01204211 Discrete Mathematics
Lecture 7a: Languages and regular expressions[1]

Jittat Fakcharoenphol

August 19, 2025

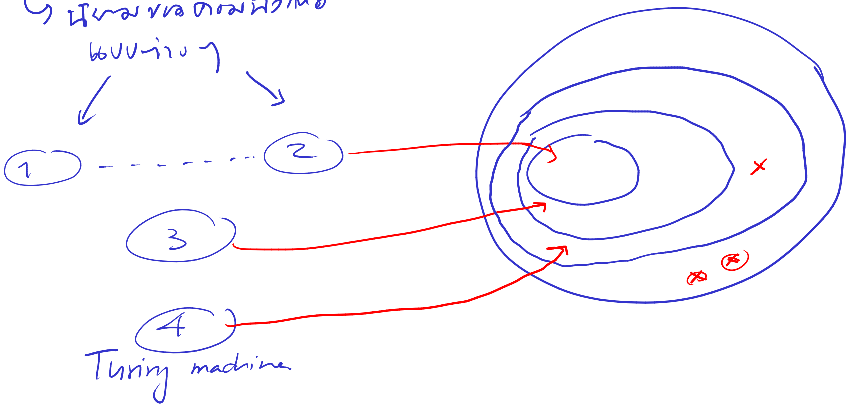
[1] Based on lecture notes of *Models of Computation* course by Jeff Erickson.

What is computation? “คิดเลข”

โมเดลการคำนวณ

↳ นิยามของคอมพิวเตอร์
แบบต่าง ๆ

ปัญหา



Models of computations

Definition $\longrightarrow$ $\boxed{\text{concept.}}$

Languages = specifications

set ของ string

{"hello", "ekitike"}

{"1+1=2", "15+31=46", ... }

{"2","3","5","7","11", ... } set ของ จน.เฉพาะ:

{0, 00, 000, 0000, ... }

$\{(a, b, c) \mid a, b, c \in \mathbb{N},\ a+b=c\}$

$x \to \boxed{C_n} \to$ Y/N

Y – อยู่ใน language

N – ไม่...

Strings

- เซตของ symbols $\Sigma = \{0, 1\}$

① ε เป็น string over Σ

② $a \cdot x$ เป็น string over Σ ถ้า x เป็น string. และ: $a \in \Sigma$

Formal definition: strings

Intuitively, a string is a *finite* sequence of symbols. However, to be able to formally prove properties of strings we need a precise definition.

Let a finite set Σ be the **alphabet**. (E.g., for bit strings, $\Sigma = \{0, 1\}$; for digits, $\Sigma = \{0, 1, \ldots, 9\}$; for English string $\Sigma = \{a, b, \ldots, z\}$.)
The following is a recursive definition of strings.

Recursive definition of strings

A **string** w over alphabet Σ is either

- the empty string ε, or
- $a \cdot x$ where $a \in \Sigma$ and x is a string.

The set of all strings over alphabet Σ is denoted by Σ^*.

Review: more recursive definitions

Lengths

For a string w, let $|w|$ be the length of w defined as

$$|w| = \begin{cases} 0 & \text{when } w = \varepsilon \\ 1 + |x| & \text{when } w = a \cdot x \end{cases}$$

Concatenation

For strings w and z, the concatenation $w \bullet z$ is defiend recursively as

$$w \bullet z = \begin{cases} z & \text{when } w = \varepsilon \\ a \cdot (x \bullet z) & \text{when } w = a \cdot x \end{cases}$$

Review: proving facts about strings

Lemma 1

For strings w and x, $|w \bullet x| = |w| + |x|$.

Proof.

Induction Hypothesis: สำหรับทุก string y ที่ $|y| < |w|$,

$$|y \cdot x| = |y| + |x|.$$

Case 1: $w = \varepsilon$.

Case 2: $w = a \cdot y$ โดยที่ y เป็น string ที่ $|y| < |w|$

พิจารณา $|w \bullet x| = |a \cdot y \bullet x|$ จากนิยาม

$= 1 + |y \bullet x|$ จากนิยามของ length

$= 1 + |y| + |x|$ $\boxed{\text{จาก I.H}}$

$= |w| + |x|$ จากนิยาม ได้ /

∎

Formal languages

A **formal language** is a set of strings over some finite alphabet Σ.

Examples:

$\{1, 10, 100, 1000, 10000, 100000, \ldots\}$

$\{1, 11, 101, 111, 1001, 1011, 1101, 1111, \ldots\}$

Careful...

langu

These are different languages: $\emptyset, \{\varepsilon\}$

And ε is not a language.

How to describe languages?

