Introduction to Finite Automata

Languages

Deterministic Finite Automata

Representations of Automata

The slides are created by Jeffrey D. Ullman
http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE

Alphabets

- An *alphabet* is any finite set of symbols.
- Examples: ASCII, Unicode, {0,1} (*binary alphabet*), {a,b,c}.

Strings

- The set of *strings* over an alphabet Σ is the set of lists, each element of which is a member of Σ.
  - Strings shown with no commas, e.g., abc.
- Σ* denotes this set of strings.
- ϵ stands for the *empty string* (string of length 0).

{a, b, c}

s1 : "f" len = 1

s2 : "" len = 0

Example: Strings

- {0,1}* = {ϵ, 0, 1, 00, 01, 10, 11, 000, 001, . . . }
- Subtlety: 0 as a string, 0 as a symbol look the same.
  - Context determines the type.

Languages

String หลายๆตัวรวมกัน

- A *language* is a subset of Σ* for some alphabet Σ.
- Example: The set of strings of 0's and 1's with no two consecutive 1's. ไม่มี 1 ติดกัน 2 ตัว.
- L = {ϵ, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, . . . }

Hmm… 1 of length 0, 2 of length 1, 3, of length 2, 5 of length 3, 8 of length 4. I wonder how many of length 5?

FA → DFA → input ทั้งหมด / input 1 ตัว 1 stage
→ NFA → input ไม่ต้องทั้งหมด

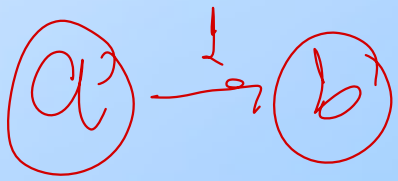
Deterministic Finite Automata

- A formalism for defining languages, consisting of:
  1. A finite set of *states* (Q, typically).
  2. An *input alphabet* (Σ, typically).
  3. A *transition function* (δ, delta typically).
  4. A *start state* (q_0, in Q, typically).
  5. A set of *final states* ($F \subseteq Q$, typically). มีน้อยกว่าหรือเท่ากับ
     - "Final" and "accepting" are synonyms.

The Transition Function

- Takes two arguments: a state and an input symbol.
- δ(q, a) = the state that the DFA goes to when it is in state *q* and input *a* is received.

$\delta(a, 1) = b$

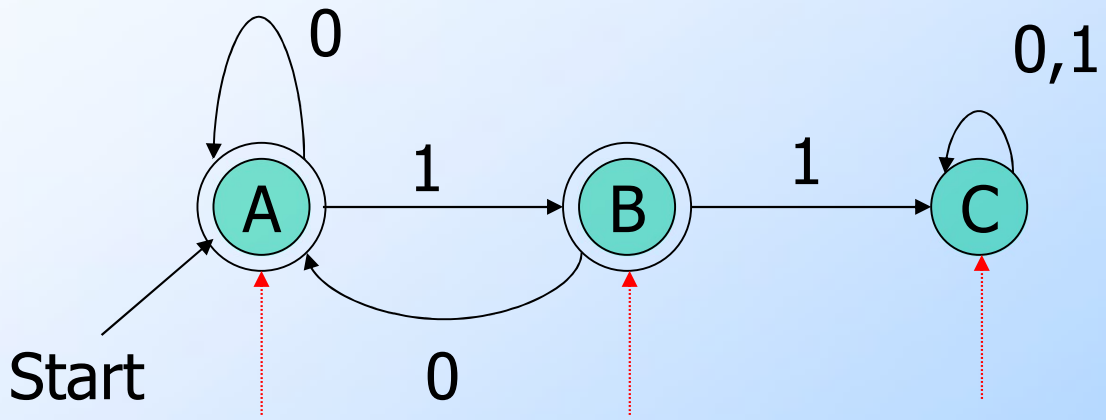


Graph Representation of DFA's

- Nodes = states.
- Arcs represent transition function.
  - Arc from state p to state q labeled by all those input symbols that have transitions from p to q.
- Arrow labeled "Start" to the start state.
- Final states indicated by double circles.

Example: Graph of a DFA

Accepts all strings without two consecutive 1's.



Previous string OK, does not end in 1.

Previous String OK, ends in a single 1.

Consecutive 1's have been seen.

