Theory of Computation
(Finite Automata)

Pramod Ganapathi
Department of Computer Science
State University of New York at Stony Brook

January 24, 2021

Contents

Contents

Deterministic Finite Automata (DFA)

Electric bulb

Problem

- Design the logic behind an electric bulb.

Electric bulb

Problem

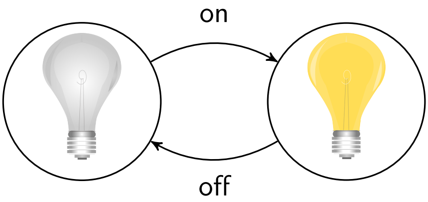
- Design the logic behind an electric bulb.

Solution

- Diagram.

- Analysis.

  States $= \{\text{nolight}, \text{light}\}$, Input $= \{\text{off}, \text{on}\}$
- Finite Automaton.



Multispeed fan

Problem

- Design the logic behind a multispeed fan.

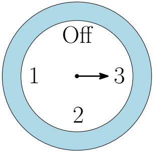
Multispeed fan

Problem

- Design the logic behind a multispeed fan.

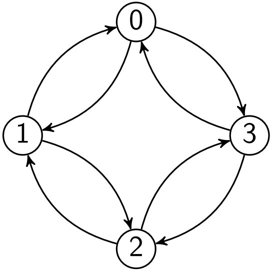
Solution

- Diagram.



- Analysis.
  States $= \{0, 1, 2, 3\}$
  Input $= \{\circlearrowright, \circlearrowleft\}$

- Finite Automaton.



Automatic doors

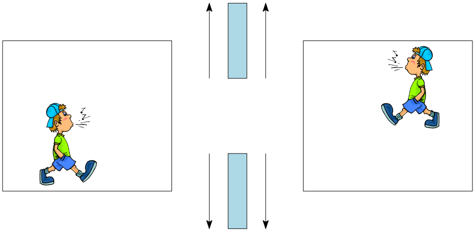
Problem

- Design the logic behind automatic doors in Walmart.

Automatic doors

Solution

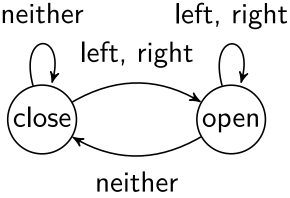
- Diagram.



- Analysis.

  States = {close, open}, Input = {left, right, neither}

- Finite Automaton.



Basic features of finite automata

- A finite automaton is a simple computer with extremely limited memory
- A finite automaton has a finite set of states
- Current state of a finite automaton changes when it reads an input symbol
- A finite automaton acts as a language acceptor i.e., outputs "yes" or "no"

Why should you care?

Deterministic Finite Automata (DFA) are everywhere.

- ATMs
- Ticket machines
- Vending machines
- Traffic signal systems
- Calculators
- Digital watches
- Automatic doors
- Elevators
- Washing machines
- Dishwashing machines
- Thermostats
- Train switches
- (CS) Compilers
- (CS) Search engines
- (CS) Regular expressions

Why should you care?

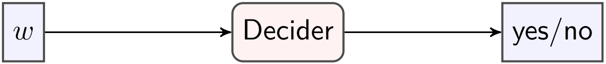
Probabilistic Finite Automata (PFA) are everywhere, too.

- Speech recognition
- Optical character recognition
- Thermodynamics
- Statistical mechanics
- Chemical reactions
- Information theory
- Queueing theory
- PageRank algorithm
- Statistics
- Reinforcement learning
- Price changes in finance
- Genetics
- Algorithmic music composition
- Bioinformatics
- Probabilistic forecasting

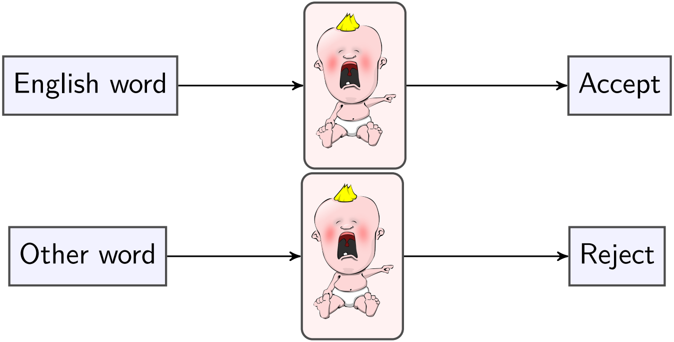
What is a decision problem?

Definition

- A decision problem is a computational problem with a 'yes' or 'no' answer.
- A computer that solves a decision problem is a decider.
  Input to a decider: A string w
  Output of a decider: Accept (w is in the language) or Reject (w is not in the language)

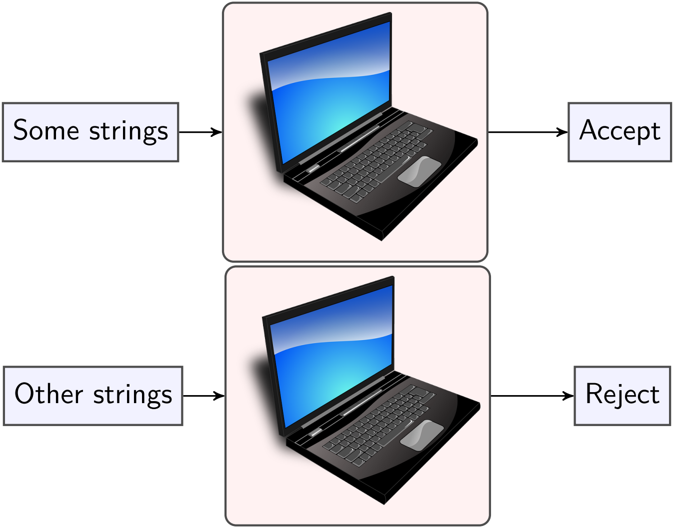


What is a decision problem?



- Language = English language = $\{\text{milk}, \text{food}, \text{sleep}, \ldots\}$ ▷ Accept
- Not in language = $\{\text{zffgb}, \text{cdcapqw}, \ldots\}$ ▷ Reject

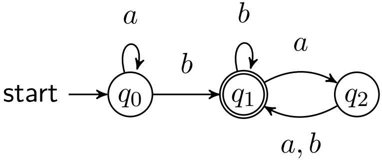
What is a decision problem?



How does a DFA work?

Problem

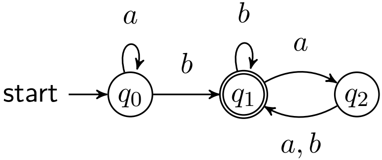
- Does the DFA accept the string $bbab$?



How does a DFA work?

Problem

- Does the DFA accept the string $bbab$?



Solution

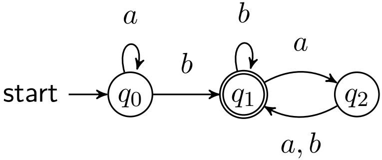
The DFA accepts the string $bbab$. The computation is:

1. Start in state q_0
2. Read b, follow transition from q_0 to q_1.
3. Read b, follow transition from q_1 to q_1.
4. Read a, follow transition from q_1 to q_2.
5. Read b, follow transition from q_2 to q_1.
6. Accept because the DFA is in an accept state q_1 at the end of the input.

How does a DFA work?

Problem

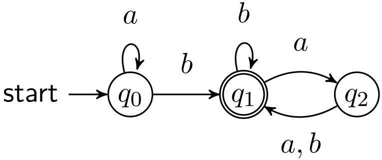
- Does the DFA accept the string $aaba$?



How does a DFA work?

Problem

- Does the DFA accept the string $aaba$?

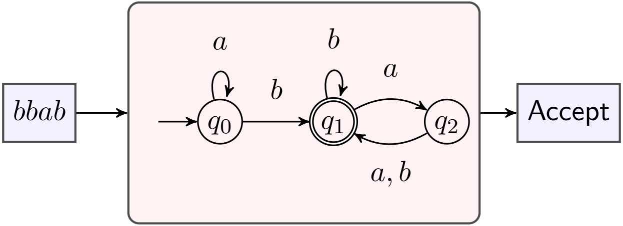


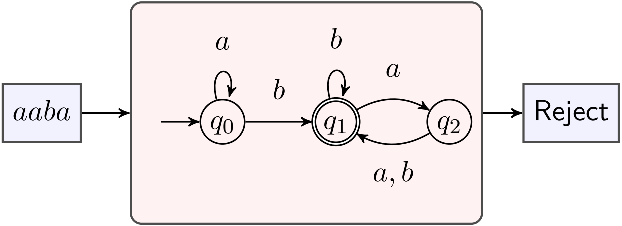
Solution

The DFA rejects the string $aaba$. The computation is:

1. Start in state q_0
2. Read a, follow transition from q_0 to q_0.
3. Read a, follow transition from q_0 to q_0.
4. Read b, follow transition from q_0 to q_1.
5. Read a, follow transition from q_1 to q_2.
6. Reject because the DFA is in a reject state q_2 at the end of the input.

