CSE 322

Exam Reviews

Basic Concepts

- Formal Languages
  - Alphabet (Σ)
  - String (Σ*)
  - Length (|x|)
  - Empty String (ε)
  - Empty Language (∅)
- Language/String Operations
  - "Regular" Operations:
    - Union (∪)
    - Concatenation (•)
    - (Kleene) Star (*)
  - Other:
    - Intersection
    - Complement
    - Reversal
    - Shuffle
    - ...

Finite Defns of Infinite Languages

- English, mathematical
- DFAs
  - States
  - Start states
  - Accept states
  - Transitions (δ function)
  - M accepts w ∈ Σ*
  - M recognizes L ⊆ Σ*
- Nondeterminism
- NFAs
  - Transitions (δ relation)
    - Missing out-edges
    - Multiple out-edges
    - ε-moves
  - N accepts w ∈ Σ*
  - N recognizes L ⊆ Σ*
- Regular Expressions
  - ∅ , ε, a ∈ Σ, ∪, •, * , ( )
- GNFAs

Key Results, Constructions, Methods

- L is regular iff it is:
  - Recognized by a DFA
  - Recognized by a NFA
  - Recognized by a GNFA
  - Defined by a Regular Expr

Proofs:

GNFA → Reg Expr
(Kleene/Floyd/Warshall: R_{ik} R_{kk}* R_{kj})

Reg Expr → NFA
(join NFAs w/ ε-moves)

NFA → DFA
(subset construction)

DFA → GNFA
(special case)

- The class of regular languages is closed under:
  - Regular ops: union, concatenation, star
  - Also: intersection, complementation, (& reversal, prefix, no-prefix, …)
- NOT closed under ⊆, ⊇
- Also: Cross-product construction (union, …)

Applications

- "globbing"
  - lpr *.txt
- pattern-match searching:
  - grep "Ruzzo.*terrific" *.txt

- Compilers:
  - Id ::= letter (letter|digit)*
  - Int ::= digit digit*
  - Float ::= d d* . d* (ε | E d d*)
  - (but not, e.g. expressions with nested, balanced parens, or variable names matched to declarations)
- Finite state models of circuits, control systems, network protocols, API's, etc., etc.

Non-Regular Languages

- Key idea: once M is in some state q, it doesn't remember how it got there.

  E.g. "hybrids":
  if $xy \in L(M)$ and
  x, x' both go to q, then
  $x'y \in L(M)$ too.

  E.g. "loops":
  if $xyz \in L(M)$ and
  x, xy both go to q, then
  $xy^iz \in L(M)$ for all $i \geq 0$.

- Cor: Pumping Lemma
- Important examples:
  $L_1 = \{ a^nb^n \mid n > 0 \}$
  $L_2 = \{ w \mid \#_a(w) = \#_b(w) \}$
  $L_3 = \{ ww \mid w \in \Sigma^* \}$
  $L_4 = \{ ww^R \mid w \in \Sigma^* \}$
  L_5 = { balanced parens }
- Also: closure under ∩, complementation sometimes useful:
  - $L_1 = L_2 \cap a^*b^*$
- PS: don't say "Irregular"

Context-Free Grammars

- Terminals, Variables/Non-Terminals
- Start Symbol S
- Rules →
- Derivations ⇒, ⇒*
- Left/right-most derivations
- Derivation trees/parse trees
- Ambiguity, Inherent ambiguity

- A key feature: recursion/nesting/matching, e.g.

  $S \rightarrow (S)S \mid \varepsilon$

Pushdown Automata

- States, Start state, Final states, stack
- Terminals (Σ), Stack alphabet (Γ)
- Configurations, Moves, $\vdash$, $\vdash^*$, push/pop

Main Results

- Every regular language is a CFL
- Closure: union, dot, *, (Reversal; ∩ w/ Reg)
- Non-Closure: Intersection, complementation
- Equivalence of CFG & PDA
  - CFG ⊆ PDA : top-down(match/expand), bottom-up (shift/reduce)
  - PDA ⊆ CFG: A_{pq}
- Pumping Lemma & non-CFL's
- Deterministic PDA != Nondeterministic PDA

Important Examples

- Some Context-Free Languages:
  - $\{ a^n b^n \mid n > 0 \}$
  - $\{ w \mid \#_a(w) = \#_b(w) \}$
  - $\{ ww^R \mid w \in \{a,b\}^* \}$
  - balanced parentheses
  - "C", Java, etc.
- Some Non-Context-Free Languages:
  - $\{ a^n b^n c^n \mid n > 0 \}$
  - $\{ w \mid \#_a(w) = \#_b(w) = \#_c(w) \}$
  - $\{ ww \mid w \in \{a,b\}^* \}$
  - "C", Java, etc.

Curiously, their complements *are* CFL's

Applications

- Programming languages and compilers
- Parsing other complex input languages
  - html, sql, …
- Natural language processing/ Computational linguistics
  - Requires handling ambiguous grammars
- Computational biology (RNA)

Turing Machines & Decidability

- TMs
  - States, Σ, δ, etc.
  - 2-way, ∞, writable tape
  - q_{acc}, q_{rej} ; both halt
  - Recognizer: halt for "yes", but may reject by looping
  - Decider: always halts, yes/ no answer
- Church-Turing Thesis: this is as good a computer as any, wrt *what* is computable
- There are (many) problems that are *not* computable
  - About TMs: E.g., A_{TM} , $HALT_{TM}$: recognizable but not decidable
  - About other systems: E.g., ambiguity of CFGs
  - About programs
- Main proof techniques: diagonalization, reduction

The big picture

Ability to specifiy and reason about abstract formal models of computational systems is an important life skill. Practice it.