01204211 Discrete Mathematics
Lecture 9b: Nonregular languages[1]

Jittat Fakcharoenphol

August 26, 2024

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

DFA: Formal definitions

A **finite-state machine** or a **deterministic finite-state automaton** (DFA) has five components:

- the input alphabet Σ,
- a finite set of states Q,
- a transition function $\delta : Q \times \Sigma \longrightarrow Q$
- a start state $s \in Q$, and
- a subset $A \subseteq Q$ of accepting states.

Acceptance

One step move: from state q with input symbol a, the machine changes its state to $\delta(q,a)$.

Extension: from state q with input string w, the machine changes its state to $\delta^*(q,w)$ defined as

$$\delta^*(q,w) = \begin{cases} q & \text{if } w = \varepsilon, \\ \delta^*(\delta(q,a),x) & \text{if } w = ax. \end{cases}$$

The signature of δ^* is $Q \times \Sigma^* \longrightarrow Q$.

accepting w

For a finite-state machine with starting state s and accepting states A, it accepts string w iff

$$\delta^*(s,w) \in A.$$

Language of a DFA

$L(M)$

For a DFA M, let $L(M)$ be the set of all strings that M accepts. More formally, for $M = (\Sigma, Q, \delta, s, A)$,

$$L(M) = \{w \in \Sigma^* \mid \delta^*(s, w) \in A\}.$$

We refer to $L(M)$ as the language of M.

Automatic languages[2]

M_1 accepts L_1
M_2 accepts L_2 $\Big\}$ ⇒ M accepts $\boxed{L_1 \cap L_2}$

Definition (for now)

A language L is **"automatic"** if there is a DFA M such that $\boxed{L(M)} = L$.

Lemma 1

If L_1 and L_2 are automatic languages over alphabet Σ, then

- $L_1 \cap L_2$,
- $L_1 \cup L_2$,
- $L_1 \setminus L_2$, *and*
- $\Sigma^* \setminus L_1$,

are also automatic.

The set of automatic languages is $\boxed{\text{closed}}$ under these boolean operations.

[2] Taken directly from Erikson's lecture notes

Other ways to combine DFAs?

Given two languages L_1 and L_2, we can combine them in various ways using Boolean operations (i.e., $\cap$, $\cup$, etc.).
What else can we do?

Other ways to combine DFAs?

Given two languages L_1 and L_2, we can combine them in various ways using Boolean operations (i.e., $\cap$, $\cup$, etc.).
What else can we do?

- Concatenation: $L_1 \bullet L_2$, defined as

Other ways to combine DFAs?

Given two languages L_1 and L_2, we can combine them in various ways using Boolean operations (i.e., $\cap$, $\cup$, etc.).
What else can we do?

- Concatenation: $L_1 \cdot L_2$, defined as

$$\{x \cdot y \mid x \in L_1, y \in L_2\}$$

Other ways to combine DFAs?

Given two languages L_1 and L_2, we can combine them in various ways using Boolean operations (i.e., $\cap$, $\cup$, etc.).
What else can we do?

- Concatenation: $L_1 \bullet L_2$, defined as

$$\{x \bullet y \mid x \in L_1, y \in L_2\}$$

- Kleene closure: L_1^*.

Interesting questions

We know that the set of automatic languages is closed under Boolean operations.

Questions

- Is it closed under concatenation?
- Is it closed under taking Kleene closure?

Interesting questions

We know that the set of automatic languages is closed under Boolean operations.

Questions

- Is it closed under concatenation?
- Is it closed under taking Kleene closure?

Spoiler:

Interesting questions

We know that the set of automatic languages is closed under Boolean operations.

Questions

- Is it closed under concatenation?
- Is it closed under taking Kleene closure?

Spoiler: Yes, it is (for both operations). We will see the proof, after we learn a required new concept.

Closure

Lemma 2

Given two automatic languages L_1 and L_2, the following languages are automatic:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$, *and*
- L_1^*.

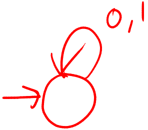
Closure

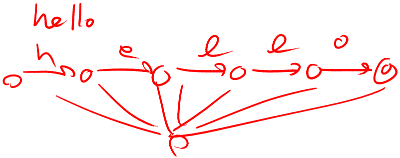
Lemma 2

Given two automatic languages L_1 and L_2, the following languages are automatic:

- $L_1 \cup L_2$,
- $L_1 \cdot L_2$, *and*
- L_1^*.

