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

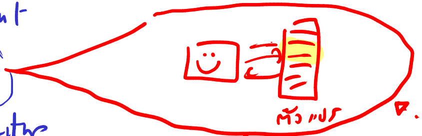
Jittat Fakcharoenphol

August 22, 2023

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

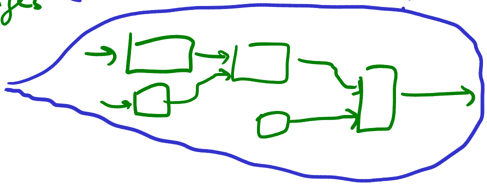
What is computation?

- เขียนโปรแกรม python → run (input → output)



model ของการคำนวณ

- Functional prog. languages
  Haskell, Lisp,
  JavaScript



เปลี่ยน model ใหม่

→ classical physics

→ quantum physics →

"quantum computer"

เปลี่ยน model ใหม่

Models of computations

Machines
- finite automata
- Turing machines

λ-calculus (functional prog)

grammars
- regular expressions
- context-free grammar

วิธีการระบุ $\boxed{\text{pattern}}$

ภาษา

Languages = specifications

(อังกฤษ จีน ญี่ปุ่น (natural language NLP.

formal language.

||

set ของ strings ที่นิยามชัดเจน

ex. { w | w แทนเลขโรมัน }

{ p | p เป็นจำนวนเฉพาะ เป็นรูป ... — }

{ p | p แบบ palindrome }

{ s | s เป็นสตริงที่ประกอบ '(' ')' ที่จับคู่กันได้ }

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.

good g · o · o · d

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

base case

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

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, $|w \cdot x| = |w| + |x|$.

Proof.

จะพิสูจน์โดยอุปนัย.

Induction Hypothesis (IH): " สำหรับทุก string s ที่ $|s| < |w|$
$|s \bullet x| = |s| + |x|$. "

Case 1: $w = \varepsilon$ $\qquad |w \bullet x| = |x|$ ตามนิยาม

$= 0 + |x|$

$= |w| + |x|$ ตามนิยาม length ✓

Case 2: $w = a \cdot y$
เมื่อ $a \in \Sigma$. . . . $\qquad |w \bullet x| = |(a \cdot y) \bullet x| = |a \cdot (y \bullet x)|$

$= 1 + |y \bullet x|$ นิยาม len

$= \underbrace{1 + |y|}_{\vdots} + |x|$ จาก IH

$\square$

Formal languages

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

Examples:

$\Sigma = \{0, 1\}$, $\Sigma^* = \{$ strings ทั่วๆ

0

$\{ w \in \Sigma^* \mid w$ ขึ้นต้นด้วย $1 \}$ ex. $\{1, 10, 1011, \ldots$

$\{ w \in \Sigma^* \mid w$ เป็นจำนวนเฉพาะ $\}$ ex $\{10, 11, 101, 111, \ldots$

rep.

$\Sigma = \{a, b\}$

$\{ w \in \underline{\Sigma^*} \mid w$ เป็น palindrome $\}$ ex. $\{a$ aba aaa, abba

bab, bbabb ...

$\vdots$

$\emptyset$, $\{\varepsilon\}$

Careful...

These are different languages: $\emptyset$, $\{\varepsilon\}$
And ε is not a language.

How to describe languages?

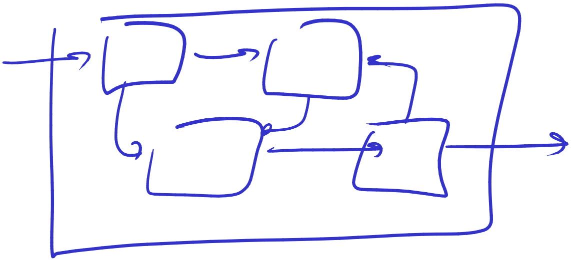
① ระบุด้วยภาษาธรรมชาติ + mathematical notation

$\{ w \mid \_\_\_\_\_\_\_\_ \}$

② วิธีที่ระบุแล้ว → ทำให้คำนวณได้ง่าย . . . .

⓪ $\{0, 01, 11, 1001\}$ ไม่ทั่วถึงทุกตัว. !!

Composition



Combining languages

set minus

If A and B are languages over alphabet Σ.

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

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

$A = \{$ HELLO, GOODBYE $\}$

$B = \{$ PITA, THAKSIN, SETTHA, CHU $\}$

$C = \{1, 2, \ldots\}$

$A \bullet C = \ldots$

- Kleene closure or Kleene star: A^*.

