Formal Languages & Automata Theory

Module 3 - Myhill Nerode Relations & Context Free Grammars

Myhill Nerode Relations

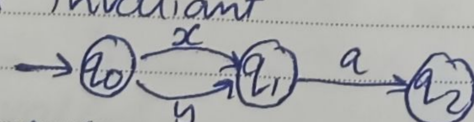
Let $L \subseteq \Sigma^*$ be a regular language and $M(Q, \Sigma, \delta, q_0, F)$ be a DFA for L, with no inaccessible states

$$\boxed{xRy \Rightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)}$$

Can be partitioned into classes called equivalence classes.
Index = no. of equivalence classes.
$\leq$ no. of states in FA.

Properties of MNR

1) Reflexive : xRx
2) Symmetric: $xRy \Rightarrow yRx$
3) Transitive: $xRy, yRz \Rightarrow xRz$
4) Right congruence / Right invariant
$xRy \Rightarrow xaRya$



$\hat{\delta}(q_0, xa) = \delta(\hat{\delta}(q_0, x), a)$
$= \delta(\hat{\delta}(q_0, y), a) = \hat{\delta}(q_0, ya)$

5) Refines L from Σ^* : $xRy \Rightarrow x \in L \Leftrightarrow y \in L$
6) Finite index : $\{x \in \Sigma^* \mid \hat{\delta}(q_0, x) = q\}$

DFA → MNR

1) No. of states
2) Equivalence classes for each state.

Myhill - Nerode Theorem

The following statements are equivalent:
1) $L \subseteq \Sigma^*$ is accepted by a DFA
2) L is union of some equivalence classes of a right invariant equivalence relation of finite index.
3) Let R_L be defined by $(x, y \in \Sigma)$, xR_Ly iff for all z in Σ^*, xz is in L when yz is in L; and of finite index.

Applications of MNT

1) To minimize DFA (Table-filling method)
2) To prove whether a language is regular or not.

Context-Free Grammar

$G = (V, T, P, S)$

$A \rightarrow x$, $A \in V$, $x \in (V \cup T)^*$

V- set of variables
T- set of symbols
P- production rules
S- Start symbol.

For regular grammar – only one variable on RHS
For CFG – More than one variable on RHS.

To check whether a string belongs to a Grammar

1) Start from start symbol, choose closest production matching the given string.
2) Replace variables with appropriate production. Repeat until string is generated or not productions left.

Derivation Tree: Ordered rooted tree graphically representing semantic info. of strings formed from CFG.
Root - Start symbol Leaves: Terminals or ϵ
Vertex - Non-terminals

left derivation tree	Right derivation tree
↓	↓
Applying production to leftmost variable	Applying production to rightmost variable.

Ambiguous Grammar: If there exists two or more derivation tree (two or more left or right) for a string w.

Simplification of CFG

1) Reduction

Phase 1: Remove variables that do not give a terminal symbol
Phase 2: Remove variables that are unreachable from start symbol.

2) Removal of unit productions

1) Choose productions of type $A \rightarrow B$
2) Add $A \rightarrow x$ if $B \rightarrow x$, then remove $A \rightarrow B$
3) Repeat until all unit productions are removed

3) Removal of Null productions

1) To remove $A \to \epsilon$, check the production which have A in RHS.
2) Replace A with ϵ on each of those productions
3) Add these to grammar.

Chomsky Normal Form

$$\boxed{\begin{array}{l} A \to a \\ A \to BC \end{array}}$$

CFG → CNF

1) Introduce $S' \to S$, where S - start symbol
2) Remove null productions ($A \to \epsilon$)
3) Remove unit productions ($A \to B$)
4) Introduce $X \to BA$ if $S \to ABA$ & Change it to $S \to AX$
5) Introduce $Y \to b$ if $S \to bA$ & change it to $S \to YA$

Greibach Normal Form

$$\boxed{\begin{array}{l} A \to b \\ A \to bC_1C_2 \ldots C_n \end{array}}$$

CFG → GNF

1) Remove null & unit productions.
2) Check if it is in CNF.
3) Non-terminals into A_i in ascending order of i
4) $A_i \to A_j x$, $i<j$
   If $i \geq j$, replace A_j with its production
   If it gives production of the form $b A_1 A_2 \ldots A_n$ - GNF.
   Else, there is left recursion. ($i \geq j$, no terminal in front)

5) If there are two or more variables on RHS, bring it to $A \to bX$ form. X contains any no. of variables.