Alternative Representation: Transition Table

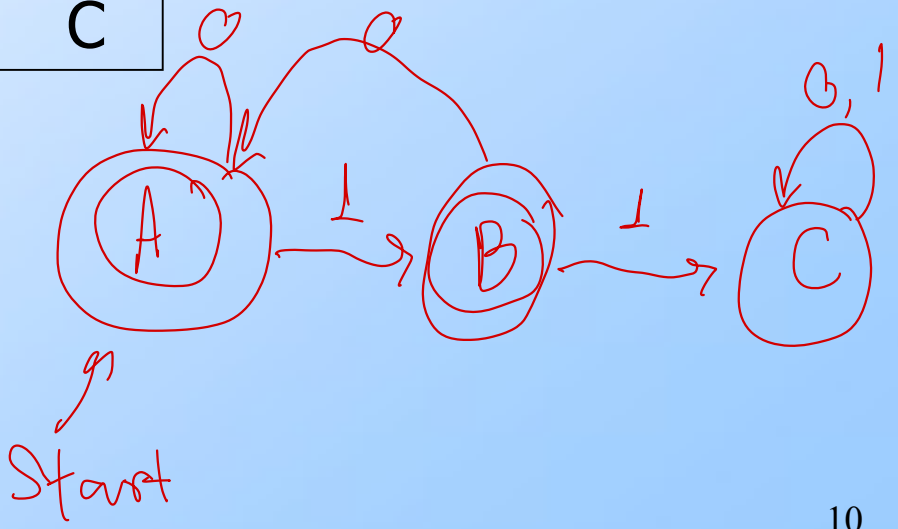
Final states starred

Arrow for start state

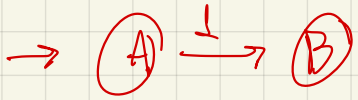
	0	1
→ * A	A	B
* B	A	C
C	C	C

Columns = input symbols

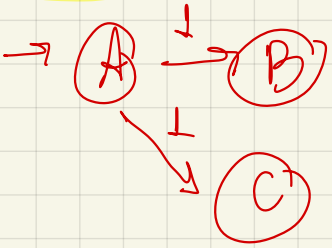
Rows = states



DFA

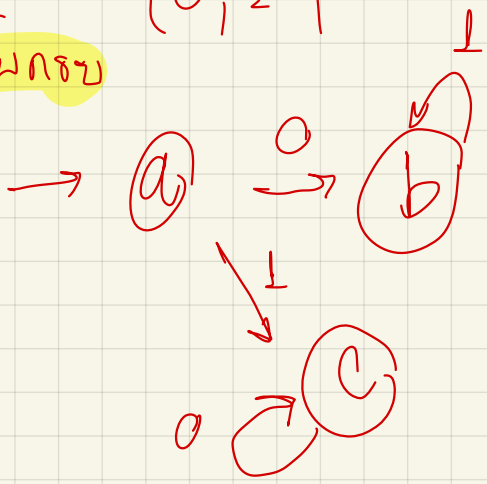


NFA

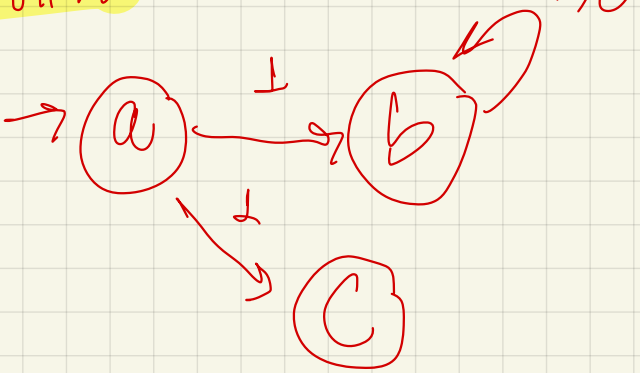


$\Sigma = \{0, 1\}$

ต้องรับครบ



รับไม่ครบก็ได้



Extended Transition Function

- We describe the effect of a string of inputs on a DFA by extending δ to a state and a string.
- Induction on length of string.
- Basis: $\delta(q, \epsilon) = q$
- Induction: $\delta(q,wa) = \delta(\delta(q,w),a)$
  - w is a string; a is an input symbol.

Extended δ: Intuition

- Convention:
  - … w, x, y, x are strings.
  - a, b, c,… are single symbols.
- Extended δ is computed for state q and inputs $a_1a_2…a_n$ by following a path in the transition graph, starting at q and selecting the arcs with labels a_1, a_2,…,a_n in turn.

