CSE 2001: Introduction to Theory of Computation

Fall 2012

Suprakash Datta

datta@cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

Course page: http://www.cs.yorku.ca/course/2001

Last class: examples of DFA

Today :

- Study limitations of DFA
- Introduce nondeterminism in finite automata. [Ch 1.2 in Sipser]

Recall: Regular Languages

The language recognized by a finite automaton M is denoted by **L(M).**

A regular language is a language for which there exists a recognizing finite automaton.

Recall: Two DFA Questions

Given the description of a finite automaton **M = (Q,Σ,δ,q,F),** what is the language **L(M)** that it recognizes?

In general, what kind of languages can be recognized by finite automata? (What are the regular languages?)

Complement of a regular language

- Swap the accepting and non-accept states of M to get M'.

- The complement of a regular language is regular.

Terminology: closure

- A set is defined to be closed under an operation if that operation on members of the set always produces a member of the same set. (adapted from wikipedia)

E.g.:

- The integers are closed under addition, multiplication.
- The integers are not closed under division
- Σ^* is closed under concatenation

- A set can be defined by closure -- Σ^* is called the (Kleene) closure of Σ under concatenation.

Terminology: Regular Operations

Pages 44-47 (Sipser)

The regular operations are:

1. Union
2. Concatenation
3. Star (Kleene Closure): For a language A,

   $A^* = \{w_1 w_2 w_3 \ldots w_k \mid k \geq 0, \text{ and each } w_i \in A\}$

Closure Properties

- Set of regular languages is closed under
  - Union
  - Concatenation
  - Star (Kleene Closure)

Union of Two Languages

Theorem 1.12: If A_1 and A_2 are regular languages, then so is $A_1 \cup A_2$.
(The regular languages are 'closed' under the union operation.)

Proof idea: A_1 and A_2 are regular, hence there are two DFA M_1 and M_2, with $A_1=L(M_1)$ and $A_2=L(M_2)$. Out of these two DFA, we will make a third automaton M_3 such that $L(M_3) = A_1 \cup A_2$.

How do we combine DFA?

Q: Can we design a DFA that somehow ``simulates'' them both and accepts when at least one of them accepts?

Ans: Yes, through a clever construction.

Proof Union-Theorem (1)

$M_1=(Q_1,\Sigma,\delta_1,q_1,F_1)$ and $M_2=(Q_2,\Sigma,\delta_2,q_2,F_2)$

Define $M_3 = (Q_3,\Sigma,\delta_3,q_3,F_3)$ by:

- $Q_3 = Q_1 \times Q_2 = \{(r_1,r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
- $\delta_3((r_1,r_2),a) = (\delta_1(r_1,a), \delta_2(r_2,a))$
- $q_3 = (q_1,q_2)$
- $F_3 = \{(r_1,r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

Proof Union-Theorem (2)

The automaton $M_3 = (Q_3,\Sigma,\delta_3,q_3,F_3)$ runs M_1 and M_2 in 'parallel' on a string w.

In the end, the final state (r_1,r_2) 'knows' if $w \in L_1$ (via $r_1 \in F_1$?) and if $w \in L_2$ (via $r_2 \in F_2$?)

The accepting states F_3 of M_3 are such that $w \in L(M_3)$ if and only if $w \in L_1$ *or* $w \in L_2$, for:
$F_3 = \{(r_1,r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

Concatenation of L_1 and L_2

Definition: $L_1 \bullet L_2 = \{ xy \mid x \in L_1 \text{ and } y \in L_2 \}$

Example: $\{a,b\} \bullet \{0,11\} = \{a0,a11,b0,b11\}$

Theorem 1.13: If L_1 and L_2 are regular languages, then so is $L_1 \bullet L_2$.
(The regular languages are 'closed' under concatenation.)

Proving Concatenation Thm.

Consider the concatenation:
{1,01,11,001,011,…} • {0,000,00000,…}
(That is: the bit strings that end with a “1”, followed by an odd number of 0’s.)

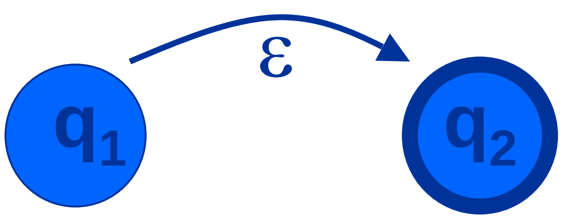
Problem is: given a string w, how does the automaton know where the L_1 part stops and the L_2 substring starts?

