COMPSCI/SFWRENG 2FA3

Discrete Mathematics with Applications II

Winter 2020

4 Finite Automata and Regular Expressions

William M. Farmer

Department of Computing and Software
McMaster University

March 8, 2020

Problem Solving (iClicker)

What is the best way to learn how to solve problems?

A. Ask other people to solve them for you.
B. Memorize solutions to problems.
C. Study theory relevant to the problems.
D. Solve problems for which you have solutions.
E. $\boxed{\text{Solve problems for which you do not have solutions.}}$

Admin — February 11

- Midterm 1.
  - Marks and solutions will be posted later this week.
- Bio sheets.
  - I have enjoyed reading your bio sheets.
  - Many of you said that you don't do much reading.
  - Can submit your bio sheet until Apr. 1 for 0.5 bonus pts.
- M&Ms.
  - M&Ms have been very useful to me; I hope they have been useful to you.
  - Common request: More examples in the lectures.
  - Assignments are meant to be exercises you haven't seen.
  - If you have questions about M&M marks, please contact Kumail at `naqvis8@mcmaster.ca`.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Assignment 4

Question 1. Construct in MSFOL a theory T of strict total orders that are dense and have minimum and maximum elements. Give two models for T.

Question 2. Construct in MSFOL a theory $T = (\Sigma_{\text{queue}}, \Gamma_{\text{queue}})$ of queues.

Looking Back

- We have covered 3 topics:
  1. Mathematical proofs
  2. Recursion and induction.
  3. Predicate logic.
- You have completed 3 assignments.
  - Written traditional proofs.
  - Used LaTeX.
- You did Midterm Test 1

Looking Forward

- We have 3 remaining topics to cover:
  1. Finite automata and regular expressions.
  2. Push-down automata and context-free languages.
  3. Turing machines and computability.
- You have 8 more assignments to do.
- There will be a midterm review in early March.
- Midterm Test 2 will be on March 11.
- The final exam will cover the entire course.

Outline

- Theory of computation.
- String operations.
- Decision problems.
- Deterministic finite automata (DFAs).
- Nondeterministic finite automata (NFAs).
- Regular expressions.
- Applications and other topics.

1. Theory of Computation

What is Theory of Computation?

- Theory of computation is the study of the foundations of computation.
- It is concerned with the following questions:
  1. What does it mean for a function to be computable?
  2. What can and cannot be computed?
  3. How does computational power depend on computational mechanisms?
  4. How do we classify computable functions?
- Various kinds of models of computation are used to study the nature of computation.
  - Examples: Automata and grammars.

Automata

- An automaton is an abstract machine that performs computations.
- We are interested in three categories of automata:
  1. Finite automata with finite memory.
  2. Push-down automata with finite memory and a stack.
  3. Turing machines with unlimited memory.

Grammars

- A grammar is a set of rules for generating the expressions in a language.
- We are interested in three categories of grammars:
  1. Regular grammars that generate regular languages.
  2. Context-free grammars that generate context-free languages.
  4. Unrestricted grammars that generate recursively enumerable languages.
- These grammars are three of the four types of grammars in the Chomsky hierarchy. The missing grammar type is:
  3. Context-sensitive grammars that generate context-sensitive languages.
- As models of computation, the three kinds of automata above are equivalent to the three kinds of grammar here.

2. String Operations

Strings

- An alphabet is a finite set Σ of symbols.
- A string over Σ is a finite sequence of the symbols in Σ.
  - The set of all strings over Σ is denoted by Σ^*.
- The empty string, denoted by ϵ, is the empty sequence.
  - $\epsilon \in \Sigma^*$ for all alphabets Σ.
  - $\emptyset^* = \{\epsilon\}$.
- A string $\langle a_0, a_2, \ldots, a_n \rangle$ is written as $a_0 a_1 \cdots a_n$ or "$a_0 a_1 \cdots a_n$".

Operations on Strings

- Concatenation:
  $\langle a_0, a_1, \ldots, a_m \rangle \langle b_0, b_1, \ldots, b_n \rangle = a_0 a_1 \cdots a_m b_0 b_1 \cdots b_n$.
- Length:
  $$|x| = \begin{cases} 0 & \text{if } x = \epsilon \\ n+1 & \text{if } x = a_0, a_1, \ldots, a_n \text{ with } n \geq 0 \end{cases}$$
- Repetition:
  $$(x)^n = \begin{cases} \epsilon & \text{if } n = 0 \\ xx\cdots x \ (n \text{ times}) & \text{if } n \geq 1 \end{cases}$$

