CISC 4090
Theory of Computation

Finite state machines & Regular languages

Professor Daniel Leeds
dleeds@fordham.edu
JMH 332

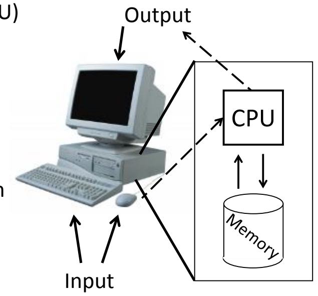
Stereotypical computer

Central processing unit (CPU) – performs all the instructions

Memory – stores data and instructions for CPU

Input – collects information from the world

Output – provides information to the world



2

Super-simple computers

Small number of potential inputs
Small number of potential outputs/actions

- Thermostat
- Elevator
- Vending machine
- Automatic door

3

Automatic door

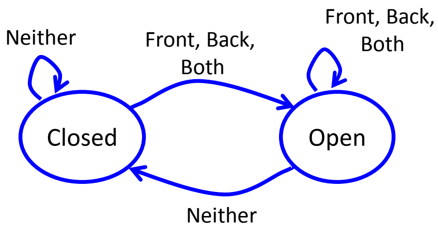
Desired behavior
- Person approaches entryway, door opens
- Person goes through entryway, door stays open
- Person is no longer near entryway, door closes
- Nobody near entryway, door stays closed

Two states: Open, Closed
Two inputs: Front-sensor, Back-sensor

Finite state machine

4

Graph and table representations



	Front	Back	Neither	Both
Closed	Open	Open	Closed	Open
Open	Open	Open	Closed	Open

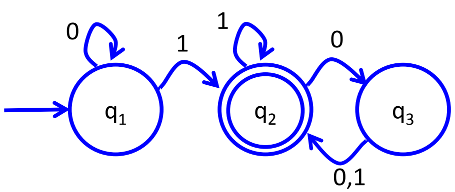
5

More finite state machine applications

- Text parsing
- Traffic light
- Pac-Man
- Electronic locks

6

Coding a combination lock



- A finite automaton M1 with 3 states
- Start state q1; accept state q2 (double circle)
- Example accepted string: 1101
- What are all strings that this model will accept?

String ending with 1 or string containing 1 and ending with 00

8

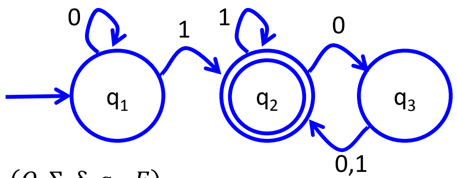
Formal definition of Finite State Automaton

Finite state automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

- Q is a finite set called states
- Σ is a finite set called the alphabet
- $\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

9

Describe M1 using formal definition



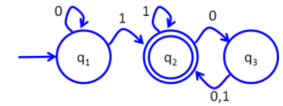
$M1 = (Q, \Sigma, \delta, q_0, F)$

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- Start state: q_1
- $F = \{q_2\}$
- $\delta =$

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

11

Language of M1



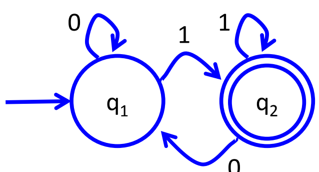
If A is set of all strings accepted by M, A is language of M
- L(M)=A

A machine may accept many strings, but only one language
- M **accepts** a string
- M **recognizes** a language

Describe L(M1)=A
- A={w|**w ends with 1 or w contains at least one 1 and ends in 00}**

13

Describe M2 using formal definition



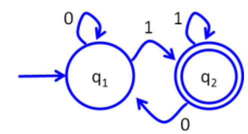
$M1 = (Q, \{0,1\}, \delta, q_0, \{q_2\})$

- $Q = \{q_1, q_2\}$
- Start state: q_1
- $\delta =$

	0	1
q1	q1	q2
q2	q1	q2

15

What is the language of M2?



