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

Jittat Fakcharoenphol

August 20, 2024

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

What is computation?

Models of computations

- ทำอะไรได้บ้าง

$\boxed{\text{ทำอะไรไม่ได้}}$

model 1 $\xleftrightarrow{\text{equivalence}}$ model 2

ง่าย/ ไม่เก่งมาก

เก่ง

Languages = specifications

formal language → set ของ string

อยากให้ตอบว่า yes accept :)

ภาษา (language)

→ ภาษาไทย, อังกฤษ } natural language (NLP)

$\{x \mid x$ เป็นจำนวนเฉพาะ$\}$

$\{x \mid x$ เป็นจำนวนคู่$\}$

$\{x : x$ เป็น palindrome$\}$

$\{p : p$ เป็นโปรแกรมภาษา python ที่รับ input แล้วทำงานไม่สิ้นสุด$\}$

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

$\varepsilon \neq \emptyset$

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, $\Sigma = \{0, 1, \ldots, 9\}$; 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

$|011\cdot112| = |011| + |112|$

$|(1\cdot\varepsilon)\cdot112| = |1| + |112|$

$|\varepsilon\cdot112| = |112| = |0| + |112|$

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

For strings w and x, $|w \bullet x| = |w| + |x|$. $|(11\cdot\varepsilon)\cdot112| =$

Proof. จะพิสูจน์ by induction บนโครงสร้างของ $\boxed{w}$

- Induction hypothesis: Assume ว่า สำหรับทุก string y ที่ $|y| < |w|$, $|y\bullet x| = |y| + |x|$.

Case 1: $w = \varepsilon$: จากนิยามของ concatenation

$w\bullet x = x$ เมื่อ $w = \varepsilon$

ดังนั้น $|w\bullet x| = |x| = 0 + |x| = |w| + |x|$ ตามต้องการ ✓

เพราะจากนิยามของ $|\varepsilon| = 0$

Case 2: $w = a\cdot y$ สำหรับบาง string y

จากนิยามของ concat

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

จาก induction hypothesis จะได้ว่า $|y\bullet x| = |y| + |x|$ ดังนั้น

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

$= 1 + |y\bullet x|$ (จากนิยาม 1.1)

$= (1 + |y|) + |x|$

$= |w| + |x|$ (จากนิยาม 1.1) □

Formal languages

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

Examples:

Careful...

empty language

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

And $\underline{\varepsilon}$ is not a language.

language of empty string

How to describe languages?

← "set" ของ string.

$\{ \mid \ldots \}$

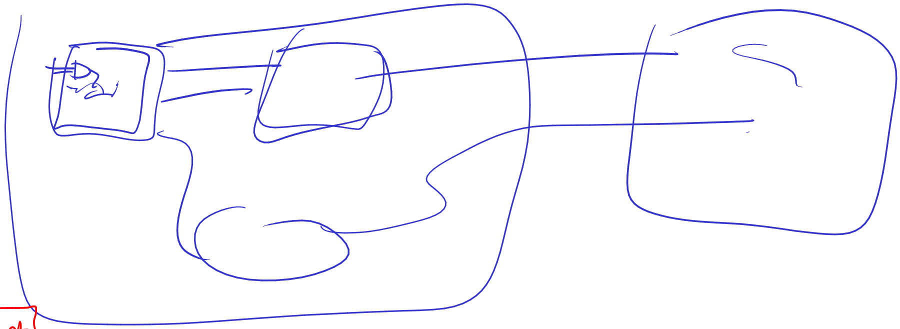
$\{ x : \text{——} \}$

- $\emptyset$
- set ที่มี string 1 string $\{\text{loveyou}\}$

⋮

Composition (concept)

← สร้างของที่ซับซ้อน จากของที่ง่าย



Set operation

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

Image

Combining languages

"hello" • "setha"

= "hello setha"

If A and B are languages over alphabet Σ.

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

$A = \{$"hello", "goodbye", $\varepsilon\}$

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

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

$\emptyset \bullet B = \emptyset$

$\{\varepsilon\} \bullet B = B$

- Kleene closure or Kleene star: A^*.