More over, $\boxed{\emptyset}$ and a language containing a single string are also automatic.





Closure

More over, $\emptyset$ and a language containing a single string are also automatic.

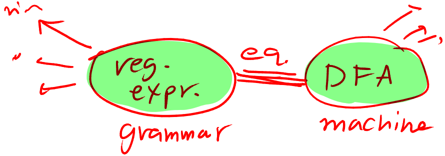
Lemma 2

These are automatic languages

- *The empty set,*
- *A language containing one string,*
- $L_1 \cup L_2$ *for automatic languages* L_1 *and* L_2*,*
- $L_1 \cdot L_2$ *for automatic languages* L_1 *and* L_2*, and*
- L^* *for automatic languages* L

Doese this look familiar?

Regular languages



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.

Thm: Language L is regular iff there exists a DFA $\boxed{M}$ such that $L(M) = L$.

$\Longrightarrow$

Every regular language is automatic

Big question:

$\Rightarrow$

Every regular language is automatic

Big question:

$\Leftarrow$

Is every automatic language regular?

Spoiler:

$\Rightarrow$

Every regular language is automatic

Big question:

$\Leftarrow$

Is every automatic language regular?

Spoiler: Yes, it is. We will see some idea on how to prove this.

$\Rightarrow$

Every regular language is automatic

Big question:

$\Leftarrow$

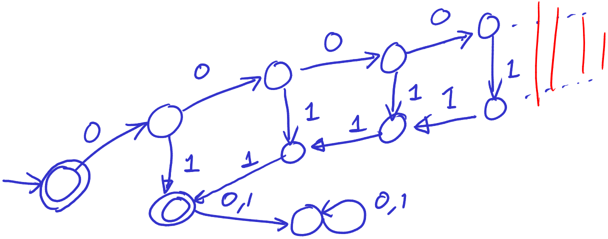
Is every automatic language regular?

Spoiler: Yes, it is. We will see some idea on how to prove this.

Theorem 3

A language L is regular if and only if there exists a DFA M such that $L(M) = L$.

Nonregular languages



Can you design a DFA that accepts strings from language

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

01

000111

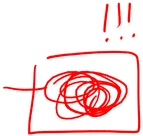
000000111111

Proof:

ถ้าได้ → มี DFA M แสดงตัวอย่าง

ถ้าไม่ได้ DFA M → !!! ???

Key idea



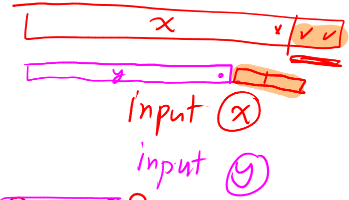
If you have finite states, you can't possibly distinguish between strings in the language and strings not in the language.

state סאיבף

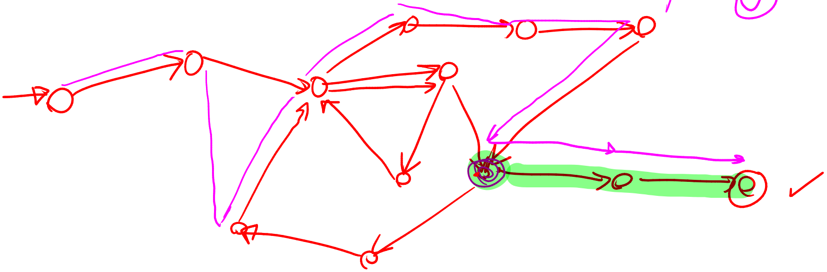
Basic question

How can you show that you need at least two states?

Basic question



How can you show that you need at least two states?
Let's see how a DFA works.

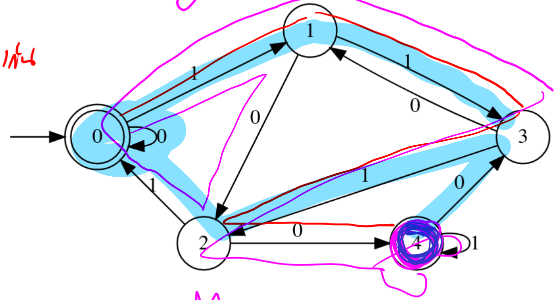


Another example

x : 1110 z

y : 10111101 z

idea:
ถ้า z input ที่ตามหลังไป ถึง state เดียวกัน
ถ้า ตามท้าย เหมือนกัน
⇒ accept / reject
จะแยกกันไม่ได้.



