CHAPTER 2.3

Pumping lemma for context-free language:

If A is a context-free language, then there is a number *p* (the pumping length) where, if *s* is any string in A of length at least *p*, then *s* may be divided into 5 pieces s = uvxyz satisfying the condition:

1. For each i >= 0, $uv^ixy^iz \in A$
2. |vy| > 0
3. |vxy| <= p

How to prove a language is not context-free

$L = 0^n1^n2^n \mid n \geq 1$ ➔ L = {012, 001122, 000111222...} ➔ Z = 001122 (n = 6, u = 0, v = 01, w = 1, x = 2, y = 2)

Case 1: |VWX| ≤ n => 2+1+1 = 4≤6. ||| Case 2: |VX| ≥ 2 + 1 = 3 ≥ 1 ||||

Case 3: uv^iwx^iy € L, i = 2 ➔ u = 0, v= 0101, w = 1, x = 22, y = 2 not in L because 1 not following by 2. ➔ L is not CFL

CHAPTER 3

Formal definition of Turing Machine.

A Turing Machine is a 7-tuple, $(Q, \Sigma, T, \partial, q_0, q_{accept}, q_{reject})$:

1. Q is a finite set of states
2. Σ is the input alphabet not containing the blank symbol
3. T is the tape alphabet
4. ∂: Q x T → Q x T x {L, R} is the transition function
5. q_0 is the start state
6. q_{accept} is the accept state
7. q_{reject} is the reject state, where $q_{accept} != q_{reject}$

Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input.

Configuration of the Turing Machine: A setting of current state, current tape contents, current head location

→ uqv configuration means: current state is q, current tape contents is uv, current head location is the first symbol of v.

Configuration C1 yields configuration C2 if the Turing machine can legally go from C1 to C2 in a single step.

Start configuration: q_0w

Accepting configuration: q_{accept}

Rejecting configuration: q_{reject}

Halting configurations: accepting and rejecting configurations.

A Turing machine M accepts input w if a sequence of configurations $C_1, C_2, C_3, \ldots, C_k$ exists where:

1. C_1 is the start configuration of M on input w
2. Each C_i yields C_{i+1} and
3. C_k is an accepting configuration

The language of M, or the **language recognized by M**, L(M), is the collection of strings that M accepts.

Call a language **Turing-recognizable** if some Turing machine recognizes it.

Three possible outcomes for a Turing machine are **accept**, **reject** and **loop**

Loop means that machine simply does not halt

Deciders are Turing machines that halt on all inputs.

A decider that recognizes some language is also said to **decide** that language.

Call a language **Turing-decidable** if some Turing machine decides it.

Every decidable language is Turing-recognizable.

How to design an algorithm for a Turing machine:

→ Example 3.7 (page 171)

How to write the sequence of configurations for a Turing machine:

→ Figure 3.8 (page 172)

Variants of the Turing machine model: the alternative definitions of Turing machines

Robustness: invariance to certain changes in the definition → the original model and its reasonable variants all have the same power

Multitape Turing machine: is an ordinary Turing machine with several tapes.

- $\partial: Q \times T^k \to Q \times T^k \times \{L, R, S\}^k$
- k: the number of tapes

Nondeterministic Turing machine: a Turing machine that may proceed according to several possibilities.

- $\partial: Q \times T \to P(Q \times T \times \{L, R\})$
- Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Enumerator: is a Turing machine with an attached printer.

HILBERT'S PROBLEM

Polynomial is a sum of terms, where each **term** is a product of certain variables and a constant, called a **coefficient**.

Root: is an assignment of values to its variables so that the value of the polynomial is 0.

Integral root: is a root where all the variables are assigned integer values.

Church -Turing thesis: the intuitive notion of algorithms equals the Turing machine algorithms.

To describe a Turing machine algorithm:

- Formal description: give details on the Turing machine's states, transition functions and so on.
- Implementation description: use English to describe the way that a Turing machine moves its head and the way it stores data on its tape.
- High-level description: use English to describe an algorithm, ignoring the implementation details

<u>CHAPTER 4</u>

A_{DFA}: is a language expressed as the acceptance problem for DFAs of testing whether a particular deterministic finite automaton accepts a given string

$A_{DFA} = \{\langle B, w\rangle \mid B$ is a DFA that accepts input string $w\}$.

