CSE 322

Exam Reviews
## Basic Concepts

### Formal Languages
- Alphabet ($\Sigma$)
- String ($\Sigma^*$)
- Length ($|x|$)
- Empty String ($\varepsilon$)
- Empty Language ($\emptyset$)

### Language/String Operations
- "Regular" Operations:
  - Union ($\cup$)
  - Concatenation ($\cdot$)
  - (Kleene) Star ($^*$)
- Other:
  - Intersection
  - Complement
  - Reversal
  - Shuffle
  - ...
Finite Defns of Infinite Languages

- English, mathematical
- DFAs
  - States
  - Start states
  - Accept states
  - Transitions (\(\delta\) function)
  - \(M\) accepts \(w \in \Sigma^*\)
  - \(M\) recognizes \(L \subseteq \Sigma^*\)
- Nondeterminism
- NFAs
  - Transitions (\(\delta\) relation)
    - Missing out-edges
    - Multiple out-edges
    - \(\varepsilon\)-moves
  - \(N\) accepts \(w \in \Sigma^*\)
  - \(N\) recognizes \(L \subseteq \Sigma^*\)
- Regular Expressions
  - \(\emptyset, \varepsilon, a \in \Sigma, \cup, \cdot, *, (, )\)
- 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:**

<table>
<thead>
<tr>
<th>GNFA</th>
<th>$\rightarrow$</th>
<th>Reg Expr</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\text{Kleene/Floyd/Warshall: } R_{ik} R_{kk}^* R_{kj})$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reg Expr</th>
<th>$\rightarrow$</th>
<th>NFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\text{join NFAs w/ } \varepsilon\text{-moves})$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NFA</th>
<th>$\rightarrow$</th>
<th>DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\text{subset construction})$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The class of regular languages is closed under:
  - Regular ops: union, concatenation, star
  - Also: intersection, complementation, ($\&$ reversal, prefix, no-prefix, … )

- **NOT closed under** $\subseteq, \supseteq$

- 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* ( \( \varepsilon \mid E \ d \ d^* \) ) 
  – (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 ∈ L(M) and x, x’ both go to q, then x’y ∈ L(M) too.

  E.g. “loops”:
  if xyz ∈ L(M) and x, xy both go to q, then xy^i z ∈ L(M) for all i ≥ 0.

• Cor: Pumping Lemma

• Important examples:
  \[ L_1 = \{ a^n b^n \mid n > 0 \} \]
  \[ L_2 = \{ w \mid \#_a(w) = \#_b(w) \} \]
  \[ L_3 = \{ w w \mid w \in \Sigma^* \} \]
  \[ L_4 = \{ w w^R \mid w \in \Sigma^* \} \]
  \[ L_5 = \{ \text{balanced parens} \} \]

• 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.

\[ S \rightarrow (S)S | \varepsilon \]
Pushdown Automata

- States, Start state, Final states, stack
- Terminals ($\Sigma$), Stack alphabet ($\Gamma$)
- Configurations, Moves, |--, |--*, push/pop
Main Results

• Every regular language is a CFL
• Closure: union, dot, *, (Reversal; $\cap w/ \text{Reg}$)
• Non-Closure: Intersection, complementation
• Equivalence of CFG & PDA
  – CFG $\subseteq$ PDA:
    top-down(match/expand), bottom-up (shift/reduce)
  – PDA $\subseteq$ CFG: $A_{pq}$
• Pumping Lemma & non-CFL’s
• Deterministic PDA $\neq$ Nondeterministic PDA
Important Examples

• Some Context-Free Languages:
  – \{ a^n b^n \mid n > 0 \}
  – \{ w \mid \#_a(w) = \#_b(w) \}
  – \{ w w^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) \}
  – \{ w w \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)
The big picture

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