Theory of Computing

Sudhan Chitgopkar

March 2, 2021

02.04.21

See 02.04 Grafstate file

02.02.21

l** Reviewing δ

DFAs

- δ: Q $\times$ $\Sigma \rightarrow$ Q
- input: (state, symbol)
- output: state

NFAs

- Not allowing ϵ transitions
  - δ: Q $\times$ $\Sigma \rightarrow$ (Q)
  - input: (state, symbol)
  - output: set of states
- If an automaton is nondeterministic, then the codomain of δ is a power set
- Allowing ϵ transitions,
  - δ: Q $\times$ ($\cup$ $\{\epsilon\}$) $\rightarrow$ (Q)
  - input: (state, symbol or ϵ)
  - output: set of states
- Note that ϵ is not a symbol and cannot belong to Σ
  - This is because symbols have a length of 1 and ϵ, by definition, has a length of 0

NFA

- General 5-tuple including
  1. Q: set of states
  2. Σ - input alphabet
  3. δ: Q $\times$ ($\Sigma \cup \{\epsilon\}$) $\rightarrow$ (Q)
  4. $q_0 \in$ Q - starting state
  5. F $\subset$ Q - set of final states
- There may exist some inputs for which it is possible that the NFA accepts the sequence if one path is taken and rejects the sequence if another path is taken
- The NFA does not have the ability to look ahead at possible states

Computation

- The NFA runs all possible branches on a given string simultaneously and independently
- A string, w, is rejected by an NFA, N, if every branch of the nondeterminism tree for N on w rejects
  - $\neg$ $\exists$ a branch that accepts
- A string, w, is accepted by an NFA, N, if $\exists$ a branch on the nondeterminism tree for which N on w accepts
- Theorem: Any language recognized by an NFA can be recognized by a DFA
  - Let N be an NFA.
  - Then $\exists$ a DFA, M, with L(M) = L(N)
  - The language recognized by a machine, M, (L(M)) is the set of exactly all strings accepted by M (no rejected strings allowed)
    - $\neg$ L(M) = set of all strings rejected by M

01.28.21

Nondeterministic Finite Automata

DFA Review

- DFA's are 5-tuples with
  - δ: Q $\times$ $\Sigma \rightarrow$ Q
  - The number of transitions, δ, is $|Q||\Sigma|$
- There is no real decision-making here, an input is simply being used alongside a rule to find an output

NFA's

- NFAs are also a 5-tuple
- δ: Q $\times$ Σ $\rightarrow$ P(Q)
- $|P(Q)| = 2^{|Q|}$

01.26.21 - 01.28.20

Closure Introduction

- The language recognized by a DFA, M, (L(M)) is the set of all string accepted by M
- Thus, M = (Q, Σ, δ, q_0,F)
- And Σ* is the universe of all possible inputs to M
- $\forall$ strings w $\in$ Σ*, M either accepts or rejects w
- It follows that L(M) $\subset$ Σ*
- And $\neg$ L(M) = Σ* - L(M)
- Therefore, M accepts every string in L(M) and rejects everything in $\neg$ L(M)

Closure Continued

- A set, A, is closed under a binary operation, OP, if $\forall$ x , y $\in$ A [x OP y $\in$ A]
- Ex. Natural Numbers ($\mathbb{N}$)
    1. $\mathbb{N}$ is closed under +
    2. $\mathbb{N}$ is closed under $\times$
    3. $\mathbb{N}$ is not closed under -
    4. $\mathbb{N}$ is not closed under \
- The class of all languages that are recognized by DFAs is closed under $\cup$

Closure Properties of DFAs

- Union ($\cup$)
- Intersection ($\cap$)
- Complement ($\neg$)
- Reverse

Applying Closure Properties

- If $L(M_1) \cup L(M_2)$ are DFAs, then $\exists$ DFA, M, with $L(M) = L(M_1) \cup L(M_2)$
- The purpose of a state machine is to make a yes/no decision

Premise:

- $M_1 = \{Q_1,\Sigma,\delta_1,q_{0_1},F_1\}$ and $M_2 = \{Q_2,\Sigma,\delta_2,q_0,F_2\}$ are DFAs
- M_1 accepts binary strings ending in 1
- M_2 accepts binary strings of odd length
- $L(M) = L(M_1) \cup L(M_2)$
  - Accepts binary strings that either end in 1 OR have odd length (or both)
- M_1: q_1, q_2 distinguished between ending in 0 and 1
- M_2: r_1, r_2 distinguished between odd and even length
- Accordingly, M must be able to distinguish between:
  - even length ending in 1
  - even length ending in 0
  - odd length ending in 1
  - odd length ending in 0