How does a DFA work?

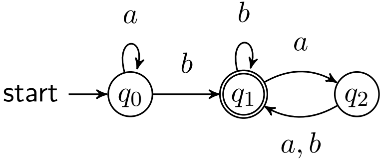




How does a DFA work?

Problem

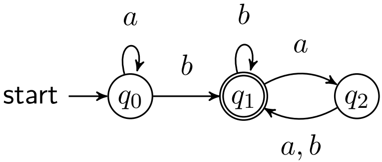
- What language does the DFA accept?



How does a DFA work?

Problem

- What language does the DFA accept?



Examples

- The DFA accepts the following strings:
  $b, ab, bb, aabbbb, ababababab, \ldots$ ▷ ends with b
  $baa, abaa, ababaaaaaa, \ldots$ ▷ ends with b followed by even a's
- The DFA rejects the following strings:
  $a, ba, babaaa, \ldots$
- What language does the DFA accept?

Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\epsilon, a, aa, aaa, aaaa, \ldots\}$

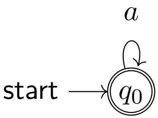
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\epsilon, a, aa, aaa, aaaa, \ldots\}$

Solution

- Language L: $\Sigma^* = \{\epsilon, a, aa, aaa, aaaa, \ldots\}$
- Expression: a^*
- Deterministic Finite Automaton (DFA) M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\}$

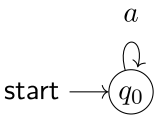
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\}$

Solution

- Language L: $\phi = \{\}$ ▷ Empty language
- Expression: ϕ
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{a, aa, aaa, aaaa, \ldots\}$

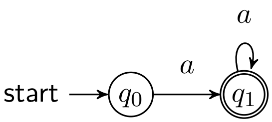
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{a, aa, aaa, aaaa, \ldots\}$

Solution

- Language L: $\Sigma^* - \{\epsilon\} = \{a, aa, aaa, aaaa, \ldots\}$
- Expression: a^+
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\epsilon\}$

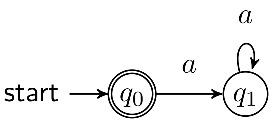
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\epsilon\}$

Solution

- Language L: $= \{\epsilon\}$
- Expression: ϵ
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{aaa\}$

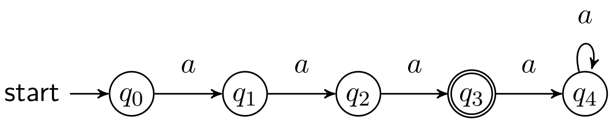
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{aaa\}$

Solution

- Language L: $\{aaa\}$
- Expression: aaa
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with even size}\}$

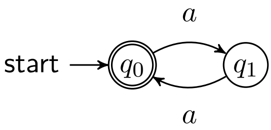
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with even size}\}$

Solution

- Language L: $\{\epsilon, aa, aaaa, aaaaaa, \ldots\}$
- Expression: $(aa)^*$
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd size}\}$

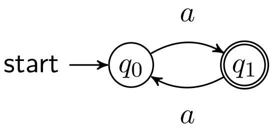
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd size}\}$

Solution

- Language L: $\{a, aaa, aaaaa, \ldots\}$
- Expression: $a(aa)^*$
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings of size divisible by 3$\}$

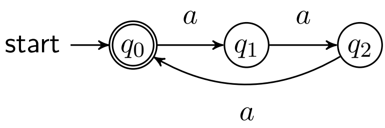
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings of size divisible by 3$\}$

Solution

- Language L: $\{\epsilon, aaa, aaaaaa, aaaaaaaaa, \ldots\}$
- Expression: $(aaa)^*$
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings of size not divisible by 3$\}$

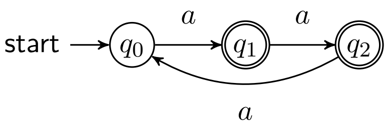
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings of size not divisible by 3$\}$

Solution

- Language L: $\{a, aa, aaaa, aaaaa, \ldots\}$
- Expression: $(a \cup aa)(aaa)^*$
- DFA M:



Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings of size divisible by 6$\}$

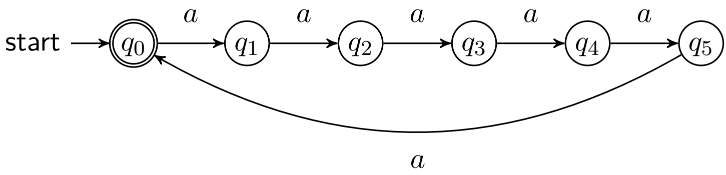
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings of size divisible by 6$\}$

Solution

- Language L: $\{\epsilon, aaaaaa, aaaaaaaaaaaa, \ldots\}$
- Expression: $(aaaaaa)^*$
- DFA M:



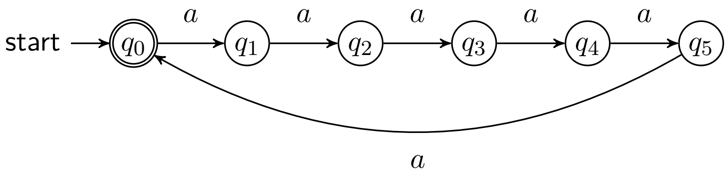
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings of size divisible by 6$\}$

Solution

- Language L: $\{\epsilon, aaaaaa, aaaaaaaaaaaa, \ldots\}$
- Expression: $(aaaaaa)^*$
- DFA M:



- Can you think of another approach?

Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings of size divisible by 6}\}$

Solution

- Let $n =$ string size
- Observation
  $n \bmod 6 = 0 \Longleftrightarrow n \bmod 2 = 0$ and $n \bmod 3 = 0$
- Idea
  Build DFA M_1 for $n \bmod 2 = 0$.
  Build DFA M_2 for $n \bmod 3 = 0$.
  Run M_1 and M_2 in parallel.
  Accept a string if both DFAs M_1 and M_2 accept the string.
  Reject a string if at least one of the DFAs M_1 and M_2 reject the string.
- It is possible to build complicated DFAs from simpler DFAs

Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with size } n \text{ where } n \bmod 4 = 2\}$

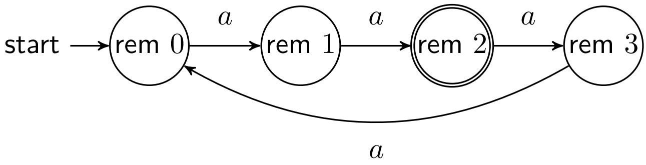
Construct DFA for $\Sigma = \{a\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with size } n \text{ where } n \bmod 4 = 2\}$

Solution

- Language L: $\{aaa, aaaaaaa, aaaaaaaaaaa, \ldots\}$
- Expression: $aa(aaaa)^*$
- DFA M:



- What about strings with size n where $n \bmod k = i$?

Construct DFA for $\Sigma = \{a\}$

More Problems

Construct a DFA that accepts all strings from the language $L = \{\text{strings with size } n\}$ such that

- $n^2 - 5n + 6 = 0$
- $n \in [4, 37]$
- n is a perfect cube
- n is a prime number
- n satisfies a mathematical function $f(n)$

Specifying a DFA

The specification of DFA consists of:

- A (finite) alphabet
- A (finite) set of states
- Which state is the start state?
- Which states are the final states?
- What is the transition from each state, on each input character?

What is a deterministic finite automaton (DFA)?

- Deterministic = Events can be determined precisely
- Finite = Finite and small amount of space used
- Automaton = Computing machine

What is a deterministic finite automaton (DFA)?

- Deterministic = Events can be determined precisely
- Finite = Finite and small amount of space used
- Automaton = Computing machine

Definition

A deterministic finite automaton (DFA) M is a 5-tuple
$M = (Q, \Sigma, \delta, q_0, F)$, where,

1. Q: A finite set (set of states). ▷ Space (computer memory)
2. Σ: A finite set (alphabet).
3. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function.
   ▷ Time (computation)
4. q_0: The start state (belongs to Q).
5. F: The set of accepting/final states, where $F \in Q$.

Acceptance and rejection of strings

Definition

- A DFA accepts a string $w = w_1 w_2 \ldots w_k$ iff there exists a sequence of states $r_0, r_1, \ldots, r_k$ such that the current state starts from the start state and ends at a final state when all the symbols of w have been read.
- A DFA rejects a string iff it does not accept it.

What is a regular language?

Definition

- We say that a DFA M accepts a language L if $L = \{w \mid M \text{ accepts } w\}$.
- A language is called a regular language if some DFA accepts or recognizes it.

Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd number of } b\text{'s}\}$

Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd number of } b\text{'s}\}$

Solution

States

- q_{odd}: DFA is in this state if it has read odd b's.
- q_{even}: DFA is in this state if it has read even b's.