$w \in A^*$ iff

① $w = \varepsilon$

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

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

$= \{\varepsilon, 0, 1, 00, 10, 01, 11, \ldots$

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

Examples

Regular languages

Notes

(10 x (20 + 30)) - 5

Definition: regular languages

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

- L is empty; $\emptyset$
- L contains one string (can be the empty string ε); {here}
- 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 $\Sigma = \{0,1\}$ $\cancel{\{01111\ldots\}}$

ϕ

$\{\varepsilon\}, \{0\}, \{010\}, \{10\}, \{0001\}, \{1001\}, \{011111\}$

$\phi \cup \{0\}, \underbrace{\{010\} \cup \{10\}}, \{0001\} \cup (\{010\} \cup \{10\})$

$\{0\} \bullet \{010\}, \phi \bullet \{10\}, \{\varepsilon\} \bullet \{10\}, (\{010\} \cup \{10\}) \bullet (\{100\} \cup \{10\})$

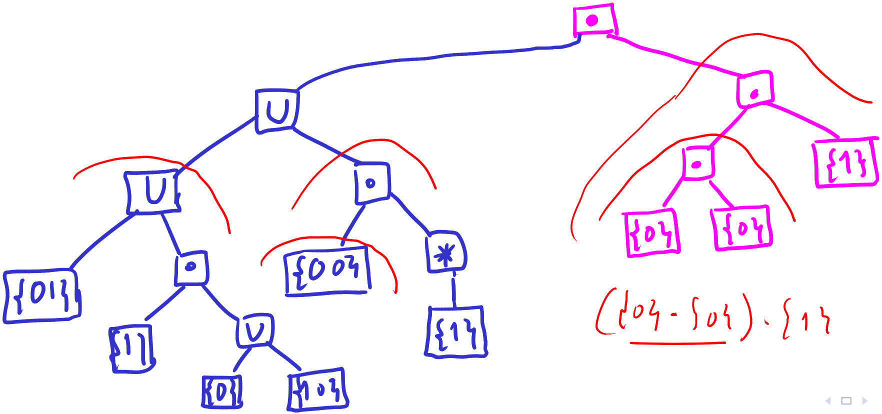
$\{0\} \bullet \{010\} = \{00010\}$, $\phi \bullet \{10\} = \cancel{\{10\}}\ \phi$, $\{\varepsilon\} \bullet \{10\} = \{10\}$

$$\left[\left[(\{01\} \cup \{10\}) \bullet \{1\}\right]^* \bullet \{0\} \bullet (\{01\} \cup \{10\})\right]^*$$

Regular expressions

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

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



regular expression tree

Regular expressions

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 $\cup$
- omit $\cdot$
- follow the precedence: Kleene star operator $*$, $\cdot$ (implicitly), and $+$.

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

Regular expressions: examples 1

$\cancel{0}\cancel{0}\underline{1}0$

เลขคู่

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

$$(0+1)^*$$

00^* 000^* $00; 000$ $(00)^*$

$0, 00, 0^n$

Regular expressions: examples 2

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

$\Sigma^* - \{010\}$ 1010

$$\boxed{\varepsilon + 1(0+1)^* + 0(\varepsilon + 0(0+1)^*) + 01(\varepsilon + 1(0+1)^*) + 010(0+1)(0+1)^*}$$

- ε
- ขึ้นต้น 1
- ขึ้นต้น 00 ...
- ขึ้นต้น 01 ...
- ขึ้นต้น 010 + ...

$(01+1)^* \dashrightarrow \varepsilon, 01, 1$

011 0101 011111

$(ab+c)^*$

ε, ab, abab, ababc, abcab, ccc, abccabccab, ...

$(10+0)^*$ 0, ε, 00010, 0010100010, ...

Subexpressions

$$0110(1+0)(1+01)^*$$

$\{0110\}$ $\{01\} \cdot \{10\}$ $\{0\} \cdot \{1\} \cdot \{1\} \cdot \{0\}$

$$\overbrace{\left(0110(1+0)\right)}^{\text{sub expression}} \cdot \overbrace{(1+01)^*}^{\text{subexpression}}$$

$$0110 \cdot (1+0)$$

Regex is everywhere

Proofs about regular expressions - structural induction

(I.H.) $\boxed{\quad}$ נכון לכל תת subexpr של R

R

Case 1: $R = \emptyset$ ✓

Case 2: $R = s$ ✓

Case 3: $R = S + T$ כאשר S & T הם reg ex.

Case 4: $R = S \circ T$

Case 5: $R = S^*$

Lemma 2

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

Proof.