String $w \in A^*$ ถ้า

(1) $w = \varepsilon$

(2) $w = x \bullet y$ สำหรับ บาง $x \in A$ และ $y \in A^*$

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

$A \bullet A \bullet A \bullet A$

Examples

string $w \in \boxed{A^*}$ ถ้า

(1) $w = \varepsilon \leftarrow$

(2) $w = x \cdot y$ สำหรับ บาง $x \in A$ และ $\underline{y \in A^*}$

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

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

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

$A^* = \{\underline{\varepsilon}, a\cdot\varepsilon, ab\cdot\varepsilon, c\cdot\varepsilon, aa, aaa, aabacab, \ldots$

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

R is a regular expression

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.

- $R = \phi$
- R is a string
- $R = X + Y$. where X & Y are
- $R = X \cdot Y$.
- $R = X^*$.

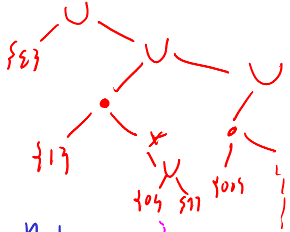
Examples

$\Sigma = \{a, b, c, d, \ldots, z\}$

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

$\{\varepsilon\} \cup (\{1\} \bullet \{0,1\}^*) \cup (\{00\} \bullet \{0,1\}^*)$
$\cup (\{011\} \bullet \{0,1\}^*) \cup \{01\}$

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



$\{ink\} \bullet (\{pm\} \cup \{hello\})$

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

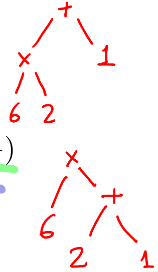
$\{0,1\}^* =$ bit string ทั้งหมด

bit string ทั้งหมด ที่ไม่มี 010 ← regular

$* \ \underline{\overbrace{\{0,1\}^*}^{reg} - \overbrace{\{010\}}^{reg}} = \boxed{\{\varepsilon\} \cup (\{1\} \bullet \{0,1\}^*) \cup (\{00\} \bullet \{0,1\}^*) \cup (\{011\} \bullet \{0,1\}^*) \cup \{01\}}$

