CSI 3104 /Winter 2011: Introduction to Formal Languages

Nejib Zaguia
Winter 2011

- I. Theory of Automata
- II. Theory of Formal Languages
- III. Theory of Turing Machines

- CSI 3104 Introduction to Formal Languages (3 hours of lecture per week, 3 credits)
- Regular languages, finite automata, transition graphs, Kleene's theorem. Finite automata with output. Context-free languages, derivation trees, normal form grammars, pumping lemma, pushdown automata, determinism. Decidability. Recursively enumerable languages, Turing machines, the halting problem. Prerequisites: MAT1361, MAT2343 or MAT2143.

- PROFESSOR: Dr. Nejib Zaguia
  - SITE 5-031, 562-5800 ext.:6782
  - zaguia@site.uottawa.ca
  - Office Hours: Tuesday10:00-11:30
- LECTURES:
  - Tuesday 19:00-22:00 ??
- MANUEL:
  - Introduction to Computer Theory, Daniel Cohen, Wiley.
  - Course notes will be available on the web page of the course: www.site.uottawa.ca\~zaguia\csi3104

Course Outline:

- Introduction, Languages, Recursive Definitions, (Chapters 1, 2, 3)
- Regular Expressions, Finite Automata (Chapters 4, 5)
- Transition Graphs, Kleene's Theorem (Chapters 6, 7)
- Nondeterministic Finite Automata (Chapter 7), Finite Automata with Output, Regular Languages (Chapters 8, 9)
- Non-regular Languages, Decidability (Chapters 10,11)
- Context-Free Grammars, Grammatical Format, (Chapters 12, 13)
- Pushdown Automata (Chapter 14)
- Context-Free Grammars = Pushdown Automata, Chapter 15 (pages 318-327)
- Non-Context-Free Languages, Context-Free Languages, (Chapters 16, 17)
- Parsing, Turing Machines, (Chapters 18 "pages 402-410 and 415-423", 19)
- Recursively Enumerable Languages, (Chapter 23)
- Review

- Evaluation
  - Assignments, 25% (late assignments not accepted)
  - Midterm, 25% (week of February 14, will take place during class time, closed book)
  - Final Exam, 50%
  - There will be approximately 4 or 5 assignments.
  - To pass the course, you must obtain at least 50% on the final exam.

Two basic questions:

What is a computer good for?

What can & can't a computer compute? Why?

Want precise answers:

Formulate unambiguous questions & proofs, using formal models of computers/computation.

Using precise, mathematical writing

- Cantor (1845-1918) theory of sets
- Hilbert (1862-1943) methodology for finding proofs
- GÖdel (1906-1978) Incompleteness theorem
- Church, Kleene, Post, Markov, von Neumann, Turing
  Which statements have proofs?
  building blocks of mathematical algorithms
- Turing (1912-1954) Universal machine and its limitations
- McCulloch, Pitts Neural nets (similar but with different limitations)
- Chomsky mathematical models for the description of languages

- Theory of computers

  Study of mathematical models
  - Abstract
  - Simplify
  - Codify

  (relate in a meaningful way to the physical world)
- Computability

Language: Some set of strings of symbols, of interest.
- E.g., all valid English words.

Machine: A formal description of a "computer".
- Based on states & transitions between states.
- Computes some output from input.

Grammar: Rules for deriving & parsing strings in some language.

We will be interested in relations such as…

Which machines recognize which languages?

Does extending a class of machines with more features also extend the class of languages about to be described?

What machines are universal? I.e., sufficiently powerful to do "anything"?

Which languages require more complicated machines?

We will be interested in properties such as…

Decidability = What questions can be answered?

Computability= What languages can be computed?

Many computation models exist.

- Can't cover them all.

