CS 138 Notes: Formal Languages & Automata

Alex Mei

Summer, 2020

1 Regular Languages

1.1 Definitions

Alphabet (Σ): finite, non-empty set of symbols in a language
String: finite sequence of symbols in the alphabet; $\mid s \mid$ denotes the length of string s; st denotes string s concatenated with string t
Substring: s is a substring of x if there exists strings y, z such that $x = ysz$
Prefix: for a string $x = sy$, s is a prefix of string x
Suffix: for a string $x = ys$, s is a suffix of string x
Reversal (s^R): for a string $s = s_1 s_2 ... s_k$, $s^R = s_k ... s_2 s_1$
Reversal Property: $(uv)^R = v^R u^R$
Language: set of strings created from an alphabet

1.2 Language Operations

Union: $A \cup B$ = elements in A or B (or both)
Concatenation: $A \cap B$ = elements in A and B
Difference: $A - B$ = elements in A and not B
Concatenation $AB = \{xy \mid x \in A, y \in B\}$
Exponentiation: $A^k = A...A$ (k times)
Star: $A^* = A^0 \cup A^1 \cup ...$
Language Property: $L \subseteq \Sigma^*$

1.3 Deterministic Finite Automata (DFA)

Definition: DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is the alphabet, δ is the transition function such that $\delta : Q \times \Sigma \to Q$, q_0 is the starting state such that $q_0 \in Q$, and F is the set of accepting states where $F \subseteq Q$

Accept: A DFA M is said to accept string w (where w only consists of characters in the alphabet) iff there exists a sequence of states $s_0...s_n$ such that $s_0 = q_0; s_i = \delta(s_{i-1}, a_i), a < i \leq n; s_n \in F$
Recognized Language: $L(M)$ denotes a language recognized by an automata; all languages recognized by DFAs are regular
DFA Properties: finite number of states; input is finite; reads a single character at a time from the left; only know current state; explicit transitions for each state and each symbol in the alphabet

1.4 Nondeterministic Finite Automata (NFA)

Definition: NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ similar to a DFA; the difference is the transition function, $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$
Nondeterminism: states do not require unique transitions from one state to another
Accept: a string is accepted by an NFA only if the entire input is read and the automata ends at an accepting state by at least one possibility
Epsilon Transition: a transition reading an empty string (valid for NFA only)

1.5 NFA to DFA

Theorem 1.3.9: Given an NFA N $= (Q, \Sigma, \delta, q_0, F)$, construct a DFA D $= (Q', \Sigma', \delta', q_0', F')$: $Q' = \mathcal{P}(Q), q_0' = q_0, F' = \{s \in Q' : s \cap F \neq \emptyset\}$, and $\delta(s, a) = \bigcup_{t \in S} closure(\delta(t, a))$
Closure Function: returns set of all possible states reachable using 0 or more epsilon transitions

1.6 Closure Properties

Definition: an operation applied to only regular languages will result in a regular language
Closed Operations: union, concatenation, complement, star, intersection

1.7 Regular Expressions

Definition: descriptive form of regular languages
Valid Expressions: $\emptyset, \epsilon, a \in \Sigma, r_1 \mid r_2, r_1 \cdot r_2, r_1*$ where r_1, r_2 are regular and $\mid$ denotes union
Order of Precedence: star, concatenation, union

1.8 Nonregular Languages

Pumping Lemma: if L is a regular language, then there exists a nonzero pumping length p such that if $w \in L$ and $\mid w \mid \geq p$ then w can be divided into 3 substrings $w = xyz$ such that the following conditions hold: $xy^iz \in L$ for $i \geq 0$, $\mid y \mid > 0$, and $\mid xy \mid \leq p$
Showing Nonregularity: use pumping lemma to arrive at a contradiction that there exists a string

in a language that cannot be pumped and arrive at a contradiction

Using Closure Properties: if a language is known to be regular, apply an operation closed under regular languages to that language with the questioned language and prove the result is nonregular to arrive at a contradiction

2 Context Free Languages

2.1 Context Free Grammars (CFG)

Variables: rules are applied and variables are substituted with values

Context Free: irrelevant of surrounding context of variables

Derivation Tree: tree to show which rules of a CFG were applied at which step

Terminals: terminating characters (symbols in the alphabet)

Definition: A CFG is a 4-tuple: (V, Σ, R, S) where V is the set of nonterminals (variables), Σ is the set of terminals, R is the set of rules $R : V \to (V \mid \Sigma)^*$, and S is the start symbol where $S \in V$ (Note: $V \cap \Sigma = \emptyset$)

Language: The language of a CFG is $L = \{w \in \Sigma^* : s \to^* w\}$

Closed Operations: union, concatenation, star, reversal

Proving Properties: use induction to prove $L(G) \subseteq L$ where L is a language of the described property

Proving Equivlence: prove both $L(G) \subseteq L$ and $L \subseteq L(G)$ to show $L = L(G)$ using induction

2.2 Chomsky Normal Form (CNF)

Definition: A CFG is in CNF iff every rule is in one of the following forms: $A \to BC, A \to a, S \to \epsilon$ where S is the start symbol and $B, C \neq S$

Converting to CNF: add new start symbol; eliminate non-singleton terminals; decompose rules with more than 2 nonterminals; remove epsilon transitions; remove unit rules