A_{DFA} is a decidable language.

$EQ_{DFA} = \{\langle A, B\rangle \mid A$ and B are DFAs and $L(A) = L(B)\}$.

EQ_{DFA} is a decidable language

$A_{NFA} = \{\langle B, w\rangle \mid B$ is an NFA that accepts input string $w\}$.

$A_{REX} = \{\langle R, w\rangle \mid R$ is a regular expression that generates string $w\}$.

$A_{CFG} = \{\langle G, w\rangle \mid G$ is a CFG that generates string $w\}$.

$E_{CFG} = \{\langle G\rangle \mid G$ is a CFG and $L(G) = \varnothing\}$. → it's a decidable language

$EQ_{CFG} = \{\langle G, H\rangle \mid G$ and H are CFGs and $L(G) = L(H)\}$. is undecidable.

Every context free language is decidable.

Prove A_{LISA}? is undecidable

Regular language → Context-free → Decidable → Recognizable

$A_{TM} = \{\langle M, w\rangle \mid M$ is a TM that accepts $w\}$. → **undecidable**

One-to-one (injective): never maps 2 different elements to the same place, f(a) ≠ f(b) whenever a ≠ b

Onto (surjective): for every b ∈ B, there's an a ∈ A such that f(a) = b

Correspondence (bijective): both one-to-one and onto

A and B are the **same size** if there is a one-to-one, onto function f: A → B

A set A is **countable** if either it is finite or it has the same size as N
The set R of real numbers is **uncountable**.

Each language A ∈ L has a unique sequence in B.The ith bit of that sequence is a 1 if si ∈ A and is a 0 if si not ∈ A, which is called the **characteristic sequence** of A

co-Turing-recognizable: the language that is the complement of a Turing-recognizable language.
A language is **decidable** iff it is Turing-recognizable and co-Turing-recognizable.

CHAPTER 5

Reducibility: the primary method for proving that problems are computationally unsolvable.
A **reduction** is a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem.

$HALT_{TM}$ = {<M, w>| M is a TM and M halts on input w}. → undecidable

E_{TM} = {<M>| M is a TM and L(M) = ∅}. → undecidable ← prove.

$REGULAR_{TM}$ = {<M>| M is a TM and L(M) is a regular language} → undecidable

EQ_{TM} = {<M_1, M_2>| M_1, M_2 are TMs and L(M_1) = L(M_2)} → undecidable

Accepting computation history: a sequence of configurations, where C1 is the start of configuration of M on w, C(l) is an accepting configuration of M, and each Ci legally follows from C(i-1) according to the rules of M.
Rejecting computation history: The same, except Cl is a rejecting configuration.
Linear bounded automaton: a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. It has limited amount of memory.
A_{LBA} = {<M, w>| M is an LBA that accepts string w}.
A_{LBA} is decidable. ← prove it.
Lemma 5.8: Let M be an LBA with q states and g symbols in the tape alphabet. There are exactly qng^n distinct configurations of M for a tape of length n.

E_{LBA} = {<M>| M is an LBA where L(M) = ∅}.

E_{LBA} is undecidable.
ALL_{CFG} = {<G>| G is a CFG and L(G) = Σ^*}.

A_{LBA} is decidable

The algorithm that decides A_{LBA} is as follows.

L = "On input <M,w>, where M is an LBA and w is a string:

1. Simulate M on w for qng^n steps or until it halts
2. If M has halted, *accept* it if has accepted and *reject* if it has *rejected*. If it has not halted, *reject*. "

EQ_{TM} is undecidable

We let TM R decide decide EQ_{TM} and construct TM S to decide E_{TM} as follows

S = "On input <M> , where M is a TM:

1. Run R on input <M,M_1>, where M_1 is a TM that rejects all inputs
2. If R accepts, *accept*; if R rejects, *reject*"

If R decides EQ_{TM} , S decides E_{TM} . But E_{TM} is undecidable by Theorem 5.2, so EQ_{TM} also must be undecidable

Lemma 5.8

Let M be an LBA with q states g symbols in the tape alphabet. There are exactly qng^n disctinct configuration of M for a tape of length n

A config consists of the state of control, position of he head, and contents of tape. Here M has q states. The length of its tape is n, so the head can be in on of n positions, and g^n possible strings of tape symbols appear on tape.

