syntax

A regular expression $R$ is defined recursively using the following rules:

1. $\varepsilon$, $\emptyset$, and $a$ are regular expressions for every $a \in \Sigma$

2. If $R_1$ and $R_2$ are regular expressions, then so are
   
   $(R_1 \cup R_2)$, $(R_1 \circ R_2)$, and $(R_1^*)$

Examples: (over $\Sigma = \{a, b, c\}$) (with simplified notation)

   $ab$  

   $ab^* c \cup (a^* b)^*$  

   $\emptyset$
Regular Expressions – Semantics

$L(R) = \text{the language a regular expression describes}$

1. $L(\emptyset) = \emptyset$
2. $L(\varepsilon) = \{\varepsilon\}$
3. $L(a) = \{a\}$ for every $a \in \Sigma$
4. $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
5. $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$
6. $L((R_1^*) = (L(R_1))^*$

Example: $L(a^* b^*) = \{a^m b^n \mid m, n \geq 0\}$
Regular Expressions Describe Regular Languages

Theorem: A language $A$ is regular if and only if it is described by a regular expression.

Theorem 1: Every regular expression has an equivalent NFA.

Theorem 2: Every NFA has an equivalent regular expression.
Regular expression -> NFA

**Theorem 1:** Every regex has an equivalent NFA

**Proof:** Induction on size of a regex

**Base cases:**

\[
\begin{align*}
R &= \emptyset \\
R &= \varepsilon \\
R &= a
\end{align*}
\]

\[
\begin{align*}
L(R) &= \emptyset \\
\varepsilon &\in \emptyset \\
\varepsilon &\in \emptyset
\end{align*}
\]

\[
\begin{align*}
\text{NFA} \\
&\rightarrow
\end{align*}
\]
Regular expression -> NFA

**Theorem 1:** Every regex has an equivalent NFA

Proof: Induction on size of a regex

What should the inductive hypothesis be?

a) Suppose **some** regular expression of length \( k \) can be converted to an NFA

\[
\text{number of characters:} \quad \left( a, \emptyset, \epsilon, (,), *, \cup, \circ \right)
\]

b) Suppose **every** regular expression of length \( k \) can be converted to an NFA

c) Suppose **every** regular expression of length **at most** \( k \) can be converted to an NFA

d) None of the above

\[
\Rightarrow \text{Every regular expression of length } k+1 \text{ has an NFA}
\]
Regular expression -> NFA

Theorem 1: Every regex has an equivalent NFA

Proof: Induction on size of a regex

Assume every regex of length \( \leq k \) has an equivalent NFA.
Show every regex of length \( k+1 \) also has an equivalent NFA.

Inductive step:

- \( R = (R_1 \cup R_2) \)
- \( R = (R_1 R_2) \)
- \( R = (R_1^*) \)
Example

Convert \((1(0 \cup 1))^*\) to an NFA

Even length strings \(\cup 1s\) in odd positions
Regular Expressions Describe Regular Languages

**Theorem:** A language $A$ is regular if and only if it is described by a regular expression

**Theorem 1:** Every regular expression has an equivalent NFA

**Theorem 2:** Every NFA has an equivalent regular expression
NFA -> Regular expression

**Theorem 2:** Every NFA has an equivalent regex

Proof idea: Simplify NFA by “ripping out” states one at a time and replacing with regexes
Generalized NFAs

- Every transition is labeled by a regex
- One start state with only outgoing transitions
- Only one accept state with only incoming transitions
- Start state and accept state are distinct
Generalized NFA Example

\[ R(q_s, q) = a^* b \]
\[ R(q_a, q) = \emptyset \]
\[ R(q, q_s) = \emptyset \]
Which of these strings is accepted?

Which of the following strings is accepted by this GNFA?

- a) $aaa$
- b) $aabb$
- c) $bbb$
- d) $bba$

Diagram:

- $q_s$ to $q$ via $a^*b$
- $q$ to $q_a$ via $a$
- $q_a$ is a loop

Path:

$q_s \xrightarrow{a^*b} q \xrightarrow{a} q_a$
NFA -> Regular expression

\[ \text{NFA} \rightarrow \text{GNFA} \rightarrow \cdots \rightarrow \text{Regex} \]

- \( k \) states
- \( k + 2 \) states
- \( k + 1 \) states
- 2 states
NFA -> GNFA

- Add a new start state with no incoming arrows.
- Make a unique accept state with no outgoing arrows.
GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state.
GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state

a) $a^* b(a \cup b)a$
b) $a^* b(a \cup b)^* a$
c) $a^* b \cup (a \cup b) \cup a$
d) None of the above
GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state.
GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state.
Non-Regular Languages
Motivating Questions

- We’ve seen techniques for showing that languages are regular:
  - Construct a DFA
  - Use closure properties
  - Construct an NFA
  - Construct a regular expression
- How can we tell if we’ve found the smallest DFA recognizing a language?
- Are all languages regular? How can we prove that a language is not regular?
An Example

\[ A = \{ w \in \{0, 1\}^* \mid w \text{ ends with } 01 \} \]

**Claim:** Every DFA recognizing \( A \) needs at least 3 states

**Proof:** Let \( M \) be any DFA recognizing \( A \). Consider running \( M \) on each of \( x = \varepsilon, y = 0, w = 01 \)

Let \( q_x = \text{state } M \text{ ends up in when reading } x \)

\[ q_y = \varepsilon \]

\[ q_w = 01 \]

**Claim 1:** \( q_x \neq q_w \) and \( q_y \neq q_w \)

**Why?** \( q_w \) is an accept state, but \( q_x \) and \( q_y \) are not accept states

**Claim 2:** \( q_x \neq q_y \)

**Why?** Assume for contradiction that \( q_x = q_y \)

Run \( M \) on \( x1 = 1 \notin \mathcal{L} \Rightarrow q_i \) is a non-accept state

\[ y1 = 01 \in \mathcal{L} \Rightarrow q_i \text{ is an accept state} \star \]
A General Technique

Definition: Strings $x$ and $y$ are **distinguishable** by $L$ if there exists a string $z$ such that exactly one of $xz$ or $yz$ is in $L$.

Ex. $x = \varepsilon$, $y = 0$

$z = 1$

$xz = \varepsilon 1 = 1 \notin L$

$yz = 01 \in L$

Definition: A set of strings $S$ is **pairwise distinguishable** by $L$ if every pair of distinct strings $x, y \in S$ is distinguishable by $L$.

Ex. $S = \{\varepsilon, 0, 01\}$