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

$\emptyset$

$0 + 1$

0^*

ε

Lemma 2

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

$P(R)$ $Q(R)$

$\forall R\, [P(R) \Rightarrow Q(R)]$

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: $R = \emptyset$. x contradiction ขัดแย้งกับ เงื่อนไขเริ่มต้น ว่า R ไม่ใช้ ϕ

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 แทน non-empty language แน่นอน

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.

$L(R) \neq \phi$

Proof. (cont.2/4)

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

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

เพราะ S & T เป็น subexpression ของ R.

ดังนั้น S represent non-empty language จาก I.H.

และ T represent non-empty lang. จาก I.H.

$L(S) \neq \emptyset$

$L(T) \neq \emptyset$

ดังนั้น เนื่องจาก R แทน language ที่เป็น union ของ non-empty lang, R จะ represent non-empty lang ด้วย..

$L(R) \neq \emptyset$

Proof. (cont.3/4)

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

$\Rightarrow$ S & T แทน non-empty languages L_S & L_T

เรียก language ที่แทนด้วย S ว่า A.

เรียก language ที่แทนด้วย T ว่า B.

ดังนั้น R จะแทน language $A \circ B$ — $L(S) \bullet L(T)$

เนื่องจาก A ไม่ empty มี $x \in A$

B ไม่ empty มี $y \in B$

ให้ x ใน $L(S)$

y ใน $L(T)$

ดังนั้น $x \circ y \in A \circ B$ (ตามนิยาม $\circ$)

$x \cdot y \in L(S) \bullet L(T)$

นั่นคือ language ที่แทนด้วย R จะไม่ empty

$L(R) \neq \phi$

Proof. (cont.4/4)

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

เนื่องจาก $\varepsilon \in$ language ที่ออกแบบด้วย $\underline{S^*}$.

R แทน language ที่ไม่ empty.

$\varepsilon \in L(S^*)$

ดัง

$L(R) \neq \emptyset$

Proof. (cont.4/4)

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

นิยาม สำหรับ reg ex R ใดๆ ให้ $L(R)$ แทน language ที่ represent โดย R

ex: 1 $L(1) = \{1\}$

$1(0+1)$ $L(1(0+1)) = \{10, 11\}$

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 $\emptyset$.

$$\forall L\,[\,\underbrace{Q(L)}_{L \neq \phi} \Rightarrow P(L)\,]$$

$$\exists R\,[\,L(R) = L \text{ וגם: } R \text{ אינו symbol } \phi\,]$$

$$\boxed{\neg Q(L) \vee P(L)}$$

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 $\emptyset$.

พิจารณา regular language L ซึ่งไม่ ..

ที่แทน L

Let R be a regular expression. We prove that if $\underbrace{L(R)} \neq \emptyset$, 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?

สำหรับทุก subexpression S ของ R

$\boxed{\text{ถ้า } L(S) \neq \emptyset \text{ จะมี reg ex } S' \text{ ที่ } S' \text{ ไม่มี symbol } \emptyset \text{ และ } L(S') = L(S)}$

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

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?

Case 1: $R = \phi$

Case 2: R เป็น single sym.

Case 3: $R = S + T$ สำหรับ regex S & T

Case 4: $R = S \bullet T$ สำหรับ regex $S \bullet T$

Case 5: $R = S^*$ สำหรับ regex S'

ถ้า $L(R) \neq \emptyset$, จะมี R' ที่ $L(R) = L(R')$ และ R' ไม่มี symbol $\emptyset$.

$(\emptyset + 1) \cdot 0^*$

$1 \cdot 0^*$

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?

- $R = \emptyset$, $L(R) = \emptyset$ ✗
- 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.

ถ้า $L(S) = \emptyset$, $L(T) \neq \emptyset$

ได้ case 4 cases.

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

จะมี S' และ T' ที่ไม่มี $\emptyset$

ได้ $R' = S' + T'$

case 1: $L(S) = \emptyset$ $\rightarrow$ ได้ $\boxed{R' = \varepsilon}$

$\boxed{\text{case 2: } L(S) \neq \emptyset}$ จะมี S' ที่ $L(S') = L(S)$ และ S' ไม่มี $\emptyset$

$R' = (S')^*$

HELLO^{R} = OLLEH

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

$\{\varepsilon, 0, 01, 100\}^R = \{\varepsilon, 0, 10, 001\}$

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

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.

ถ้า L เป็น regular language
L^R เป็น regular ...

$\boxed{(L^*)^R = (L^R)^*}$

Δ พิจาร language L ที่ represent ด้วย regular expression R. จะแสดงว่า L^R เป็น regular language
วิธีสร้าง reg. ex S ที่ $L(S) = L^R$

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