Will concentrate on 3 groups of models:

- Each proven pragmatically useful.
- Presented in order of increasing power.
    1. Regular languages & Finite automata
    2. Context-free languages & Push-down automata
    3. Recursively-enumerable languages & Turing machines

I. Automata II. Formal Languages III. Turing Machines

	Language Defined by	Corresponding Accepting Machine	Nondeter-minism= Determinism	Language Closed Under	What Can Be Decided?	Examples of Applications
I.	Regular expression	Finite automaton, transition graph	Yes	Union,product Kleene star, intersection, complement	Equivalence, emptiness, finiteness, membership	Text editors, sequential circuits, verification
II.	Context-free grammar	Pushdown automaton	No	Union, product, Kleene star	Emptiness, finiteness, membership	Parsing, compilers
III.	Type 0 grammar	Turing machine, Post machine, Pushdown automaton	Yes	Union, product, Kleene star	Not much	Computers

- Definitions
- alphabet – a finite set of symbols, denoted Σ
- letter – "characters" an element of an alphabet Σ
- word – a finite sequence of letters from the alphabet Σ
- Σ^* - the set of all words on Σ
- Λ (empty string) – a word without letters
- language – a set of words formed from the alphabet Σ (a subset of Σ^*)

Two examples

	English-Words	English-Sentences
alphabet	Σ ={a,b,c,d,…}	Σ =words in dictionary + space + punctuation marks
letter	letter	word
word	word	sentence
language	all the words in the dictionary	all English sentences

- It is very difficult to define the complete English language with a finite number of rules.
- We cannot too simply list all acceptable sentences.
  - Grammatical rules are not enough:
    - « I ate two Mondays »

- In general, the interesting languages have the following properties:
  - Well defined « without ambiguity ».
  - Using a formula, a property or a finite set of rules, We should be able to recognize in a finite time, whether any given word is in the language.

- Example L_1: Σ ={x}
- L_1={x, xx, xxx, xxxx,…} or $L_1=\{x^n \mid n=1, 2, 3,…\}$
- Σ^* = {Λ, x, xx, xxx, xxxx,…} $=\{x^n \mid n=0, 1, 2, 3,…\}$

We denote $x^0 = \Lambda$

L_2 = {w in Σ^*: w has an odd number of characters}
= {x, xxx, xxxxx, … }
$= \{x^n \mid n=1, 3, 5, … \}$

- Operations on Words:

length – number of letters in a word
length(xxxxx) = 5
length(1025)=4
length(Λ)=0

Σ= {0, 1}
L1 = set of all words in Σ^* starting with 1 and with length at most three
= {1, 10, 11, 101, 100, 110, 111}

reverse

reverse(xxx)=xxx
reverse(157)=751
reverse(acb)=bca

Example: PALINDROME

Σ={a, b}

PALINDROME:={Λ and w in Σ^*| reverse(w) = w}

= {Λ, a, b, aa, bb, aaa, aba, bbb, bab, ... }

Concatenation of two words – two words written down side by side. A new word is formed.

u=xx v=xxx uv=xxxxx
u = abb v=aa uv=abbaa

$u = u_1u_2...u_m$ $v=v_1v_2 \dots v_n$

$uv= u_1u_2...u_m v_1v_2 \dots v_n$

factor – one of the words in a concatenation

Property: length(uv) = length(u) + length(v)

- Concatenation of two languages-

The concatenation of two languages L_1 and L_2, L_1L_2, is the set of all words which are a concatenation of a word in L_1 with a word in L_2.

L_1L_2 = {uv: u is in L_1 and v is in L_2}

Example: Σ={0, 1}

L_1 = {u in Σ^*: the number of zeros in u is even}

L_2 = {u in Σ^*: u starts with a 0 and all the remaining characters are 1's}

L_1L_2 = {u in Σ^*: the number of zeros in u is odd}

- Closure of an alphabet Σ, Kleene star *

  Given an alphabet Σ, the closure of Σ (or Kleene star), denoted Σ^*, is the language containing all words made up of finite sequences of letters from Σ, including the empty string Λ.
- Examples:
  - $\Sigma = \{x\}$ $\Sigma^* = \{\Lambda, x, xx, xxx, \ldots\}$
  - $\Sigma = \{0, 1\}$ $\Sigma^* = \{\Lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
  - $\Sigma = \{a, b, c\}$ $\Sigma^* = ?$

- closure or Kleene star of a set of words (a language)

  It is a generalization of Σ^* to a more general set of words.

  Let Σ be an alphabet and let L be a set of words on Σ.

  L* is the language formed by concatenating words from L, including the empty string Λ.