Operations on Sets of Strings

- Let $A, B \subseteq \Sigma^*$.
- The usual set-theoretic operations: union $(A \cup B)$, intersection $(A \cap B)$, and complement $(\sim A)$.
- Concatenation: $AB = \{xy \mid x \in A \text{ and } y \in B\}$.
  - Notice that $A\emptyset = \emptyset A = \emptyset$.
- Power:
  $$A^n = \begin{cases} \{\epsilon\} & \text{if } n = 0 \\ AA^{n-1} & \text{if } n \geq 1 \end{cases}$$
- Asterate: $A^* = \bigcup_{n \geq 0} A^n = A^0 \cup A^1 \cup A^2 \cup \cdots$.
  - Also called the Kleene star or Kleene closure.
- Positive asterate: $A^+ = \bigcup_{n \geq 1} A^n = A^1 \cup A^2 \cup \cdots$.
  - $A^+ = A^* \setminus \{\epsilon\}$.

Monoids (iClicker)

Which of the following mathematical structures is not a monoid?

A. $(\Sigma^*, \epsilon, \text{string-concatenation})$.
B. $(\mathcal{P}(\Sigma^*), \{\epsilon\}, \text{set-concatenation})$.
C. $(\mathcal{P}(\Sigma^*), \emptyset, \cup)$.
D. $(\mathcal{P}(\Sigma^*), \Sigma^*, \cap)$.
E. $\boxed{\text{None of the above.}}$

$\mathcal{P}(S)$, the power set of S, is the set of all subsets of S.

3. Decision Problems

Decision Problems

- A decision problem is a problem to determine the answer to a yes-or-no question about a given input.
  - For example, "Given a Σ-formula A, is A closed?" is a decision problem.
  - A decision problem can be identified with a function from the inputs to yes or no, true or false, 1 or 0, etc.
  - Many problems can be formulated as decision problems.
- A solution of a decision problem is an algorithm that, for each input, returns as output a "yes" or "no" that correctly answers the question.
- A solution to a decision problem is thus a computable function.

Decidability

- A decision problem is decidable if there exists a computable function that solves it.
- Gottfried Leibniz (1646–1716) postulated:
  1. The characteristica universalis, a universal language in which all scientific ideas could be expressed.
  2. The calculus ratiocinator, a computer that could compute the truth or falsity of statements expressed in the characteristica universalis.
- Alonzo Church (1903–95) and Alan Turing (1912–54) showed independently in 1936 that there are undecidable decision problems!
  - This shows that Leibniz's grand decision problem "Given a scientific statement S, is S true?" is undecidable!
- Examples of undecidable decision problems are the Entscheidungsproblem and the halting problem.

Decision Problems formalized as Strings

- A decision problem can often be formalized as the decision problem of whether a string is the member of a particular set $S \subseteq \Sigma^*$ for some alphabet Σ.
- A solution to the decision problem is then a computable function $f_S : \Sigma^* \to \{\text{yes}, \text{no}\}$ such that, for all $x \in \Sigma^*$,

  $$f_S(x) = \text{yes iff } x \in S.$$
- Automata solve decision problems of this kind.

Example: Theories

- Let T be a theory.
- Let Σ be the variable symbols, logical constant symbols, nonlogical constant symbols, and punctuation symbols used in T.
- Each formula of T is represented by a string in Σ^*.
- Let $S \subseteq \Sigma^*$ be the set of strings in S that represent formulas A of T such that $T \models A$.
- Thus the

  decision problem of whether a formula is valid in T

  is formalized as the

  decision problem of whether a string is a member of S.

Lecture Participation (iClicker)

How often do you attend the lectures?

A. I attend nearly all the lectures.
B. I attend more than half of the lectures.
C. I attend less than half of the lectures.
D. I rarely attend the lectures.

Discussion Session Participation (iClicker)

How often do you attend the discussion sessions?

A. I attend nearly all the discussion sessions.
B. I attend more than half of the discussion sessions.
C. I attend less than half of the discussion sessions.
D. I rarely attend the discussion sessions.

Tutorial Participation (iClicker)

How often do you attend the tutorials?

A. I attend nearly all the tutorials.
B. I attend more than half of the tutorials.
C. I attend less than half of the tutorials.
D. I rarely attend the tutorials.

Exercise Participation (iClicker)

How many exercises do you do?