Example: Extended Delta

	0	1
A	A	B
B	A	C
C	C	C

$$\delta(B,011) = \delta(\delta(B,01),1) = \delta(\delta(\delta(B,0),1),1) =$$

$$\delta(\delta(A,1),1) = \delta(B,1) = C$$

Delta-hat

- In book, the extended δ has a "hat" to distinguish it from δ itself.
- Not needed, because both agree when the string is a single symbol.
- $\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a)$

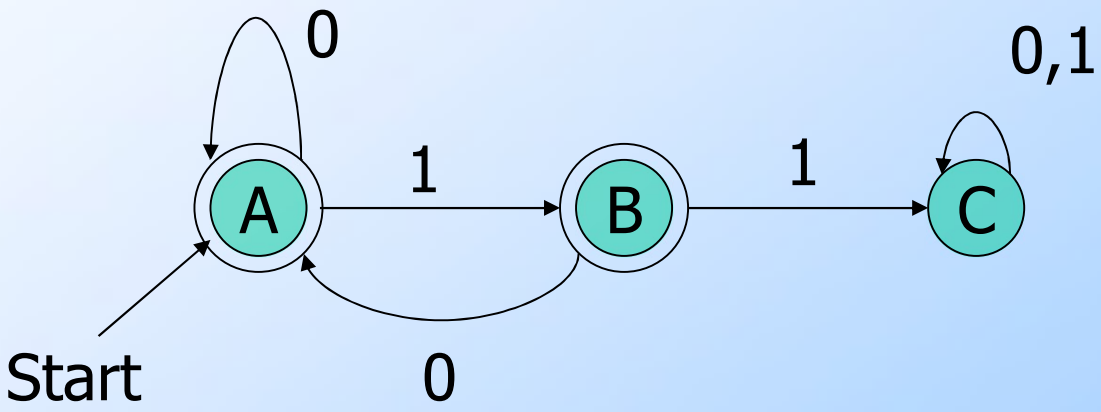
Extended deltas

Language of a DFA

- Automata of all kinds define languages.
- If A is an automaton, L(A) is its language.
- For a DFA A, L(A) is the set of strings labeling paths from the start state to a final state.
- Formally: L(A) = the set of strings w such that $\delta(q_0, w)$ is in F.

Example: String in a Language

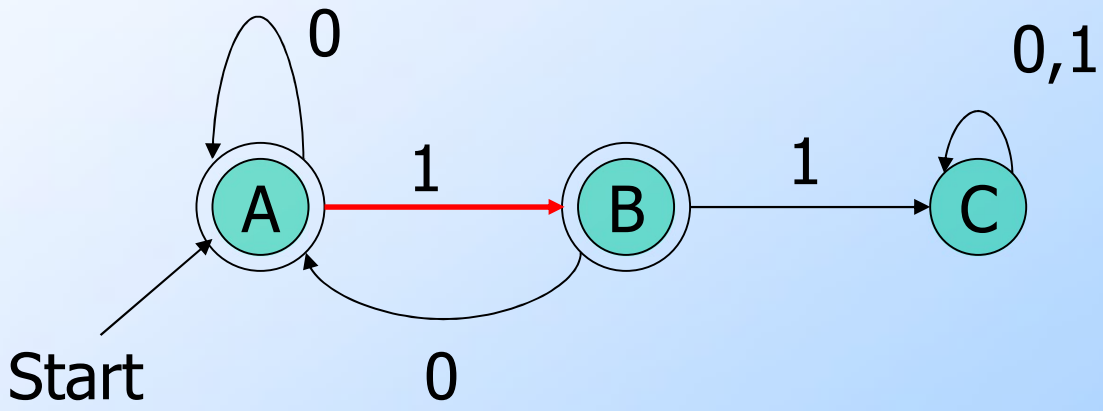
String 101 is in the language of the DFA below.
Start at A.



Example: String in a Language

String 101 is in the language of the DFA below.