$A = \boxed{\{ \text{hello} \}}$

$Z = \emptyset$

$B = \{ a, an, the \}$

$C = \{ \text{hunsen}, \text{thaksin}, \text{prayuth}, \text{ink} \}$

$A \cup B \cup C$

$B \bullet C = \{ \text{ahunsen}, \text{anhuse}, \ldots$

$\text{athn}, \text{an}\ldots, \ldots \}$

$\boxed{\{ 0^n \mid n \geq 0 \}}$

$= \{ \varepsilon, 0, 00, 000, 0000, \ldots$

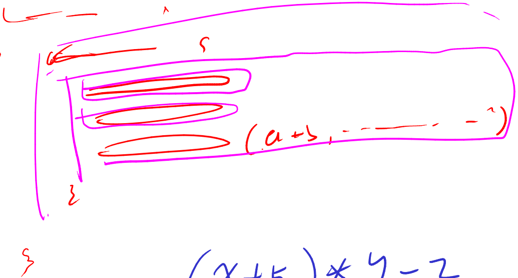
$\boxed{\{ 0^n 1^n \mid n \geq 0 \}}$

$= \{ \varepsilon, 01, 0011, 000111, \ldots$

$\boxed{\{ 0^p \mid p \text{ เป็นจน.เฉพาะ} \}}$

คณิตศาสตร์.

Composition



$(x+5)*y-2$

decomposition

LU

Combining languages

If A and B are languages over alphabet Σ.

- Basic set operations: $A \cup B$, $A \cap B$, $\bar{A} = \Sigma^* \setminus A$.
- Concatenation: $A \bullet B$.

$$A \bullet B = \{ xy \mid x \in A, y \in B \}$$

$A = \{ a, an, the \}$

$A^* = \{ \varepsilon, a, an, the,$
$aa, ana, thea,$
$anan, aan, thean,$

- Kleene closure or Kleene star: A^*.

string $w \in A^*$ เมื่อ

1. $w = \varepsilon$.
2. $w = x \bullet y$ เมื่อ $x \in A$ และ $y \in A^*$

Also $\boxed{A^+} = A \bullet A^*$

Examples

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

$$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$$

$$(\Sigma^*) \cdot \{0\}$$

$\vdots$

Regular languages

Definition: regular languages

A language L is **regular** if and only if it satisfies one of the following conditions:

- L is empty;
- L contains one string (can be the empty string ε);
- L is a union of two regular languages;
- L is the concatenation of two regular languages; or
- L is the Kleene closure of a regular language.

Examples

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

$\{\}$

$\{000\}$

$\left(\underline{\underline{\{000\}} \cup \underline{\{11\}}}\right) \bullet \{\} =$

$\underline{\{0,1\}}^*,$

$\downarrow$

$(\underline{\{0\}} \cup \underline{\{1\}})^*$

$\underline{\{00\}}^* = \{\varepsilon, 00, 0000, 000000, \vdots$

$\{000, 11\} \bullet \{\} = \cancel{\{000, 11\}} = \{000, 11\} \bullet \{\varepsilon\}$

$\{\}$ ×

Regular expressions

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

$$((\{01\} \cup (\{1\} \bullet (\{0\} \cup \{10\}))) \cup (\{00\} \bullet (\{1\})^*)) \bullet ((\{0\} \bullet \{0\}) \bullet \{1\})$$

Regular expressions

표현 (describe) regular language

Regular language

set $\rightarrow$ $((\{01\} \cup (\{1\} \cdot (\{0\} \cup \{10\}))) \cup (\{00\} \cdot (\{1\})^*)) \cdot ((\{0\} \cdot \{0\}) \cdot \{1\})$

is represented as

$$(01 + 1(0 + 10) + 00(1)^*)001$$

Regular expressions

- omit braces around one-string sets
- use $+$ instead of $\cup$
- omit $\bullet$
- follow the precedence: Kleene star operator $*$, $\bullet$ (implicitly), and $+$.

Remark: $+$ *and* $\bullet$ *are associative, i.e.,* $(A + B) + C = A + (B + C)$ *and* $(A \bullet B) \bullet C = A \bullet (B \bullet C)$.

Regular expressions: examples 1

Strings ที่ลงท้ายด้วย 000000000

$\{0,1\}^*$

$(0+1)^*000000000$

(b) strings that contain 010

$(0+1)^*010(0+1)^*$

<u>substring</u> $\sim\sim$ 010 . — — —

<u>subsequence</u> $\sim\sim$ 0 — 1 — — . 0 —

(c)

$(0+1)^*0(0+1)^*1(0+1)^*0(0+1)^*$

Regular expressions: examples 2

All strings over $\{0, 1\}$ except 010.

0101

$$0^* + 1^* + 1(0+1)^* + 00(0+1)^* + 011(0+1)^* + 010(0+1)(0+1)^*$$

Models

grammar	machine
• regular expressions	• finite state automata.