M

ถ้า DFA M เดินได้ไปสิ้นสุดของ x กับของ y และไปถึงที่ state เดียวกัน

M จะ accepts x กับ y ?

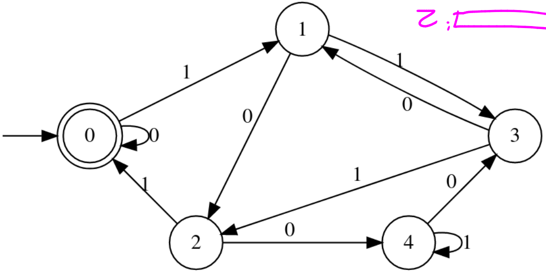
Another example

x:

y:

M reaches the same state for both x & y.

z:



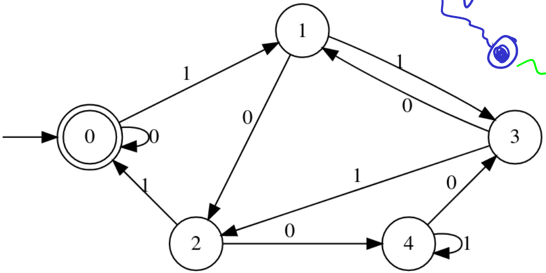
x z

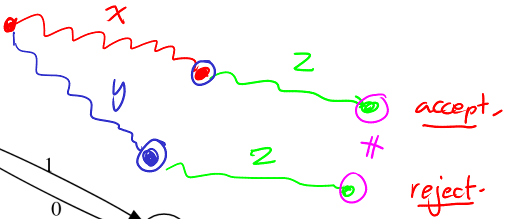
y z

If string x and y reach the same state in a DFA, for any string z, both xz and yz must reach the same state.

→ accept / reject

Another example





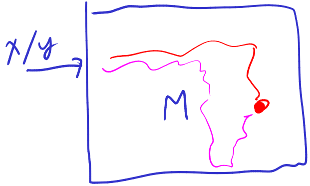
If string x and y reach the same state in a DFA, for any string z, both xz and yz must reach the same state.
In other words, a DFA accepts xz iff it accepts yz.

Contraposition: ถ้า $\boxed{x}$ & $\boxed{y}$ ที่ xz & yz ไปถึงคนละ state, x กับ y จะต้องไปถึงคนละ state.

Basic question

How can you show that you need at least two states?

พิจารณา DFA M ที่ $L(M) = \{0^n 1^n \mid n \geq 0\}$



$x = 0000$

$y = 000$

- $z = 01111$

  $xz = 000001111 \notin L(M)$

  $yz = 00001111 \in L(M)$

- $z = 111$

  $xz = 0000111 \notin L(M)$ ✗

  $yz = 000111 \in L(M)$ ✓

  $\Rightarrow$ xz กับ yz จะต้องไปที่คนละ state.

  $\Rightarrow$ x กับ y ต้องไปที่คนละ state.

- $z = 1111$

  $xz = 00001111 \in L(M)$

  $yz = 0001111 \notin L(M)$

  $\Rightarrow$ xz & yz จะต้องไปที่คนละ state ใน M

  $\Rightarrow$ x & y ไปที่คนละ state ใน M.

Distinguishing suffixes

Consider language $L = \{0^n 1^n \mid n \geq 0\}$.
Consider $x = 0$ and $y = 00$. Consider suffix $z = \underline{11}$.

$xz = 011 \notin L$

$yz = 0011 \in L$

Distinguishing suffixes

Consider language $L = \{0^n 1^n \mid n \geq 0\}$.
Consider $x = 0$ and $y = 00$. Consider suffix $z = 11$.
We have that

$$xz = 011 \notin L,$$

but

$$yz = 0011 \in L.$$

Distinguishing suffixes

Consider language $L = \{0^n 1^n \mid n \geq 0\}$.
Consider $x = 0$ and $y = 00$. Consider suffix $z = 11$.
We have that

$$xz = 011 \notin L,$$

but

$$yz = 0011 \in L.$$

What can you say about a DFA M such that $L(M) = L$?

Distinguishing suffixes

$\{x_1, x_2, x_3, x_4, \ldots, x_m\}$

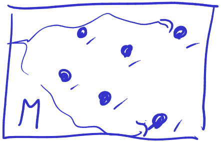
Consider language $L = \{0^n 1^n \mid n \geq 0\}$.
Consider $x = 0$ and $y = 00$. Consider suffix $z = 11$.
We have that

$$xz = 011 \notin L,$$

but

$$yz = 0011 \in L.$$

What can you say about a DFA M such that $L(M) = L$?



