CS304: Automata and Formal Languages

Lec 3

Refresher on Basic Maths, Recursion, and Operations on Strings and Languages

Rachit Nimavat

Aug 1, 2025

Outline

Announcement

10min in-class **quiz** in next class!

Syllabus: everything until (and including) induction

Alphabet, Strings, and Languages

Alphabet Σ: A finite, non-empty set of symbols

- e.g. the binary alphabet $\Sigma = \{0, 1\}$

String w: A finite sequence of symbols taken from an alphabet

- e.g. $w = 001$ is a string over Σ
- ε is an empty string with $|\varepsilon| = 0$ symbols
- $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \dots,\}$ is the set of all possible strings over Σ

Language L: A set of strings over a particular alphabet

- For any language L over Σ, $L \subseteq \Sigma^*$
- Languages $\equiv$ Functions

  Each language L can be thought of as an indicator function
  Each boolean function f can be thought of as a language

Induction

Useful to prove properties about structures that are defined **recursively**

Core Idea:

Imagine an infinitely tall ladder

Base Case: If you can climb the first step (prove the statement for the smallest value)

Inductive Step: And if you show that if a step can be climbed, the next one can also be climbed (prove that if the statement holds for k, it holds for $k+1$)

Conclusion: Then an infinitely tall ladder can be climbed (the statement holds for all values)

Otherway round: Use recursion to build "infinite ladders"

Recursion

Core Idea: Imagine an infinitely tall ladder

Induction

Base Case: If you can climb the first step (prove the statement for the smallest value)

Inductive Step: And if you show that if a step can be climbed, the next one can also be climbed (prove that: if the statement holds for k, it also holds for $k+1$)

Conclusion: Then the ladder can be climbed (the statement holds for all values)

Recursion

Base Case: If first step exists (state the simplest/smallest members of the set)

Recursive Step: And if you show how to build the next step when standing on the previous step (provide rules to build ‘next level’ of elements from existing ones)

Closure: Then the ladder can be built (completes the description of the set)

Example - Recursive Definition of Σ^*

Formalize the definition of the set of all strings over an alphabet.

Fix an alphabet Σ. Then, Σ^* is the unique set satisfying the following properties:

Base Case: The empty string $\varepsilon \in \Sigma^*$

Recursive Step: For each string $w \in \Sigma^*$ and a symbol $a \in \Sigma$, $wa \in \Sigma^*$ (recall, wa is concatenation of strings w and a)

Closure: A string is in Σ^* only if it can be derived from the base case by a finite number of applications of the recursive step

HW: Show that Σ^* (as defined above) is indeed the set of all strings over Σ.

Hint:

(Show (i) $\Sigma^* \subseteq \text{AllStrings}(\Sigma)$ and (ii) $\text{AllStrings}(\Sigma) \subseteq \Sigma^*$. (i) is easy, (ii) uses induction.)

Another Example

$\Sigma = \{0,1\}$ and $L = \{0, 01, 011, 0111, 01111, \ldots,\}$

Base Case: $0 \in L$

Recursive Step: $w \in L \implies w1 \in L$

Closure: L is the set of all strings that can be derived from the base case by a finite number of applications of the recursive step

HW: Write recursive definition of $L = \{\varepsilon, 01, 0011, 000111, 00001111, \ldots,\}$ **over** $\Sigma = \{0,1\}$

Operations on Strings: Concatenation and Reversal

Concat: If x and y are strings, xy is their concatenation

e.g. $x = 010$ and $y = 101$

- $xy = 010101$
- $xx = 010010$
- shorthand for **string powers**: $x^2 = xx = 010010$
- for a string w, define $w^n := \underbrace{ww \ldots w}_{n \text{ times}}$

HW: Formally define w^n.

Hint: $w^0 := \varepsilon$ and $w^{k+1} := w^k w$

Reversal: For a string w, its reverse w^R is defined as:

- If $w = \varepsilon$, $\varepsilon^R := \varepsilon$
- If $w = xa$, for string x and character a, $w^R := ax^R$

HW: Write a recursive C/Python program to find string reversals. Test your program with $w =$ STRESSED.

Operations on Languages: Union, Intersection, and Complement

Recall, languages are sets. All set operations apply naturally.

Union

For languages L_1 and L_2, define $L_1 \cup L_2 := \{w \mid w \in L_1 \vee w \in L_2\}$

Intersection

For languages L_1 and L_2, define $L_1 \cap L_2 := \{w \mid w \in L_1 \wedge w \in L_2\}$

Complement

For a language over Σ, define $\bar{L} = L^c := \{w \in \Sigma^* \mid w \notin L\} = \underbrace{\Sigma^* \setminus L}_{\text{“set minus”}}$

Operations on Languages: Concatenation

Similar to set "cross-product" or "cartesian-product"

For languages L_1 and L_2, define $L_1 L_2 := \{xy \mid x \in L_1 \wedge y \in L_2\}$

e.g. $L_1 = \{0, 1\}$ and $L_2 = \{a, b\}$ then $L_1 L_2 = \{0a, 0b, 1a, 1b\}$

note: L_1 and L_2 may not be over same alphabet

What is the alphabet of $L_1 L_2$?

If L_1 over Σ_1 and L_2 over Σ_2, then language $L_1 L_2$ is over $\Sigma_1 \cup \Sigma_2$

Language Powers: Similar to string powers,

$L^2 = LL$, $L^3 = LLL$, and so on

What is L^0? $L^0 = \{\varepsilon\}$

HW: Formally define L^n

Hint: $L^0 = \{\varepsilon\}$, $L^{k+1} = L^k L$

Operations on Languages: Kleene Star

Consider a language L over alphabet Σ. We defined $L^n = \underbrace{LL\ldots L}_{n \text{ times}}$

Useful Shorthand to denote 0 or more occurrences (known as Kleene Star pronounced: clay-knee):

$$L^* = L^0 \cup L^1 \cup L^2 \ldots = \bigcup_{i=0}^{\infty} L^i$$

L^* is the set of all strings formed by concatenating zero or more strings from L

Notice:

- $\varepsilon \in L^*$
- L^* is also a language over Σ
- Can also view Σ as a language (consisting of all possible single-character strings)
- Σ^* is the language consisting of all possible strings over Σ (excatly as we defined!)

HW: Convince yourself that $L \subseteq L^* \subseteq \Sigma^*$. When are these inequalities tight?

Reminder

10min in-class **quiz** in next class!
Syllabus: everything until (and including) induction