2.3 Pushdown Automata

Description: NFA + stack memory

Definition: A PDA is a 6-tuple: $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Σ is the input alphabet, Γ is the tape alphabet, and $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$

Accept: A PDA accepts input w if $w = w_1 w_2 ... w_n$ such that $w_i \in \Sigma_\epsilon$, there exists a sequence of states $s_0, s_1, ..., s_n \in Q$, and there exists strings $y_0, y_1, ...y_n \in \Gamma$ representing the state of the stack such that $s_0 = q_0, y_0 = \epsilon, s_n \in F$, and $(s_i, b) \in \delta(s_{i-1}, w_i, a)$ where $y_{i-1} = at, y_i = bt$, for $a, b \in \Gamma_\epsilon$, and $t \in \Gamma^*$

Note: a PDA with deterministic ability is not the same as a nondeterministic PDA

2.4 CFG to PDA

Idea: Push $ and start symbol to stack; repeat the following: if top of stack is a nonterminal, pop it and push the right side of the nonterminal, if the top of the stack is a terminal, match it with current input character, and if the top of the stack is $, go to the accept state

2.5 Non-Context Free Languages

Pumping Lemma: if L is a CFL, then there exists a nonzero pumping length p such that if $w \in l$ and $\mid w \mid \geq p$, then w can be divided into 5 pieces $uvxyz$ such that $uv^ixy^iz \in L$ for $i \geq 0$, $\mid vy \mid > 0$, and $\mid vxy \mid \leq p$

3 Unrestricted Languages

3.1 Definitions

Subsets: Regular $\subseteq$ Linear $\subseteq$ Deterministic Context Free $\subseteq$ Context Free $\subseteq$ Context Sensitive $\subseteq$ Unrestricted
Context Sensitive: has rules such as $uAv \rightarrow uyw$ such that $y \in (v \cup \Sigma)^*$

3.2 Turing Machines (TM)

Description: DFA + Infinite Linked List
Results: A TM halts when it transitions to an explicit accept or reject state; a TM may loop infinitely and never halt
Definition: A TM is a 7-tuple: $(Q, \Sigma, \Gamma, \delta, q_0, q_accept, q_reject)$ where Γ is the tape alphabet; $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
Configuration: The configuration of a TM is uqv where $u \in \Gamma^*$ is the tape contents left of the current position, $q \in Q$ is the current state, and $v \in Gamma$ is the tape contents of current position and everything to the right (excluding blank symbols); the configuration uaq_ibv yields uq_jacv iff $\delta(q_i, b) = (q_j, c, K)$ and the configuration uaq_ibv yields $uacq_jv$ iff $\delta(q_i, b) = (q_j, c, R)$
Standard Turing Machine: input starts at left most character; no blank characters to left of input; the left most end yields the same position if attempting to move left
Recognized Languages: languages recognized by TMs are Turing-Recognizable Languages (synonyms with semi-decidable/enumerable languages)
Decider: a turing machine that is guaranteed to halt for all inputs
Decidable Languages: a language decidable by a Turing machine

Subsets: Decidable $\subseteq$ Turing Recognizable $\subseteq$ Non TR Language
Closure Operations (TR): union, intersection, concatenation, star
Closure Operations (Decidable): union, intersection, concatenation, star, complement
Property: A language is decidable iff it is Turing recognizable and the complement is Turing recognizable

3.3 TM Variants

Definition: Variant of a TM able to be simulated by standard TM (via proof that standard can be simulated by variant and vice versa)
MultiTape TM: multiple tapes with transitions that specify movement for each tape
Two Way Infinite TM: tape that extends infinitely to the left and right
Other Variants: Nondeterministic TM, lambda calculus, recursively enumerable functions, unrestricted grammars, combinatory logic, programming languages

4 Decidability

Convention: input $< I >$ is an encoding suitable for putting on the TM tape
A_{DFA}: decides whether a TM can simulate a DFA for an input string; if the simulation ends in one of the DFA's accept states, accept; otherwise, reject; the TM can keep track of current state and input position by storing on tape and using the delta function to update current state and input position
A_{NFA}: decides whether a TM can simulate a NFA for an input string; can convert the NFA to a DFA and use the TM constructed for the DFA to decide
A_{RE}: decides whether a TM can simulate a RE for an input string; can convert the RE to a DFA and use the TM constructed for the DFA to decide
E_{DFA}: decides whether a TM can simulate DFA that only accept the empty set
EQ_{DFA}: decides whether two DFA are equivalent using $< (D_1 - D_2) \cup (D_2 - D_1) >$ and determining whether the result is empty (accept), or nonempty (reject)
A_{CFG}: decides whether a TM can simulate a CFG on an input string; convert the CFG into CNF and determine all derivations in $2* \mid w \mid -1$ steps (unless w is length 0, in which we complete 1 derivation). If any derivation yields w, accept (utilizes CNF theorem)
A_{PDA}: convert PDA into CFG to simulate PDA as a CFG above
E_{CFG} decides whether a TM can simulate empty CFGs; mark all terminals and mark non terminals A such that $A \rightarrow u_1 u_2 ... u_k$ such that each u_i has already been marked; if the start symbol is marked, reject; else, accept

5 Undecidability

Methodology: use proof by contradiction by assuming a certain problem is first decidable
Rice's Theorem: and property of a TM equivalent is undecidable