ภาษาใดๆ

Definition

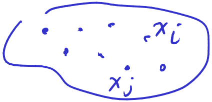
For strings x and y, string z is a **distinguishing suffix** with respect to L if exactly one of xz and yz is in L.

z ??

z เป็นหลักฐานว่า ถ้ามี DFA M สำหรับ L
M เมื่อรับ input x กับ y จะต้องไปจบคนละ state.

Fooling sets

any language.



$\forall (x_i, x_j)$ $\exists$ str z ที่ z ไม่...

A **fooling set** for a language L is set F of strings such that every pair of strings in F has a distinguishing suffix.

Example: The set $\{0, 00, 000\}$ is a fooling set for $L = \{0^n1^n \mid n \geq 0\}$.

0, 00 · $z = 1$ ✓

01 ✓ 001 ×

0, 000 · $z = 111$ ✓

0111 × 000111 ✓

00, 000 · $z = 11$ ✓

0011 ✓ 00011 ×

- จะพิสูจน์ว่า language L ไม่เป็น regular
  - พิสูจน์ว่ามี infinite fooling set สำหรับ language L.
- fooling set
- distinguishing suffix ②

A large fooling set

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

Lemma 4 // F

The set $\{0^n \mid n \geq 0\}$ *is a fooling set for* $L = \{0^n1^n \mid n \geq 0\}$.

Proof.

ให้ $F = \{0^n \mid n \geq 0\}$. พิจารณา $x = 0^i \in F$, และ $y = 0^j \in F$ โดยที่ $i \neq j$

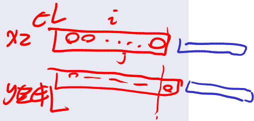
สมมติว่า $i < j$ (โดยไม่เสียความเป็นทั่วไป)

ให้ $z = 1^i$ จะได้ว่า

- $xz = 0^i 1^i \in L$
- $yz = 0^j 1^i \notin L$

ดังนั้น z เป็น distinguishing suffix ของ x และ y.

$\Rightarrow F$ เป็น fooling set



Note:
A **fooling set** for a language L is set F of strings such that every pair of strings in F has a distinguishing suffix.

Observation

If language L has an infinite fooling set, L is not regular

→ ถ้ามี DFA M สำหรับ language L,
จำนวน state ใน M จะต้องมากกว่า ขนาดของ fooling set ใดๆ ของ L.

⇒ contradiction. ⇒ ไม่มี DFA M สำหรับ L.

Observation

If language L has an infinite fooling set, L is not regular

Lemma 5

Language $L = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

We previously establish that the set $F = \{0^n \mid n \geq 0\}$ is a fooling set for L.
Since F has infinite size, from the observation, we know that L is not regular. □

Lemma 6

reverse $\quad \{0110,\ 00,\ \underbrace{1011}_{w}\underbrace{1101}_{w^R}, \dots\}$

For $\Sigma = \{0, 1\}$*, the language* $L = \{ww^R \mid w \in \Sigma^*\}$ *is not regular.* ← palindrome.

Proof.

ให้ $F = \{0^n \mid n \geq 0\}$ จะพิสูจน์ว่า F เป็น fooling set ของ L

พิจารณา $x = 0^i \in F$ และ $y = 0^j \in F$ ที่ $i \neq j$.

สมมติว่า $i < j$ โดยไม่เสียความเป็นทั่วไป

ให้ $z = 110^j$ จะได้ว่า . $xz = 0^i 11 0^j \notin L$

แต่ . $yz = 0^j 11 0^j \in L$, นั่นคือ z เป็น

distinguishing suffix สำหรับ x, y ใดๆ ใน F ที่ $x \neq y \Rightarrow F$ เป็น fooling set ขนาด infinite ของ L $\Rightarrow$ นั่นแสดงว่า L ไม่เป็น regular $\square$

00000 11 000 ✗
000 11 000 ✓
000 1 000 ✓
000 000 ✓

$x_1, x_2, \dots$

x_i z
y_j z

Note: แสดงว่ามี fooling set F ขนาด infinite.

$L = \{0^{2^n} \mid n \geq 0\}$: Proof (1/2)

$\{0, 00, 0000, 00000000, 0^{16}, 0^{32}, \ldots\}$

Lemma 7

For $\Sigma = \{0\}$, the language $L = \{0^{2^n} \mid n \geq 0\}$ is not regular.