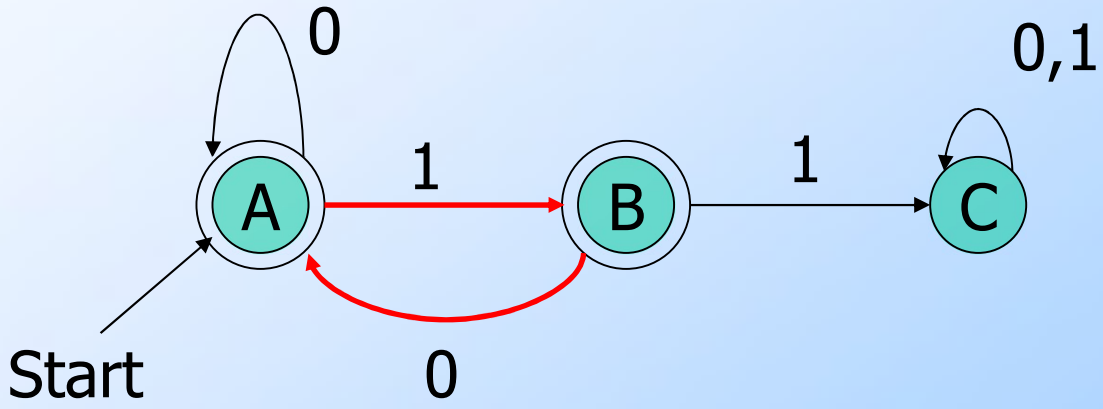
Follow arc labeled 1.



Example: String in a Language

String 101 is in the language of the DFA below.

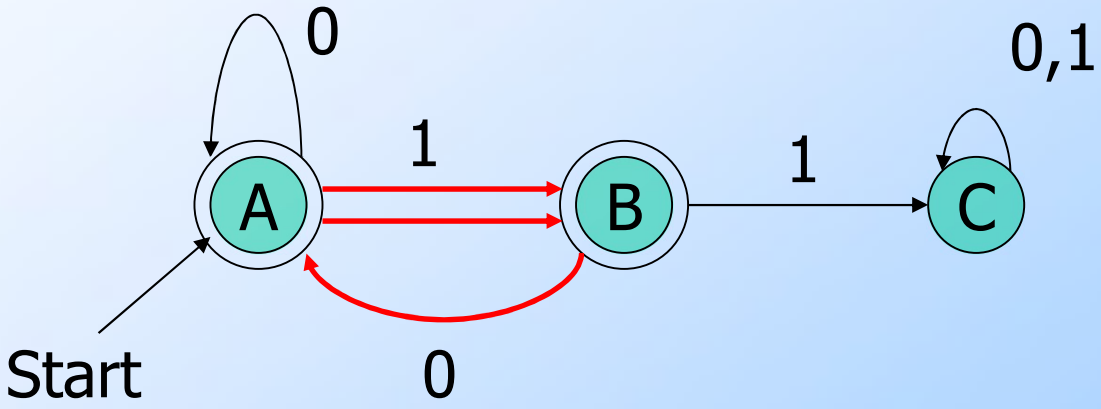
Then arc labeled 0 from current state B.



Example: String in a Language

String 101 is in the language of the DFA below.