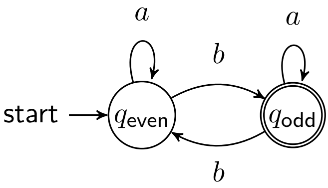
Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings with odd number of b's$\}$

Solution

- Language L: $\{$strings with odd number of b's$\}$
- Expression: $a^*b(a \cup ba^*b)^*$ or $a^*ba^*(ba^*ba^*)^*$
- DFA M:



Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with odd number of } b\text{'s}\}$

Solution (continued)

- DFA M is specified as
  Set of states is $Q = \{q_{\text{even}}, q_{\text{odd}}\}$
  Set of symbols is $\Sigma = \{a, b\}$
  Start state is q_{even}
  Set of accept states is $F = \{q_{\text{odd}}\}$
  Transition function δ is:

δ	a	b
q_{even}	q_{even}	q_{odd}
q_{odd}	q_{odd}	q_{even}

Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings containing } bab\}$

Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings containing } bab\}$

Solution

States

- q_b: DFA is in this state if the last symbol read was b, but the substring bab has not been read.
- q_{ba}: DFA is in this state if the last two symbols read were ba, but the substring bab has not been read.
- q_{bab}: DFA is in this state if the substring bab has been read in the input string.
- q: In all other cases, the DFA is in this state.

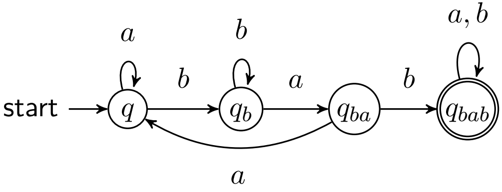
Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings containing } bab\}$

Solution (continued)

- Language L: $\{\text{strings containing } bab\}$
- Expression: $(a^*b^+aa)^*bab(a \cup b)^*$
- DFA M:



Construct DFA for $\Sigma = \{a, b\}$

Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings containing } bab\}$

Solution (continued)

- DFA M is specified as
  Set of states is $Q = \{q, q_b, q_{ba}, q_{bab}\}$
  Set of symbols is $\Sigma = \{a, b\}$
  Start state is q
  Set of accept states is $F = \{q_{bab}\}$
  Transition function δ is:

δ	a	b
q	q	q_b
q_b	q_{ba}	q_b
q_{ba}	q	q_{bab}
q_{bab}	q_{bab}	q_{bab}

Closure properties of regular languages

Properties

Let L_1 and L_2 be regular languages.

Then, the following languages are regular.

- Complement. $\overline{L_1} = \{x \mid x \in \Sigma^* \text{ and } x \notin L_1\}$.
- Union. $L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$.
- Intersection. $L_1 \cap L_2 = \{x \mid x \in L_1 \text{ and } x \in L_2\}$.
- Concatenation. $L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$.
- Star. $L_1^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in L_1\}$.

Closure properties for languages

Language	Operation $L_1 \cup L_2$	$L_1 \cap L_2$	L'	$L_1 L_2$	L^*	L^R	L^T
Regular	✓	✓	✓	✓	✓	✓	✓
DCFL	✗	✗	✓	✗	✗	✗	✗
CFL	✓	✗	✗	✓	✓	✓	✓
Recursive	✓	✓	✓	✓	✓	✓	✗
R.E.	✓	✓	✗	✓	✓	✓	✓

- $L_1 \cup L_2 =$ Union of L_1 and L_2
- $L_1 \cap L_2 =$ Intersection of L_1 and L_2
- $L' =$ Complement of L
- $L_1 L_2 =$ Concatenation of L_1 and L_2
- $L^* =$ Powers of L
- $L^R =$ Reverse of L
- $L^T =$ Finite transduction of L (may include: intersection/shuffle/perfect-shuffle/quotient with arbitrary regular languages)

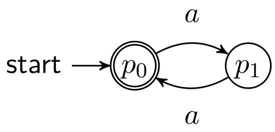
Construct DFA for $L_1 \cup L_2$

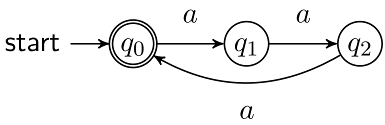
Problem

- Construct a DFA that accepts all strings from the language $L = \{\text{strings with size multiples of 2 or 3}\}$ where $\Sigma = \{a\}$

Solution

- Language $L_1 = \{\text{strings with size multiples of 2}\}$
- Language $L_2 = \{\text{strings with size multiples of 3}\}$

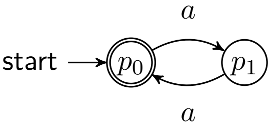


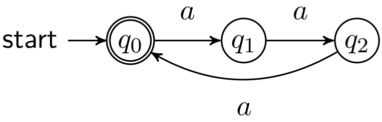


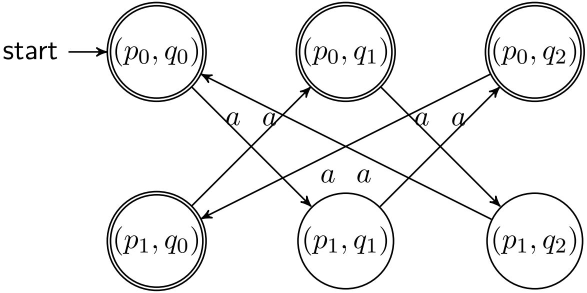
Construct DFA for $L_1 \cup L_2$

Solution (continued)

- Language $L_1 \cup L_2 = \{$strings with size multiples of 2 and 3$\}$







Construct DFA for $L_1 \cup L_2$

Union

- Let M_1 accept L_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  Let M_2 accept L_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- Let M accept $L_1 \cup L_2$, where $M = (Q, \Sigma, \delta, q, F)$. Then
  $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$ ▷ Cartesian product
  $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \quad \forall (r_1, r_2) \in Q, a \in \Sigma$
  $q_0 = (q_1, q_2)$
  $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

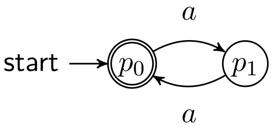
Construct DFA for $L_1 \cap L_2$

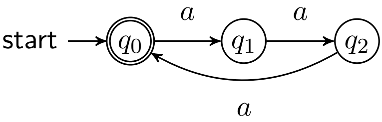
Problem

- Construct a DFA that accepts all strings from the language $L = \{$strings with size multiples of 2 and 3$\}$ where $\Sigma = \{a\}$

Solution

- Language $L_1 = \{$strings with size multiples of 2$\}$
- Language $L_2 = \{$strings with size multiples of 3$\}$

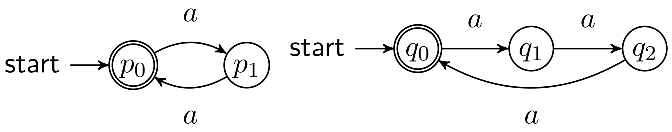


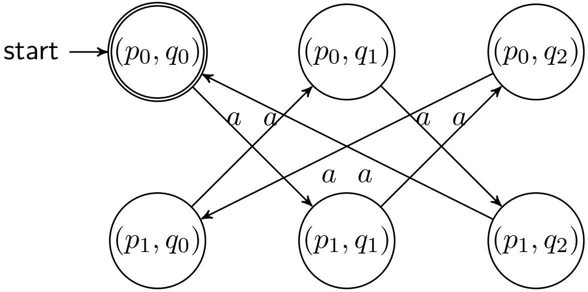


Construct DFA for $L_1 \cap L_2$

Solution (continued)

- Language $L_1 \cap L_2 = \{$strings with size multiples of 2 and 3$\}$





Construct DFA for $L_1 \cap L_2$

Intersection

- Let M_1 accept L_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  Let M_2 accept L_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
- Let M accept $L_1 \cap L_2$, where $M = (Q, \Sigma, \delta, q, F)$. Then
  $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$ $\triangleright$ Cartesian product
  $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \;\; \forall (r_1, r_2) \in Q, a \in \Sigma$
  $q_0 = (q_1, q_2)$
  $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$

Problems for practice

Problems

Assume $\Sigma = \{a, b\}$ unless otherwise mentioned.