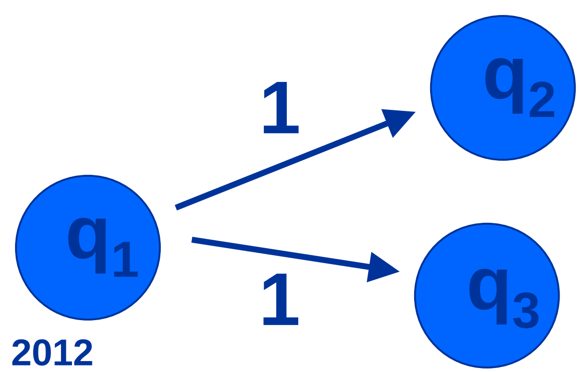
We need an M with ‘lucky guesses’.

Nondeterminism

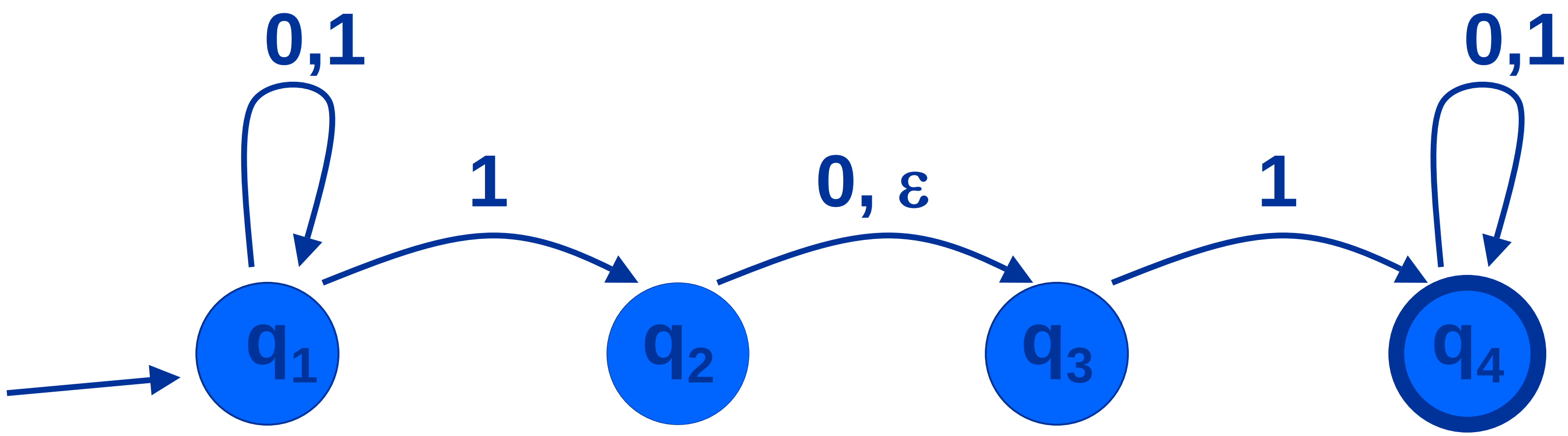
Nondeterministic machines are capable of being lucky, no matter how small the probability.

