


01204211 Discrete Mathematics

Lecture 7a: Languages and regular expressions¹

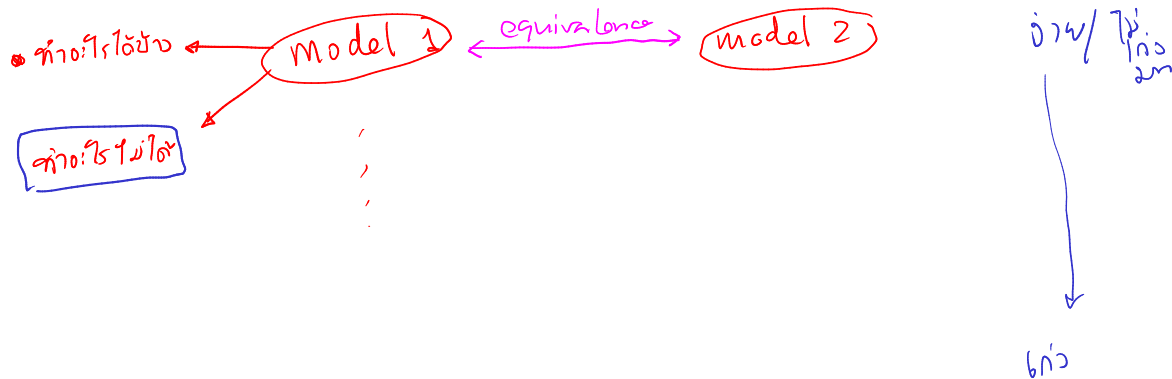
Jittat Fakcharoenphol

August 20, 2024

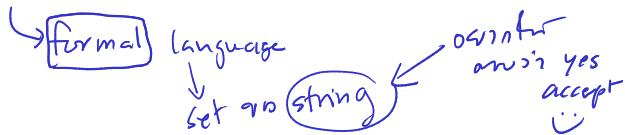
¹Based on lecture notes of *Models of Computation* course by Jeff Erickson. 

What is computation?

Models of computations



Languages = specifications



~~טבעי (language)~~

→ טבעי / טבעי } natural language (NLP)

$\{x \mid x \text{ is a word}\}$

$\{x \mid x \text{ is a sentence}\}$

$\{x : x \text{ is a palindrome}\}$

$\{p : p \text{ is a program that takes input and outputs}\}$

$\{\varepsilon\} \quad \{\} = \emptyset$

$\varepsilon \neq \emptyset$

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, $\Sigma = \{0, 1, \dots, 9\}$; for English string $\Sigma = \{a, b, \dots, z\}$.)

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 $w \cdot z$ is defined recursively as

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

Review: proving facts about strings

$$|011 \cdot 112| = |011| + |112|$$

$$|(1 \cdot \epsilon) \cdot 112| = |1| + |112|$$

Lemma 1 $\forall w, x$ def $|x|$.

For strings w and x $|w \cdot x| = |w| + |x|$. $|(11 \cdot \epsilon) \cdot 112| = |\epsilon \cdot 112| = |112| = |0| + |112|$

Proof. គិតដោយ induction លើ $|w|$

► Induction hypothesis: Assume វា ជាការពិត ចំពោះ string y ទី
 $|y| < |w|$, $|y \cdot x| = |y| + |x|$.

→ Case 1: $w = \epsilon$: an empty concatenation

$$w \cdot x = x \quad \text{when } w = \epsilon$$

$$\text{ចំពោះ } |w \cdot x| = |x| = 0 + |x| = |w| + |x|$$

ឆ្លើយតប $|\epsilon| = 0$

→ Case 2: $w = a \cdot y$ ជា string មួយ

an empty concat

$$w \cdot x = a \cdot (y \cdot x)$$

$$\begin{aligned} |w \cdot x| &= |a \cdot (y \cdot x)| \\ &= 1 + |y \cdot x| \quad (\text{question 1.1}) \\ &= (1 + |y|) + |x| \\ &= |w| + |x| \quad (\text{question 1.1}) \square \end{aligned}$$

an induction hypothesis គិតថា $|y \cdot x| = |y| + |x|$ ត្រឹមត្រូវ

Formal languages

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

Examples:

Careful...