Construct DFA's for the following languages and generalize:

- $L = \{w \mid |w| = 2\}$
- $L = \{w \mid |w| \leq 2\}$
- $L = \{w \mid |w| \geq 2\}$
- $L = \{w \mid n_a(w) = 2\}$
- $L = \{w \mid n_a(w) \leq 2\}$
- $L = \{w \mid n_a(w) \geq 2\}$
- $L = \{w \mid n_a(w) \bmod 3 = 1\}$
- $L = \{w \mid n_a(w) \bmod 2 = 0$ and $n_b(w) \bmod 2 = 0\}$
- $L = \{w \mid n_a(w) \bmod 3 = 2$ and $n_b(w) \bmod 2 = 1\}$
- $L = \{w \mid n_a(w) \bmod 5 = 3, n_b(w) \bmod 3 = 2,$ and $n_c(w) \bmod 2 = 1\}$ for $\Sigma = \{a, b, c\}$
- $L = \{w \mid n_a(w) \bmod 3 \geq n_b(w) \bmod 2\}$

Problems for practice

Problems (continued)

- $L = \{b \mid \text{binary number } b \bmod 3 = 1\}$ for $\Sigma = \{0, 1\}$
- $L = \{t \mid \text{ternary number } t \bmod 4 = 3\}$ for $\Sigma = \{0, 1, 2\}$
- $L = \{w \mid w \text{ starts with } a\}$
- $L = \{w \mid w \text{ contains } a\}$
- $L = \{w \mid w \text{ ends with } a\}$
- $L = \{w \mid w \text{ starts with } ab\}$
- $L = \{w \mid w \text{ contains } ab\}$
- $L = \{w \mid w \text{ ends with } ab\}$
- $L = \{w \mid w \text{ starts with } a \text{ and ends with } b\}$
- $L = \{w \mid w \text{ starts and ends with different symbols}\}$
- $L = \{w \mid w \text{ starts and ends with the same symbol}\}$
- $L = \{w \mid \text{every } a \text{ in } w \text{ is followed by a } b\}$
- $L = \{w \mid \text{every } a \text{ in } w \text{ is never followed by a } b\}$

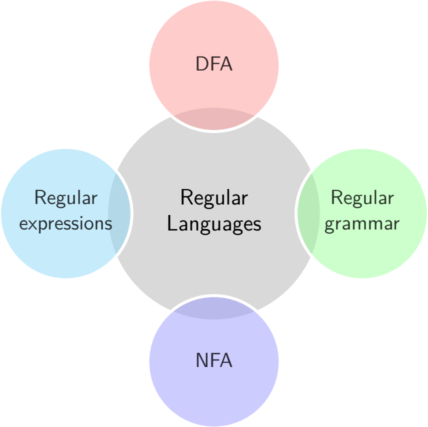
Problems for practice

Problems (continued)

- $L = \{w \mid \text{every } a \text{ in } w \text{ is followed by } bb\}$
- $L = \{w \mid \text{every } a \text{ in } w \text{ is never followed by } bb\}$
- $L = \{w \mid w = a^m b^n \text{ for } m, n \geq 1\}$
- $L = \{w \mid w = a^m b^n \text{ for } m, n \geq 0\}$
- $L = \{w \mid w = a^m b^n c^\ell \text{ for } m, n, \ell \geq 1\}$ for $\Sigma = \{a, b, c\}$
- $L = \{w \mid w = a^m b^n c^\ell \text{ for } m, n, \ell \geq 0\}$ for $\Sigma = \{a, b, c\}$
- $L = \{w \mid \text{second symbol from left end of } w \text{ is } a\}$
- $L = \{w \mid \text{second symbol from right end of } w \text{ is } a\}$
- $L = \{w \mid w = a^3 b x a^3 \text{ such that } x \in \{a, b\}^*\}$

Equivalence of different computation models

- Two machines or computational models are computationally equivalent if they accept/recognize the same language.
- The following models are computationally equivalent:
  DFA, regular expressions, NFA, and regular grammars.



Closure properties for languages

Language	Operation				
	$L_1 \cup L_2$	$L_1 \cap L_2$	$\bar{L}$	$L_1 \circ L_2$	L^*
DFA	Easy	Easy	Easy	Hard	Hard
Regex	Easy	Hard	Hard	Easy	Easy
NFA	Easy	Hard	Hard	Easy	Easy

- $L_1 \cup L_2 =$ Union of L_1 and L_2
- $L_1 \cap L_2 =$ Intersection of L_1 and L_2
- $\bar{L} =$ Complement of L
- $L_1 \circ L_2 =$ Concatenation of L_1 and L_2
- $L^* =$ Powers of L

Regular Expressions

Example

Example

- Arithmetic expression.
  $(5+3) \times 4 = 32 =$ Number
- Regular expression.
  $(a \cup b)a^* = \{a, b, aa, ba, aaa, baa, \ldots\} =$ Regular language

Application

- Regular expressions in Linux.
  Used to find patterns in filenames, file content etc.
  Used in Linux tools such as awk, grep, and Perl.
  Google search: `http://www.googleguide.com/advanced_operators_reference.html`

What is a regular expression?

Definition

- The following are regular expressions.
  $\epsilon, \phi, a \in \Sigma$.
- If R_1 and R_2 are regular expressions, then the following are regular expressions.
  (Union.) $R_1 \cup R_2$
  (Concatenation.) $R_1 \circ R_2$
  (Kleene star.) R_1^*

Examples

Regular language	Regular expression
$\{\}$	ϕ
$\{\epsilon\}$	ϵ
$\{a\}$	a
$\{a, b\}$	$a \cup b$
$\{a\}\{b\}$	ab
$\{a\}^* = \{\epsilon, a, aa, aaa, \ldots\}$	a^*
$\{aab\}^*\{a, ab\}$	$(aab)^*(a \cup ab)$
$(\{aa, bb\} \cup \{a, b\}\{aa\}^*\{ab, ba\})^*$	$(aa \cup bb \cup (a \cup b)(aa)^*(ab \cup ba))^*$

Equality

- Two regular expressions are equal if they describe the same regular language. E.g.:
  $(a^*b^*)^* = (a \cup b)^*ab(a \cup b)^* \cup b^*a^* = (a \cup b)^* = \Sigma^*$

Examples

Examples

Let $\Sigma = a \cup b$, $R^+ = RR^*$, and $R^k = \underbrace{R \cdots R}_{k \text{ times}}$

- $L = \{w \mid |w| = 2\}$
  $R = \Sigma\Sigma$
- $L = \{w \mid |w| \leq 2\}$
  $R = \epsilon \cup \Sigma \cup \Sigma\Sigma$
- $L = \{w \mid |w| \geq 2\}$
  $R = \Sigma\Sigma\Sigma^*$
- $L = \{w \mid n_a(w) = 2\}$
  $R = b^*ab^*ab^*$
- $L = \{w \mid n_a(w) \leq 2\}$
  $R = b^* \cup b^*ab^* \cup b^*ab^*ab^*$
- $L = \{w \mid n_a(w) \geq 2\}$
  $R = b^*ab^*ab^*(ab^*)^*$

Rules

Beware of ϕ and ϵ

Suppose R is a regular expression.

- $R \cup \phi = R$
- $R \circ \epsilon = R$
- $R \cup \epsilon$ may not equal R
  (e.g.: $R = 0$, $L(R) = \{0\}$, $L(R \cup \epsilon) = \{0, \epsilon\}$)
- $R \circ \phi$ may not equal R
  (e.g.: $R = 0$, $L(R) = \{0\}$, $L(R \circ \phi) = \phi$)

Rules

Rules

Suppose R_1, R_2, R_3 are regular expressions. Then

- $R_1\phi = \phi R_1 = \phi$
- $R_1\epsilon = \epsilon R_1 = R_1 \cup \phi = \phi \cup R_1 = R_1$
- $R_1 \cup R_1 = R_1$
- $R_1 \cup R_2 = R_2 \cup R_1$
- $R_1(R_2 \cup R_3) = R_1R_2 \cup R_1R_3$
- $(R_1 \cup R_2)R_3 = R_1R_3 \cup R_2R_3$
- $R_1(R_2R_3) = (R_1R_2)R_3$
- $\phi^* = \epsilon$
- $(\epsilon \cup R_1)^* = (\epsilon \cup R_1)^+ = R_1^*$
- $R_1^*(\epsilon \cup R_1) = (\epsilon \cup R_1)R_1^* = R_1^*$
- $R_1^*R_2 \cup R_2 = R_1^*R_2$
- $R_1(R_2R_1)^* = (R_1R_2)^*R_1$
- $(R_1 \cup R_2)^* = (R_1 * R_2)^*R_1^* = (R_2^*R_1)^*R_2^*$

Construct a regex for $\Sigma = \{a, b\}$

Problem

- Construct a regular expression to describe the language $L = \{w \mid n_a(w) \text{ is odd}\}$

Construct a regex for $\Sigma = \{a, b\}$

Problem

- Construct a regular expression to describe the language $L = \{w \mid n_a(w) \text{ is odd}\}$

Solution

- Incorrect expressions.

  $b^*ab^*(ab^*a)^*b^*$ ▷ Why?

  $b^*a(b^*ab^*ab^*)^*$ ▷ Why?

- Correct expressions.

  $b^*ab^*(b^*ab^*ab^*)^*$ ▷ Why?

  $b^*ab^*(ab^*ab^*)^*$ ▷ Why?

  $b^*a(b^*ab^*a)^*b^*$ ▷ Why?

  $b^*a(b \cup ab^*a)^*$ ▷ Why?

  $(b \cup ab^*a)^*ab^*$ ▷ Why?

