Chap. 1 Regular Languages

- **Language** = set of strings from some alphabet
- Language $L$ is **accepted by** FA $M$ if after last symbol both:
  - For any string in $L$, $M$ ends in accept state
  - For any string not in $L$, $M$ does not end in accept state
- **Regular Language (RL)**: any language accepted by a DFA
  - Also called **Regular Expressions (regex)**
- Question: Is there a way of describing all, and only, languages accepted by a FA? I.e. is there a syntax for RLs?
  - Can we build “larger” languages from “smaller” ones?
- Answer to all above: YES
- (p.44) Possible set operations on languages $A$ and $B$:
  - **Union**: $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$
  - **Intersection**: $A \cap A B = \{ x \mid x \in A \text{ and } x \in B \}$
  - **Complementation of $B$ wrt $A$**: $A / B = \{ x \mid x \in A \text{ and } x \not\in B \}$
- (p.44) Possible operations on strings in languages
  - **Concatenation**: $A \circ B = \{ xy \mid xy \text{ a string where } x \in A \text{ and } y \in B \}$
  - **Star**: $A^* = \{ x \mid x = \varepsilon \text{ or } x = x_1 x_2 \ldots x_k \text{ where } k \geq 1 \text{ and all } x_k \in A \}$
  - **Plus**: $A^+ = \{ x \mid x = x_1 x_2 \ldots x_k \text{ where } k \geq 1 \text{ and all } x_k \in A \}$
- P. 45 Examples of above operations on some simple sets
• Fundamental Question: if we apply any of above operations to known RLs, are we guaranteed to get another RL?
  • Are we guaranteed we can build an FA that accepts result
  • Answer: YES if set of RLs is closed under the operation
• **Closure**: A set is closed under some operation if applying it to any member(s) of the set returns another member of set
  • i.e. can we build a FA (DFA or NFA) that accepts any language created by applying specified operation
  • Typical proof process: by construction
    • Assume language A1 accepted by FA M1, A2 by M2:
    • Show how to build an M (typically using M1 and M2 as pieces) that accepts all strings from any combination of sets A1 and A2 using that operation
  • i.e **Set of all RLs is closed under these operations**
  • Assume following in closure proofs
    • A1 accepted by DFA M1, and M1 = (Q1, Σ, δ1, q1, F1)
    • A2 accepted by DFA M2, and M2 = (Q2, Σ, δ2, q2, F1),
    • Q1 ∩ Q2 = φ (i.e. no common states)
      • We can always “rename” states to prevent confusion
(p.45,46) Prove closure under \( U \) by constructing new DFA \( M \)

- Construct \( M = (Q, \Sigma, \delta, q_0, F) \)
  - \( Q = Q_1 \times Q_2 \)
    - i.e. states in \( M \) are “named” as tuples \((r_1, r_2)\)
      - \( r_1 \) in \( Q_1 \), \( r_2 \) in \( Q_2 \)
  - \( \Sigma \) same for all 3 machines
  - \( \delta( (r_1,r_2), a) = ( \delta_1(r_1,a), \delta_2(r_2,a) ) \)
  - \( q_0 = (q_1, q_2) \)
  - \( F = \{ (r_1, r_2) | r_1 \epsilon F_1 \text{ or } r_2 \epsilon F_2 \} \)

- New machine keeps track of states of both machines
  - If either ends up in their \( F \), then accept
  - If neither accept, then reject

- Do example: \( A_1 = \) set of even # of a’s, \( A_2 = \) odd # of b’s

(p.47) To show we’ve proven closure, must show:
  - If \( w \) is accepted by either \( M_1 \) or \( M_2 \), it is accepted by \( M \)
  - If \( w \) is accepted by \( M \), it is accepted by either \( M_1 \) or \( M_2 \)
  - Both of above are fairly obvious by construction