A_{LBA} is decidable

The algorithm that decides A_{LBA} is as follows

L= " On input <M,w>, where M is an LBA and w is a string:

1. Simulate M on w for qng^n steps or until it halts.
2. If M has halted, accept f it has accepted and rejected if it has rejected. If it has not halted, reject

E_{LBA} is undecidable

Suppose TM R decides E_{LBA}. Constructs TM S to decide A_{TM} as follows

S = "On input <M,w> where M is a TM and w is a string

1. Construct LBA B from M and w as described in the proof idea
2. Run R on input <B>
3. If R rejects, accept; if R accepts, reject.

$HALT_{TM}$ is undecidable

Let's assume for the purpose of obtaining a contradiction that TM R decides HALT.
We construct TM S to decide A^{TM} with S operating as follows
S-= "On input <M, w>, an encoding of a TM M and a string w:

1. Run TM R on input <M,w>
2. If R rejects, reject.
3. If R accepts, simulate M on w until it halts.
4. IF M has accepted, accept; if M has rejected, reject

E_{TM} is undecidable

Let's write the modified machine described in the proof idea using our standard notation. We call it M_1

M_1 = "On input x:

1. If x is not w, reject
2. If x = w, run M on input w and accept if M does

Assume that TM R decides E TM and construct TM S that decides A TM as follows
S = "On input <M,w>, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M1 just described
2. Run R on input <M1>
3. If R accepts, reject; If R rejects, accept

If R were decider for E TM, S would be decider for A TM. A decider for a A TM cannot exist, so we know that E TM must be undecidable.

Regular TM is undecidable

We let R be a TM that decides REGULAR TM and construct TM S to decide A TM. Then S works in the following manner.
S = "On input <M,w>, where M is a TM and w is a string:

1. Construct the following TM M_2
   M_2 = "On input x:
   1. If x has the form 0^n1^n, accept
   2. If x does no have this form, run M on input w and accept if M accepts w"
   3. Run R on input<M_2>
   4. If R accepts, accept; if R rejects, reject"

EQ_{TM} is undecidable

We let TM R decide EQ_{TM} were decidable, E_{TM} as follows
S = "On input <M>, where M is a TM:

1. Run R on input <M,M_1>, where M_1 is a TM that rejects all
2. If R accepts, accept; if R rejects, reject"

Chuong Vu
9/29
Quiz 1

Foundation
Krishnan

83 90

① Difference between
- Set, Subset, Proper subset, Power set

Let R be a set R = {1, 2, 3}
- subset {1} ⊆ R, {2} ⊆ R, {3} ⊆ R
- Proper subset A = {1,2,3} ∈ R, A is proper subset of R, A ⊆ R
- Power set contain all subset of set
  P(R) = {∅, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}}

- Domain is the set of all possible input to a function. f(n)
  Range is the output of function f(n)
- Empty set = {∅} / Empty string = { } or ε
- Function is a object that set up an input and output relationship
  Relation. A predicate, most typically when the domain is a set of k-tuples

② - {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}
- {0, 1, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}
- {1, 8, 27, 64, 125} -1.5

③. DFA: Determinate Finite Automaton, means that it can only be in and transition to one state at a time.
NFA: Nondeterminate Finite Automaton, means that it can transition to and be in multiple states at once.
-0.5 Need formal definition

Language: A set of a strings.
Regular Language: Let A and B be language. We define the regular language operation union, concatenation, and star.

Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

Concatenation: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

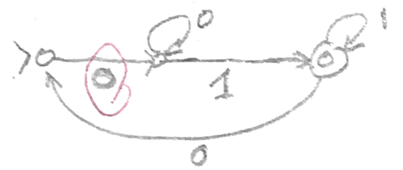
Star $A^* = \{x_1 x_2 x_3 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

(4) $\Sigma = \{0, 1\}$

w is a strings of the form $0^* 1 1^*$ } draw a

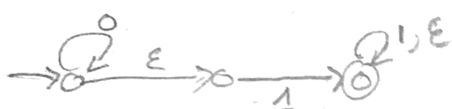
$0^* 1 1^* = 00\ldots 11\ldots 1$

DFA:



-5

NFA:



1 - 0

2 - 10

3 - 5

4 - 5

-10

Viet Tran Quoc Hoang
01607460

72/100

Department of Computer Science
University of Massachusetts Lowell
COMP.3040 Foundations of Computer Science
Fall 2017
Quiz 1 [5%]
9/28/2017

Domain: set of all possible inputs of a function

Range: set of all possible output of a function

set: a group of ~~project~~ objects presented as an unit i.e set $A = \{1, 2, 3\}$

subset: a set ~~which~~ whose elements also belong to a different set

proper subset: set A is a subset of set B but set A is not equal to set B

power set: a set containing all possible sets $P(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{2,3\}, \{1,2\}, \{1,2,3\}, \{1,3\}\}$

1. *[30 points]* Differentiate between
   - Set, Subset, Proper Subset, Power Set
   - Domain, Range
   - Empty set, Empty string
   - Function, Relation
2. *[15 points]* Write a formal description for the following sets
   - set containing 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024
   - set containing 0-9, a-f
   - set containing 1, 8, 27, 64, 125
3. *[20 points]* Give the formal definition for
   - DFA
   - NFA
   - Regular Language
   - Regular Expression
4. *[5 points]* A NFA is more powerful than a DFA.
   - True
   - False

   -5

5. *[5 points]* Every NFA can be converted to a DFA.
   - True
   - False

   -5

6. *[5 points]* A language is Regular if a DFA or NFA exists which can recognize it.
   - True
   - False

   -5

7. *[5 points]* All regular languages are infinite–contain infinite number of strings }
   - True

   -5

- False

8. *[15 points]* The class of regular languages is closed under the following operations:

Viet Tran Quoc Hoang - 01607460 Viet_tran1@student.uml.edu

(1) Set : a group of objects presented as an unit i.e set $A = \{1,2,3\}$
$B = \{1,2,3,4,5\}$

Subset : a set whose element also belong to a diff. set ie. $A \subseteq B$

Proper subset : set A is a subset of set B while set A is not equat to set B

Power set : a set containing all possible sets

Function :

$P(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{1,3\}$
$\{1,2,3\}\}$

Domain : set of possible input for a function

Range : set of possible output for a function

Function : establish a relationship between input and output
g takes input and produce output

Relation : ~~property~~ ~~which~~ how all elements suppose to interact with another elements

(2) $1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024$ (2^5, 2^6, 2^7, 2^8, 2^9, 2^{10})

$\{x \mid x = 2^i, 0 \leq i \leq 10\}$

0-9, a-f

$\{y \mid$ y is the hexadecimal values for decimal values of 0 to 15$\}$

1^3 2^3 3^3 4^3 5^3
1, 8, 27, 64, 125

$\{z \mid i^3, 1 \leq i \leq 5\}$

(3)
A regular language can be expressed by a DFA, containing string that said DFA can accept (A) (L) = n Language A String = n
A regular expression is used to search string following a strict pattern

-3

Formal Definition

3) DFA $(Q, \Sigma, \delta, q_0, F)$
a) Q is a finite set called starting state
Σ is a finite set call alphabet
δ $Q \times \Sigma \rightarrow Q$ is the transition function
$q_0 \in Q$ is the start state
$F \subseteq Q$ is the set of final state

-5 (4) True, a NFA is more > DFA

-5 (5) False, same NFA cannot be converted to DFA

-2 (6) ~~True~~ False, only DFA to determine reg. language

-2 (7) True

(8) Concatenation $A \circ B$ $A \times B$
Multiplication $A * B$
Union $A \cup B$

-2

3) NFA $(Q, \Sigma, \delta, q_0, F)$
b) Q is ~~not~~ finite set call starting state
Σ is a infinite set called alphabet
δ $Q \times \Sigma \rightarrow P(Q)$ is the transition function
$q_0 \in Q$ is the start state
$F \subseteq Q$ is the final state