A. I do all the exercises.

B. I do nearly all the exercises.

C. I do about half of the exercises.

D. I do very few of exercises.

Admin — February 25

- Midterm Test 1.
  - Stage 1 average: 71.3%.
  - State 2 average: 86.3%.
  - Problems with scan sheets.
  - Penalties for incorrect student numbers and version numbers and incomplete erasures will be imposed for Midterm Test 2.
- finsm system.
  - For creating and simulating DFAs and NFAs.
  - Developed at Mac by Chris Schankula and Lucas Dutton.
  - Web interface at https://finsm.io.
  - Will be demonstrated in the tutorials.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Assignment 5

Question 1. Construct a deterministic finite automaton M for the alphabet $\Sigma = \{a\}$ such that $L(M)$ is the set of all strings in Σ^* whose length is divisible by either 2 or 5. Present M as a transition diagram.

Question 2. Construct a deterministic finite automaton M for the alphabet $\Sigma = \{0, 1\}$ such that $L(M)$ is the set of all strings x in Σ^* for which $\#0(x)$ is divisible by 2 and $\#1(x)$ is divisible by 3. Present M as a transition diagram.

Review

- Theory of computation.
- String operations.
- Decision problems.
- Deterministic finite automata (DFAs).
- Nondeterministic finite automata (NFAs).

4. Deterministic Finite Automata

Finite-State Transition Systems

- A finite-state transition system is a system model such that:
  - The system is always in one of finitely many states.
  - In response to external inputs, the system instantaneously changes state by one of finitely many state transitions.
- Many physical systems are engineered to behave like finite-state transition systems.
  - Example: A modern computer.
- Finite-state transition systems are themselves modeled by finite automata.

Deterministic Finite Automata [1/2]

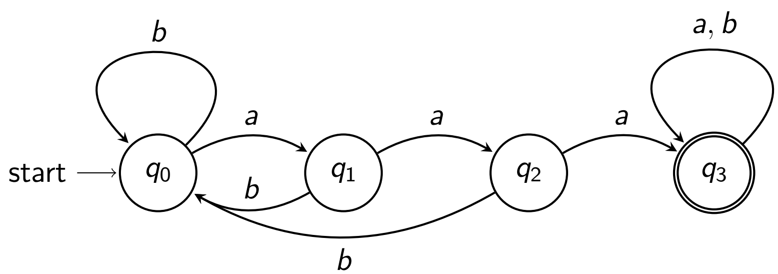
- A deterministic finite automaton (DFA) is a tuple $M = (Q, \Sigma, \delta, s, F)$ where:
  1. Q is a finite set of elements called states.
  2. Σ is a finite set of symbols called the input alphabet.
  3. $\delta : Q \times \Sigma \to Q$ is the transition function.
  4. $s \in Q$ is the start state.
  5. $F \subseteq Q$ is the set of final states.
- The function $\hat{\hat{\delta}} : Q \times \Sigma^* \to Q$ defined recursively by
  1. $\hat{\hat{\delta}}(q, \epsilon) = q$ where $q \in Q$ and
  2. $\hat{\hat{\delta}}(q, xa) = \delta(\hat{\hat{\delta}}(q, x), a)$ where $q \in Q$, $x \in \Sigma^*$, and $a \in \Sigma$

  extends δ to strings over Σ.
- M can be described by either a transition table or transition diagram.

DFA Example 1: Transition Table

	$Q \backslash \Sigma$	a	b
start $\to$	q_0	q_1	q_0
	q_1	q_2	q_0
	q_2	q_3	q_0
final $\to$	q_3	q_3	q_3

DFA Example 1: Transition Diagram



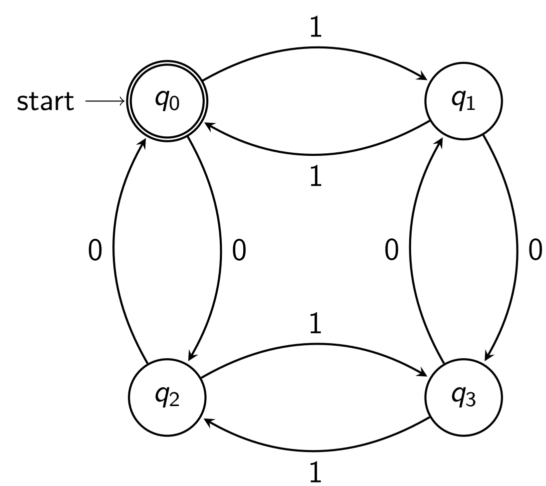
Deterministic Finite Automata [2/2]

- A string $x \in \Sigma^*$ is accepted by M if $\hat{\delta}(s, x) \in F$ and is rejected by M if $\hat{\delta}(s, x) \notin F$.
- The set or language accepted by M, written $L(M)$, is the set of all stings accepted by M. That is,
  $$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(s, x) \in F\}.$$
- $A \subseteq \Sigma^*$ is a regular set or regular language if $A = L(M)$ for some DFA M.
- Examples:
  1. $L(M_1) = \{x \in \{a, b\}^* \mid aaa \text{ is a substring of } x\}$ where M_1 is the DFA presented in Example 1.
  2. $L(M_2) = \{x \in \{0, 1\}^* \mid \#0(x) \equiv \#1(x) \equiv 0 \text{ mod } 2\}$ where M_2 is the DFA presented in Example 2 below.
- Two DFAs are equivalent if they accept the same language.