Proof.

ให้ $F = L$. จะพิสูจน์ว่า F เป็น fooling set ของ L.

พิจารณา $x = 0^{2^i} \in F$ และ $y = 0^{2^j} \in F$ ที่ $i < j$

ให้ $z = 0^{2^i}$ พิจารณา

$$xz = 0^{2^i}0^{2^i} = 0^{2 \cdot 2^i} = 0^{2^{i+1}} \in L$$

$$yz = 0^{2^j}0^{2^i} = 0^{2^j+2^i}$$ สังเกตว่า

$$2^j < \boxed{2^j + 2^i} < 2^j + 2^j = 2^{j+1}$$

← เพราะ $2^i > 0$ ← เพราะ $2^i < 2^j$

ซึ่งหมายถึง $2^j + 2^i$ ไม่เป็นกำลังของ 2 $\Rightarrow yz \notin L$. ดังนั้น z เป็น distinguishing suffix ของ x & y $\Rightarrow F$ เป็น fooling set

x

y

0^4 0^4

0^8 0^4

$L = \{0^{2^n} \mid n \geq 0\}$: Proof 2

Lemma 8

For $\Sigma = \{0\}$, the language $L = \{0^{2^n} \mid n \geq 0\}$ is not regular.

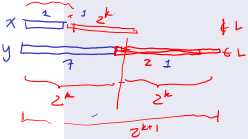
Proof.

ให้ $F = \{0^m \mid m \geq 0\}$ จะพิสูจน์ว่า F เป็น fooling set ของ L

กำหนดให้

ให้ x & y เป็น string ใน F ที่ $x \neq y$ ดังนั้น $x = 0^i$, $y = 0^j$ และ $i < j$ (ไม่เสียความทั่วไป)

จะแสดงว่ามี z ที่เป็น distinguishing suffix ของ x & y



ให้ k เป็นจำนวนเต็มที่ $2^k > j$ ให้ $z = 0^{2^k+2^k-j}$

- $yz = 0^j \cdot 0^{2^k+2^k-j} = 0^{2\cdot 2^k} = 0^{2^{k+1}} \in L$
- $xz = 0^i \cdot 0^{2^k+2^k-j} = 0^{2^k} \cdot 0^{2^k-j+i}$ จะไม่อยู่ใน L

เพราะ $2^k < \boxed{2^k+2^k-j+i} < 2^k+2^k = 2^{k+1}$,

ดังนั้น F เป็น fooling set ของ L

แสดงว่า L มี infinite fooling set L จึงไม่เป็น regular

$\square$

$L = \{0^{2^n} \mid n \geq 0\}$: Proof 3

Lemma 9

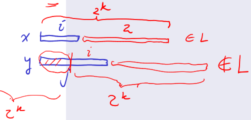
For $\Sigma = \{0\}$, the language $L = \{0^{2^n} \mid n \geq 0\}$ is not regular.

Proof.

ให้ $F = \{0^m \mid m \geq 0\}$ จะพิสูจน์ว่า F เป็น fooling set ของ L

ให้ x & y เป็น string ใน F ที่ $x \neq y$ ดังนั้น $x = 0^i$, $y = 0^j$ และ $i < j$ (ไม่เสียความเป็นทั่วไป)

จะแสดงว่ามี z ที่เป็น distinguishing suffix ของ x & y



- ให้ k เป็น จน.เต็ม ที่ $2^k > j$, ให้ $z = 0^{2^k - i}$

พิจารณา $xz = 0^i \cdot 0^{2^k - i} = 0^{2^k} \in L$ ✓

- $yz = 0^j \cdot 0^{2^k - i} = 0^{2^k - i + j}$ ซึ่ง $\notin L$ เพราะ

$$2^k < 2^k - i + j < 2^k - i + 2^k \leq 2^k + 2^k = 2^{k+1}.$$

ดังนั้น F เป็น fooling set ของ L. เนื่องจาก L มี infinite fooling set, L จึงไม่เป็น regular

□

$L = \{0^p \mid p$ is prime$\}$: Proof 1

Lemma 10

For $\Sigma = \{0\}$, the language $L = \{0^p \mid p$ is prime$\}$ is not regular.

Proof.

$\square$

$L = \{0^p \mid p$ is prime$\}$: Proof 2

Lemma 11

For $\Sigma = \{0\}$, the language $L = \{0^p \mid p$ is prime$\}$ is not regular.

Proof.

$\square$