Regular expressions

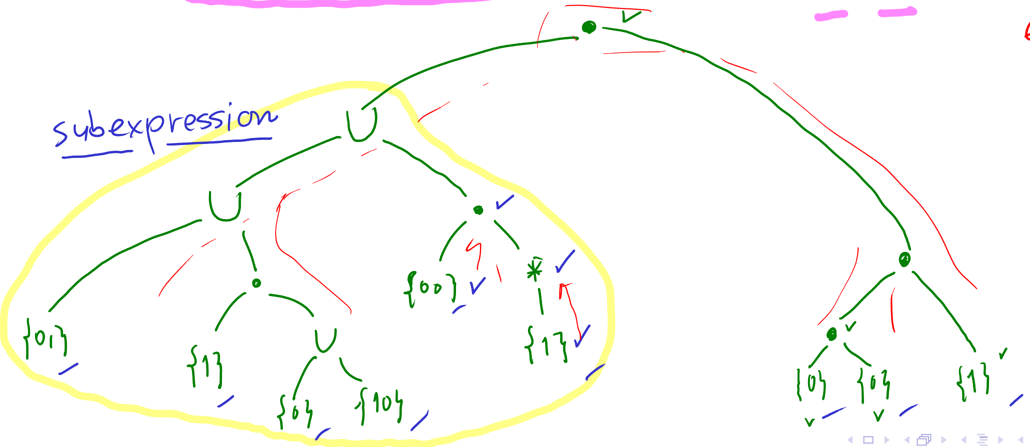
$6 \times 2 + 1$



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

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

subexpression



Regular expressions

001^*00 $\qquad (0+1)^*$

Regular language

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

is represented as

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

$A \cdot B \cdot C$

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

Regular expressions

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

$A + B \cdot C$

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

$(A+B)C$

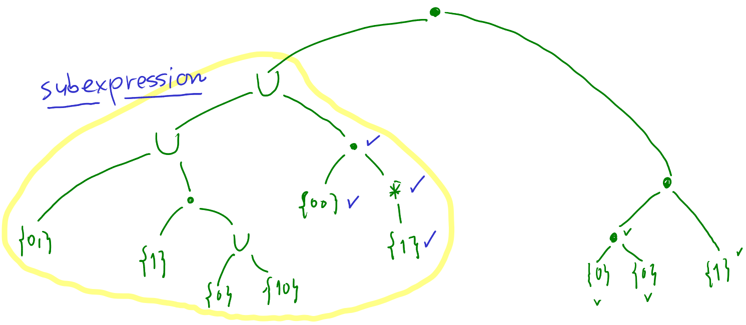
Regular expressions: examples 1

Regular expressions: examples 2

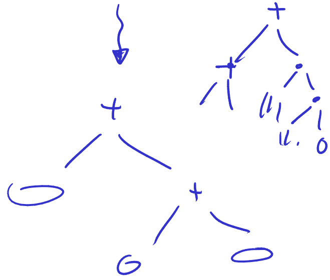
All strings over $\{0, 1\}$ except 010.

Subexpressions

$0101^* + 01 + 110$



ทราบ tree



ถ้าดูทุกตัว subexpression ของ string , มีได้หลายแบบ.

Regex is everywhere

Proofs about regular expressions - structural induction

- พิจารณา regular expression R ใดๆ

Induction Hypothesis: Assume ว่า สำหรับทุก subexpression X ของ R, ____________ เป็นจริง

Case 1: $R = \emptyset$

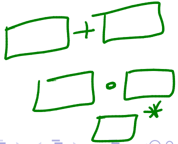
Case 2: R เป็น string ใดๆ

Case 3: $R = X + Y$ เมื่อ X & Y เป็น regex

Case 4: $R = X \circ Y$ เมื่อ X & Y regex

Case 5: $R = X^*$ เมื่อ X เป็น regex

✓ $\emptyset$ ✓ hello



Proofs about regular expressions - structural induction

induction

$P(i) = $ "___ i ___"

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

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

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

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

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

induction

w is a string

or Case 1: $w = \varepsilon$

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

$w = 0\,1\,0\,1\,1\,0\,1\,0 \cdot \varepsilon$

Case 1: $P(\varepsilon)$

Case 2: $\boxed{\forall x.\ \forall a \in \Sigma\ P(x) \Rightarrow P(a \cdot x)}$

Case: $w = a\,x$

Lemma 2

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

Proof. พิจารณา reg expression $\boxed{R}$ ที่ไม่มีเครื่องหมาย ϕ
จะพิสูจน์ว่า R แทนภาษาที่ไม่ empty

Case 1: $R = \phi$ กรณีนี้เป็นไปไม่ได้ เพราะว่า R ไม่มีเครื่องหมาย ϕ

Case 2: R เป็น string บางตัว, สังเกตว่า R แทนภาษาที่ไม่ empty ตามต้องการ

Case 3: R เป็น union ของ 2 regular expression ที่เล็กกว่า regular expression X และ Y ที่

$R = X + Y$ . . . .

Case 4: R เป็น concat ของ X กับ Y ที่เล็กกว่า $R = X \circ Y$
ทั้งคู่ reg ex . . . . . .

Case 5: $R = X^*$ สำหรับ X ที่เป็น reg expr . . . .

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: $R = \emptyset$.

เป็นไปไม่ได้ เพราะ สมมุติว่า R ไม่ใช้ $\emptyset$

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.

R จะแทนภาษาที่ ไม่ empty เพราะ มีสมาชิก 1 ตัว ที่เป็นสมาชิกของภาษา

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.

ให้ L(R) language ที่แทนด้วย R

Proof. (cont.2/4)

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

เนื่องจาก R ไม่มี symbol $\emptyset$, S และ T ไม่มี $\emptyset$ ด้วย

จาก I.H. S และ T แทน non-empty language

เนื่องจาก L(R) เป็น union ของ non-empty set,

L(R) จะเป็น non-empty ด้วย.