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

And ε is not a language.

empty language



language of empty string



How to describe languages? \leftarrow "set" or string. $\left\{ \begin{array}{l} \{ \dots \} \\ \boxed{x: \dots} \end{array} \right.$

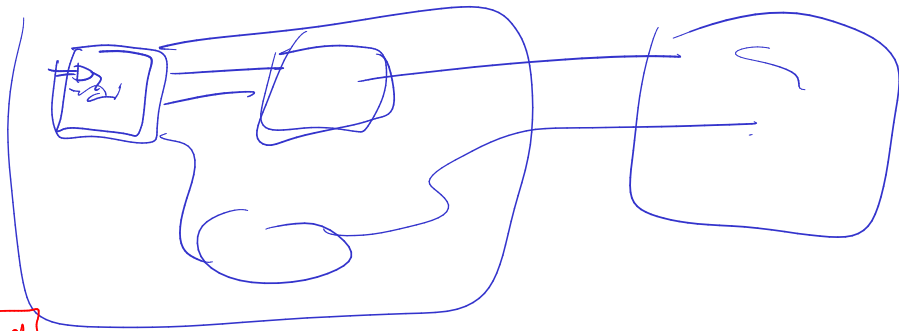
$\rightarrow \bullet \phi$

$\rightarrow \bullet$ set of string 1 string $\{ \text{loveyou} \}$

\vdots

Composition (concept)

← સરવાળાં જુલો જોઈ શકાય



Set point

$A \cup B$, $A \cap B$, $A - B$

↑ ↑

Image

Combining languages

If A and B are languages over alphabet Σ .

"hello" • "setha"
= "hello setha"

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

► Concatenation: $A \cdot B$.
set set

$A = \{\text{"hello"}, \text{"goodbye"}, \epsilon\}$

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

$B = \{\text{"setha"}, \text{"pom"}, \text{"ink"}\}$

$$\begin{aligned}\phi \cdot B &= \phi \\ \{\epsilon\} \cdot B &= B\end{aligned}$$

► Kleene closure or Kleene star: A^* .

String $w \in A^*$ if

(1) $w = \epsilon$

(2) $w = x \cdot y$ જો $x \in A$ and $y \in A^*$

Also $A^+ = A \cdot A^*$

$$A \cdot A \cdot A \cdot A$$

Examples

String $w \in A^*$ ជា

(1) $w = \epsilon$ ←

(2) $w = x \cdot y$ តាំងបើ $x \in A$ ឬ $y \in A^*$

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

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$$

$$A = \{a, ab, c\}$$

$$A^* = \{\underline{\epsilon}, a\epsilon, ab\epsilon, c\epsilon, aa, aab, aabacab, \dots\}$$

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.



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.

R is a
regular expression

- $R = \emptyset$
- R is a string
- $R = X + Y$ X & Y are
- $R = XY$
- $R = X^*$

Examples

$\{\}$, $\{ink\}$, $\{pm\}$

$\{ink\} \cup \underbrace{\{\{taksin\} \cup \{prayut\}\}}_{reg}$

$\{ink\} \cdot \underbrace{(\{pm\} \cup \{hello\})}$

$\{0\}^* = \{\epsilon, 0, 00, 000, \dots\} = \{0^n \mid n \geq 0\}$

$\{0,1\}^* = \text{bit string strings}$

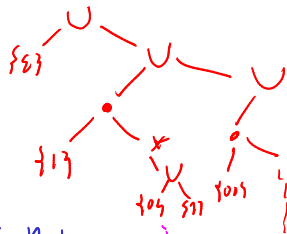
bit string like '010' 010 ← regular

$\underbrace{\{0,1\}^*}_{reg} - \underbrace{\{010\}}_{reg}$

$= \{ \epsilon \} \cup (\{1\} \cdot \{0,1\}^*) \cup (\{00\} \cdot \{0,1\}^*) \cup (\{011\} \cdot \{0,1\}^*) \cup \underline{\{01\}}$

$\Sigma = \{a, b, c, \dots, z\}$

$\{\epsilon\} \cup (\{1\} \cdot \{0,1\}^*) \cup (\{00\} \cdot \{0,1\}^*) \cup (\{011\} \cdot \{0,1\}^*) \cup \underline{\{01\}}$

