435 Notes

Introduction to The Theory of Computation Chap. 1

* **Learning Objectives** •Define the 3 basic concepts in the Theory of Computation:•Machine (Automaton, Turing Machine), •Formal Language (Regular language, Context‐Free language, etc.), and •Grammar (Regular grammar, Context‐Free grammar, etc.)•Evaluate expressions involving operations on strings and on languages.•Generate strings from simple grammars.•Construct grammars to generate simple languages.•Describe the essential components of an automaton.•Design grammars to describe simple programming constructs.
* **Mathematical Preliminaries**
* Sets: basic notation, operations (union, intersection, difference, and complementation), disjoint sets, power set, partitions.
* Functions and Relations: domain, range, total function, partial function, order of magnitude, equivalence relations.
* Graphs and Trees: vertices, edges, walk, path, simple path, cycle, loop, root vertex, parent, child, leaves, depth, height.
* Proof Techniques: proof by deduction, proof by induction, proof by contradiction.
* Proof by Deduction•A proof where a statement is proved to be true based on well‐known mathematical principles; i.e. establish facts through reasoningor make conclusions about a particular instance by referring to a general rule or principle.•It may use the algebraic symbols and construct logical arguments from known facts to show that something is true for all instances.•Example: Prove that the difference between the squares of any two consecutive integers is equal to the sum of those integers.Proof) Choose any two consecutive integers, nand n+1.Then, take the squares of these integers: n2and (n+1)2 = n2+ 2n+1.The difference between these squares is (n2+ 2n+1) ‐n2= 2n+1 (A)The sum of the original two consecutive integers is: n+ (n+1) = 2n+1 (B).Therefore, the given claim is true since the above (A) and (B) are equal. Q.E.D
* 8/27/20204Proof by Induction•A proof by which the truth of a number of statements can be inferred from the truth of a few specific instances.•Suppose we have a sequence of statements P1, P2, ... , and we want to prove Pkto be true, for all k ≥1. Suppose the following holds:1. For some k ≥1, the starting statement(s) P1, (P2, ..., Pk) are true.2. The problem is s.t.for any n≥ k, the truths of P1, P2,..., Pnimply the truth of Pn+1. Use induction to show that every statement in this sequence is true.•Base case: For some k ≥1, the starting statement(s) P1, (P2, ..., Pk) are true.•Inductive Hypothesis (I.H.): Assume that P1, P2, ..., Pn, n≥k ≥1 are true.•Inductive Step: Prove Pn+1is true using Inductive Hypothesis and Base case.Therefore, the given statement Pkis true for all k ≥1.
* Proof by Induction (cont.)•Example: A binary tree of height hhas at most 2hleaves. Proof by Induction) Let l(h) denote the maximum number of leaves of a binary tree of height h. Claim: Show that l(h)≤2h.Basis: h= 0. l(0) = 1 = 20since a tree of height 0 has a root only, i.e. it has at most one leaf. Thus, l(h)≤2hfor h=0.Inductive Hypothesis: Assume that l(h)≤2h is true for h = 0, 1, ..., n.Inductive Step: Let’s prove that a binary tree of height h+1 has at most 2h+1leaves, i.e. l(h+1)≤2h+1 .To get a binary tree of height h+1 from one of height h, we can create it by merging at most two binary trees TH, TR, of height h, adding a new root.Thus, l(h+1) = l(TH ) + l(TR) = l(h)+l(h) = 2⋅l(h).Hence,l(h+1) = 2⋅l(h) ≤2⋅2h = 2h+1by I.H. The claim is true for h+.Therefore, l(h)≤2h for all h ≥0. i.e. A binary tree of height h has at most 2hleaves for any height h. Q.E.D.