Proof. (cont.3/4)

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

เนื่องจาก R ไม่มี symbol $\emptyset$, S และ T ไม่มี $\emptyset$ ด้วย

จาก I.H. S และ T แทน non empty language

ดังนั้น มี $s \in L(S)$, $t \in L(T)$

จากนิยามของ $S \bullet T$ จะได้ว่า $s \bullet t \in L(S) \bullet L(T)$

ดังนั้น $s \cdot t \in L(R)$ ดังนั้น R ไม่ empty

Proof. (cont.4/4)

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

จากนิยามของ $*$, $\varepsilon \in L(S^*) = L(R)$ 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: พิจารณา regular language L ใดๆ เนื่องจาก L เป็น regular
จะมีบาง regular expression R ที่ represent L.

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

จะหา R' ที่ $\emptyset$-free
ที่ $L(R') = L(\underline{\qquad})$.

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 ที่แทน non-empty regular language ใดๆ .

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 structural 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?

- $R = \emptyset$ ✗ เป็นไปไม่ได้ เพราะ assumption ว่า $L(R) \neq \emptyset$
- R เป็น string ✓ จะได้ว่า $R' = R$ และ R' ไม่มีเครื่องหมาย $\emptyset$
- * $R = S + T$ เมื่อ S & T เป็น regular exp.
- $R = S \bullet T$ เมื่อ S & T ———
- $R = S^*$ เมื่อ S เป็น reg exp.

แต่เนื่องจาก $L(R) \neq \emptyset$
จะได้ว่า $L(S) \cup L(T) \neq \emptyset$

Case 1: $L(S) \neq \emptyset$, $L(T) \neq \emptyset$
จาก I.H. มี S' ที่ $\emptyset$-free และ $L(S') = L(S)$
และ มี T' ที่ $\emptyset$-free, $L(T') = L(T)$
ให้ $R' = S' + T'$

Case 2: มีแค่ S หรือ T ที่เป็น nonempty.
สมมติว่า เป็น S ที่ $L(S) \neq \emptyset$
I.H. มี S' ที่ $\emptyset$-free ที่ $L(S') = L(S)$
ให้ $R' = S'$ ———

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?

- $R = \emptyset$
- 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:

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

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.

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

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

$(11244)^R$

$(1 \cdot 1244)^R$

$=$

$(1244)^R \bullet 1$

$=$

$(244)^R \bullet 1 \cdot 1$

$(44)^R \bullet 2 \bullet 1 \cdot 1$

$(4)^R \cdot 4211$

$(\varepsilon)^R 44211.$

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

- Can assume $(L^*)^R = (L^R)^*$

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

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

Proof: พิจารณา $w \in (L^R)^*$, จะแสดงว่า $w \in (L^*)^R$.

I.H. Assume ว่า ทุกๆ substring y ของ w ถ้า $y \in (L^R)^*$ แล้ว $y \in (L^*)^R$
ที่ $y \neq w$

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

$(\text{uoy})(\text{evol})$

Case 1: $w = \varepsilon$
เนื่องจาก $\varepsilon \in L^*$, $\varepsilon = \varepsilon^R \in (L^*)^R$.

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

เนื่องจาก $x \in L^R$ จะมี u ที่ $x = u^R$ และ $u \in L \Rightarrow x^R \in L$
และ $x^R \in L^*$ ①

จาก I.H แล้วดู $y \in (L^R)^*$ จะได้ว่า $y \in (L^*)^R$
นั่นคือ $y^R \in L^*$ ②

จาก ① และ ② $[x^R \in L^*, y^R \in L^*]$ จะได้ว่า $y^R \cdot x^R \in L^*$

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

หรือ $x \cdot y \in (L^*)^R$ ตามต้องการ ∎

ทด $x \cdot y$ L^R $(L^R)^*$

$(L^*)^R$

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