DFA Example 2: Transition Table

	$Q \setminus \Sigma$	0	1
start, final $\rightarrow$	q_0	q_2	q_1
	q_1	q_3	q_0
	q_2	q_0	q_3
	q_3	q_1	q_2

DFA Example 2: Transition Diagram



5. Nondeterministic Finite Automata

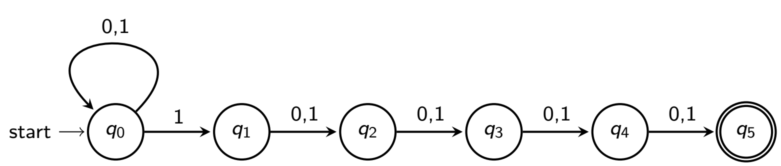
Nondeterministic Finite Automata [1/2]

- A nondeterministic finite automaton (NFA) is a tuple $N = (Q, \Sigma, \Delta, S, F)$ where:
  1. Q is a finite set of elements called states.
  2. Σ is finite set of symbols called the input alphabet.
  3. $\Delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function.
  4. $S \subseteq Q$ is the set of start states.
  5. $F \subseteq Q$ is the set of final states.
- The function $\hat{\Delta} : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$ defined recursively by
  1. $\hat{\Delta}(A, \epsilon) = A$ where $A \in \mathcal{P}(Q)$ and
  2. $\hat{\Delta}(A, xa) = \bigcup_{q \in \hat{\Delta}(A,x)} \Delta(q, a)$ where $A \in \mathcal{P}(Q)$, $x \in \Sigma^*$, and $a \in \Sigma$

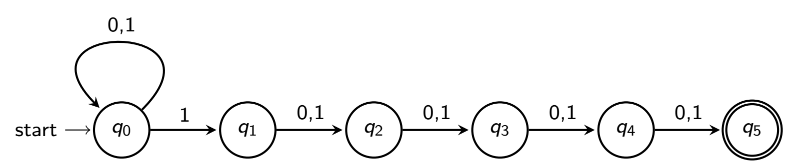
  extends Δ to strings over Σ.
- NFAs were introduced in 1959 by Michael Rabin (1931–) and Dana Scott (1932–).

NFA Example 1 : Transition Diagram

- Let $\Sigma = \{0, 1\}$ and $L = \{x \in \Sigma^* \mid \text{the fifth symbol from the right in } x \text{ is } 1\}$.
- The following NFA accepts L:



States in an NFA (iClicker)



Which of the states in this NFA are different than states in an DFA?

A. q_0.
B. q_5.
C. $\boxed{q_0 \text{ and } q_5.}$
D. All the states are different.

NFA Example 1: Transition Table

	$Q \backslash \Sigma$	0	1
start $\rightarrow$	q_0	$\{q_0\}$	$\{q_0, q_1\}$
	q_1	$\{q_2\}$	$\{q_2\}$
	q_2	$\{q_3\}$	$\{q_3\}$
	q_3	$\{q_4\}$	$\{q_4\}$
	q_4	$\{q_5\}$	$\{q_5\}$
final $\rightarrow$	q_5	$\emptyset$	$\emptyset$

Rejection (iClicker)

Which of the following statements is false?

A. A DFA must process an entire string to reject it.
B. $\boxed{\text{An NFA must process an entire string to reject it.}}$
C. The empty string is rejected by a DFA or NFA (without ϵ-transitions) iff the start state is not a final state.
D. Every string is rejected by a DFA or NFA if all the final states are inaccessible.

Nondeterministic Finite Automata [2/2]

- A string $x \in \Sigma^*$ is accepted by N if $\hat{\Delta}(S, x) \cap F \neq \emptyset$ and is rejected by N if $\hat{\Delta}(S, x) \cap F = \emptyset$.
- The set or language accepted by N, written $L(N)$, is the set of all stings accepted by N.
- A DFA and an NFA are equivalent if they accept the same language.
- Proposition 1. If a DFA $M = (Q, \Sigma, \delta, s, F)$ accepts a language L, then the NFA $N = (Q, \Sigma, \Delta, \{s\}, F)$ where $\Delta(q, a) = \{\delta(q, a)\}$ also accepts L.
- Theorem 1. If an NFA accepts a language L, then there is a DFA that also accepts L.