* Proof by Contradiction•A proof that determines the truth of a statement by assuming the proposition is false, then working to show its falsity until the result of that assumption is a contradiction.•A disproof by counterexample also belongs to it.•Example:Disprove that for any a, b∈Z, if a2= b2, then a= b.By CounterExample) Z is the set of all positive or negative integers. If an aand bs.t.a≠b but a2= b2 , then the statement is disproved.Choose any integer for a,then choose b= ‐a. Then, a2= b2 =(‐a)2 , but a ≠b(= ‐a). e.g.) a = 4, b = ‐4 ◊a2= b2 ⇔42= (‐4)2 = 16, but a≠b Thus, the given statement is false: ais not necessarily equal to b.Q.E.D
* Proof by Contradiction (cont.)•Example: For all integers n, if n3+5 is odd, then nis even.Proof) Let nbe any integer. Suppose that n3+5 and nare both odd. Then, there exist integers jand ks.t.n3+5= 2k+1 and n=2j+1.Substituting for nwe have:2k+1 = n3+5 = (2j+1)3+5= 8j3+ 3(2j)21 + 3(2j)(1)2+ 13+ 52k= 8j3+ 12j2+ 6j+ 5Dividing by 2 and rearrange it yields \* k‐4j3-6j2-3j=5/2\*\*‐‐impossible because 5/2 \*\* is a non‐integer rational number while \* is an integer by the closure properties for integer.Thus, the assumption ‘nis odd’ is false, i.e. nmust be even
* Formal Languages:Basic Concepts•Alphabet:a set of symbols, i.e. Σ= {a, b}•String:a finite sequence of symbols from Σ, such as v = aba and w = abaaa•So, any string u ∈∑\*•Empty string: λ, ε•Substring, prefix, suffix•Operations on strings:•Concatenation: vw = abaabaaa•Reverse: wR= aaaba•Repetition: v2= abaaba andv0= λ(empty string)•Length of a string: |v| = 3 and |λ| = 0
* Formal Languages: Property•Example 1.8: For the strings u, v, |uv| = |u| + |v|.Proof by Induction) First, let’s define the length of a string recursively:|a| = 1, |ua| = |u| + 1, for any a∈Σand any string u on Σ\*. Base case: For all uof any length and all vof length 1, i.e. |v|=1, |uv| = |u| + 1 = |u|+|v| where v∈Σ. Holds.Inductive Hypothesis (I.H.): Assume that |uv| = |u| + |v| for all uof any length and all vof length k ≤n, i.e. |v| ≤n.Inductive Step: For any vof |v|= n+1, rewrite vas v= wawhere |w| = n. Then, |v| = |w| + 1, |uv| = |uwa|= | uw| + 1.By I. H. , |uw| = |u| + |w| since |w|= n, so that|uv| = |uwa|= |uw| + 1 = |u| + |w| + 1 = |u| + |wa| = |u|+|v|.Therefore, |uv| = |u| + |v| for all uand vof any length. Q.E.D
* Formal Languages: Definitions•Σ\*= a set of all strings formed by concatenating zeroor moresymbols in Σ.•Σ+= a set of all non‐empty strings formed by concatenating symbols in Σ.In other words, Σ+= Σ\* ‐{ λ}•A formal languageis any subset of Σ\*Example 1.10: Σ= {a, b}L1= { anbn| n≥ 0 }andL2= { ab, aa }•A string in a language is also called a sentenceof the language.
* **Formal Languages: Set Operations**
* A language is a set of strings. Thus, set operations are defined as usual.
* If L1= { anbn| n ≥ 0 }andL2={ ab, aa } whereΣ= {a, b}
* Union:L1 ∪L2 = { aa, λ, ab, aabb, aaabbb, ... }•
* Intersection:L1 ∩L2 = { ab}
* Difference:L1 ‐L2 = { λ, aabb, aaabbb, ... } = { anbn| n = 0 orn ≥ 2 }
* Complement: 𝐿ଶൌΣ∗െL2 =Σ∗ െሼ𝑎𝑏,𝑎𝑎ሽ
* Find L2–L1 ?
* **Formal Languages: Other Operations**
* Reversal of all strings in a language:
  + LR= { wR| w∈L}
* Concatenation of strings from two languages, and
* L1L2 = { xy| x∈L1, y∈L2}
* Concatenation of strings from the same language:
* LL= L2= { xy| x∈L, y∈L}
* Star‐Closure:L\*= L0ᴜ L1 ᴜ L2 ᴜ L3 ᴜ ...whereL0 = { λ}, L1 = L, L2 = L⋅L, etc.
