Fundamental Question about R.L.

Q: Given a string w and a language L, does w ∈ L? (Membership question)

For R.L., standard representations: for L is

F.A., R.E. or R.G.

Thus: Given a standard representation; for any R.L. on Σ and any w ∈ Σ*, ∃ an algorithm to answer Q above.

Proof: Use M(L) to test

Thus: ∃ algo. to determine if a R.L. is empty, finite on

Proof: If ∃ a simple path from initial to a final state ⇒

Find all the vertices that are the base L * ≠ φ

If any of these are on any path of some cycle. If any of these are on any path from initial to any final state, then R.L. is infinite, otherwise finite.

Equality of Languages.

For L₁, L₂ ∈ R.L., ∃ algo to assess if L₁ = L₂

Proof:

L₃ = (L₁ ∩ L₂) ∪ (L̅₁ ∩ L₂) (by closure L₂ is regular + we can get its DFA)

L₃ = φ iff L₁ = L₂

Identifying non-regular languages

Regular language can represent as L using finite # of states.

Pigeonhole Principle

For m boxes, n objects, if n > m, ∃ one box that contains more than one object.

L = {aⁿbⁿ : n ≥ 0} is not regular

| L₀ = δ*(S₀, a) |
| δ*(S₀, a) = φ, δ*(S₀, aⁿ) = φ (and n ≠ m) |
| and δ*(S₀, bⁿ) = φ for b ∈ Σ, but this means |
| aⁿbⁿ ∈ L (a contradiction). |
| aⁿbⁿ ∉ L (a contradiction). |
A pumping lemma

→ (A transition graph on k vertices will have cycles
  for cycles of length more than n)

→ string in L can be broken into 3 parts with middle
  part pumped (repeated).

Thus: L is an \( \infty \)-R.L. \( \Rightarrow \) \( \exists \) \( n > 0 \) \( \in \mathbb{I}^+ \) s.t

\[ \forall w \in L, |w| \geq n, \quad w = xy^2z \]

with \( |xy| \leq n \); \( |y| \geq 1 \)

\[ s.t. \ y^i \in L \quad \& \ i = 0, 1, 2, \ldots \]

Direct consequence of pigeon-hole principle.

\( \therefore \) show \( L = \{a^n b^n ; n \geq 0 \} \subseteq \mathbb{R.L} \)

\( \therefore \) show \( L = \{w w^R ; w \in \mathbb{E}^* \} \subseteq \mathbb{R.L} \quad \& \ \exists = \{a, b\} \)

\( \therefore \) show \( L = \{w w^R ; w \in \mathbb{E}^* \} \subseteq \mathbb{R.L} \quad \& \ \exists = \{a, b\} \)

Given \( a \ddots b \ddots a \)

\[ a \cdots a \quad b \cdots b \quad a \cdots a \]

\[ x^y \]

Here again \( y \) will have only \( a \).

\( \therefore \) take \( i > 0 \), and we will get a

string \( \in L \).
Content Free Languages (C.F.L)
(related to programming languages)

RG = left side single var., rt. side special structure
CFG " " " " " " free

\[ G = (V, \Sigma, S, P) \in \text{CFG}, \text{ if } P \]
\[ A \to \alpha \]
\[ A \in V, \alpha \in (V\Sigma)^* \]
CFL is \( L(G) \)

Every RG is also CFG, \( : L(RG) \) is also \( L(CFG) \Rightarrow L(RG) \subseteq L(CFG) \)

(1) \( G: S \to aSb \) \( \Rightarrow L(G) = \{a^n b^n : n \geq 0\} \)

(2) \( G: S \to aSa \)
\( S \to aSb \)
\( S \to \lambda \)
\( L(G) = \{ww^R : w \in \{a, b\}^*\} \)

Derivation of language from Grammar

Take an example of a non-linear grammar
\( G: \]
1) \( S \to AB \)
2) \( A \to aaA \)
3) \( A \to \lambda \)
4) \( B \to bB \)
5) \( B \to \lambda \)

Consider (leftmost derivation): leftmost variable is replaced every

\[ S \Rightarrow AB \Rightarrow aaAB \Rightarrow aAB \Rightarrow aAb \]
(rightmost)

\[ S \Rightarrow AB \Rightarrow ABb \Rightarrow aaABB \Rightarrow aaAb \Rightarrow aab \]

Derivation Tree