BU CS 332 – Theory of Computation

Lecture 5:
• Closure Properties
• Regular Expressions

Reading:
Sipser Ch 1.2-1.3

Mark Bun
September 20, 2022

https://forms.gle/aCxHoyMogZNUpLw96
Last Time

• Nondeterministic Finite Automata
• NFAs vs. DFAs
  • Subset construction: NFA -> DFA
Closure Properties
An Analogy

In algebra, we try to identify operations which are common to many different mathematical structures

Example: The integers \( \mathbb{Z} = \{ ..., -2, -1, 0, 1, 2, ... \} \) are closed under
- Addition: \( x + y \)
- Multiplication: \( x \times y \)
- Negation: \( -x \)
- ...but NOT Division: \( x / y \)

We’d like to investigate similar closure properties of the class of regular languages
Operations on languages

Let \( A, B \subseteq \Sigma^* \) be languages. Define

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

**Theorem:** The class of regular languages is closed under all six of these operations, i.e., if \( A \) and \( B \) are regular, applying any of these operations yields a regular language.
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:
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) $(F, \Sigma, \delta, q_0, Q)$

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

c) $(Q, \Sigma, \delta', q_0, F)$ 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, then $A \circ B$ 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.

$L(M_A) = A \quad L(M_B) = B$
Closure under Concatenation

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

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

\[ L(M_A) = A \quad \text{and} \quad L(M_B) = B \]
A Mystery Construction

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

Theorem. If \( A \) is regular, then \( A^* \) is also regular.

\[ L(M) = A \]
Closure under Star

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

**Theorem.** If \( A \) is regular, then \( A^* \) 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”

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

$(a \circ b)$, $(((a \circ (b^*)) \circ c) \cup (((a^*) \circ b))^*)$, $(\emptyset^*)$
Regular Expressions – Semantics

$L(R)$ = 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) \( \{ a, b \}^* \)
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. $\{w \mid w \text{ has length at least 3 and its third symbol is 0}\}$

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

• For alphabet $\Sigma$, the regex $\Sigma$ represents $L(\Sigma) = \Sigma$

• For regex $R$, the regex $R^+ = RR^*$
**Regexes in the Real World**

```bash
$ grep '^xy*z' myfile
xyz
xyzde
xz
xyyz
xyyz
$ grep '^x.*z' myfile
xyz
xyzde
xxz
xxx
xz
x
xz
x
x
$ grep '^x\z' myfile
x*z
$ grep '\\' myfile
\n```

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 -> NFA

Theorem 1: Every regex has an equivalent NFA
Proof: Induction on size of a regex

Base cases:

\[ R = \emptyset \]

\[ R = \varepsilon \]

\[ R = a \]
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

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:

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

\[ R = (R_1 R_2) \]

\[ R = (R_1^*) \]
Example

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