Coding M

- Consider $Q = Q_1 \cdot Q_2 = \{q_1r_1,q_2r_1,q_1r_2,q_2r_2\}$
- wherein
  - q_1r_1 = even string ending in 0
  - q_1r_2 = odd string ending in 0
  - q_2r_1 = even string ending in 1
  - q_2r_2 = odd string ending in 1
- Ex. $\delta(q_1r_1,1) = q_1r_2$
- Applying this logic to a DFA, we know that
  - Q={q1r1,q1r2,q2r1,q2r2};
  - S={0,1};
  - d:Q *sigma → Q;
  - d(q1r1,0)=q1r2;
  - d(q1r1,1)=q2r2;
  - d(q1r2,0)=q1r1;
  - d(q1r2,1)=q2r1;
  - d(q2r1,0)=q1r2;
  - d(q2r1,1)=q2r2;
  - d(q2r2,0)=q1r1;
  - d(q2r2,1)=q2r1;
  - q0=q1r1;
  - F={q1r2,q2r1,q2r2};

Derivation

- Construct M and show that L(M) = $L(M_1) \cup L(M_2)$.
- $Q = Q_1 \cdot Q_2$
- let $q_i \in Q_1$ and let $r_j \in Q_2$. and let $c \in \Sigma$
- and $q_i r_j \in Q$
- thus, $\delta(q_i r_j,c) = \delta_1(q_i,c)\delta_2(r_j,c)$
- and $q_0 = q_{01},q_{02}$
- so that $F=\{q_i r_j\text{: } q_i \in F_1 \cup r_j \in F_2\}$

Correctness

- Show that M accepts exactly the strings that are accepted by M_1 or M_2
- If $w \in \Sigma^*$. tjem w is accepted by M and w is accepted by either M_1 or M{2}
- To do this, we can organize Σ^* into strings of length 0, length 1, length 2...
- We solve with mathematical induction, which is how we prove recurrence relationships

Mathematical Induction

- Need base case and induction hypothesis
- Induction hypothesis says something is true about k, where k is the length of strings
- δ^* (q_0,w) is the ending state of M on w where
- δ^* $(q_0,w) = \delta^*{}_1(q_{01},w)\delta^*{}_2(q_{02},w)$
- Induction Hypothesis: If $|w| = k$, then $\delta^*(q_0,w) \in F \iff \delta^*{}_1(q_{0_1},w) \in F_1$ or $\delta^*{}_2(q_{02},w)$ inf F_2
- Let $x \in \Sigma$.
- Then $|w \cdot x| = k+1$
- And $\delta^*(q_{0,wx}) = \delta^*{}_1(q_{01},wx)\delta^*{}_2(q_{02},wx)$
- If $\delta^*{}_1(q_{01},wx) \in F_1$, $\delta^*(q_{0,wx}) \in F$
- Similarly, if delta$^*{}_2(q_{02},wx)$, then $\delta^*(q_{0,wx}) \in F$

01.21.21

Deterministic Finite Automata

- We know that $\delta = \mathrm{Q} \times \Sigma \to \mathrm{Q}_2$
- We want a function that takes a starting state and a string, then returns the state after the machine has read that string
- Let's define $\delta* = \mathrm{Q} \times \Sigma* \to \mathrm{Q}$
  - $\delta*$ takes a state and a string
  - δ takes a state and a symbol
- Now, we need a recursive definition
  - Base case:
    - Let $\mathrm{q_i} \in \mathrm{Q}$
    - $\delta*(\mathrm{q_i},\epsilon) = \mathrm{q_i}$
  - Recursive step:
    - If $\mathrm{q_i} \in \mathrm{Q}$, $\mathrm{w} \in \Sigma*$, and $\mathrm{c} \in \Sigma$
    - then $\delta*(\mathrm{q}, \mathrm{w} \cdot \mathrm{c}) = \delta(\delta*(\mathrm{q_i},\mathrm{w}),\mathrm{c})$

Aside: On Σ and $\Sigma*$

- $\Sigma*$ is the universe of all strings over Σ
  - $\Sigma = \{0,1\}$
  - $\Sigma* = \{\epsilon,0,1,00,01,11,10,000,\ldots\}$
- We can see this recursively
  - Base step: $\epsilon \in \Sigma*$
  - Recursive step:
  - Let $\mathrm{w} \in \Sigma*$, let $\mathrm{c} \in \Sigma$
  - Then $\mathrm{w} \times \mathrm{c} \in \Sigma*$
- We can see this recursion graphically

w	c	w · c	step
ϵ		-	base
ϵ	1	1	recursive
1	0	10	recursive
10	1	101	recursive

Recursively Testing 101

- Solve $\delta*(\mathrm{q_1},101)$
- $\delta(\delta*(\mathrm{q_1},10),1)$
- $\delta*(\mathrm{q_1},10)$
- $\delta(\delta*(\mathrm{q_1},1),0)$

