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

Jittat Fakcharoenphol

August 19, 2025

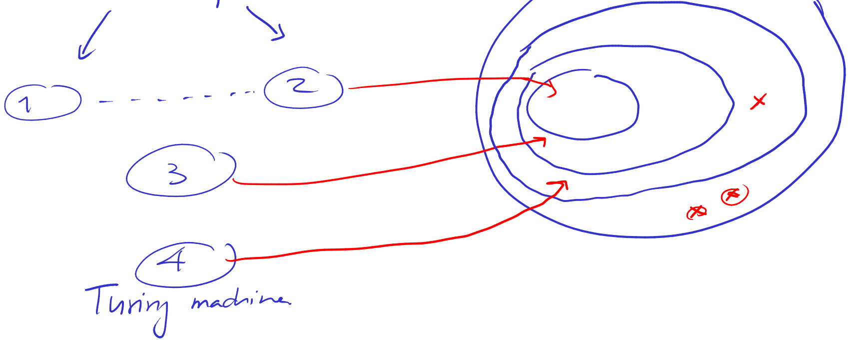
[1] Based on lecture notes of *Models of Computation* course by Jeff Erickson.

What is computation? "คิดเอง"

โมเดลการคำนวณ

↳ นิยามของคอมพิวเตอร์แบบต่างๆ

ปัญหา



Models of computations

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

Languages = specifications

set ของ string

$$\{\text{"hello"}, \text{"ekitike"}\}$$

$$\{\text{"1+1=2"}, \text{"15+31=46"}, \ldots\}$$

$$\{\text{"2"}, \text{"3"}, \text{"5"}, \text{"7"}, \text{"11"}, \ldots\}$$ set ของ จน.เฉพาะ:

$$\{0, 00, 000, 0000, \ldots\}$$

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

$x \rightarrow \boxed{\text{Cน}} \rightarrow$ Y/N

Y – อยู่ใน langu[?]

N – ไม่[?]...

Strings

- เซตของ symbols Σ $= \{0, 1\}$

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 $|y| < |w|$,

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

Case 1: $w = \varepsilon$.

Case 2: $w = a \cdot y$ แต่ 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 แล้ว ✓

$\square$

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?

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

$Z = \emptyset$

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

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

$A \cup B \cup C$

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

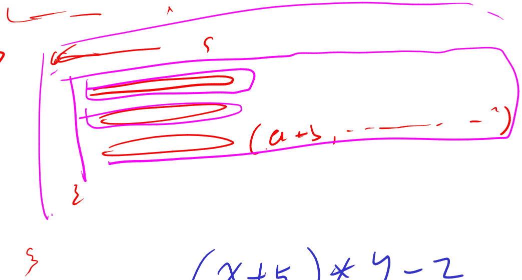
$\boxed{\{0^n \mid n \geq 0\}}$
$=$
$\{\varepsilon, 0, 00, 000, 0000, \ldots$

$\boxed{\{0^n1^n \mid n \geq 0\}}$
$=$
$\{\varepsilon, 01, 0011, 000111, \ldots$

$\boxed{\{0^p \mid p \text{ เป็นจำนวนเฉพาะ}\}}$

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 $A^+ = A \bullet A^*$

Examples

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

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

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

$\vdots$

Regular languages

$\{\} \circ \{hello\} = \emptyset$

$\{a\} \circ \{c, d, a\} = \{ac, ad, aa\}$

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.

w ... regular expr

1. $w = \phi$
2. $w = x$ ... x ... string
3. $w = S + T$ ... S & T ... Regexpr
4. $w = S \circ T$ ... S & T ...
5. $w = S^*$

Examples

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

$\{\}$

$\{000\}$

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

$\underline{\{0,1\}^*}$, $\{\underline{00}\}^* = \{\varepsilon, 00, 0000, 000000, \vdots$

$\downarrow$

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

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

$\{\}$ ×

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

→ อธิบาย (describe) regular language

Regular language

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

is represented as

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

$001 + 10 + 00^*$

Regular expressions

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

$001 + [10 + 00^*]$

$[001 + 10] + 00^*$

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

$\{0,1\}^*$

$(0+1)^* 000000000$

substring ~~~ 010 ---

subsequence ~~~ 0 --- 1 --- 0

(b) strings that contain 010

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

(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

(d)

(e)

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

grammar

- regular expressions

machine

- finite state automata

Subexpressions

$\underbrace{01}+\underbrace{111}$

regular expression w ใดๆ
ที่ไม่ใช่ $\emptyset$ จะแทน language ที่ non empty.

$\forall w \in \boxed{\text{Regular expression}}$, ($w$ ไม่ใช่ $\emptyset$) $\Rightarrow$ (w จะแทน language ที่ไม่ empty)

$\boxed{\text{I.H.}}$ สำหรับทุกๆ subexpression S ของ w, ถ้า S ไม่ใช่ $\emptyset$, S จะแทน nonempty language.

Case 1: $w = \emptyset$ —

Case 2: $w = x$ เพราะ x เป็น string

Case 3: $w = S+T$ เมื่อ S & T เป็น regex

Case 4: $w = S \bullet T$ . . .