  Proof. Use the subset construction to produce the DFA.
- Corollary 1. DFAs and NFAs accept the same class of languages — the class of regular languages.

Equivalence of DFAs and NFAs (iClicker)

What can we say about a DFA and a NFA that are equivalent?

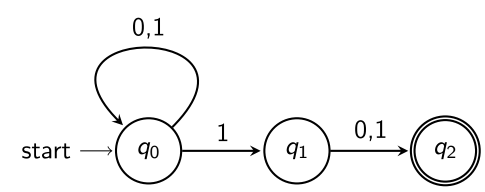
A. $\boxed{\text{They accept the same language.}}$
B. They have roughly the same number of states.
C. The ease of construction is about the same for both of them.
D. The ease of verifying that a string is accepted is about the same for both of them.

Subset Construction

- Let $N = (Q_N, \Sigma, \Delta_N, S_N, F_N)$ be an NFA. Using the subset construction, we can construct a DFA M that is equivalent to N.
- Main idea: Each state of M is a set of states of N.
  - M may have as many as 2^n states when N has n states.
- By the subset construction, $M = (Q_M, \Sigma, \delta_M, s_M, F_M)$ where:
  1. $Q_M = \mathcal{P}(Q_N)$.
  2. $\delta_M(A, a) = \hat{\Delta}_N(A, a)$ for $A \subseteq Q_N$ and $a \in \Sigma$.
  3. $s_M = S_n$.
  4. $F_M = \{A \subseteq Q_N \mid A \cap F_N \neq \emptyset\}$.
- Lemma 1. For all $A \subseteq Q_N$ and $x \in \Sigma^*$,
  $\hat{\delta}_M(A, x) = \hat{\Delta}_N(A, x)$.
- Theorem 2. N and M are equivalent.

Subset Construction Example [1/3]

- Let $\Sigma = \{0, 1\}$ and $L = \{x \in \Sigma^* \mid$ the second symbol from the right in x is $1\}$.
- The following NFA N accepts L:



Subset Construction Example [2/3]

The transition table for the NFA N is:

	$Q \backslash \Sigma$	0	1
start →	q_0	$\{q_0\}$	$\{q_0, q_1\}$
	q_1	$\{q_2\}$	$\{q_2\}$
final →	q_2	$\emptyset$	$\emptyset$

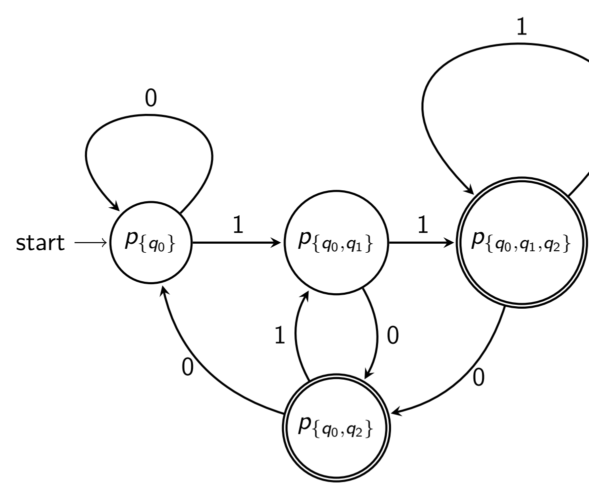
The transition table for an equivalent DFA M is:

	$\mathcal{P}(Q) \backslash \Sigma$	0	1
	$\emptyset$	$\emptyset$	$\emptyset$
start →	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
	$\{q_1\}$	$\{q_2\}$	$\{q_2\}$
final →	$\{q_2\}$	$\emptyset$	$\emptyset$
	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
final →	$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
final →	$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
final →	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

The green states are inaccessible and can be removed.

Subset Construction Example [3/3]

The transition diagram for the DFA M is:



Admin — February 26

- Midterm course review.
  - Survey on Avenue.
    - Open until 11:59 on Tuesday, March 10.
  - Discussion sessions with instructor.
    - Four sessions next week by invitation.
    - 1.0 percentage point bonus for attending a session.
- Lecturing style: Blackboard vs. slides.
- Engineering Graduate Studies Coffee House.
  - Thursday, Feb. 27, at 5:30-7:00 PM in the JHE Lobby.
  - Interested students can register at https://www.eng.mcmaster.ca/events/engineering-grad-studies-2020-coffee-house-fair.
- Office hours: To see me please send me a note with times.
- Are there any questions?