Construct a regex for $\Sigma = \{a, b\}$

Problem

- Construct a regular expression to describe the language $L = \{w \mid w \text{ ends with } b \text{ and does not contain } aa\}$

Construct a regex for $\Sigma = \{a, b\}$

Problem

- Construct a regular expression to describe the language $L = \{w \mid w \text{ ends with } b \text{ and does not contain } aa\}$

Solution

- A string not containing aa means that every a in the string:
  - is immediately followed by b, or
  - is the last symbol of the string
- Each string in the language has to end with b.
- Hence, every a in each string of the language is immediately followed by b
- Regular expression is: $(b \cup ab)^+$

Construct a regex to recognize identifiers in C

Problem

- Identifiers are the names you supply for variables, types, functions, and labels.
- Construct a regular expression to recognize the identifiers in the C programming language i.e., $L = \{\text{identifiers in C}\}$

Construct a regex to recognize identifiers in C

Problem

- Identifiers are the names you supply for variables, types, functions, and labels.
- Construct a regular expression to recognize the identifiers in the C programming language i.e., $L = \{\text{identifiers in C}\}$

Solution

- C identifier = FirstLetter OtherLetters
  FirstLetter = English letter or underscore
  OtherLetters = Alphanumeric letters or underscore
- Let $L = \{a, \ldots, z, A, \ldots, Z\}$ and $D = \{0, 1, \ldots, 9\}$
- Regular expression is:
  R = FirstLetter $\circ$ OtherLetters
  FirstLetter $= (L \cup \_)$
  OtherLetters $= (L \cup D \cup \_)$

Construct a regex to recognize decimals in C

Problem

- Construct a regular expression to recognize the decimal numbers in the C programming language i.e.,
  $L = \{\text{decimal numbers in C}\}$
- Examples: $14, +1, -12, 14.3, -.99, 16., 3E14, -1.00E2, 4.1E-1$, and $.3E+2$

Construct a regex to recognize decimals in C

Problem

- Construct a regular expression to recognize the decimal numbers in the C programming language i.e.,
  $L = \{\text{decimal numbers in C}\}$
- Examples: $14, +1, -12, 14.3, -.99, 16., 3E14, -1.00E2, 4.1E-1$, and $.3E+2$

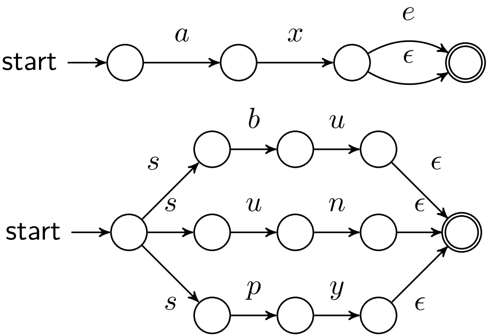
Solution

- C decimal number = Sign Decimals Exponent
- Let $D = \{0, 1, \ldots, 9\}$
- Regular expression is:
  $R =$ Sign $\circ$ Decimals $\circ$ Exponent
  Sign $= (+ \cup - \cup \epsilon)$
  Decimals $= (D^+ \cup D^+.D^* \cup D^*.D^+)$
  Exponent $= (\epsilon \cup E$ Sign $D^+)$

Nondeterministic Finite Automata (NFA)

Example NFA's

Examples



Difference	DFA	NFA
Multiple transitions	1 exiting arrow	≥ 0 exiting arrows
Epsilon transitions	✗	✓
Missing transitions	No missing transitions	Missing transitions mean transitions to sink/reject state

What is the intuition behind nondeterminism?

Intuition

Nondeterministic computation = Parallel computation
(NFA searches all possible paths in a graph to the accept state)

- When NFA has multiple choices for the same input symbol, think of it as a process forking multiple processes for parallel computation.
- A string is accepted if any of the parallel processes accepts the string.

Nondeterministic computation = Tree of possibilities
(NFA magically guesses a right path to the accept state)

- Root of the tree is the start of the computation.
- Every branching point is the decision-making point consisting of multiple choices.
- Machine accepts a string if any of the paths from the root of the tree to a leaf leads to an accept state.

Why care for NFA's?

Uses of NFA's

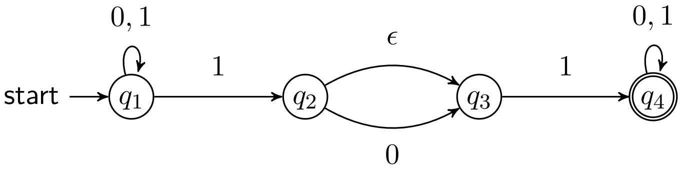
- Constructing NFA's is easier than directly constructing DFA's for many problems.
  Hence, construct NFA's and then convert them to DFA's.
- NFA's are easier to understand than DFA's.

Construct NFA for $\Sigma = \{0, 1\}$

Problem

- Construct a NFA that accepts all strings from the language $L = \{$strings containing 11 or 101$\}$

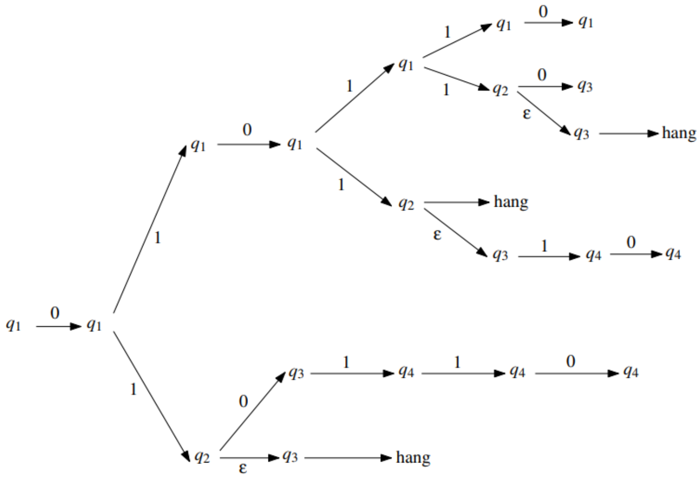
Solution



- How does the machine work for the input 010110?
- What is the equivalent DFA for solving the problem?

Construct NFA for $\Sigma = \{0, 1\}$

Solution (continued)



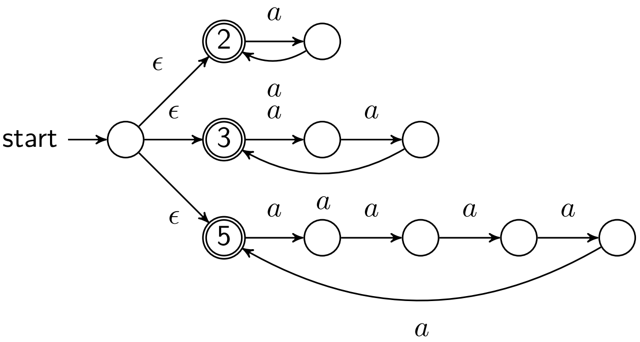
Source: Anil Maheshwari and Michiel Smid's Theory of Computation

Construct NFA for $\Sigma = \{a\}$

Problem

- Construct a NFA that accepts all strings from the language $L = \{$strings of size multiples of 2 or 3 or 5$\}$

Solution



- What is the equivalent DFA for solving the problem?

What is a nondeterministic finite automaton (NFA)?

- Nondeterministic = Event paths cannot be determined precisely
- Finite = Finite and small amount of space used
- Automaton = Computing machine

Definition

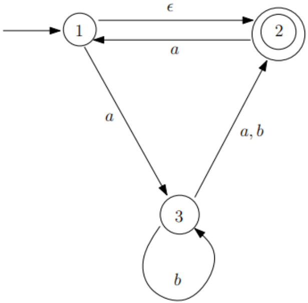
A nondeterministic finite automaton (NFA) M is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where,

1. Q: A finite set (set of states). ▷ Space (computer memory)
2. Σ: A finite set (alphabet).
3. $\boxed{\delta : Q \times (\Sigma \cup \epsilon) \to P(Q)}$ is the transition function, where $P(Q)$ is the power set of Q. ▷ Time (computation)
4. q_0: The start state (belongs to Q).
5. F: The set of accepting/final states, where $F \in Q$.

Convert NFA to DFA

Problem

- Convert the NFA to a DFA.



Source: Anil Maheshwari and Michiel Smid's Theory of Computation

Construct DFA for the given NFA

Solution

- NFA M is specified as
  Set of states is $Q = \{1, 2, 3\}$
  Set of symbols is $\Sigma = \{a, b\}$
  Start state is 1
  Set of accept states is $F = \{1\}$
  Transition function δ is:

δ	a	b	ϵ
1	$\{3\}$	ϕ	$\{2\}$
2	$\{1\}$	ϕ	ϕ
3	$\{2\}$	$\{2, 3\}$	ϕ

- How do you convert this NFA to DFA?

Construct DFA for the given NFA

Solution

- NFA M is specified as
  Set of states is $Q = \{1, 2, 3\}$
  Set of symbols is $\Sigma = \{a, b\}$
  Start state is 1
  Set of accept states is $F = \{1\}$
  Transition function δ is:

δ	a	b	ϵ
1	$\{3\}$	ϕ	$\{2\}$
2	$\{1\}$	ϕ	ϕ
3	$\{2\}$	$\{2, 3\}$	ϕ

- How do you convert this NFA to DFA?
  If NFA has states Q, then construct a DFA with states $P(Q)$.