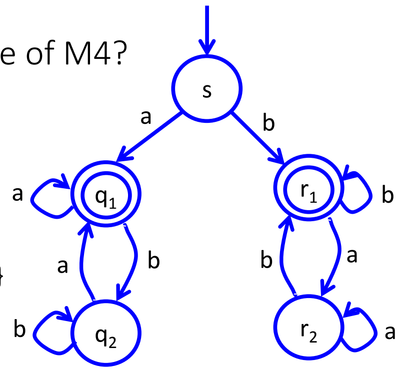
L(M2)={w| w ends with at least one 1}

17

What is the language of M4?

(page 38, Ex. 1.11)

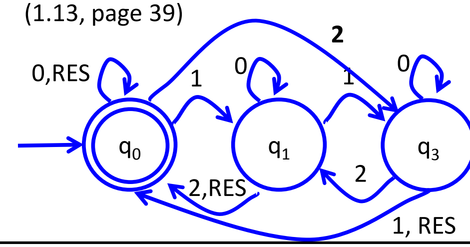
L(M4)={w| w ends and begins with same letter (either a or b)}



Perform modulo arithmetic

Let Σ={RESET, 0, 1, 2}

Construct M5 to accept a string only if the sum of each input symbol is multiple of 3, and RESET sets the sum back to 0 (1.13, page 39)



More modulo arithmetic

Generalize M5 to accept if sum of symbols is a multiple of *i* instead of 3

$(\{q0, q1, q2, q3, \cdots, q_{i-1}\}, \{0,1,2,\text{RESET}\}, \delta, q0, F)$

$\delta(qj, RESET) = q0$

$\delta(qj, 0) = qj$

$\delta(qj, 1) = qk$ for k = j+1 mod i

$\delta(qj, 2) = qk$ for k = j+2 mod i

Definition of M accepting a string

Let M $= (Q, \Sigma, \delta, q0, F)$ be a finite automaton and let $w = w_1 w_2 \cdots w_n$

Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with 3 conditions

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, 1, \cdots, n-1$
- $r_n \in F$

Regular languages

Definition: a language is called a regular language if some finite automaton recognizes it

equivalently

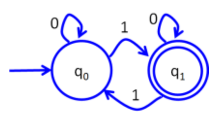
All of the strings in a regular language are accepted by some finite automaton

Designing finite automata (FAs)

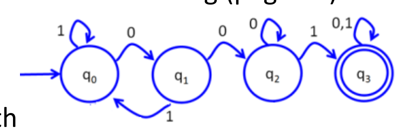
- Determine what you need to remember
  - How many states needed for your task?
- Set start and finish states
- Assign transitions
- Check your solution
  - Should accept $w \in L$
  - Should reject $w \notin L$
  - Be careful about ε!

FA design practice!

- FA to accept language where number of 1's is odd (page 43)

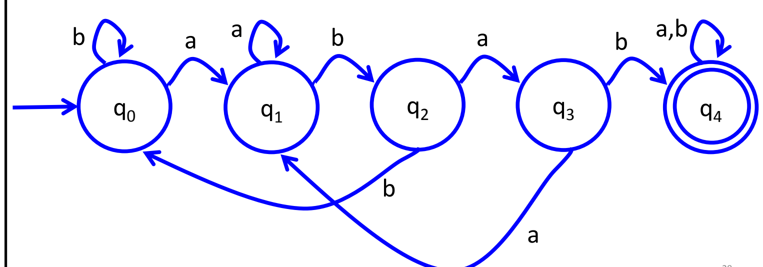


- FA to accept string with 001 as substring (page 44)



- FA to accept string with substring abab **(next page!)**

FA to accept string with substring abab



Regular operations

Let A and B be languages. We define 3 regular operations:

- Union: $A \cup B = \{x | x \in A \text{ or } x \in B\}$
- Concatenation: $A \cdot B = \{xy | x \in A \text{ and } y \in B\}$
- Star: $A^* = \{x_1 x_2 \cdots x_k | k \geq 0 \text{ and each } x_i \in A\}$
  - Repeat a string 0 or more times

31

Examples of regular operations

Let $A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$

What is:

- $A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$
- $A \cdot B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$
- $A^* = \{\varepsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badgood}, \text{badbad}, \cdots\}$

33

Closure

A collection of objects is closed under an operation if applying that operation to members of the collection returns an object in the collection

Regular languages are closed under $\cup$, $\cdot$, $*$

34

Closure of Union

Theorem 1.25: The class of regular languages is closed under the union operation

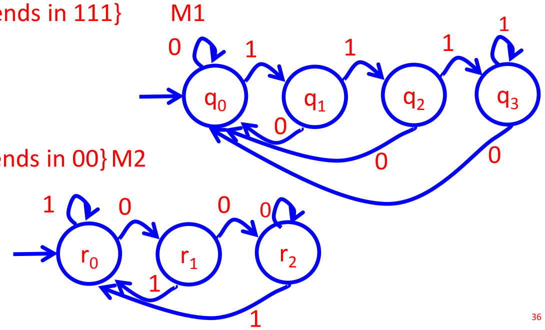
Proof by construction

35

Example union

A = {w | w ends in 111} M1

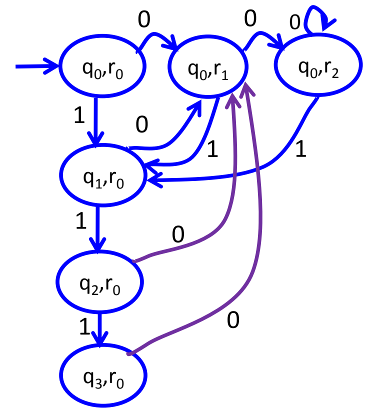
B = {w | w ends in 00} M2



Example union

AUB M5

Simulate M1 and M2 states



Closure of Union – Proof by Construction

Let us assume M1 recognizes language L1

- Define M1 as $M1 = (Q, \Sigma, \delta_1, q_0, F_1)$

Let us assume M2 recognizes language L2

- Define M2 as $M2 = (R, \Sigma, \delta_2, r_0, F_2)$

Proof by construction: Construct M3 to recognize $L3 = L1 \cup L2$

- Let M3 be defined as $M3 = (S, \Sigma, \delta_3, s_0, F_3)$

Closure of Union – Proof by Construction

- Let M3 be defined as $M3 = (S, \Sigma, \delta_3, s_0, F_3)$

Use each state of M3 to simulate being in a state of M1 and another state in M2 simultaneously

M3 states: $S = \{(q_i, r_j) \mid q_i \in Q \text{ and } r_j \in R\}$

Start state: $s_0 = (q_0, r_0)$

Accept state: $F_3 = \{(q_i, r_j) \mid q_i \in F_1 \text{ or } r_j \in F_2\}$

Transition function: $\delta_3\left((q_i, r_j), x\right) = \left(\delta_3(q_i, x), \delta_3(q_j, x)\right)$

Closure of Concatenation

Theorem 1.26: The class of regular languages is closed under the concatenation operation

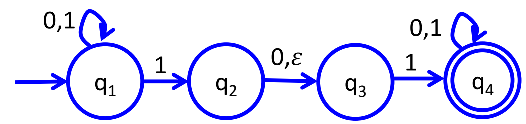
- If A1 and A2 are regular languages, then so is $A1 \cdot A2$
- Challenge: How do we know when M1 ends and M2 begins?

Determinism vs. non-determinism

Determinism: Single transition allowed given current state and given input

Non-determinism:

- multiple transitions allowed for current state and given input
- transition permitted for null input ε



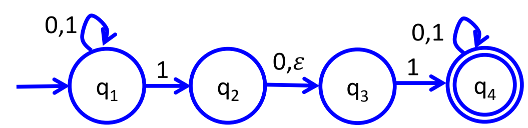
NFA in action

- When there is a choice, follow all paths – like cloning
- If there is no forward arrow, path terminates and clone dies (no accept)
- NFA will “accept” if at least one path terminates at accept

Alternative thought:

- Magically pick best path from the set of options

The language of M10



- List some accepted strings

  110 – at third entry, we’re in states {q_1,q_3, and q_4}
- What is L(M10)?