ϵ-Transitions

- An ϵ-transition is special NFA state transition labeled with ϵ,
  $$p \xrightarrow{\epsilon} q,$$
  that can take place without reading an input symbol.
- ϵ-transitions are convenient but do not widen the set of languages that can be accepted by NFAs.
  - ϵ-transitions are especially convenient for building NFAs out of smaller NFAs.
- Theorem 3. Let N be an NFA with ϵ-transitions that accepts a language L. Then there is an NFA N' without ϵ-transitions that also accepts L.

  Proof. Let $N = (Q, \Sigma, \Delta, S, F)$. Define $N' = (Q, \Sigma, \Delta', E(S), F)$ where $E(A)$ is the "ϵ-closure" of $A \subseteq Q$ and $\Delta'(q, a) = E(\Delta(q, e))$ for all $q \in Q$ and $a \in \Sigma$. Then $L(N) = L(N')$.

6. Regular Expressions

Regular Expressions

- Let Σ be a finite alphabet.
- A regular expression over Σ is defined inductively by:
  1. $\emptyset$ is a regular expression over Σ.
  2. ϵ is a regular expression over Σ.
  3. a is regular expression over Σ for each $a \in \Sigma$.
  4. If α and β are regular expressions over Σ, then $(\alpha + \beta)$, $(\alpha\beta)$, and (α^*) are regular expressions over Σ.
- We will omit parentheses by assuming that $*$ has a higher precedence than concatenation and that concatenation has a higher precedence than $+$.
- Stephen Kleene (1909–1994), a student of Church, invented regular expressions in 1951.

Regular Expressions as an Inductive Set

- Let RegExp be the inductive set defined by the following constructors:
  1. EmptySet : RegExp.
  2. EmptyString : RegExp.
  3. Symbol : $\Sigma \to$ RegExp.
  4. Union : RegExp $\times$ RegExp $\to$ RegExp.
  5. Concatenation : RegExp $\times$ RegExp $\to$ RegExp.
  6. Asterate : RegExp $\to$ RegExp.

Regular Expressions as Patterns

- A regular expression α over Σ can be viewed as a pattern that matches a set $L(\alpha) \subseteq \Sigma^*$ called the language of α.
- $L(\alpha)$ is defined by pattern matching as following:
  1. $L(\emptyset) = \emptyset$.
  2. $L(\epsilon) = \{\epsilon\}$.
  3. $L(a) = \{a\}$.
  4. $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$.
  5. $L(\alpha\beta) = L(\alpha)L(\beta)$.
  6. $L(\alpha^*) = (L(\alpha))^*$.
- Two regular expressions α and β over Σ are equivalent if $L(\alpha) = L(\beta)$.

Admin — March 3

- Midterm course review.
  - Survey on Avenue.
    - Open until 11:59 on Tuesday, March 10.
  - Discussion sessions with instructor:
    - Mon, Mar 2, at 4:30 in T13 105.
    - Tue, Mar 3, at 5:30 in T13 105.
    - Wed, Mar 4, at 6:30 in T13 105.
    - Thu, Mar 5, at 4:30 in T13 105.

    (1.0 percentage point bonus for attending a session.)
- Exercises.
  - Doing the exercises is the best way to learn the material!
  - The solutions for the Week N Exercises will be posted after Assignment N-2 is marked.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Midterm Test 2

- Midterm Test 2 will be held on Wednesday, March 11, at 7:00–9:00 PM in MDCL 1305.
- Same format as Midterm Test 1.
- Will cover the first four topics.
- The lecture on March 11 will be a review session.
- A sample test will be posted next Monday; solutions will be posted next Tuesday evening.
- There will be TA review sessions next Monday and Tuesday.
- Penalties:
  - 15% penalty for incorrect or missing student numbers or version numbers.
  - 5% penalty for incomplete erasures.

Assignment 6

Question 1. Let $L = \{a^m b^n c^p \mid 0 \leq m, n, p\}$. Construct an NFA N (without ϵ-transitions) and an NFA N' with ϵ-transitions such that $L(N) = L(N') = L$. Present each of N and N' as both a transition table and a transition diagram.

Question 2. Construct a DFA M with no inaccessible states that is equivalent to the NFA defined by the following transition table:

	$Q \backslash \Sigma$	0	1
start →	p	$\{q, s\}$	$\{q\}$
final →	q	$\{r\}$	$\{q, r\}$
	r	$\{s\}$	$\{p\}$
final →	s	$\{\}$	$\{p\}$

Present M as both a transition table and a transition diagram.