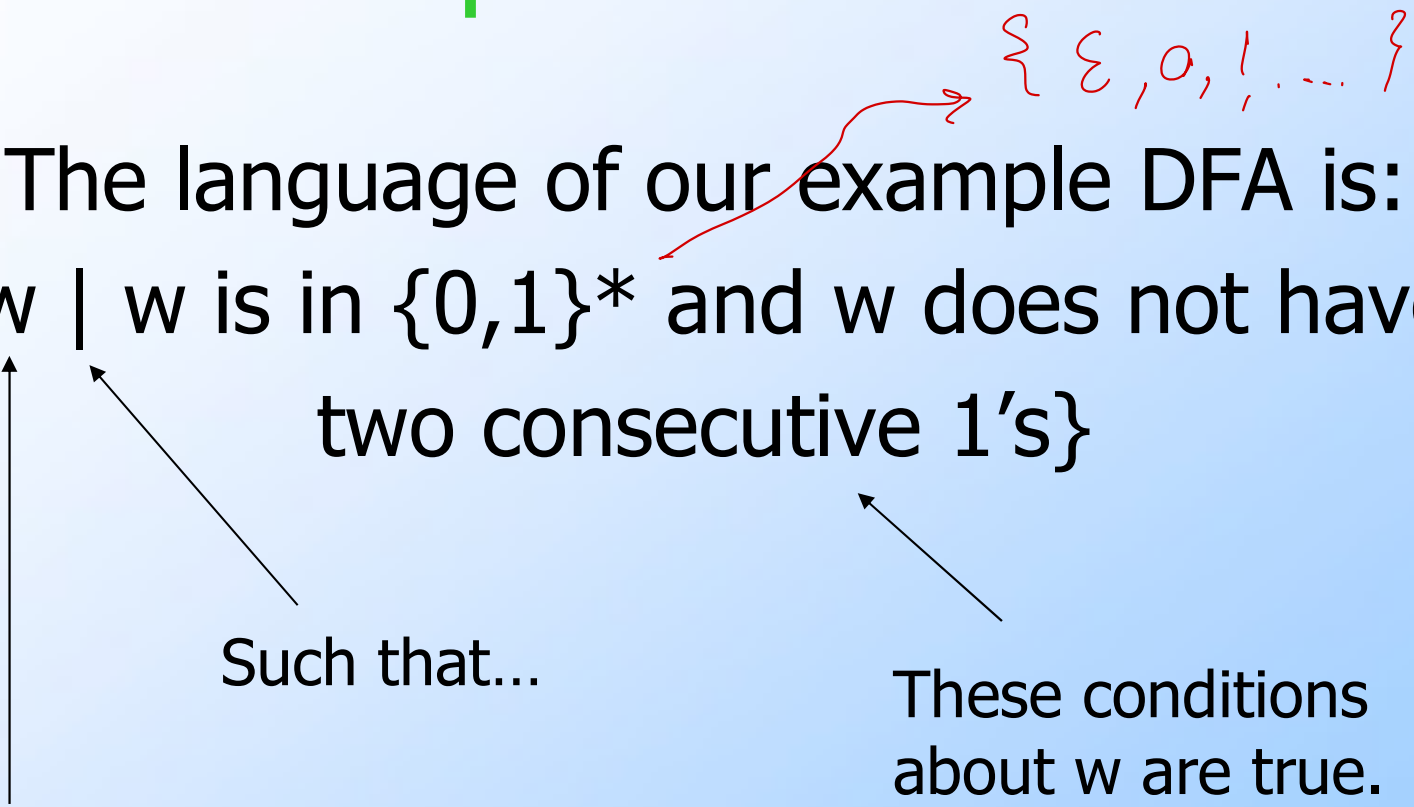
Finally arc labeled 1 from current state A. Result is an accepting state, so 101 is in the language.



Example – Concluded

- The language of our example DFA is:

{w | w is in {0,1}* and w does not have two consecutive 1's}



Regular Languages

- A language L is *regular* if it is the language accepted by some DFA.
  - Note: the DFA must accept only the strings in L, no others.
- Some languages are not regular.
  - Intuitively, regular languages "cannot count" to arbitrarily high integers.

Example: A Nonregular Language

$L_1 = \{0^n 1^n \mid n \geq 1\}$

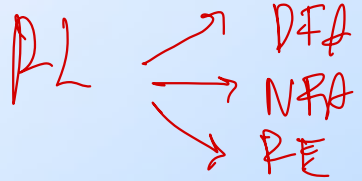
- Note: a^i is conventional for i a's.
  - Thus, $0^4 = 0000$, e.g.
- Read: "The set of strings consisting of n 0's followed by n 1's, such that n is at least 1.
- Thus, $L_1 = \{01, 0011, 000111, \ldots\}$

Another Example

L_2 = {w | w in {(,)}* and w is *balanced* }

- Note: alphabet consists of the parenthesis symbols '(' and ')'.
- Balanced parens are those that can appear in an arithmetic expression.
  - E.g.: (), ()(), (()), (()()),…

But Many Languages are Regular



- Regular Languages can be described in many ways, e.g., regular expressions.
- They appear in many contexts and have many useful properties.
- Example: the strings that represent floating point numbers in your favorite language is a regular language.

Example: A Regular Language

L_3 = { w | w in {0,1}* and w, viewed as a binary integer is divisible by 23}

- The DFA:
  - 23 states, named 0, 1,…,22.
  - Correspond to the 23 remainders of an integer divided by 23.
  - Start and only final state is 0.

Transitions of the DFA for L_3

- If string w represents integer i, then assume $\delta(0, w) = i\%23$.
- Then w0 represents integer 2i, so we want $\delta(i\%23, 0) = (2i)\%23$.
- Similarly: w1 represents 2i+1, so we want $\delta(i\%23, 1) = (2i+1)\%23$.
- Example: $\delta(15,0) = 30\%23 = 7$; $\delta(11,1) = 23\%23 = 0$.

Key idea: design a DFA by figuring out what each state needs to remember about the past.

Another Example

L_4 = { w | w in {0,1}* and w, viewed as the reverse of a binary integer is divisible by 23}

- Example: 01110100 is in L_4, because its reverse, 00101110 is 46 in binary.
- Hard to construct the DFA.
- But theorem says the reverse of a regular language is also regular.