* Positive Closure: L+= L1 ᴜ L2 ᴜ L3 ᴜ ...
* **Example: Other Operations**
* If L1= { anbn| n≥ 0 } andL2= { ab, aa }
  + Reversal :L2R= { ba, aa }, L1R= { bnan| n≥ 0 }
  + Concatenation:L1L2 = { ab, aa, abab, abaa, aabbab, aabbaa, ... }
  + Concatenation:L2L2 = L22= { abab, abaa, aaab, aaaa}
  + Star‐Closure:L2\*= L20ᴜ L21 ᴜ L22 ᴜ L23 ᴜ ...
* Positive Closure: L2+= L21 ᴜ L22 ᴜ L23 ᴜ ...
* Find (L2–L1)R ?
* **Grammars: Definition**
* A rule to describe the strings in a language.
* In English grammar:
* <sentence> →<noun phrase> < predicate>,
* <noun phrase> →<article> <noun>,
* <predicate> →<verb>,
* <article> →a | the,
* <noun> →boy | dog,
* <verb> →runs | walks.
* Example: a boy walks, the dog runs.
* **Grammars: Definition**
* A rule to describe the strings in a language, i.e. a syntax of a language –not a semantics.
* Def. 1.1: A grammar G is defined as a quadruple G = (V, T, S, P)where V: a finite set of variable or non‐terminal symbols T: a finite set of terminal symbolsS (∈V): a variable called the start symbolP: a finite set of productions (i.e. rules)•Example 1.11:V = { S } T = { a, b }P = { S →aSb, S →λ}◊L(G) = { anbn| n≥ 1 } s->-1
* **Grammars: Derivation of Strings**
* Beginning with the start symbol, strings are derived by repeatedly replacing variable symbols with the expression on the right‐hand side of any applicable production.
* Any applicable production can be used, in arbitrary order, until the string contains no variable symbols.
* Sample derivation using grammar in Ex. 1.11:S →aSb, S →λS⇒aSb(applying 1stproduction)⇒aaSbb(applying 1stproduction)⇒aabb(applying 2ndproduction)
* **The Language generated by a Grammar**
* Def. 1.2: For a given grammar G=(V, T, S, P), the language generated by G, L(G) = { w∈T\* | 𝑆⇒∗𝑤}is the set of all strings derived from the start symbol.
* To show a language Lis generated by G: L =L(G)
* Show every string in L can be generated by G and∀w ∈L →∀w∈L(G).
* Show every string generated by G is in L.∀w ∈L(G) →∀w∈L.
* A given language can normally be generated by different grammars.
* **The Language generated by a Gramma**r
* For convenience, productions with the same left‐hand sides are written on the same line: S →A| B ⇔S →A, S →B
* Example 1.13: For a given grammar G=(V, T, S, P) with productions S→SS| λ| aSb| bSa,find L(G) = ?L(G) = { w| ? }
* **Equivalence of Grammars**
* Two grammars, G1and G2, are equivalent if they generate the same language: L(G1) = L(G2).
* For convenience, productions with the same left‐hand sides are written on the same line: S →A| B (= S →A, S →B)
* Example 1.11:G1= (V, T, S, P) where V = { S }, T = { a, b }, P = {S →aSb| λ }
* Example 1.14:G2= (V, T, S, P) where V = { A, S }, T = { a, b }, P = { S →aAb| λA →aAb| λ }G1and G2 are equivalent since L(G1) = L(G2).
* Automata
* An Automaton is an abstract mathematical model of a (von Neumann) digital computer.
* An automaton consists of
* An input mechanism
* A control unit
* Possibly, a storage mechanism
* Possibly, an output mechanism
* Control unit can be in any number of internal states, as determined by a next‐state or transition function.
* **Application: Grammars for Programming Languages**
* The syntax of constructs in a programming language is commonly described with grammars.
* Assume that in a hypothetical programming language,
* Identifiers consist of digits and the letters a, b, or c
* Identifiers must begin with a letter
* Productions for a sample grammar:<id> →<letter> <rest><rest> →<letter> <rest> | <digit> <rest> | λ<letter> →a| b| c<digit> →0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
* Ref.) CSci465. Principles of Translation. Compiler