Case 5: $w = S^*$ เมื่อ S เป็น regex

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.

$P \Rightarrow Q$

Proof.

[illegible]

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: $R = \emptyset$. × เป็นไปไม่ได้!

เนื่องจาก assumption ว่า R ไม่มี $\emptyset$, R จะไม่เท่ากับ $\emptyset$ กรณีนี้จึงเป็นไปไม่ได้

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.

ในกรณีนี้, R จะแทน nonempty language ที่มี string 1 ตัว

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. $L(R) \neq \phi$

Induction hypothesis: Every subexpression S of R that does not use the symbol $\emptyset$ represents a non-empty language. $(L(S) \neq \phi)$

Case 1: $R = \emptyset$.

Case 2: R is a single string.

Goal: R แทน non-empty language.

Proof. (cont.2/4)

Case 3: $R = S + T$ for some regular expressions S and T.

ข้อความ	เหตุผล
1. S ไม่มี ∅	เนื่องจาก R ไม่มี ∅ และ S เป็น subexpression ของ R
2. S แทน non-empty language	I.H. และ 1
3. T ไม่มี ∅	
4. T แทน non-empty language	
5. R แทน non-empty language	เนื่องจาก R แทน language ที่เป็น union ของ language ที่แทนโดย S และ T และ 4, 2

- เนื่องจาก S และ T เป็น subexpression ของ R และ R ไม่มี ∅ จะได้ว่า S และ T ไม่มี ∅ ด้วย
- จาก I.H จึงสรุปได้ว่า [S และ T แทน non empty language]

$$L(S) \neq \emptyset, \quad L(T) \neq \emptyset$$

- เนื่องจาก R แทน language ที่เป็น union ของ language ที่แทนโดย S และ T จะได้ว่า R แทน language ที่ non empty ด้วย เช่นเดียวกัน

เนื่องจาก $L(R) = L(S) \cup L(T)$, และ $L(S) \neq \emptyset$, $L(T) \neq \emptyset$

$\Rightarrow L(R) \neq \emptyset$

โดย $\boxed{L(R)}$ เป็น language ที่แทนโดย R

Proof. (cont.3/4)

Case 4: $R = S \bullet T$ for some regular expressions S and T.

- เนื่องจาก S และ T เป็น subexpr ของ R, S & T จะไม่ใช่ ϕ
- จาก I.H. จะได้ว่า $L(S) \neq \phi$, $L(T) \neq \phi$
- เนื่องจาก $L(R) = L(S) \bullet L(T)$

  และมีบางสตริง $x \in L(S)$ กับบางสตริง $y \in L(T)$

  จะได้ว่า $xy \in L(S) \bullet L(T)$ ดังนั้น

  $L(R) \neq \phi$

Proof. (cont.4/4)

Case 5: $R = S^*$ for some regular expression S.

เนื่องจาก $\varepsilon \in S^*$,

ดังนั้น $L(R) \neq \phi$.

ดังนั้น สำหรับทุกๆ regular expr R ที่ไม่ใช่ ϕ, $L(R) \neq \phi$. ∎

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 $\emptyset$.

พิจารณา regular lange R (set) ใดๆ

I.H. สำหรับทุก subset ของ R.

ex

$$(\emptyset \cup \{10\})^* \cup (\emptyset \cdot \{1\}^*)^*$$

$$\downarrow$$

$$(\emptyset + 10)^* + (\underbrace{\emptyset \cdot 1^*}_{\emptyset})^*$$

$$(10)^* + \{\varepsilon\}$$

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. สำหรับทุกๆ subexpression S ของ R, if $L(S) \neq \emptyset$,
จะมี S' ที่ $L(S') = L(S)$ และ S' ไม่มี $\emptyset$.

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?

- $R = \emptyset$
- $R = x$ then x [illegible]
- $R = S + T$
- $R = S \circ T$
- $R = S^*$

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?

- $R = \emptyset$, ไม่ต้อง $L(R) \neq \emptyset$, ตาม ×.
- R is a single string. ✓
- $R = S + T$ for some regular expressions S and T.
- $R = S \cdot T$ for some regular expressions S and T.
- $R = S^*$ for some regular expression S.

$\boxed{\text{Case 1}}$ $L(S) \neq \emptyset$, $L(T) \neq \emptyset$

จะมี S' และ T' ที่ไม่มี $\emptyset$

และ $L(S') = L(S)$ &

$L(T') = L(T)$

ได้ $R' = S' + T'$.

Case 2: $L(S) = \emptyset$, $L(T) = \emptyset$;

จะได้ว่า $L(R) = L(S) \cup L(T) = \emptyset$ ซึ่ง

เป็นไปไม่ได้. (ตัดแล้ว)

Case 3: $L(S) = \emptyset$ และ $L(T) \neq \emptyset$ ω

$L = \{10, 11, 111, 100\}$

$L^R = \{01, 11, 111, 001\}$

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

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

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.

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

$(\text{love} \cdot \text{you})^R = \text{you}^R \cdot \text{love}^R$

$= \text{uoy evol}$

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