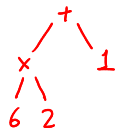


Regular expressions

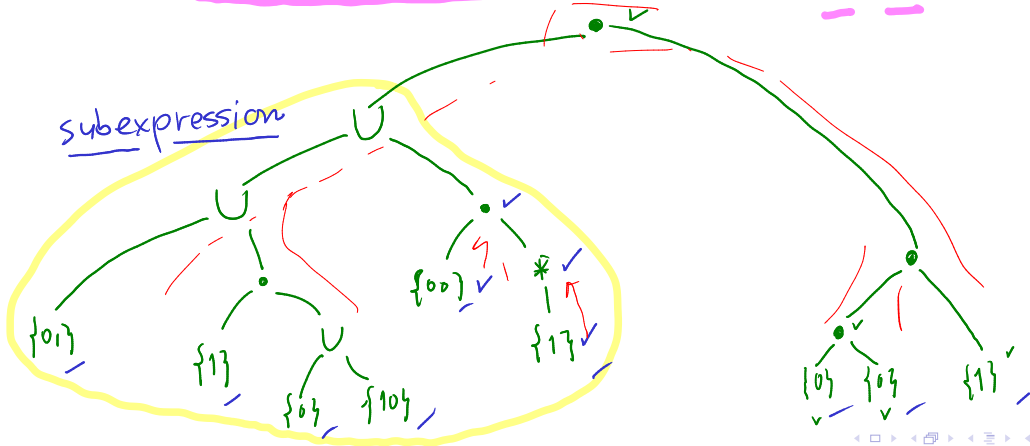
Let $\Sigma = \{0, 1\}$. Consider

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

$$6 \times 2 + 1$$



subexpression



Regular expressions

001^{*}00

(0+1)^{*}

Regular language

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

is represented as

↓

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

Regular expressions

- ▶ omit braces around one-string sets
- ▶ use + instead of ∪
- ▶ omit •
- ▶ follow the precedence: Kleene star operator *, • (implicitly), and +.

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

$A \cdot B \cdot C$

$(A \cdot B) \cdot C$ $A \cdot (B \cdot C)$

$A + B \cdot C$

$(A + B)C$

Regular expressions: examples 1

Regular expressions: examples 2

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

Regex is everywhere

Regex

Search

Proofs about regular expressions - structural induction

- Given regular expression R then

Induction Hypothesis: Assume it's given any subexpression X of R , _____ it's given

Case 1: $R = \emptyset$

Case 2: R is a string

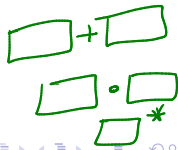
Case 3: $R = \underline{X} + \underline{Y}$ where X & Y are regex

Case 4: $R = \underline{X} \cdot \underline{Y}$ where X & Y are regex

Case 5: $R = \underline{X}^*$ where X is a regex

✓
 \emptyset

✓
hello



Proofs about regular expressions - structural induction

induction

$$P(i) = \text{" } ______ i ______ \text{"}$$

$$P(0), P(1), P(2), \dots$$

$$\boxed{\forall i \in \mathbb{N}, P(i)}$$

$$0 \Rightarrow 1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 6$$

$$P(2) \Rightarrow P(3)$$

$$\boxed{\forall i \geq 0, P(i) \Rightarrow P(i+1)}$$

induction

w interesting

or Case 1: $w = \epsilon$

Case 2: $w = a \cdot x$ for x interesting.

$$w = 01011010 \cdot \epsilon$$

Case 1: $P(\epsilon)$ *

$$\boxed{\forall x. P(x) \Rightarrow P(a \cdot x)}$$

$\forall a \in \Sigma$

Case: $w = a \cdot x$

Lemma 2

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

Proof. พิสูจน์ว่า reg expression R ที่ไม่ใช้ \emptyset

๑: พิสูจน์ว่า R บรรยายภาษาที่ไม่ empty

Case 1: $R = \emptyset$ กรณีนี้ไม่ใช้ \emptyset เลย และ \emptyset ไม่เป็นภาษา

Case 2: R เป็น string ธรรมดา, สังเกตว่า R บรรยายภาษาที่ไม่ empty ธรรมดา

Case 3: R เป็น union ของ 2 regular expression
ที่สร้างจาก regular expression X กับ Y ที่
 $R = X + Y$

Case 4: R เป็น concat ของ X กับ Y ที่ $R = X \cdot Y$
ทั้ง X และ Y เป็น reg ex

Case 5: $R = X^*$ สังเกตว่า X เป็น reg ex

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: $R = \emptyset$.

במקרה זה: R לא \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.

מסתבר כי כל ביטוי רגולרי שאינו \emptyset מייצג שפה לא ריקה
 R אינו \emptyset ולכן הוא מייצג שפה לא ריקה

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.

$L(R)$ language defined by R

Proof. (cont.2/4)

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

if R is symbol \emptyset , S and T is \emptyset

I.H. S and T are non empty language

$L(R)$ is union of non empty set,
 $L(R)$ is non-empty.