A nondeterministic finite automaton has transition rules/possibilities like





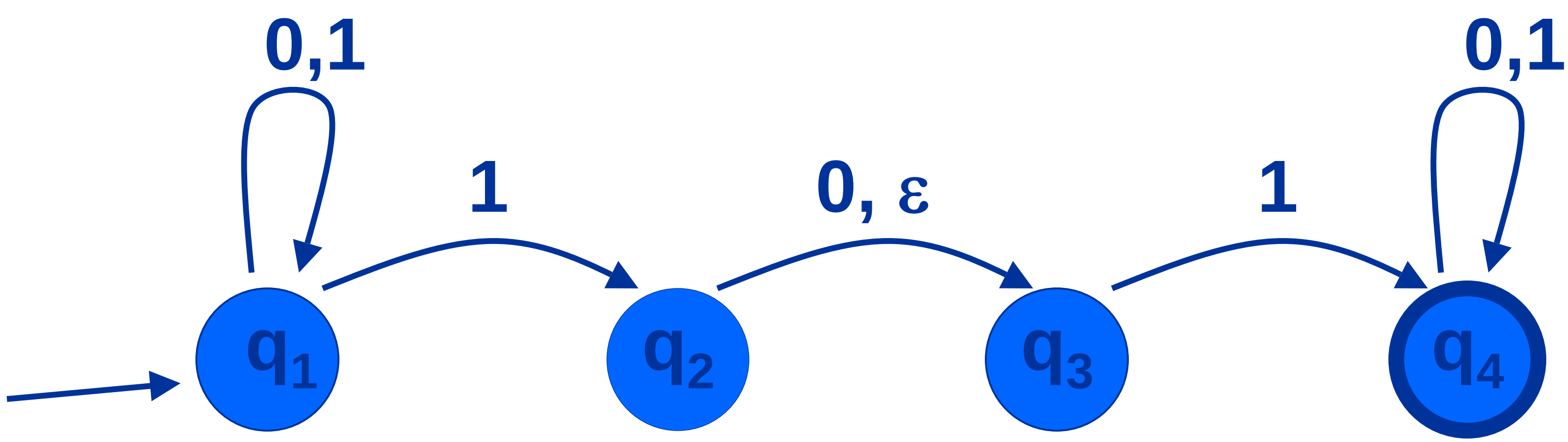
A Nondeterministic Automaton



This automaton accepts "0110", because there is a possible path that leads to an accepting state, namely:

$$q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_4 \rightarrow q_4$$

A Nondeterministic Automaton



The string 1 gets rejected: on "1" the automaton can only reach: $\{q_1,q_2,q_3\}$.

Nondeterminism ~ Parallelism

For any (sub)string w, the nondeterministic automaton can be in a set of possible states.

If the final set contains an accepting state, then the automaton accepts the string.

“The automaton processes the input in a parallel fashion. Its computational path is no longer a line, but a tree.” (Fig. 1.28)

Are NFA more powerful than DFA?

- NFA can solve every problem that DFA can (DFA are also NFA)
- Need proof
- Let us define NFA formally

Nondeterministic FA (def.)

- A nondeterministic finite automaton (NFA) M is defined by a 5-tuple $M=(Q,\Sigma,\delta,q_0,F)$, with
  - Q: finite set of states
  - Σ: finite alphabet, $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$
  - δ: transition function $\delta:Q\times\Sigma_\varepsilon\to\mathcal{P}(Q)$
  - $q_0\in Q$: start state
  - $F\subseteq Q$: set of accepting states

Nondeterministic $\delta: Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$

The function $\delta: Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the crucial difference. It means:
"When reading symbol "a" while in state q, one can go to one of the states in $\delta(q,a) \subseteq Q$."

The ε in $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ takes care of the empty string transitions.

Recognizing Languages (def)

A nondeterministic FA **M = (Q,Σ,δ,q,F)** accepts a string **w** = $w_1 \ldots w_n$ if and only if we can rewrite w as $y_1 \ldots y_m$ with $y_i \in \Sigma_\varepsilon$ and there is a sequence $r_0 \ldots r_m$ of states in Q such that:

1) $r_0 = q_0$

2) $r_{i+1} \in \delta(r_i, y_{i+1})$ for all $i=0,\ldots,m-1$

3) $r_m \in F$

NFA drawing conventions

- Not all transitions are labeled
- Unlabeled transitions are assumed to go to a reject state from which the automaton cannot escape

NFA examples

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

1. Strings ending in 01
2. String containing 01

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

1. Strings ending in ab, bc, ca

Closure under regular operations

Union (new proof):