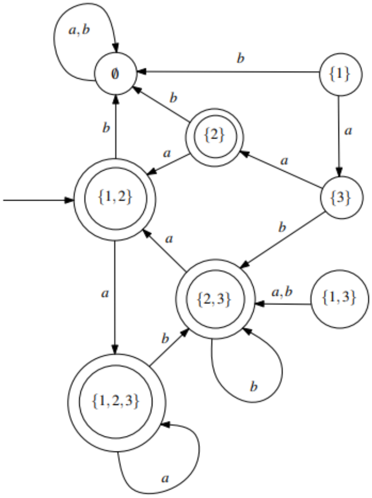
Construct DFA for the given NFA

Solution (continued)

- $\phi \xrightarrow{a} \phi$
- $\phi \xrightarrow{b} \phi$
- $\{1\} \xrightarrow{a} \{3\}$
- $\{1\} \xrightarrow{b} \phi$
- $\{2\} \xrightarrow{a} \{1,2\}$
- $\{2\} \xrightarrow{b} \phi$
- $\{3\} \xrightarrow{a} \{2\}$
- $\{3\} \xrightarrow{b} \{2,3\}$
- $\{1,2\} \xrightarrow{a}$?
- $\{1,2\} \xrightarrow{b}$?
- $\{1,3\} \xrightarrow{a}$?
- $\{1,3\} \xrightarrow{b}$?
- $\{2,3\} \xrightarrow{a}$?
- $\{2,3\} \xrightarrow{b}$?
- $\{1,2,3\} \xrightarrow{a}$?
- $\{1,2,3\} \xrightarrow{b}$?

Construct DFA for the given NFA

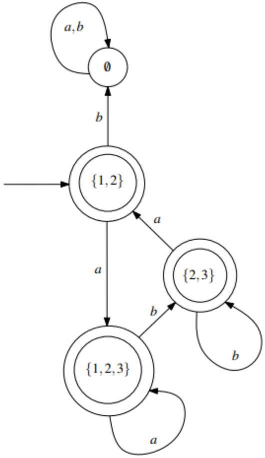
Solution (continued)



Source: Anil Maheshwari and Michiel Smid's Theory of Computation

Construct DFA for the given NFA

Solution (continued)



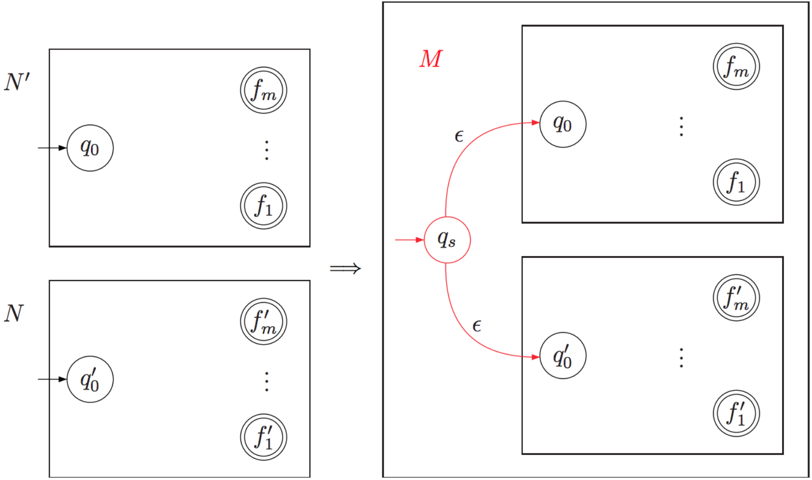
Source: Anil Maheshwari and Michiel Smid's Theory of Computation

Construct DFA for the given NFA

Convert NFA to DFA

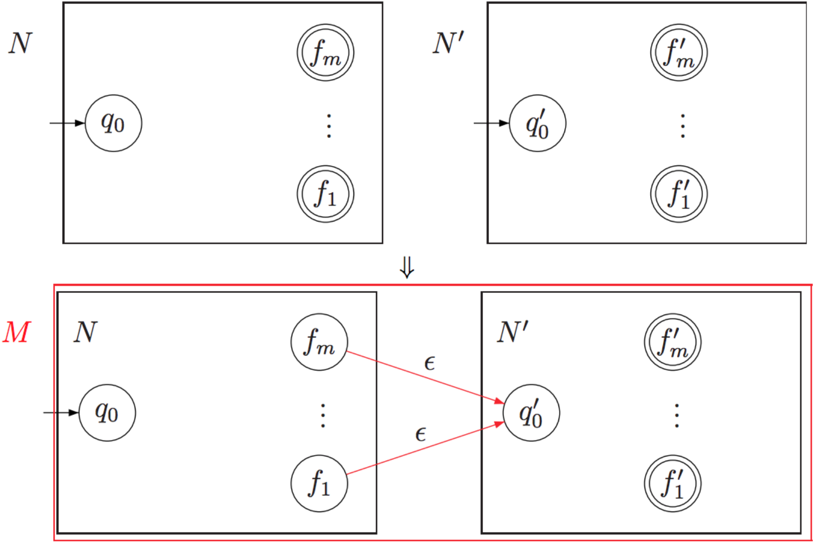
- Let $N = (Q, \Sigma, \delta, q, F)$ be the NFA.
  Let $M = (Q', \Sigma, \delta', q', F')$ be the DFA. Then
- $Q' = P(Q)$ ▷ Power set of Q
  $q' = C_\epsilon(\{q\})$ ▷ ϵ-closure of the start state
  $F' = \{S \in Q' \mid S \cap F \neq \phi\}$ ▷ $S \cap F \neq \phi$ means that S contains at least one accept state of N
  $\delta' : Q' \times \Sigma \to Q'$ is defined as follows:
  For all state $S \in Q'$ and for all letter $a \in \Sigma$,
  $\delta'(S, a) = \bigcup_{s \in S} C_\epsilon(\delta(s, a))$

Union of NFA



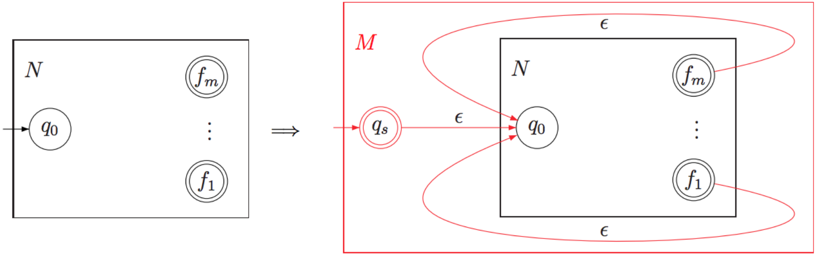
Source: Margaret Fleck and Sariel Har-Peled's Notes on Theory of Computation

Concatenation of NFA



Source: Margaret Fleck and Sariel Har-Peled's Notes on Theory of Computation

Star of NFA



Source: Margaret Fleck and Sariel Har-Peled's Notes on Theory of Computation

Construct a NFA for $(aa \cup aab)^*b$

Problem

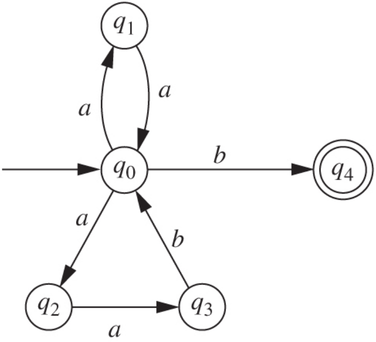
- Construct a NFA for the regular expression $(aa \cup aab)^*b$.

Construct a NFA for $(aa \cup aab)^*b$

Problem

- Construct a NFA for the regular expression $(aa \cup aab)^*b$.

Solution



Source: John Martin's Introduction to Languages and the Theory of Computation.

Construct a NFA for $(aab)^*(a \cup aba)^*$

Problem

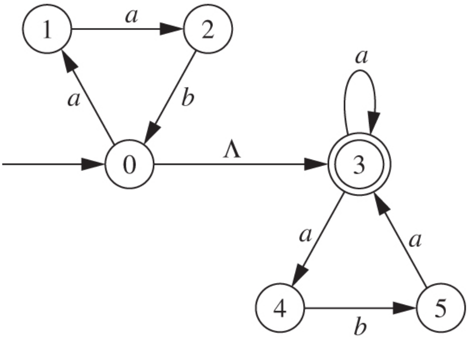
- Construct a NFA for the regular expression $(aab)^*(a \cup aba)^*$.

Construct a NFA for $(aab)^*(a \cup aba)^*$

Problem

- Construct a NFA for the regular expression $(aab)^*(a \cup aba)^*$.

Solution



Source: John Martin's Introduction to Languages and the Theory of Computation.