  $L^* = \{u \text{ in } \Sigma^*: u = u_1u_2\ldots u_m$, where $u_1, u_2, \ldots, u_m$ are in L$\}$

- Examples:
  - L = {a, ab}

  $L^* = \{\Lambda, a, aa, ab, aaa, aab, aba, aaaa, \ldots\}$

  abaaababa $\in$ L* (ab|a|a|ab|ab|a factors)

  $L^* = \{\Lambda$ plus all sequences of a's and b's except those that start with b and those that contain a double b}

  Is the factoring always unique?

- L = {xx, xxx}
  xxxxxxx ∈ L*

xx|xx|xxx   xx|xxx|xx   xxx|xx|xx

L* = {Λ and all sequences of more than one x}
  = {x^n : x≠1}

If L = ϕ   then   L* = {Λ}

- The Kleene closure L*, of a language L, always produces an infinite language unless L is empty or L={Λ}.

- Example:
  S = {a, b, ab}   T = {a, b, bb}
  S* = T* although S ≠ T
  ab|a|a|ab|ab|a
  a|b|a|a|a|b|a|b|a

- Definition: L+, Σ+
  Le language with all concatenations that contain at least
    1 word from L
    1 letter from Σ
  (L* without Λ)

  If Λ is a member of L, L* = L+. Otherwise L* = L+ - {Λ}.

- Examples:
  - Σ = {x}   Σ+ = {x, xx, xxx, …}
  - S = {aa, bbb, Λ}   S+ = {aa, bbb, Λ, aaaa, aabbb, …}

  (N.B. aΛ = a)

Theorem 1: For any set of words S, we have S*=S**.

- Definitions:

  equality of sets S = T: S ⊂ T et T ⊂ S

  subsets S ⊂ T: for all x in S, x is also in T

- Example: S = {a,b}

  aaba, baaa, aaba ∈S*

  aaba|baaa|aaba ∈S**

  a|a|b|a|b|a|a|a|a|a|b|a ∈S*

- Proof of Theorem 1: S*=S** :
  - Case 1: S** ⊂ S*

  Every word in S** is made up of factors from S* (definition of Kleene star). Every word in S* is made up of factors from S. Therefore, every word in S** is also a word in S*. Thus S**⊂S*.

  - Case 2: S* ⊂ S**

  For any set A, we can show that A ⊂ A*. Let *w* be a word in A. Then *w* is certainly in A*. If we consider S* as our set A, we can conclude S* ⊂ S**.

  By definition of equality of sets, we have S* = S**.

RECURSIVE DEFINITIONS

- A new method to define languages: recursive definition

  3 steps:

1. Specify the basic words (base case).
2. Rules for constructing new words from ones already known (recursive case).
3. Declare that no word except those constructed by following rules 1 and 2 are in the language.

The same method could be used to define sets in general.

- Example

EVEN is the set of all whole numbers divisible by 2.

EVEN = {2n | n = 1, 2, 3, 4, …}

EVEN is defined by the rules:

1. 2 is in EVEN.
2. If x is in EVEN, x+2 is in EVEN.
3. The only elements in EVEN are the ones that are constructed by following rules 1 and 2.

- Theorem: 12 is in EVEN
  - Divisible by 2? Yes, 12/2 = 6.
  - 12 = 2n? Yes, n = 6.
- Rules of the recursive definition?
  - Rule 1: 2 $\in$ EVEN
  - Rule 2: x=2, 2+2 = 4 $\in$ EVEN
  - Rule 2: x=4, 4+2 = 6 $\in$ EVEN
  - Rule 2: x=6, 6+2 = 8 $\in$ EVEN
  - Rule 2: x=8, 8+2 = 10 $\in$ EVEN
  - Rule 2: x=10, 10+2 = 12 $\in$ EVEN

- Another equivalent recursive definition for the set EVEN
  - 2 is in EVEN.
  - If x and y are in EVEN, x+y is in EVEN.

Using the alternative definition

- Rule 1: 2 $\in$ EVEN
- Rule 2: x=2, y=2, 2+2 = 4 $\in$ EVEN
- Rule 2: x=4, y=4. 4+4 = 8 $\in$ EVEN
- Rule 2: x=4, y=8, 4+8 = 12 $\in$ EVEN

Example: Recursive definition of the set POLYNOMIAL

- All numbers are in POLYNOMIAL.
- The variable x is in POLYNOMIAL.
- If p and q are in POLYNOMIAL, p+q, p – q, (p), and pq are also in POLYNOMIAL.
- The only elements in POLYNOMIAL are the ones that are constructed by following rules 1, 2, and 3.

