Lecture 5:

• Closure Properties
• Regular Expressions

Reading:
Sipser Ch 1.2-1.3

Mark Bun
September 16, 2021
Last Time

• NFAs vs. DFAs
  • Subset construction: NFA -> DFA

• Intro to closure properties of regular languages
Closure Properties
Operations on languages
Let $A, B \subseteq \Sigma^*$ be languages. Define

\begin{align*}
\text{Union: } A \cup B &= \{ x \mid x \in A \lor x \in B \} \\
\text{Concatenation: } A \circ B &= \{ xy \mid x \in A, y \in B \} \\
\text{Star: } A^* &= \{ w_1 w_2 \ldots w_n \mid n \geq 0 \text{ and } w_i \in A \} \\
&= \varepsilon \cup A \cup A A \cup A A A \cup \ldots \\
\text{Complement: } \overline{A} \\
\text{Intersection: } A \cap B \\
\text{Reverse: } A^R &= \{ a_1 a_2 \ldots a_n \mid a_n \ldots a_1 \in A \}
\end{align*}

**Theorem:** The class of regular languages is closed under all six of these operations (i.e., if $A$ and $B$ are both regular, all of the above langs are regular)
Proving Closure Properties
Complement

Complement: $\overline{A} = \{ w \mid w \notin A \} $

**Theorem:** If $A$ is regular, then $\overline{A}$ is also regular

**Proof idea:**

\begin{align*}
\text{A regular } & \implies \exists \text{ a NFA } M_A \text{ recognizes } A \\
\text{Goal: construct new NFA } N \text{ s.t. } N \text{ recognizes } \overline{A}
\end{align*}

\[ \text{N accepts } w \iff \text{M}_{\overline{A}} \text{ rejects } w \]
Complement, Formally

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing a language $A$. Which of the following represents a DFA recognizing $\overline{A}$?

a) $\left( \overline{F}, \Sigma, \delta, q_0, Q \right)$

b) $\left( Q, \Sigma, \delta, q_0, Q \setminus F \right)$, where $Q \setminus F$ is the set of states in $Q$ that are not in $F$

c) $\left( Q, \Sigma, \delta', q_0, F \right)$ where $\delta'(q, s) = p$ such that $\delta(p, s) = q$

d) None of the above
Closure under Concatenation

Concatenation: $A \circ B = \{ xy \mid x \in A, y \in B \}$

Theorem. If $A$ and $B$ are regular, $A \circ B$ is also regular.

Proof idea: Given DFAs $M_A$ and $M_B$, construct NFA by

• Connecting all accept states in $M_A$ to the start state in $M_B$.
• Make all states in $M_A$ non-accepting.

$L(M_A) = A$

$L(M_B) = B$
Closure under Concatenation

Concatenation: \( A \circ B = \{ xy \mid x \in A, y \in B \} \)

**Theorem.** If \( A \) and \( B \) are regular, \( A \circ B \) is also regular.

**Proof idea:** Given DFAs \( M_A \) and \( M_B \), construct NFA by

- Connecting all accept states in \( M_A \) to the start state in \( M_B \).
- Make all states in \( M_A \) non-accepting.

![Diagram of NFA construction for concatenation](image)
Given DFAs $M_A$ recognizing $A$ and $M_B$ recognizing $B$, what does the following NFA recognize?

a) $A \cup B$
b) $A \circ B$
c) $A \cap B$
d) $\{\varepsilon\} \cup A \cup B$
Closure under Star

Star: $A^* = \{ a_1 a_2 ... a_n \ | \ n \geq 0 \text{ and } a_i \in A \}$

**Theorem.** If $A$ is regular, $A^*$ is also regular.
Closure under Star

Star: $A^* = \{ a_1a_2...a_n | n \geq 0 \text{ and } a_i \in A \}$

**Theorem.** If $A$ is regular, $A^*$ is also regular.
On proving your own closure properties

You’ll have homework/test problems of the form “show that the regular languages are closed under some operation”

Given $A \cup B$, show that if $A, B$ are regular, then $A \cup B$ is also regular.