Review

- Equivalence of DFAs and NFAs.
- Subset construction.
- Regular expressions.
- Thompson's construction.

Identifiers (iClicker)

Which of the following regular expressions matches the set of identifiers of a programming language?

A. $(a + \cdots + z + A + \cdots + Z)^*$.

B. $(a + \cdots + z + A + \cdots + Z + 0 + \cdots + 9)^*$.

C. $(a + \cdots + z + A + \cdots + Z + 0 + \cdots + 9)^+$.

D. $\boxed{(a + \cdots + Z)(a + \cdots + Z + 0 + \cdots + 9)^*.}$

Regular Expressions (iClicker)

Which of the following regular expressions matches the set of words in an English dictionary that contain "oat", "boat", or "stoat"?

A. $(oat + boat + stoat)^*$.

B. $(a + \cdots + Z + \texttt{"-"})^*(oat + boat + stoat)^*$.

C. $(a + \cdots + Z + \texttt{"-"})^*(b + st)oat(a + \cdots + Z + \texttt{"-"})^*$.

D. $\boxed{(a + \cdots + Z + \texttt{"-"})^*(\epsilon + b + st)oat(a + \cdots + Z + \texttt{"-"})^*.}$

Kleene Algebras

- A Kleene algebra is a mathematical structure

  $$(K, 0, 1, +, \cdot, {}^*)$$

  where $0, 1 \in K$, $+ : K \times K \to K$, $\cdot : K \times K \to K$, and ${}^* : K \to K$ such that the axioms on the next slide are satisfied.
  - $a \cdot b$ is usually written as simply ab.
- Formulated in 1994 by Dexter Kozen (1951–), the author of our textbook AC, this set of axioms is the first finite axiomatization of Kleene algebras.
- Examples:
  1. $(\mathcal{P}(\Sigma^*), \emptyset, \{\epsilon\}, \cup, \text{concatenation}, {}^*)$.
  2. The set of regular expressions over Σ in which equivalent regular expressions are consider equal.

Axioms of a Kleene Algebras

Associativity of $+$:	$x + (y + z) = (x + y) + z$
Commutativity of $+$:	$x + y = y + x$
Idempotence of $+$:	$x + x = x$
Identity for $+$:	$x + 0 = x$
Associativity of $\cdot$:	$x(yz) = (xy)z$
Identity for $\cdot$:	$x1 = 1x = x$
Annihilator for $\cdot$:	$x0 = 0x = 0$
Distributivity:	$x(y + z) = xy + xz$
	$(x + y)z = xz + yz$
Asterate properties:	$1 + xx^* = x^*$
	$1 + x^*x = x^*$
	$y + xz \leq z \Rightarrow x^*y \leq z$
	$y + zx \leq z \Rightarrow yx^* \leq z$

Note: $a \leq b$ stands for $a + b = b$.

Properties of Regular Expressions (iClicker)

Which of the following is not a valid property of regular expressions (or Kleene algebras)?

A. $\boxed{\emptyset^* = \emptyset.}$

B. $\alpha + \alpha = \alpha.$

C. $\alpha + \beta = \beta + \alpha$

D. $\boxed{\alpha\beta = \beta\alpha.}$

Equivalence of Regular Expressions and FAs

- Theorem 4. If a regular expression matches a language L, there is an NFA with ϵ-transitions that accepts L.

  Proof. Use Thompson's construction to produce the NFA with ϵ-transitions.
- Theorem 5. If a DFA accepts a language L, there is a regular expression that matches L.

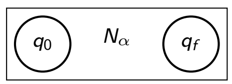
  Proof. Use Kleene's algorithm to produce the regular expression.
- Corollary 2. Regular expressions match the same class of languages that finite automata (DFAs and NFAs) accept — the class of regular languages.

  Proof. Use in order Theorem 4, Theorem 3, Theorem 1, and Theorem 5

Proof of Theorem 4 [1/5]

- We will prove a stronger theorem that implies Theorem 4.
- Theorem 6. Let α be a regular expression over Σ that matches a language L. Then there is an NFA N_α with ϵ-transitions that accepts L such that:
  1. N_α has one start state q_0.
  2. N_α has one final state q_f.
  3. There are at most two transitions from each state in N_α.
  4. There are no transitions from the final state in N_α.

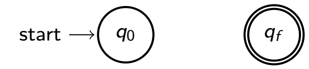
  N_α is represented graphically as:



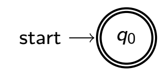
- The proof will be by structural induction on α using the construction named after Ken Thompson (1943–present).