- Theorem: $5x^3$-8x+7 is in POLYNOMIAL
- Rule 1: 5 $\in$ POLYNOMIAL
- Rule 2: x $\in$ POLYNOMIAL
- Rule 3: 5x $\in$ POLYNOMIAL
- Rule 3: 5xx = $5x^2$ $\in$ POLYNOMIAL
- Rule 3: $5x^2x = 5x^3$ $\in$ POLYNOMIAL
- Rule 1: 8 $\in$ POLYNOMIAL
- Rule 3: 8x $\in$ POLYNOMIAL
- Rule 3: $5x^3 - 8x$ $\in$ POLYNOMIAL
- Rule 1: 7 $\in$ POLYNOMIAL
- Rule 3: $5x^3 - 8x + 7$ $\in$ POLYNOMIAL

Is this derivation unique?

OTHER EXAMPLES

- Σ+, Σ={x}
  - Rule 1: x $\in$ Σ+
  - Rule 2: If w is in Σ+, then xw is in Σ+
- Σ*, Σ={x}
  - Rule 1: $\Lambda \in \Sigma^*$
  - Rule 2: If w is in Σ^*, then xw is in Σ^*
- S-ODD, Σ={x}
  - Rule 1: x $\in$S-ODD
  - Rule 2: If w is in S-ODD then xxw is in S-ODD

- Kleene closure S* of a language S
  - Rule 1: Λ is in S*. All words in S are in S*.
  - Rule 2: If x and y are in S*, their concatenation xy is also in S*.

- POSITIVE
  - Rule 1: 1,2,3,4,5,6,7,8,9 are in POSITIVE
  - Rule 2: If w is in POSITIVE, w0, w1, w2, w3, w4, w5, w6, w7, w8, w9 are also words in POSITIVE

AE (Arithmetic Expressions)

- Σ = {0,1,2,3,4,5,6,7,8,9,+, –,*,/,(,)}
  - Rule 1: All numbers are in AE.
  - Rule 2: If x is in AE, the following words are also in AE:
    1. (x)
    2. – x (as long as x does not already start with a – sign)

- Rule 3: If x and y are in AE, the following words are also in AE:
  1. x + y (as long as y does not already start with a – sign)
  2. x – y (same condition)
  3. x * y
  4. x / y
  5. x ** y

Theorem 1: There is no word in AE that begins or ends with the symbol /.

- Proof:

1. No number contains the symbol /. So / is not introduced by rule 1.
2. Suppose that x does not begin or end with /. Neither (x) nor – x begin or end with /. (Rule 2)
3. Suppose that x and y do not begin or end with /. Then there is no expression introduced by rule 3 that begins or ends with /.

All words in AE are constructed by applying the rules 1,2,3.

Thus, no word in AE begins or ends with \.

- FORMULAS (Formulas of propositional logic)
  - Rule 1: All Latin letters are in FORMULAS.
  - Rule 2: If p is in FORMULAS, (p) et $\neg$p are also in FORMULAS.
  - Rule 3: If p and q are in FORMULAS, p $\rightarrow$ q is in FORMULAS.

Define operations (or functions) recursively

- Successor function
- Addition
- Product
- Factorial

- Function successor $\sigma : N \rightarrow N$
- D = {0, 1, 2, …, 9}

d	0	1	2	3	4	5	6	7	8	9
m(d)	1	2	3	4	5	6	7	8	9	0

(i) If v is in D then
If $v \neq 9$ then $\sigma (v) = m(v)$
If v= 9 then $\sigma (v) = 10$
(ii) If v = wd where d is in D then
- If $d \neq 9$ then $\sigma (v) = wm(d)$
- If d=9 then $\sigma (v) = \sigma (w)0$

CSI 3104 /Winter 2011: Recursive Definitions

- Addition of two positive integers:
  - (i) m + 0 = m for every positive integer m
  - (ii) $m + \sigma (n) = \sigma (m+n)$
- Product of two positive integers:
  - (i) m * 0 = 0 for every positive integer m
  - (ii) $m * \sigma (n) = m*n + m$

CSI 3104 /Winter 2011: Recursive Definitions

- The factorial function
  - Rule 1: 0! = 1
  - Rule 2: n! = n(n-1)!
- Recursive programs