Proof. (cont.3/4)

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

បើ R ជា symbol \emptyset , S ឬ T ជា \emptyset ផង

ឡើយ I.H. S ឬ T មាន non empty language

យើងដឹង $s \in L(S)$, $t \in L(T)$

ដូច្នេះ $s \cdot t \in L(S) \cdot L(T)$

ដូច្នេះ $s \cdot t \in L(R)$ ដូច្នេះ R ជា non empty

Proof. (cont.4/4)

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


กรณีนี้ $\epsilon \in L(S^*) = L(R)$ ดังนั้น $L(R)$ ไม่ empty.

Proof. (cont.4/4)


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

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

Lemma 2

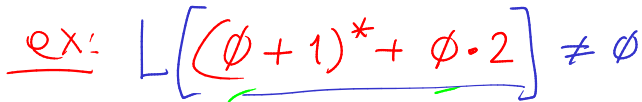
Every regular expression that does not use the symbol \emptyset represents a non-empty language. 

Lemma 3

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

Proof: Given regular language L for which L is regular
there is regular expression (R) that represent L .

ex: $L[(\emptyset + 1)^* + \emptyset \cdot 2] \neq \emptyset$



∴ R' is \emptyset -free

∴ $L(R') = L(\text{---})$

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 *if $L(R)$ is non-empty regular language.*

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?

Structural

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?

- ▶ $R = \emptyset$ ✗ *in the induction we assume that $L(R) \neq \emptyset$*
- ▶ R is a string ✓ *if R is a string, then $R' = R$ b/c R is \emptyset -free*
- ✗ ▶ $R = S + T$ *if S & T are regular exp.* b/c $L(R) \neq \emptyset$
- ▶ $R = S \cdot T$ *if S & T are regular exp.* if $L(S) \cup L(T) \neq \emptyset$
- ▶ $R = S^*$ *if S is a regular exp.* Case 1: $L(S) \neq \emptyset, L(T) \neq \emptyset$

Case 2: if S & T are \emptyset -free.
 if S is \emptyset -free, then $L(S) \neq \emptyset$
 I.H. if S' is \emptyset -free, then $L(S') = L(S)$
 let $R' = S'$

by I.H. if S' is \emptyset -free, then
 $L(S') = L(S)$
 b/c if T' is \emptyset -free, then
 $L(T') = L(T)$
 let $R' = S' + T'$

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 .
- ▶ $R = S \cdot T$ 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:

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

For a language \underline{L} , the reversal of L is defined as

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

You may assume the following facts.

- ▶ $\underline{L^*} \cdot \underline{L^*} = \underline{L^*}$ 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 .

$$L = \{\text{hello, world, good, bad}\}$$

$$L^R = \{\text{olleh, dlrow, doog, dab}\}$$

Prove that if L is regular,
 L^R is regular.

• Can assume $(L^*)^R = (L^R)^*$

$$\begin{aligned} & (11244)^R \\ & (1 \cdot 1244)^R \\ & \quad \parallel \\ & (1244)^R \cdot 1 \\ & \quad \parallel \\ & (244)^R \cdot 1 \cdot 1 \\ & (44)^R \cdot 2 \cdot 1 \cdot 1 \\ & (4)^R \cdot 4211 \\ & (\varepsilon)^R 44211. \end{aligned}$$

$$\text{def: } L^R = \{w^R \mid w \in L\}$$

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

Proof: Assume $w \in (L^R)^*$, then $w \in (L^*)^R$.

I.H. Assume for any substring y of w in $y \in (L^R)^*$ then $y \in (L^*)^R$
if $y \neq w$

Case 1: $w = \epsilon$

then $\epsilon \in L^*$, $\epsilon = \epsilon^R \in (L^*)^R$.

(love you)^R
(uoy)(evol)

Case 2: $w = x \cdot y$ then $x \in L^R$ then $y \in (L^R)^*$

then $x \in L^R$ if u is $x = u^R$ then $u \in L \Rightarrow x^R \in L$
then I.H. then $y \in (L^R)^*$ then $y \in (L^*)^R$ then $x^R \in L^*$ ①

then $y^R \in L^*$ ②

then ① then ② $[x^R \in L^*, y^R \in L^*]$ then $y^R \cdot x^R \in L^*$

then $y^R \cdot x^R = (x \cdot y)^R$ then $(x \cdot y)^R \in L^*$
then $x \cdot y \in (L^*)^R$

no $x \cdot y \in (L^R)^*$
 $(L^*)^R$

Goal: $x \cdot y \in (L^*)^R \Leftrightarrow (x \cdot y)^R \in L^*$