(d)

(e)

$$0^* + 1^*(0^* + 00^*1^*)$$

Subexpressions

Regex is everywhere

Proofs about regular expressions - structural induction

Lemma 2

Every regular expression that does not use the symbol $\emptyset$ represents a non-empty language.

Proof.

Lemma 2

Every regular expression that does not use the symbol $\emptyset$ represents a non-empty language.

Proof.

Let R be a regular expression that does not use the symbol $\emptyset$. We prove by (structural) induction that R represents a non-empty language.

Lemma 2

Every regular expression that does not use the symbol $\emptyset$ represents a non-empty language.

Proof.

Let R be a regular expression that does not use the symbol $\emptyset$. We prove by (structural) induction that R represents a non-empty language.

Induction hypothesis: Every subexpression of R that does not use the symbol $\emptyset$ represents a non-empty language.

Lemma 2

Every regular expression that does not use the symbol $\emptyset$ represents a non-empty language.

Proof.
Let R be a regular expression that does not use the symbol $\emptyset$. We prove by (structural) induction that R represents a non-empty language.
Induction hypothesis: Every subexpression of R that does not use the symbol $\emptyset$ represents a non-empty language.
Case 1: $R = \emptyset$.

Lemma 2

Every regular expression that does not use the symbol $\emptyset$ represents a non-empty language.

Proof.

Let R be a regular expression that does not use the symbol $\emptyset$. We prove by (structural) induction that R represents a non-empty language.

Induction hypothesis: Every subexpression of R that does not use the symbol $\emptyset$ represents a non-empty language.

Case 1: $R = \emptyset$.

Case 2: R is a single string.

Lemma 2

Every regular expression that does not use the symbol $\emptyset$ represents a non-empty language.

Proof.

Let R be a regular expression that does not use the symbol $\emptyset$. We prove by (structural) induction that R represents a non-empty language.

Induction hypothesis: Every subexpression of R that does not use the symbol $\emptyset$ represents a non-empty language.

Case 1: $R = \emptyset$.

Case 2: R is a single string.

Proof. (cont.2/4)
Case 3: $R = S + T$ for some regular expressions S and T.

Proof. (cont.3/4)
Case 4: $R = S \bullet T$ for some regular expressions S and T.

Proof. (cont.4/4)
Case 5: $R = S^*$ for some regular expression S.

Proof. (cont.4/4)
Case 5: $R = S^*$ for some regular expression S.

In every case, the language $L(R)$ is non-empty.

Lemma 3

Every non-empty regular language is represented by a regular expression that does not use the symbol $\emptyset$.

Lemma 3

Every non-empty regular language is represented by a regular expression that does not use the symbol $\emptyset$.

Let R be a regular expression.

Lemma 3

Every non-empty regular language is represented by a regular expression that does not use the symbol $\emptyset$.

Let R be a regular expression. We prove that if $L(R) \neq \emptyset$, then there exists a regular expression R' such that $L(R) = L(R')$ and R' does not contain $\emptyset$.

Lemma 3

Every non-empty regular language is represented by a regular expression that does not use the symbol $\emptyset$.

Let R be a regular expression. We prove that if $L(R) \neq \emptyset$, then there exists a regular expression R' such that $L(R) = L(R')$ and R' does not contain $\emptyset$.
We prove by induction. What should the induction hypothesis be?

I.H.: For every subexpression S of R, if $L(S) \neq \emptyset$, there exists an $\emptyset$-free regular expression S' such that $L(S) = L(S')$.

I.H.: For every subexpression S of R, if $L(S) \neq \emptyset$, there exists an $\emptyset$-free regular expression S' such that $L(S) = L(S')$.

What are the cases that we have to consider?

I.H.: For every subexpression S of R, if $L(S) \neq \emptyset$, there exists an $\emptyset$-free regular expression S' such that $L(S) = L(S')$.

What are the cases that we have to consider?

- $R = \emptyset$
- R is a single string.
- $R = S + T$ for some regular expressions S and T.
- $R = S \bullet T$ for some regular expressions S and T.
- $R = S^*$ for some regular expression S.

(E-ex1-6) For string w, the reversal w^R is defined recursively as follows:

$$w^R = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ x^R \cdot a & \text{if } w = ax \text{ for some symbol } a \text{ and some string } x \end{cases}$$

For a language L, the reversal of L is defined as

$$L^R = \{w^R \mid w \in L\}.$$

You may assume the following facts.

- $L^* \cdot L^* = L^*$ for every language L.
- $(w^R)^R = w$ for every string w.
- $(x \cdot y)^R = y^R \cdot x^R$ for all strings x and y.

Prove that $(L^R)^* \subseteq (L^*)^R$.