What would Sipser do?

- Give the “proof idea”: Explain how to take machine(s) recognizing regular language(s) and create a new machine
- Explain in a few sentences why the construction works
- Give a formal description of the construction
- No need to formally prove that the construction works
Regular Expressions
Regular Expressions

• A different way of describing regular languages
• A regular expression expresses a (possibly complex) language by combining simple languages using the regular operations

“Simple” languages: $\emptyset, \{\varepsilon\}, \{a\}$ for some $a \in \Sigma$

Regular operations:

**Union:** $A \cup B$

**Concatenation:** $A \circ B = \{ab \mid a \in A, b \in B\}$

**Star:** $A^* = \{a_1a_2...a_n \mid n \geq 0$ and $a_i \in A\}$
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 $\Sigma = \{a, b, c\}$)

$(a \circ b)$
$((((a \circ (b^*)) \circ c) \cup (((a^*) \circ b))^*))$  \hspace{1cm} (\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. $L((R_1^*) = (L(R_1))^*$
Regular Expressions – Example

$L(((a^*) \circ (b^*))) = \$

a) \{a^n b^n \mid n \geq 0\}
b) \{a^m b^n \mid m, n \geq 0\}
c) \{(ab)^n \mid n \geq 0\}
d) \{a, b\}^*$

1) \(L(a) = \{a^n \mid n \geq 0\}\)
   \(L(b) = \{b^n \mid n \geq 0\}\)

2) \(L(a^*) = \{a^m \mid m \geq 0\}\)
   \(L(b^*) = \{b^n \mid n \geq 0\}\)

3) \(L((a^*) \circ (b^*)) = \)
   \(L((a^*)) \circ L((b^*)) = \)
   \(\{a^m b^n \mid m \geq 0, n \geq 0\}\)
Simplifying Notation

• Omit $\circ$ symbol: $(ab) = (a \circ b)$

• Omit many parentheses, since union and concatenation are associative:

$$(a \cup b \cup c) = (a \cup (b \cup c)) = ((a \cup b) \cup c)$$

• Order of operations: Evaluate star, then concatenation, then union

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

Let $\Sigma = \{0, 1\}$

1. $\{w \mid w \text{ contains exactly one } 1\}$

2. $\{w \mid w \text{ has length at least } 3 \text{ and its third symbol is } 0\}$

3. $\{w \mid \text{every odd position of } w \text{ is } 1\}$
Syntactic Sugar

• For alphabet $\Sigma$, the regex $\Sigma$ represents $L(\Sigma) = \Sigma$
  
  $\Sigma = \{ a, b, c, \ldots, z \}$

  means
  $\Sigma = (a \cup b \cup c \ldots \cup z)$

• For regex $R$, the regex $R^+ = RR^*$

  
  $L(R^+) = \{ a \cdot a \cdot \ldots \cdot a^n \mid n \geq 1 \}$

  and $a \in L(R)$ for $i \geq 3$
Regexes in the Real World

`grep` = globally search for a regular expression and print matching lines
Equivalence of Regular Expressions, NFAs, and DFAs
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 \(-\rightarrow\) NFA

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

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

**Base cases:**

\[
\begin{align*}
R &= \emptyset \\
L(R) &= \emptyset \\
\rightarrow &\rightarrow \emptyset \\

R &= \varepsilon \\
L(R) &= \{ \varepsilon \} \\
\rightarrow &\rightarrow \varepsilon \\

R &= \alpha \\
L(R) &= \{ \alpha \} \\
\rightarrow &\rightarrow \alpha \varepsilon \rightarrow
\end{align*}
\]
Regular expression -> NFA

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

Proof: Induction on size of a regex

Assume IN, show every regex of size $k+1$ has an equiv. NFA

What should the inductive hypothesis be?

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

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
Regular expression -> NFA

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

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

Inductive step:

\[ R = (R_1 \cup R_2) \]

\[ R = (R_1 R_2) \]

\[ R = (R_1^*) \]