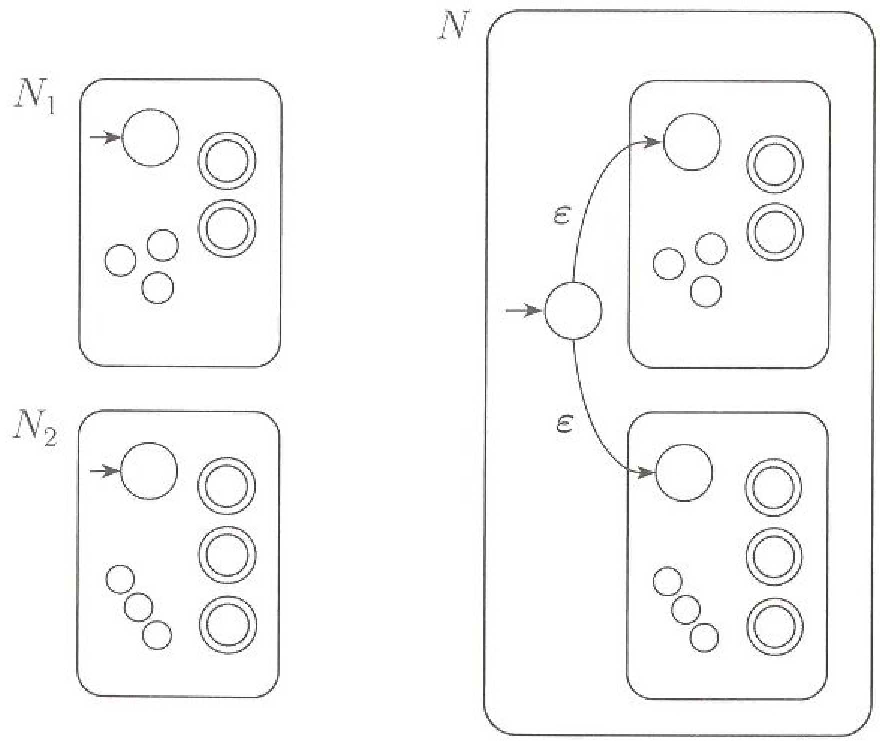


FIGURE 1.46
Construction of an NFA N to recognize $A_1 \cup A_2$

Closure under regular operations

Concatenation:

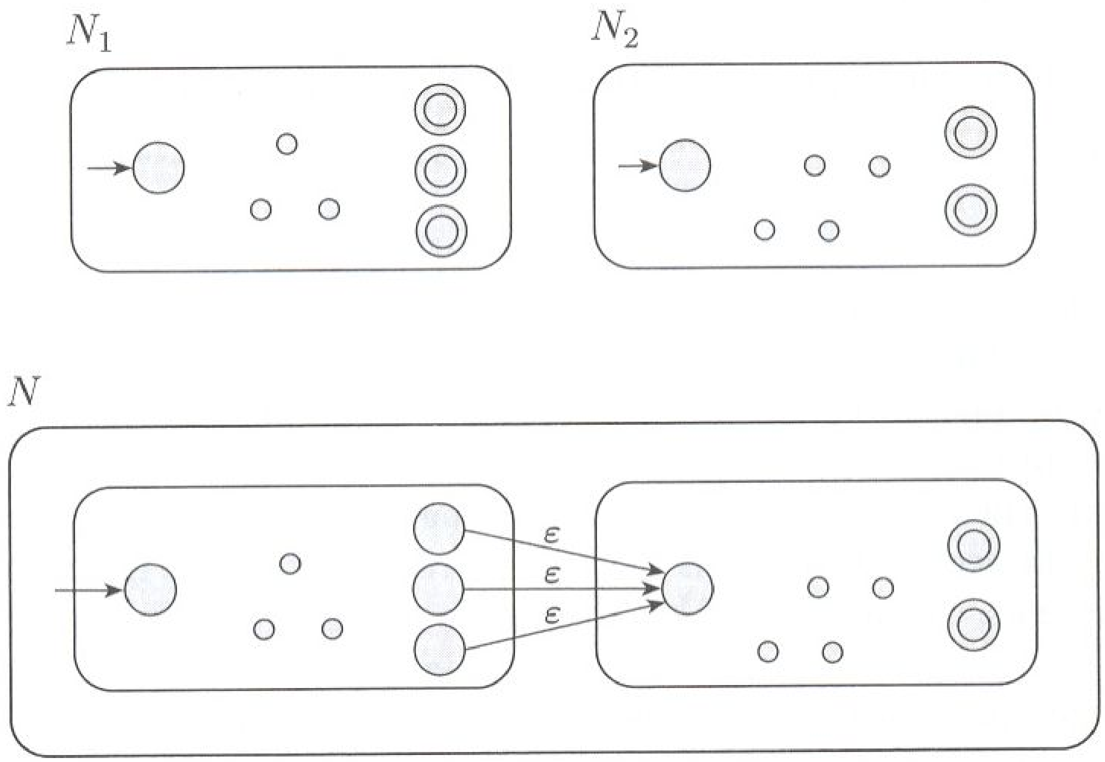


FIGURE **1.48**
Construction of N to recognize $A_1 \circ A_2$

Closure under regular operations

Star:

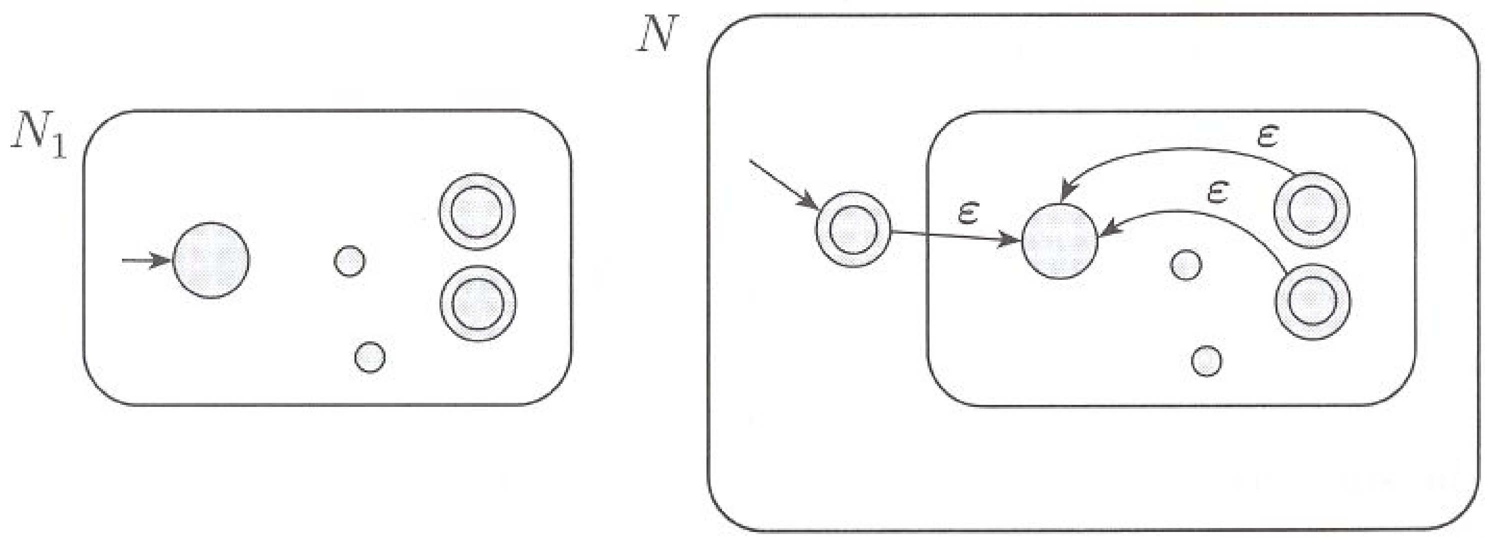


FIGURE 1.50
Construction of N to recognize A^*

Incorrect reasoning about RL

- Since $L_1 = \{w |\ w=a^n, n \in \mathbf{N}\}$, $L_2 = \{w |\ w = b^n, n \in \mathbf{N}\}$ are regular, therefore $L_1 \bullet L_2 = \{w |\ w=a^n b^n, n \in \mathbf{N}\}$ is regular
- If L_1 is a regular language, then $L_2 = \{w^R |\ w \in L_1\}$ is regular, and Therefore $L_1 \bullet L_2 = \{w\, w^R \mid w \in L_1\}$ is regular

Exercises

[Sipser 1.7 in 3rd Ed, 1.5 in 2nd Ed]: Give NFAs with the specified number of states that recognize the following languages over the alphabet $\Sigma=\{0,1\}$:

1. { w | w ends with 00}, three states
2. {0}; two states
3. { w | w contains even number of 0s, or exactly two 1s}, six states
4. $\{0^n \mid n \in N\}$, one state

Exercises - 2

Prove the following result:
"If L_1 and L_2 are regular languages, then $L_1 \cap \overline{L_2}$ is a regular language too."

Describe the language that is recognized by this nondeterministic automaton: