Formal Languages, Automata and Computation

Regular Expressions

Carnegie Mellon University in Qatar

Summary

- Nondeterminism
  - Clone the FA at choice points

SUMMARY

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify

Summary

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify
- Nondeterministic FA

SUMMARY

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify
- Nondeterministic FA
  - Multiple transitions from a state with the same input symbol

SUMMARY

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify
- Nondeterministic FA
  - Multiple transitions from a state with the same input symbol
  - ϵ-transitions

Summary

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify
- Nondeterministic FA
  - Multiple transitions from a state with the same input symbol
  - ϵ-transitions
- NFAs are equivalent to DFAs

Summary

- Nondeterminism
  - Clone the FA at choice points
  - Guess and verify
- Nondeterministic FA
  - Multiple transitions from a state with the same input symbol
  - ϵ-transitions
- NFAs are equivalent to DFAs
  - Determinization procedure builds a DFA with up to 2^k states for an NFA with k states.

Closure Theorems

Theorem

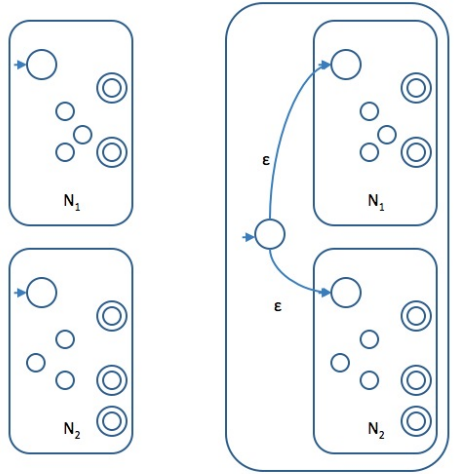
The class of regular languages is closed under the union *operation.*

Closure Theorems

Theorem

The class of regular languages is closed under the union operation.

Proof Idea Based on NFAs



CLOSURE THEOREMS

THEOREM

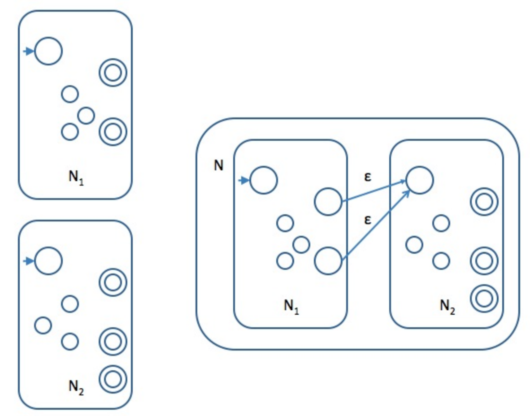
The class of regular languages is closed under the concatenation operation.

Closure Theorems

Theorem

The class of regular languages is closed under the concatenation operation.

Proof Idea based on NFAs



Closure Theorems

Theorem

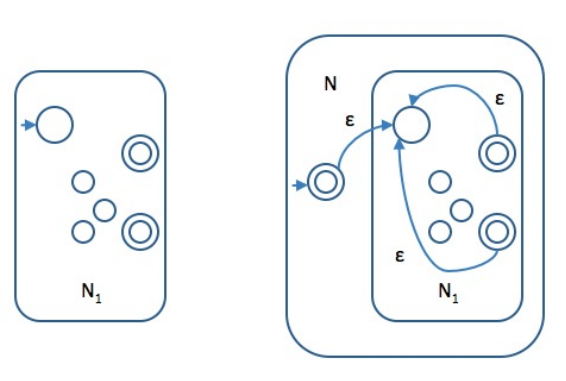
The class of regular languages is closed under the star operation.

Closure Theorems

Theorem

The class of regular languages is closed under the star operation.

Proof Idea based on NFAs



Regular Expressions

- DFAs are finite descriptions of (finite or infinite) sets of strings

- DFAs are finite descriptions of (finite or infinite) sets of strings
  - Finite number of symbols, states, transitions

Regular Expressions

- DFAs are finite descriptions of (finite or infinite) sets of strings
  - Finite number of symbols, states, transitions
- Regular Expressions provide an algebraic expression framework to describe the same class of strings

Regular Expressions

- DFAs are finite descriptions of (finite or infinite) sets of strings
  - Finite number of symbols, states, transitions
- Regular Expressions provide an algebraic expression framework to describe the same class of strings
- Thus, DFAs and Regular Expressions are equivalent.

Regular Expressions

- For every regular expression, there is a corresponding regular set or language

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expressions

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expression	**Regular Set**

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expression	**Regular Set**
ϕ	$\{\}$

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expression	**Regular Set**
ϕ	$\{\}$
a for a $\in \Sigma$	$\{a\}$

Regular Expressions

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expression	**Regular Set**
ϕ	$\{\}$
a for a $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$

Regular Expressions

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expression	**Regular Set**
ϕ	$\{\}$
a for a $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$

Regular Expressions

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expression	**Regular Set**
ϕ	$\{\}$
a for a $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 R_2)$	$L(R_1) L(R_2)$

Regular Expressions

- For every regular expression, there is a corresponding regular set or language
- R, R_1, R_2 are regular expressions; $L(R)$ denotes the corresponding regular set

Regular Expression	**Regular Set**
ϕ	$\{\}$
a for a $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 R_2)$	$L(R_1)L(R_2)$
(R^*)	$L(R)^*$

Regular Expressions – More Syntax

Regular Expression	Regular Set
ϕ	$\{\}$
a for $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
(R^*)	$L(R)^*$

- Also some books use $R_1 + R_2$ to denote union.

Regular Expressions– More Syntax

Regular Expression	Regular Set
ϕ	$\{\}$
a for $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
(R^*)	$L(R)^*$

- Also some books use $R_1 + R_2$ to denote union.
- In $(\ldots)$, the parenthesis can be deleted

Regular Expressions– More Syntax

Regular Expression	Regular Set
ϕ	$\{\}$
a for $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
(R^*)	$L(R)^*$

- Also some books use $R_1 + R_2$ to denote union.
- In $(\ldots)$, the parenthesis can be deleted
  - In which case, interpretation is done in the precedence order: star, concatenation and then union.

Regular Expressions– More Syntax

Regular Expression	Regular Set
ϕ	$\{\}$
a for $\in \Sigma$	$\{a\}$
ϵ	$\{\epsilon\}$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
(R^*)	$L(R)^*$

- Also some books use $R_1 + R_2$ to denote union.
- In $(\ldots)$, the parenthesis can be deleted
  - In which case, interpretation is done in the precedence order: star, concatenation and then union.
- $R^+ = RR^*$ and R^k for k-fold concatenation are useful shorthands.

Regular Expression Examples

Regular Expression		Regular Language
0^*10^*	$\rightarrow$	

Regular Expression Examples

Regular Expression		**Regular Language**
$\mathbf{0}^*\mathbf{1}\mathbf{0}^*$	$\rightarrow$	$\{\omega \mid \omega \text{ contains a single } \mathsf{1}\}$

Regular Expression Examples

Regular Expression		Regular Language
$\mathbf{0}^*\mathbf{1}\mathbf{0}^*$	$\rightarrow$	$\{\omega \mid \omega \text{ contains a single } \mathbf{1}\}$
$(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^*$	$\rightarrow$	

Regular Expression Examples

Regular Expression		**Regular Language**
$\mathbf{0}^*\mathbf{1}\mathbf{0}^*$	$\rightarrow$	$\{\omega \mid \omega$ contains a single $\mathsf{1}\}$
$(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^*$	$\rightarrow$	$\{\omega \mid \omega$ has at least one $\mathsf{1}\}$

Regular Expression		Regular Language
$\mathbf{0}^*\mathbf{1}\mathbf{0}^*$	$\rightarrow$	$\{\omega \vert \omega$ contains a single $\mathsf{1}\}$
$(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^*$	$\rightarrow$	$\{\omega \vert \omega$ has at least one $\mathsf{1}\}$
$\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*\mathbf{0} \cup \mathbf{1}(\mathbf{0} \cup \mathbf{1})^*\mathbf{1} \cup \mathbf{0} \cup \mathbf{1}$	$\rightarrow$	

Regular Expression Examples

Regular Expression		Regular Language
$\mathbf{0}^*\mathbf{1}\mathbf{0}^*$	$\rightarrow$	$\{\omega \mid \omega$ contains a single $\mathbf{1}\}$
$(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^*$	$\rightarrow$	$\{\omega \mid \omega$ has at least one $\mathbf{1}\}$
$\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*\mathbf{0} \cup \mathbf{1}(\mathbf{0} \cup \mathbf{1})^*\mathbf{1} \cup \mathbf{0} \cup \mathbf{1}$	$\rightarrow$	$\{\omega \mid \omega$ starts and ends with the same symbol$\}$

REGULAR EXPRESSION EXAMPLES

Regular Expression		Regular Language
$\mathbf{0}^*\mathbf{1}\mathbf{0}^*$	$\rightarrow$	$\{\omega \mid \omega$ contains a single $1\}$
$(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^*$	$\rightarrow$	$\{\omega \mid \omega$ has at least one $1\}$
$\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*\mathbf{0} \cup \mathbf{1}(\mathbf{0} \cup \mathbf{1})^*\mathbf{1} \cup \mathbf{0} \cup \mathbf{1}$	$\rightarrow$	$\{\omega \mid \omega$ starts and ends with the same symbol$\}$
$(\mathbf{0}^*\mathbf{1}\mathbf{0}^*\mathbf{1})^*\mathbf{0}^*$	$\rightarrow$	

Regular Expression Examples

Regular Expression		Regular Language
$\mathbf{0}^*\mathbf{1}\mathbf{0}^*$	$\rightarrow$	$\{\omega \mid \omega$ contains a single $\mathbf{1}\}$
$(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})^*$	$\rightarrow$	$\{\omega \mid \omega$ has at least one $\mathbf{1}\}$
$\mathbf{0}(\mathbf{0} \cup \mathbf{1})^*\mathbf{0} \cup \mathbf{1}(\mathbf{0} \cup \mathbf{1})^*\mathbf{1} \cup \mathbf{0} \cup \mathbf{1}$	$\rightarrow$	$\{\omega \mid \omega$ starts and ends with the same symbol$\}$
$(\mathbf{0}^*\mathbf{1}\mathbf{0}^*\mathbf{1})^*\mathbf{0}^*$	$\rightarrow$	$\{\omega \mid n_1(\omega)$ is even$\}$

- All strings with at least one pair of consecutive 0s

Writing Regular Expressions

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$

Writing Regular Expressions

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$
- All strings such that fourth symbol from the end is a 1

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$
- All strings such that fourth symbol from the end is a 1
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})$

Writing Regular Expressions

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$
- All strings such that fourth symbol from the end is a 1
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})$
- All strings with no pair of consecutive 0s

WRITING REGULAR EXPRESSIONS

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$
- All strings such that fourth symbol from the end is a 1
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})$
- All strings with no pair of consecutive 0s
  - $(\mathbf{1}^*\mathbf{011}^*)^*(\mathbf{0} \cup \epsilon) \cup \mathbf{1}*$

Writing Regular Expressions

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$
- All strings such that fourth symbol from the end is a 1
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})$
- All strings with no pair of consecutive 0s
  - $(\mathbf{1}^*\mathbf{011}^*)^*(\mathbf{0} \cup \epsilon) \cup \mathbf{1}*$
  - Strings consist of repetitions of 1 or 01 or two boundary cases: $(\mathbf{1} \cup \mathbf{01})^*(\mathbf{0} \cup \epsilon)$

Writing Regular Expressions

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$
- All strings such that fourth symbol from the end is a 1
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})$
- All strings with no pair of consecutive 0s
  - $(\mathbf{1}^*\mathbf{011}^*)^*(\mathbf{0} \cup \epsilon) \cup \mathbf{1}*$
  - Strings consist of repetitions of 1 or 01 or two boundary cases: $(\mathbf{1} \cup \mathbf{01})^*(\mathbf{0} \cup \epsilon)$
- All strings that do not end in 01.

Writing Regular Expressions

- All strings with at least one pair of consecutive 0s
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{00}(\mathbf{0} \cup \mathbf{1})^*$
- All strings such that fourth symbol from the end is a 1
  - $(\mathbf{0} \cup \mathbf{1})^*\mathbf{1}(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})(\mathbf{0} \cup \mathbf{1})$
- All strings with no pair of consecutive 0s
  - $(\mathbf{1}^*\mathbf{011}^*)^*(\mathbf{0} \cup \epsilon) \cup \mathbf{1}*$
  - Strings consist of repetitions of 1 or 01 or two boundary cases: $(\mathbf{1} \cup \mathbf{01})^*(\mathbf{0} \cup \epsilon)$
- All strings that do not end in 01.
  - $(\mathbf{0} \cup \mathbf{1})^*(\mathbf{00} \cup \mathbf{10} \cup \mathbf{11}) \cup \mathbf{0} \cup \mathbf{1} \cup \epsilon$

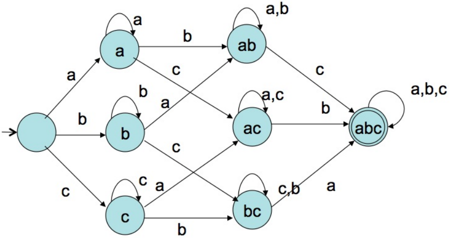
Writing Regular Expressions

- All strings over $\Sigma = \{a, b, c\}$ that contain every symbol at least once.

Writing Regular Expressions

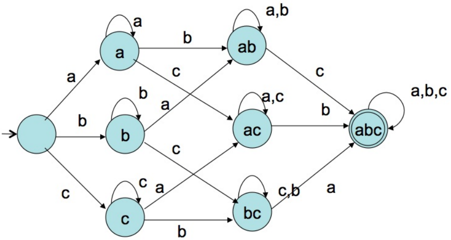
- All strings over $\Sigma = \{a, b, c\}$ that contain every symbol at least once.
- $(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{a}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{b}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{c}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^* \cup$
  $(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{a}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{c}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{b}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^* \cup$
  $(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{b}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{a}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{c}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^* \cup$
  $(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{b}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{c}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{a}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^* \cup$
  $(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{c}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{a}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{b}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^* \cup$
  $(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{c}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{b}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*\mathbf{a}(\mathbf{a} \cup \mathbf{b} \cup \mathbf{c})^*$

- All strings over $\Sigma = \{a, b, c\}$ that contain every symbol at least once.



Writing Regular Expressions

- All strings over $\Sigma = \{a, b, c\}$ that contain every symbol at least once.



- DFAs and REs may need different ways of looking at the problem.
  - For the DFA, you count symbols
  - For the RE, you enumerate all possible patterns

RE IDENTITIES

- $\mathbf{R} \cup \phi = \mathbf{R}$

RE Identities

- $\mathbf{R} \cup \phi = \mathbf{R}$
- $\mathbf{R}\epsilon = \epsilon\mathbf{R} = \mathbf{R}$

RE Identities

- $\mathbf{R} \cup \phi = \mathbf{R}$
- $\mathbf{R}\epsilon = \epsilon\mathbf{R} = \mathbf{R}$
- $\phi^* = \epsilon$

RE Identities

- $\mathbf{R} \cup \phi = \mathbf{R}$
- $\mathbf{R}\epsilon = \epsilon\mathbf{R} = \mathbf{R}$
- $\phi^* = \epsilon$
- Note that we do not have explicit operators for intersection or complementation!

DIGRESSION: REs IN REAL LIFE

- Linux/Unix Shell, Perl, Awk, Python all have built in regular expression support for pattern matching functionality
- See `http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html`

Digression: REs in Real life

- Linux/Unix Shell, Perl, Awk, Python all have built in regular expression support for pattern matching functionality
- See `http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html`
- Mostly some syntactic extensions/changes to basic regular expressions with some additional functionality for remembering matches

Digression: REs in Real life

- Linux/Unix Shell, Perl, Awk, Python all have built in regular expression support for pattern matching functionality
- See `http://www.wdvl.com/Authoring/Languages/Perl/PerlfortheWeb/perlintro2_table1.html`
- Mostly some syntactic extensions/changes to basic regular expressions with some additional functionality for remembering matches
- Substring matches in a string!
- Search for and download *Regex Coach* to learn and experiment with regular expression matching

Equivalence with Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Equivalence with Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Lemma- The *if* part

If a language is described by a regular expression, then it is regular

Equivalence with Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Lemma- The *if* part

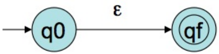
If a language is described by a regular expression, then it is regular

Proof Idea

Inductively convert a given regular expression to an NFA.

Regular Expression	Corresponding NFA
ϕ	

Converting REs to NFAs: Basis Cases

Regular Expression	Corresponding NFA
ϕ	
ϵ	

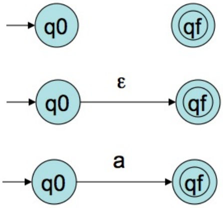
CONVERTING REs TO NFAs: BASIS CASES

Regular Expression **Corresponding NFA**

ϕ

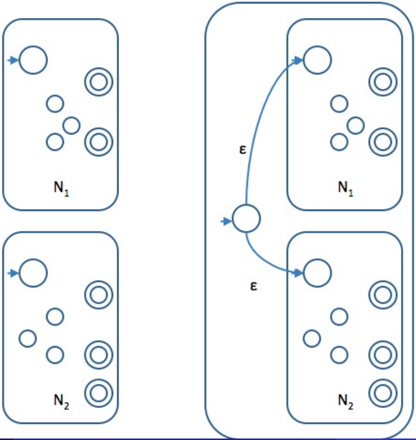
ϵ

a for $a \in \Sigma$



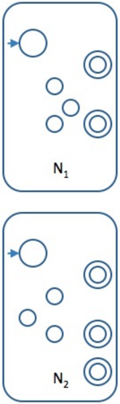
Union

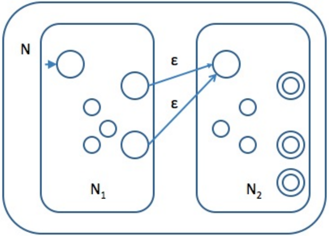
- Let N_1 and N_2 be NFAs for R_1 and R_2 respectively. Then the NFA for $\mathbf{R_1} \cup \mathbf{R_2}$ is



Concatenation

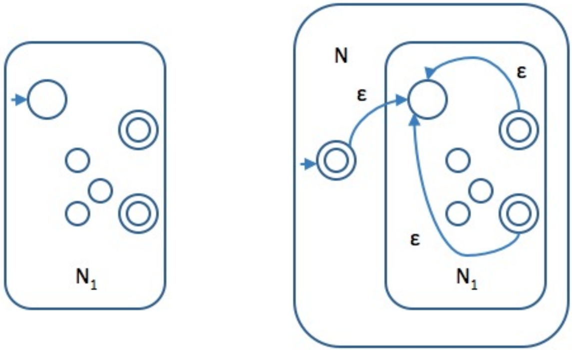
- Let N_1 and N_2 be NFAs for R_1 and R_2 respectively. Then the NFA for $\mathbf{R_1R_2}$ is





Star

- Let N be NFAs for R. Then the NFA for $\mathbf{R}^*$ is



RE TO NFA CONVERSION EXAMPLE

- Let's convert $(\mathbf{a} \cup \mathbf{b})^* \mathbf{aba}$ to an NFA.

RE to NFA to DFA

- Regular Expression $\rightarrow$ NFA (possibly with ϵ-transitions)

RE TO NFA TO DFA

- Regular Expression $\rightarrow$ NFA (possibly with ϵ-transitions)
- NFA $\rightarrow$ DFA via determinization

Equivalence with Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Equivalence with Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Lemma – The *only if* part

If a language is regular then it is described by a regular expression

EQUIVALENCE WITH FINITE AUTOMATA

THEOREM

A language is regular if and only if some regular expression describes it.

LEMMA – THE *only if* PART

If a language is regular then it is described by a regular expression

PROOF IDEA

- Generalized transitions: label transitions with regular expressions

Equivalence with Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Lemma – The *only if* part

If a language is regular then it is described by a regular expression

Proof Idea

- Generalized transitions: label transitions with regular expressions
- Generalized NFAs (GNFA)

Equivalence with Finite Automata

Theorem

A language is regular if and only if some regular expression describes it.

Lemma – The *only if* part

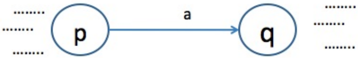
If a language is regular then it is described by a regular expression

Proof Idea

- Generalized transitions: label transitions with regular expressions
- Generalized NFAs (GNFA)
- Iteratively eliminate states of the GNFA one by one, until only two states and a single generalized transition is left.

Generalized Transitions

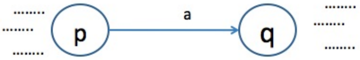
- DFAs have single symbols as transition labels



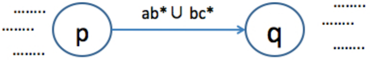
  - If you are in state p and the next input symbol matches a, go to state q

Generalized Transitions

- DFAs have single symbols as transition labels

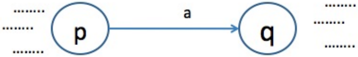


  - If you are in state p and the next input symbol matches a, go to state q
- Now consider



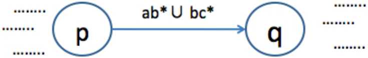
GENERALIZED TRANSITIONS

- DFAs have single symbols as transition labels



  - If you are in state p and the next input symbol matches a, go to state q

- Now consider



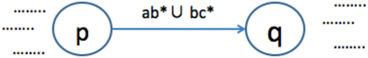
  - If you are in state p and a prefix of the remaining input matches the regular expression $\mathbf{ab}^* \cup \mathbf{bc}^*$ then go to state q

Generalized Transitions and NFA

- A generalized transition is a transition whose label is a regular expression

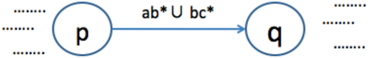
Generalized Transitions and NFA

- A generalized transition is a transition whose label is a regular expression



Generalized Transitions and NFA

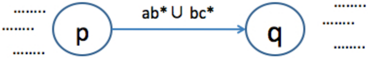
- A generalized transition is a transition whose label is a regular expression



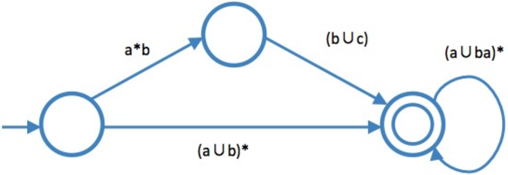
- A Generalized NFA is an NFA with generalized transitions.

Generalized Transitions and NFA

- A generalized transition is a transition whose label is a regular expression



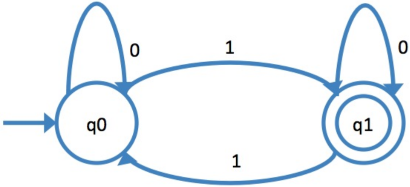
- A Generalized NFA is an NFA with generalized transitions.



  - In fact, all standard DFA transitions are generalized transitions with regular expressions of a single symbol!

Generalized Transitions

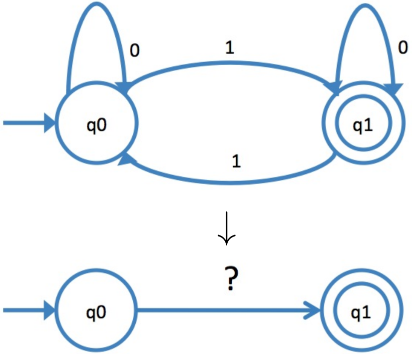
- Consider the 2-state DFA



$\downarrow$

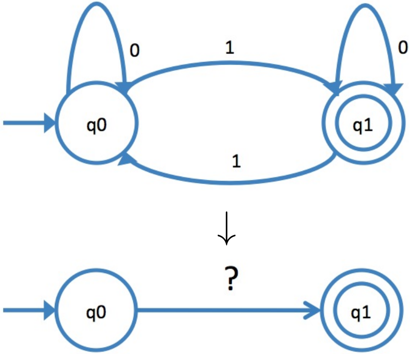
Generalized Transitions

- Consider the 2-state DFA



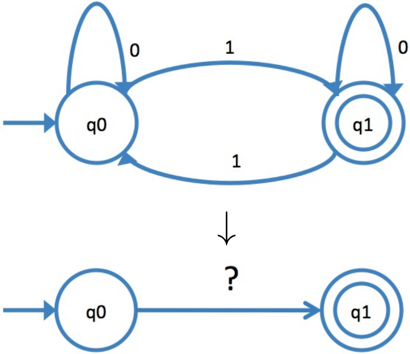
Generalized Transitions

- Consider the 2-state DFA



- **0*****1** takes the DFA from state q_0 to q_1

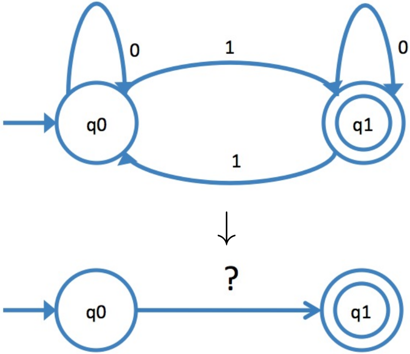
- Consider the 2-state DFA



- $\mathbf{0}^*\mathbf{1}$ takes the DFA from state q_0 to q_1
- $(\mathbf{0} \cup \mathbf{10}^*\mathbf{1})^*$ takes the machine from q_1 back to q_1

Generalized Transitions

- Consider the 2-state DFA



- $\mathbf{0}^*\mathbf{1}$ takes the DFA from state q_0 to q_1
- $(\mathbf{0} \cup \mathbf{10}^*\mathbf{1})^*$ takes the machine from q_1 back to q_1
- So $\mathbf{?} = \mathbf{0}^*\mathbf{1}(\mathbf{0} \cup \mathbf{10}^*\mathbf{1})^*$ represents all strings that take the DFA from state q_0 to q_1

Generalized NFAs

- Take any NFA and transform it into a GNFA
  - with only two states: one start and one accept

Generalized NFAs

- Take any NFA and transform it into a GNFA
  - with only two states: one start and one accept
  - with one generalized transition

Generalized NFAs

- Take any NFA and transform it into a GNFA
  - with only two states: one start and one accept
  - with one generalized transition
- then we can “read” the regular expression from the label of the generalized transition (as in the example above)