Non-Regular Languages

Regular or non-regular languages

Problems

Let $\Sigma = \{a, b\}$ unless mentioned otherwise. Check if the languages are regular or non-regular (✗):

- $L = \{w \mid w = a^n \text{ and } n \leq 10^{100}\}$
- $L = \{w \mid w = a^n \text{ and } n \geq 1\}$
- $L = \{w \mid w = a^m b^n \text{ and } m, n \geq 1\}$
- $L = \{w \mid w = a^* b^*\}$
- $L = \{w \mid w = a^n b^n \text{ and } n \geq 1\}$
- $L = \{ww^R \mid |w| = 3\}$
- $L = \{ww^R \mid |w| \geq 1\}$
- $L = \{w \mid w = w^R \text{ and } |w| \geq 1\}$
- $L = \{w \mid w = a^{2i+1} b^{3j+2} \text{ and } i, j \geq 1\}$
- $L = \{w \mid w = a^n \text{ and } n \text{ is a square}\}$
- $L = \{w \mid w = a^n \text{ and } n \text{ is a prime}\}$
- $L = \{w \mid w = a^i b^{j^2} \text{ and } i, j \geq 1\}$

Regular or non-regular languages

Problems

Let $\Sigma = \{a, b\}$ unless mentioned otherwise. Check if the languages are regular or non-regular (✗):

- $L = \{w \mid w = a^n$ and $n \leq 10^{100}\}$
- $L = \{w \mid w = a^n$ and $n \geq 1\}$
- $L = \{w \mid w = a^m b^n$ and $m, n \geq 1\}$
- $L = \{w \mid w = a^* b^*\}$
- $L = \{w \mid w = a^n b^n$ and $n \geq 1\}$ .... ✗
- $L = \{ww^R \mid |w| = 3\}$
- $L = \{ww^R \mid |w| \geq 1\}$ .... ✗
- $L = \{w \mid w = w^R$ and $|w| \geq 1\}$ .... ✗
- $L = \{w \mid w = a^{2i+1} b^{3j+2}$ and $i, j \geq 1\}$
- $L = \{w \mid w = a^n$ and n is a square$\}$ .... ✗
- $L = \{w \mid w = a^n$ and n is a prime$\}$ .... ✗
- $L = \{w \mid w = a^i b^{j^2}$ and $i, j \geq 1\}$ .... ✗

Regular or non-regular languages

Problems (continued)

- $L = \{w \mid n_a(w) = n_b(w)\}$
- $L = \{w \mid n_a(w) \bmod 3 \geq n_b(w) \bmod 5\}$
- $L = \{w \mid w = a^i b^j \text{ and } j > i \geq 1\}$
- $L = \{wxw^R \mid x \in \Sigma^*, |w|, |x| \geq 1, \text{ and } |x| \leq 5\}$
- $L = \{wxw^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1\}$
- $L = \{xww^R y \mid x, y \in \Sigma^* \text{ and } |w|, |x|, |y| \geq 1\}$
- $L = \{xww^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1\}$
- $L = \{ww^R y \mid y \in \Sigma^* \text{ and } |w|, |y| \geq 1\}$

Regular or non-regular languages

Problems (continued)

- $L = \{w \mid n_a(w) = n_b(w)\}$ .... ✗
- $L = \{w \mid n_a(w) \bmod 3 \geq n_b(w) \bmod 5\}$
- $L = \{w \mid w = a^i b^j \text{ and } j > i \geq 1\}$ .... ✗
- $L = \{wxw^R \mid x \in \Sigma^*, |w|, |x| \geq 1, \text{ and } |x| \leq 5\}$ .... ✗
- $L = \{wxw^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1\}$
- $L = \{xww^Ry \mid x, y \in \Sigma^* \text{ and } |w|, |x|, |y| \geq 1\}$
- $L = \{xww^R \mid x \in \Sigma^* \text{ and } |w|, |x| \geq 1\}$ .... ✗
- $L = \{ww^Ry \mid y \in \Sigma^* \text{ and } |w|, |y| \geq 1\}$ .... ✗

How to prove that certain languages are not regular?

Pumping lemma

- Many languages are not regular.
- Pumping lemma is a method to prove that certain languages are not regular.

Pumping property

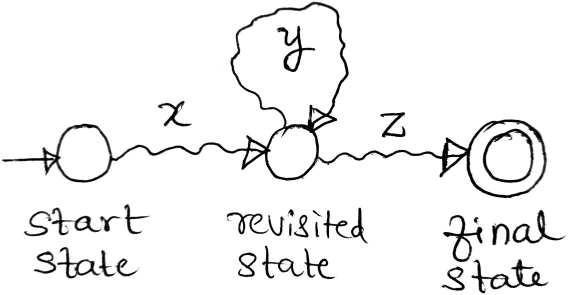
- If a language is regular, then it must have the pumping property.
- If a language does not have the pumping property, then the language is not regular. ▷ Proof by contraposition

How to prove languages non-regular using pumping lemma?

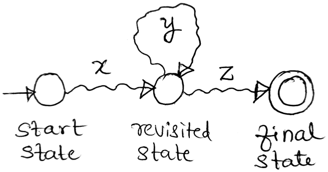
- Proof by contradiction.
  Assume that the language is regular.
  Show that the language does not have the pumping property.
  Contradiction! Hence, the language has to be non-regular.

Pumping property of regular languages

- Suppose a DFA M with s number of states accepts a very long string w such that $|w| \geq s$ from a language L.
- From pigeonhole principle, at least one state is visited twice.
- This implies that the string went through a loop.



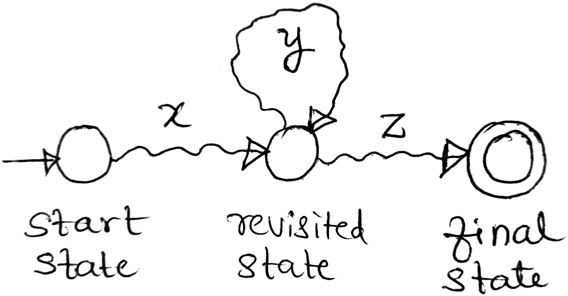
Pumping property of regular languages



Observations

- Suppose string w has more characters than the number of states in the DFA, i.e., $|w| \geq s$
- String w can be split into three parts i.e., $w = xyz$ where
  x: string before the first loop
  y: string of the first loop
  z: string after the first loop (might contain loops)
- Loop must appear i.e., $|y| \geq 1$
  (x and z can be empty)
- Loop must appear in the first s characters of w i.e., $|xy| \leq s$

Pumping property of regular languages



Idea

- An infinite number of strings can be pumped with loop length and they must also be in the language.
- Formally, for all $i \geq 0$, xy^iz must be in the language.
- xz, xyz, $xyyz$, $xyyyz$, etc must also belong to the language.

Pumping lemma for regular languages

Theorem

Suppose L is a regular language over alphabet Σ. Suppose L is accepted by a finite automaton M having s states. Then, every long string $w \in L$ satisfying $|w| \geq s$ can be split into three strings $w = xyz$ such that the following three conditions are true.

- $|xy| \leq s$.
- $|y| \geq 1$.
- For every $i \geq 0$, the string xy^iz also belongs to L.

$L = \{a^n b^n \mid n \geq 0\}$ is non-regular

Problem

- Prove that $L = \{a^n b^n \mid n \geq 0\}$ is not a regular language.

$L = \{a^n b^n \mid n \geq 0\}$ is non-regular

Problem

- Prove that $L = \{a^n b^n \mid n \geq 0\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = a^s b^s$.
- Let $w = xyz =$ | a^p | a^q | $a^r b^s$ |
  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, $xy^i z$ must belong to L for all $i \geq 0$.
- But, $xyyz$ is not in L.
  Reason: $xyyz = a^p a^q a^q a^r b^s = a^{s+q} b^s \notin L$.
  $xyyz$ has more a's than b's.
- Contradiction! Hence, L is not regular.

$L = \{w \mid n_a(w) = n_b(w)\}$ is non-regular

Problem

- Prove that $L = \{w \mid n_a(w) = n_b(w)\}$ is not a regular language.

$L = \{w \mid n_a(w) = n_b(w)\}$ is non-regular

Problem

- Prove that $L = \{w \mid n_a(w) = n_b(w)\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = (ab)^s$.
- Let $w = xyz =$ | ϵ | $(ab)^1$ | $(ab)^{s-1}$ |
- We have $|xy| \le s$ and $|y| \ge 1$.
- Also, xy^iz must belong to L for all $i \ge 0$.
- xy^iz belongs to L for all $i \ge 0$.
- No contradiction! Hence, L is regular.

$L = \{w \mid n_a(w) = n_b(w)\}$ is non-regular

Problem

- Prove that $L = \{w \mid n_a(w) = n_b(w)\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = (ab)^s$.
- Let $w = xyz = $ | ϵ | $(ab)^1$ | $(ab)^{s-1}$ |
- We have $|xy| \leq s$ and $|y| \geq 1$.
- Also, xy^iz must belong to L for all $i \geq 0$.
- xy^iz belongs to L for all $i \geq 0$.
- No contradiction! Hence, L is regular.

Mistakes

Incorrect solution! Why? Multiple reasons:

1. If we cannot find a contradiction, that does not prove anything.
2. We must try for all possible values of x, y such that $|xy| \leq s$.
3. The chosen string $(ab)^s$ is a bad string to work on.