- Proof of closure under intersection is simple: change \( F \)!
• Since DFAs=NFAs, L is regular iff accepted by some NFA
• (p. 59-60) **Alternative construction proof of U** using NFAs
  • A1 accepted by NFA N1, and N1 = (Q1, ∑, δ1, q1, F1)
  • A2 accepted by NFA N2, and N2 = (Q2, ∑, δ2, q2, F1),
  • Construct N = (Q, ∑, δ, q0, F) to recognize A1 U A2
    • Q = {q0} U Q1 U Q2
    • F = F1 U F2
    • δ(q,a) =
      • = δ1(q, a) if q ε Q1
      • = δ2(q, a) if q ε Q2
      • = {q1, q2} if q = q0 and a = ε
      • = Φ if q = q0 and a ≠ ε
  • New starting state q0 “guesses correctly” which other machine to start – without looking at any input
• Proving ◦ or * is “harder” – we don’t know when to stop string from one language and start other!
• Really need nondeterminism!
• (p. 60) Proof that RLs are **closed under concatenation**
  • See Fig. 1.48 on p. 61
  • ε edge from each final state of N1 to start state of N2
  • N “guesses” when to hop from N1 to N2
  • A1 accepted by NFA N1, and N1 = (Q1, ∑, δ1, q1, F1)
  • A2 accepted by NFA N2, and N2 = (Q2, ∑, δ2, q2, F1)
  • Construct N = (Q, ∑, δ, q0, F) to recognize A1 ∘ A2
    • Q = Q1 U Q2
    • q0 = q1 (from N1)
    • F = F2 (from N2)
    • δ(q,a) =
      • = δ1(q, a) if q ∈ Q1 and q not in F1
      • = δ1(q, a) if q ∈ F1 and a ≠ ε
      • = δ1(q, a) U {q2} if q ∈ F1 and a = ε
      • = δ2(q, a) if q ∈ Q2 and any a
• (p. 62) Proof that RLs are **closed under Kleene star**
  
  • See Fig. 1.50 on p. 62
    • Add $\varepsilon$ edge from each final state back to start
    • Again guess correctly when to restart $N_1$
  
  • $A_1$ accepted by NFA $N_1$, and $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  
  • Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1^*$
    
    - $Q = \{q_0\} \cup Q_1$
    - $q_0$ = a new state
    - $F = \{q_0\} \cup F_1$
      - $\{q_0\}$ for empty set when 0 copies
    - $\delta(q,a) =$
      - $= \delta_1(q, a)$ if $q \in Q_1$ and $q$ not in $F_1$
      - $= \delta_1(q, a)$ if $q \in F_1$ and $a \neq \varepsilon$
      - $= \delta_1(q, a) \cup \{q_1\}$ if $q \in F_1$ and $a = \varepsilon$
      - $= \{q_1\}$
      - $= \emptyset$ if $q = Q_0$ and $a \neq \varepsilon$
• (p63 – Section 1.3) Regular Expressions

• Example: describing arithmetic expressions:

  <op1> -> + | -
  <op2> -> * | /

  <factor> -> <number> | (<arith-expr>) | <factor>^<factor>
  <term> -> <factor> | <term> <op2> <factor>
  < arith-expr > -> <term> | < arith-expr > <op1> <term>

  • Notice this defines a precedence for operators:
    • Do inside () first
    • Do * or / first before + or –
    • Do + or - last

• (p. 64) Describing regular expressions R (no precedence)

  <regex> -> φ | ε | ... any member of Σ ...

  | ( <regex> U <regex> )
  | ( <regex> ◦ <regex> )
  | ( <regex>* )

  • Note: this demands () all the time
  • No assumed precedence
  • Normal Precedence rules – drop unnecessary ()
    • Do inside () first
    • Do * first, then ◦, then U

• Examples p. 65 Example 1.53
• Redo of BNF to “build-in” precedence

\[ \langle \text{basic-regex} \rangle \rightarrow \phi \mid \varepsilon \mid \ldots \text{any member of } \Sigma \ldots \]

\[ \langle \text{regex-factor} \rangle \rightarrow \langle \text{basic-regex} \rangle \mid ( \langle \text{regex} \rangle ) \]

\[ \mid \langle \text{regex-factor} \rangle^* \]

\[ \langle \text{regex-term} \rangle \rightarrow \langle \text{regex-factor} \rangle \]

\[ \mid \langle \text{regex-term} \rangle \circ \langle \text{regex-factor} \rangle \]

\[ \langle \text{regex} \rangle \rightarrow \langle \text{regex-term} \rangle \]

\[ \mid \langle \text{regex} \rangle \cup \langle \text{regex} \rangle \]