Proof of Theorem 4 [2/5]

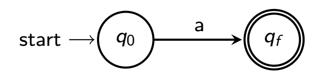
Base case 1: $\alpha = \emptyset$. Then N_α is:



Base case 2: $\alpha = \epsilon$. Then N_α is:

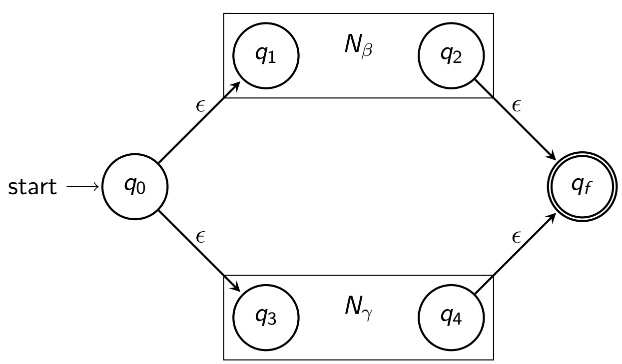


Base case 3: $\alpha = a \in \Sigma$. Then N_α is:



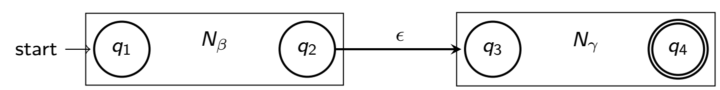
Proof of Theorem 4 [3/5]

Induction step 1: $\alpha = \beta + \gamma$. Assume N_β and N_γ are NFAs with ϵ-transitions that accept $L(\beta)$ and $L(\gamma)$ and satisfy the four conditions of the theorem. Then N_α is:



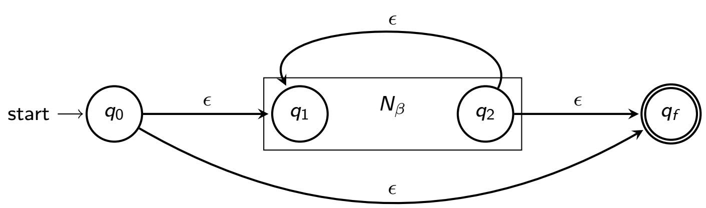
Proof of Theorem 4 [4/5]

Induction step 2: $\alpha = \beta\gamma$. Assume N_β and N_γ are NFAs with ϵ-transitions that accept $L(\beta)$ and $L(\gamma)$ and satisfy the four conditions of the theorem. Then N_α is:



Proof of Theorem 4 [5/5]

Induction step 3: $\alpha = \beta^*$. Assume N_β is a NFA with ϵ-transitions that accepts $L(\beta)$ and satisfies the four conditions of the theorem. Then N_α is:



Closure Properties of Regular Languages

Regular languages are closed under:

1. Union.
   - L_1 and L_2 are regular implies $L_1 \cup L_2$ is regular.
2. Concatenation.
   - L_1 and L_2 are regular implies $L_1 L_2$ is regular.
3. Asterate.
   - L is regular implies L^* is regular.
4. Complementation.
   - $L \subseteq \Sigma^*$ is regular implies $\sim L \subseteq \Sigma^*$ is regular.
5. Intersection.
   - L_1 and L_2 are regular implies $L_1 \cap L_2$ is regular.

7. Applications and Other Topics

Applications of Finite Automata

- Lexical analyzers.
  - The set T of tokens (strings that represent meaningful symbols) of a programming languages L is usually a regular set.
  - A lexical analyzer is a module in a compiler for L based on a FA that decides whether a given string is in T.
  - A lexical analyzer is automatically generated from a regular expression α matching T (by, e.g., $\alpha \mapsto$ NFA with ϵ-transitions $\mapsto$ DFA $\mapsto$ minimum-state DFA).
- Text editing.
  - String search and replacement is done by:
    1. Writing a regular expression that represents the set of strings to be matched.
    2. The regular expression is converted to an NFA with ϵ-transitions.
    3. The NFA with ϵ-transitions is directly simulated.

Other Topics

1. State minimization.
   - There is a simple algorithm that will collapse a DFA M into an minimum-state DFA M' that is equivalent to M.
2. Pumping lemma.
   - Used to identify nonregular languages.
3. Myhill-Nerode theorem.
   - A language L is regular iff a certain relation R_L has a finite number of equivalence classes.
4. Finite automata with output.
   - Moore machines.
   - Mealy machines.
5. Two-way finite automata.
   - Equivalent to standard one-way finite automata.