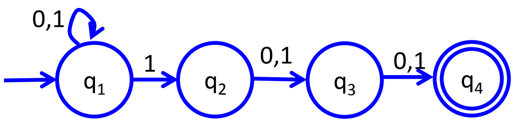
  {w | w contains 11 or 101}

NFA construction practice

Build an NFA that accepts all strings over {0,1} with 1 in the third position from the end

NFA construction practice

Build an NFA that accepts all strings over {0,1} with 1 in the third position from the end



If path is at q_4 and you receive more input, your path terminates

NFA -> DFA

Build an NFA that accepts all strings over {0,1} with 1 in the third position from the end

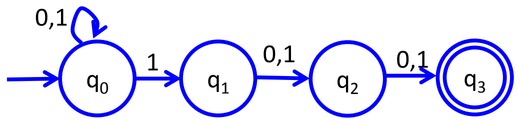
Can we construct a DFA for this?

Formal definition of Nondeterministic Finite Automaton

Similar to DFA: a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

- Q is a finite set called states
- Σ is a finite set called the alphabet
- $\delta: Q \times \Sigma\varepsilon \longrightarrow P(Q)$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Describe M10 using formal definition



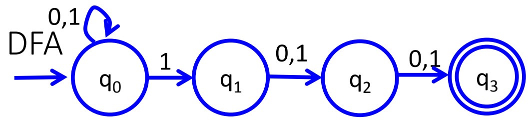
$M1 = (Q, \Sigma, \delta, q_0, F)$

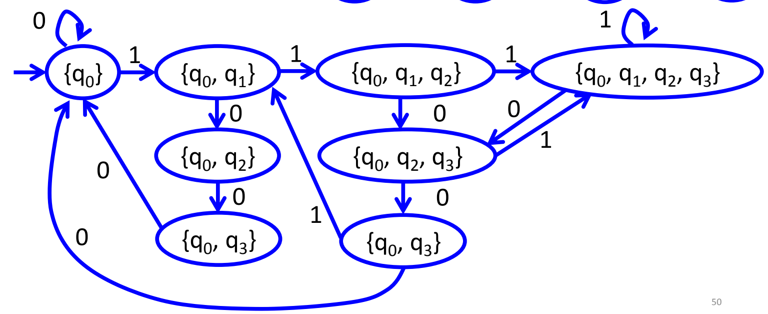
- $Q = \{\mathbf{q_0}, \mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}\}$
- $\Sigma = \{\mathbf{0}, \mathbf{1}\}$
- Start state: $\mathbf{q_0}$
- $F = \{\mathbf{q_3}\}$
- $\delta =$

	0	1	ε
q_0	$\{q_0\}$	$\{q_0,q_1\}$	{}
q_1	$\{q_2\}$	$\{q_2\}$	{}
q_2	$\{q_3\}$	$\{q_3\}$	{}
q_3	{}	{}	{}

49

Describe M10 as DFA





50

Equivalence of NFAs and DFAs

NFAs and DFAs recognize the same class of languages

Two machines are equivalent if they recognize the same language

Every NFA has an equivalent DFA

51

Equivalence of NFAs and DFAs

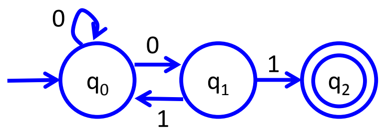
NFA $\quad N1 = (Q, \Sigma, \delta, q_0, F)$

Define DFA $\quad M1 = (R, \Sigma, \delta^D, r_0, F^D)$

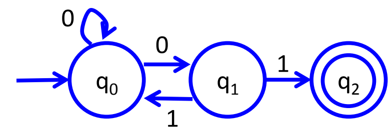
- **R=P(Q)** --- $R = \{\{\}, \{q_0\}, \ldots, \{q_n\}, \{q_1,q_2\}, \ldots \{q_{n-1},q_n\}, \ldots\}$
  every combination of states in Q
- $r_0=\{q_0\}$
- $F^D = \{s \in R \mid s \text{ contains at least 1 accept state for N1}\}$
- $\delta^D(r_i, x)$ **Consider all states q_j in r_i, find r_k that is union of outputs for N1's $\delta(q_j, x)$ for all q_j**