TODO Complete Recursion Sequence

01.19.21

Tuples & DFAs

- Tuples are sequences which are always finite in length
- The deterministic finite automaton shown is a 5-tuple:
  1. Q: finite nonempty set of states
     - state: configuration of logic of a machine
  2. Σ (Sigma) - input alphabet
     - alphabet: a finite, nonempty set of symbols where symbols are an object of length 1
  3. δ (Delta) - transition function
  4. $Q_0 \in Q$ - starting state
  5. $F \subset Q$ - set of final states
- For this deterministic finite automaton,
  - δ: $Q \times \Sigma \rightarrow Q_2$

  Represented as a table,

Step	State	Input	Transition
1	Q_1	1	$Q_1 \rightarrow Q_2$
2	Q_2	0	$Q_2 \rightarrow Q_1$
3	Q_1	1	$Q_1 \rightarrow Q_2$
4	Q_2	1	$Q_2 \rightarrow Q_2$

Domains & Codomains

- Domain: set of all possible function inputs
- Codomain: set of all possible outputs

Strings

- In computer science, strings are character arrays
- In mathematics, strings are sequences of symbols
- Specifically a string over an alphabet, Σ, is a sequence of symbols belonging to Σ
- ϵ is the empty string
- Concatenation: If $w_1, w_2 \in \Sigma$, $w_1 \cdot w_2 = w_1w_2$
- If $c \in \Sigma$, then $\epsilon \cdot c = c \cdot \epsilon = c$

TODO Review Recursive Definitions

- Base step: a step that can not be broken down any further, a fact that is always true regardless of the input
- Recursive step:
- Defining the length of a string over Σ
  - Base: $|\epsilon| = 0$
  - Recursive:
    * let w be a string over Σ, and c $\in \Sigma$
    * then $|\mathrm{w} \cdot \mathrm{c}| = |\mathrm{w}| + 1$
- Using this to define $|1011|$,
  1. $|1011| = |101 \cdot 1| = |101| + 1 =$
  2. $|10 \cdot 1| + 1 = |10| + 1 + 1 =$
  3. $|1 \cdot 0| + 1 + 1 = |1| + 1 + 1 + 1 =$
  4. $|\epsilon \cdot 1| + 1 + 1 + 1 =$
  5. $|\epsilon| + 1 + 1 + 1 + 1 =$
  6. $0 + 1 + 1 + 1 + 1 = 4$

Languages

- Languages over Σ - a set of finite strings over Σ
- Langauges recognized by an automaton, M, L(M) is the language accepted by M
- $\emptyset$ is the empty language
- $\epsilon \neq \emptyset$
- $\epsilon \neq \{\epsilon\}$
- ϵ is not a symbol in any alphabet

01.14.21

Automaton (automata)

- Self running machine requiring a continuous power source
  - Historically used power sources include water, steam, and electricity
- Course revolves around defining the mathematics powering machines

The Mathematics of Automata

Mathematicians & History

- Cantor defines sets as collections of objects
- Cantor also argues that infinites can be of different magnitudes - there are infinitely more real numbers than natural numbers
- Goedel eventually derives his incompleteness theorem
  - No logical system that contains the natural numbers can prove its own soundness
  - Every sound logical system containing the natural numbers contains valid statements that cannot be proved or disproved
- In 1936, Turing proves The Halting Problem is not decidable, it is impossible
  - The Halting Problem is an algorithm that can analyze any other algorithm and determine whether or not it goes into an infinite loop
- Turing creates the turing machine as an object consisting of sets and processes wherein the object can use any finite process to complete an action.
- Turing machine sets the basis for a computer, which leads to a series of important questions:
  - What can & can't a machine do?
  - What does it mean for a problem ot be harder than another?
  - What does it mean for a machine to be more powerfule than another?

Sequential Logic

- Sentential Logic- based on boolean results
  - Predicated on AND, OR, NOT
  - XOR, XAND, etc. can be derived using the above

Necessary Review

- Textbook Ch. 0
- Logic Statements
- Set Theory
- Functions

Functions

- Functions - something that maps objects from one set to another
- Given f: a $\rightarrow$ b;
  - Everything in a is mapped to something in b

    - For every x, such that x is an element of a, there exists a y, such that y is an element of b
  - No one point in the domain can be mapped to two different points in the codomain
    - Logically, you can't have a function that takes in one input and returns two different outputs
    - If f maps x → y1 and → y2, y1 = y2

  -$\forall$ x $\in$ A y_1,y_2 $\in$ B [f(x)=y_1 $\wedge$ f(x)=y_2 $\rightarrow$ y_1 = y_2]

TODO Types of Functions - Definition & Logical Statement

- Injective Functions
- Surjective Functions
- Proof by Induction ($\forall$)
- Proof by Contradiction ($\neg\exists$)

Finite Automaton (Finite State Machine)

- States are logical confirgurations
- States are generally based upon input
- Purpose of a state machine is to make a yes/no decision