$L = \{w \mid n_a(w) = n_b(w)\}$ is non-regular

Problem

- Prove that $L = \{w \mid n_a(w) = n_b(w)\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = a^s b^s$.
- Let $w = xyz = $ | a^p | a^q | $a^r b^s$ |
  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, $xy^i z$ must belong to L for all $i \geq 0$.
- But, $xyyz$ is not in L.
  Reason: $xyyz = a^p a^q a^q a^r b^s = a^{s+q} b^s \notin L$.
  $xyyz$ has more a's than b's.
- Contradiction! Hence, L is not regular.

$L = \{w \mid n_a(w) = n_b(w)\}$ is non-regular

Problem

- Prove that $L = \{w \mid n_a(w) = n_b(w)\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = a^s b^s$.
- Let $w = xyz = $ | a^p | a^q | $a^r b^s$ |
  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, xy^iz must belong to L for all $i \geq 0$.
- But, $xyyz$ is not in L.
  Reason: $xyyz = a^p a^q a^q a^r b^s = a^{s+q} b^s \notin L$.
  $xyyz$ has more a's than b's.
- Contradiction! Hence, L is not regular.

Takeaway

1. Reduction! Reduce a problem to another. Reuse its solution.

Superset of a non-regular language

Problem

- $\{a^n b^n\}$ is a subset of $\{w \mid n_a(w) = n_b(w)\}$.
  We used the fact that $\{a^n b^n\}$ is non-regular to prove that $\{w \mid n_a(w) = n_b(w)\}$ is non-regular.
  Is a superset of a non-regular language non-regular?

Superset of a non-regular language

Problem

- $\{a^n b^n\}$ is a subset of $\{w \mid n_a(w) = n_b(w)\}$.
  We used the fact that $\{a^n b^n\}$ is non-regular to prove that $\{w \mid n_a(w) = n_b(w)\}$ is non-regular.
  Is a superset of a non-regular language non-regular?

Solution

- No!
  Σ^* is a superset of every non-regular language.
  But, it is regular.

$L = \{w \mid n_a(w) = n_b(w)\}$ is non-regular

Problem

- Prove that $L = \{w \mid n_a(w) = n_b(w)\}$ is not a regular language.

Solution (without using pumping lemma)

- Suppose L is regular.
- We know that $L' = \{w \mid w = a^i b^j, i, j \geq 0\}$ is regular.
- As regular languages are closed under intersection, $L \cap L'$ must also be regular.
- We see that $L \cap L' = \{w \mid w = a^n b^n \text{ and } n \geq 0\}$.
- But, this language was earlier proved to be non-regular.
- Contradiction! Hence, L is not regular.

$L = \{ww\}$ is non-regular

Problem

- Prove that $L = \{ww\}$ is not a regular language.

$L = \{ww\}$ is non-regular

Problem

- Prove that $L = \{ww\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $ww = a^s a^s$.
- Let $ww = xyz =$

a^p	a^1	$a^{s-p-1}a^s$

- We have $|xy| \leq s$ and $|y| \geq 1$.
- Also, xy^iz must belong to L for all $i \geq 0$.
- But, $xyyz$ is not in L.
  Reason: $xyyz = a^p a^1 a^1 a^{s-p-1} a^p = a^{s+1} a^s \notin L$.
  $xyyz$ has odd number of a's.
- Contradiction! Hence, L is not regular.

$L = \{ww\}$ is non-regular

Problem

- Prove that $L = \{ww\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $ww = a^s a^s$.
- Let $ww = xyz =$ | a^p | a^1 | $a^{s-p-1}a^s$ |
- We have $|xy| \leq s$ and $|y| \geq 1$.
- Also, xy^iz must belong to L for all $i \geq 0$.
- But, $xyyz$ is not in L.
  Reason: $xyyz = a^p a^1 a^1 a^{s-p-1} a^p = a^{s+1} a^s \notin L$.
  $xyyz$ has odd number of a's.
- Contradiction! Hence, L is not regular.

Mistakes

Incorrect solution! Why?

1. We must try all possible values of x, y such that $|xy| \leq s$.
2. Try pumping with $y \in \{a^2, a^4, \ldots\}$ such that $|y| \leq s$.

$L = \{ww\}$ is non-regular

Problem

- Prove that $L = \{ww\}$ is not a regular language.

$L = \{ww\}$ is non-regular

Problem

- Prove that $L = \{ww\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $ww = a^s b^s a^s b^s$.
- Let $ww = xyz =$ | a^p | a^q | $a^r b^s a^s b^s$ |
  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, $xy^i z$ must belong to L for all $i \geq 0$.
- But, $xyyz$ is not in L.
  Reason: $xyyz = a^p a^q a^q a^r b^s a^s b^s = a^{s+q} b^s a^s b^s \notin L$.
- Contradiction! Hence, L is not regular.

$L = \{w \mid w = a^n, n$ is a square$\}$ is non-regular

Problem

- Prove that $L = \{w \mid w = a^{n^2}\}$ is not a regular language.

$L = \{w \mid w = a^n, n$ is a square$\}$ is non-regular

Problem

- Prove that $L = \{w \mid w = a^{n^2}\}$ is not a regular language.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = a^{s^2}$.
- Let $w = xyz =$

a^p	a^q	$a^r a^{s^2-s}$

  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, xy^iz must belong to L for all $i \geq 0$.
- But, $xyyz$ is not in L.
  Reason: $xyyz = a^p a^q a^q a^r a^{s^2-s} = a^{s^2+q} \notin L$.
  Because, $s^2 < s^2 + q < (s+1)^2$.
- Contradiction! Hence, L is not regular.

$L = \{w \mid w = a^n, n$ is prime$\}$ is non-regular

Problem

- Prove that $L = \{w \mid w = a^n, n \text{ is prime}\}$ is not regular.

$L = \{w \mid w = a^n, n$ **is prime**$\}$ **is non-regular**

Problem

- Prove that $L = \{w \mid w = a^n, n \text{ is prime}\}$ is not regular.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = a^m$, where m is prime and $m \geq s$.
- Let $w = xyz =$ | a^p | a^q | a^r |
  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = m$.
- Also, xy^iz must belong to L for all $i \geq 0$.
- But, $xy^{m+1}z$ is not in L.
  Reason: $xy^{m+1}z = a^p a^{q(m+1)} a^r = a^{m(q+1)} \notin L$.
- Contradiction! Hence, L is not regular.

$L = \{w \mid w = a^m b^n, m > n\}$ is non-regular

Problem

- Prove that $L = \{w \mid w = a^m b^n, m > n\}$ is not regular.

$L = \{w \mid w = a^m b^n, m > n\}$ is non-regular

Problem

- Prove that $L = \{w \mid w = a^m b^n, m > n\}$ is not regular.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = a^{s+1} b^s$.
- Let $w = xyz =$ | a^p | a^q | $a^r b^s$ |
  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s + 1$.
- Also, $xy^i z$ must belong to L for all $i \geq 0$.
- But, xz is not in L. ▷ Pumping down
  Reason: $xz = a^p a^r b^s = a^{p+r} b^s \notin L$.
  Because, $p + r \leq s$ i.e., $\#a$'s is not greater than $\#b$'s.
- Contradiction! Hence, L is not regular.

$L = \{w \mid w = a^m b^n, m \neq n\}$ is non-regular

Problem

- Prove that $L = \{w \mid w = a^m b^n, m \neq n\}$ is not regular.

$L = \{w \mid w = a^m b^n, m \neq n\}$ is non-regular

Problem

- Prove that $L = \{w \mid w = a^m b^n, m \neq n\}$ is not regular.

Solution

- Suppose L is regular. Then it must satisfy pumping property.
- Suppose $w = a^s b^{s+s!}$.
- Let $w = xyz =$ | a^p | a^q | $a^r b^{s+s!}$ |
  where $|xy| \leq s$, $|y| \geq 1$, and $p + q + r = s$.
- Also, $xy^i z$ must belong to L for all $i \geq 0$.
- But, $xy^i z$ is not in L for some i.
  We pump a^q to get $a^{s+s!} b^{s+s!}$.
  Reason: $xy^i z = a^p a^{qi} a^r b^{s+s!} = a^{s+(i-1)q} b^{s+s!} \notin L$.
  This means $(i-1)q = s! \implies i = s!/q + 1$.
- Contradiction! Hence, L is not regular.

$L = \{w \mid w = a^m b^n, m \neq n\}$ **is non-regular**

Problem

- Prove that $L = \{w \mid w = a^m b^n, m \neq n\}$ is not regular.

Solution (without using pumping lemma)

- Suppose L is regular.
- We know that $L' = \{w \mid w = a^i b^j, i, j \geq 0\}$ is regular.
- Let $L'' = \{w \mid w = a^n b^n, n \geq 0\}$.
- As regular languages are closed under intersection and complementation, $L = L' - L'' = L' \cap \bar{L}''$ is regular.
  This implies that L'' is regular.
- But, the language L'' was earlier proved to be non-regular.
- Contradiction! Hence, L is not regular.