52

Consider NFA N1

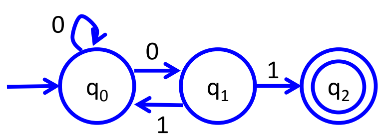


Language:
L(N1)={w | w begins with 0, ends with 01, every 1 in w is preceded by a 0}



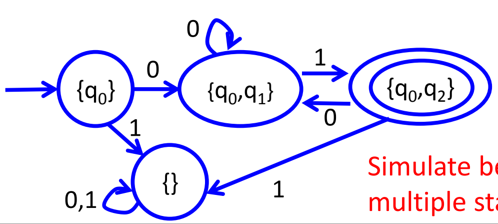
Convert NFA N1 to DFA M1

N1



Convert NFA N1 to DFA M1

N1



M1

Simulate being in multiple states at once

Union Closure with NFAs

- Proofs by construction – fewer states!
- Any NFA proof applies to DFA

Given two regular languages A_1 and A_2 recognized by N1 and N2 respectively, construct N to recognize $A_1 \cup A_2$

Let's consider two languages

L1: start with 0, end with 1

L2: start with 1, end with 0

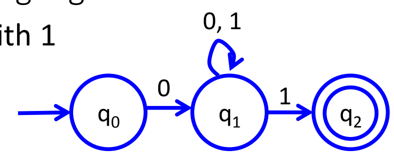
Construct machines for each languages

Construct machines N3 to recognize L1 U L2

Let's consider two languages

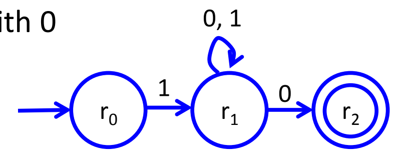
L1: start with 0, end with 1

N1

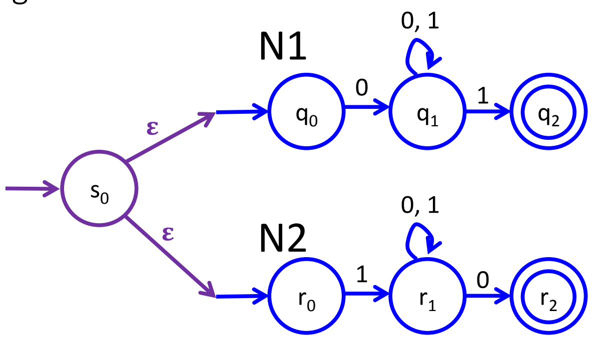


L2: start with 1, end with 0

N2



N3 recognizes L1 U L2



Closure of regular languages under union

This is a good example of how to write up a general proof by construction

Let N1 = $(Q, \Sigma, \delta_1, q_0, F_1)$ recognize L1

Let N2 = $(R, \Sigma, \delta_2, r_0, F_2)$ recognize L2

N3 = $(Q_3, \Sigma, \delta_3, s_0, F_3)$ will recognize L1 U L2 iff

$Q_3 = Q \cup R \cup \{s_0\}$

Start state: s_0

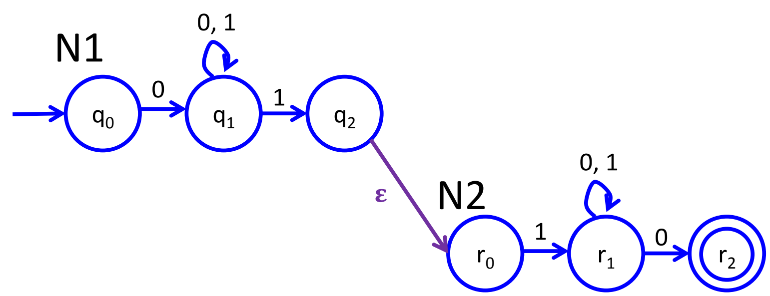
$F_1 = F_2 \cup F_3$

$$\delta_3(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q \\ \delta_2(q, a) & \text{if } q \in R \\ \{q_0, r_0\} & \text{if } q = s_0 \text{ and } a = \varepsilon \end{cases}$$

Closure under concatenation

Given two regular languages A_1 and A_2 recognized by N1 and N2 respectively, construct N to recognize $A_1 \cdot A_2$

Concatenation: $L_1 \cdot L_2$



Closure of regular languages under concatenation

This is a good example of how to write up a general proof by construction

Let N1 = $(Q, \Sigma, \delta_1, q_0, F_1)$ recognize L1

Let N2 = $(R, \Sigma, \delta_2, r_0, F_2)$ recognize L2

N3 = $(Q_3, \Sigma, \delta_3, s_0, F_3)$ will recognize $L_1 \cdot L_2$ iff

$Q_3 = Q \cup R$

Start state: q_0

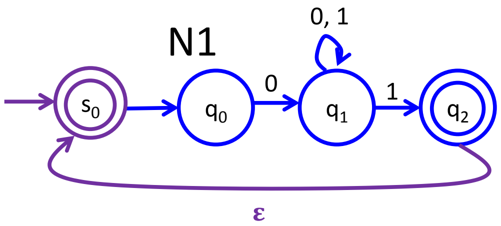
$F_1 = F_3$

$$\delta_3(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q \\ \delta_2(q, a) & \text{if } q \in R \\ r_0 & \text{if } q \in F_1 \text{ and } a = \varepsilon \end{cases}$$

Closure under star

Prove if A_1 is regular, A_1^* is also regular

Star: L_1^*



Closure of regular languages under star

This is a good example of how to write up a general proof by construction

Let N1 = $(Q, \Sigma, \delta_1, q_0, F_1)$ recognize L1

N3 = $(Q_3, \Sigma, \delta_3, s_0, F_3)$ will recognize L1* iff

$Q_3 = Q \cup \{s_0\}$

Start state: s_0

$F_1 = F_3 \cup \{s_0\}$

$$\delta_3(q, a) = \begin{cases} \delta_1(q, a) & \text{if } q \in Q \\ q_0 & \text{if } q = s_0 \text{ and } a = \varepsilon \\ s_0 & \text{if } q \in F_1 \text{ and } a = \varepsilon \end{cases}$$

Regular expressions

A regular expression is description of a set of possible strings using a single characters and possibly including regular operations

Examples:

- $(0 \cup 1)0^*$
- $(0 \cup 1)^*$

Regular expressions – formal definition

R is a regular expression if R is

- a, for some a in alphabet Σ
- ε
- $\emptyset$
- $R1 \cup R2$, where R1 and R2 are regular expressions
- $R1 \cdot R2$, where R1 and R2 are regular expressions
- $R1^*$, where R1 is a regular expression

This is a recursive definition

Examples of Regular Expressions

- 0^*10^*={1, 010, 100, 00100, 001, …} =
  {w | w contains exactly one 1}
- $\Sigma^*1\Sigma^*$={1,11,01,011,001,110,111,…} =
  {w | w contains at least one 1}
- $01 \cup 10$={01, 10}
- $(0 \cup \varepsilon)(1 \cup \varepsilon)$={01,0,1,$\varepsilon$}

71

FA can recognize any Regular Expression

Theorem: A language is regular if and only if some regular expression describes it

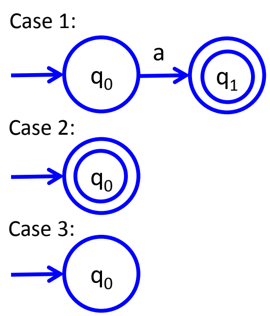
- Prove: If a language is described by a regular expression, then it is regular
- Prove: If a language is regular, then it is described by a regular expression

72

Prove if language described regular expression, it is regular (recognized by FSA)

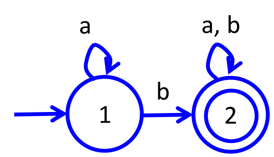
Each regular expression is either

- Case 1: $a \epsilon \Sigma$
- Case 2: ε
- Case 3: ∅
- Case 4: R1 ∪ R2 – Theorem 1.45
- Case 5: R1 · R2 – Theorem 1.47
- Case 6: R1* – Proven on slide 50



73

Converting from FSA to Regular Expression



$a^*b(a \cup b)^*$

74