Equivalence of FA

→ Create sets of both FA

Any set of Non Equivalent

make sure (F, NF) → should not be created

→ If all sets are processed that means Equivalence is found

F, NFA F, PA NF, NPA

Consider the following two DFAs M and M' over {0, 1} given in Fig. 5.23. Determine whether M and M' are equivalent.

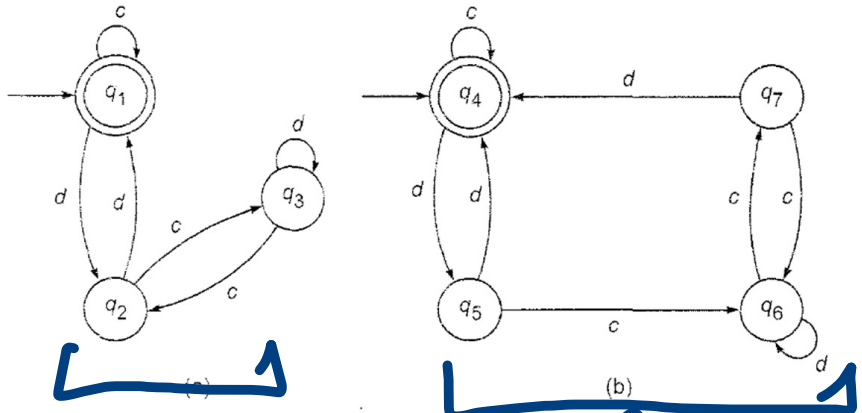


Fig. 5.23 (a) Automaton M and (b) automaton M'.

A B

	c	d
→ q_1, q_4 (F, F)	$\{q_1, q_4\}$ (F)	$\{q_2, q_5\}$ (NF)
q_2, q_5	$\{q_3, q_6\}$ (NF)	$\{q_1, q_4\}$
q_3, q_6	$\{q_2, q_7\}$ (NF)	$\{q_3, q_6\}$
q_2, q_7	$\{q_3, q_6\}$	$\{q_1, q_4\}$ (F)

⇒ No Combination as dead Combination

→ Every Combination of NF, NF

↳ Both A & B are Equivalent

Tuples of Grammar

→ Non Terminals ⇒ Variables ⇒ Generating Rules

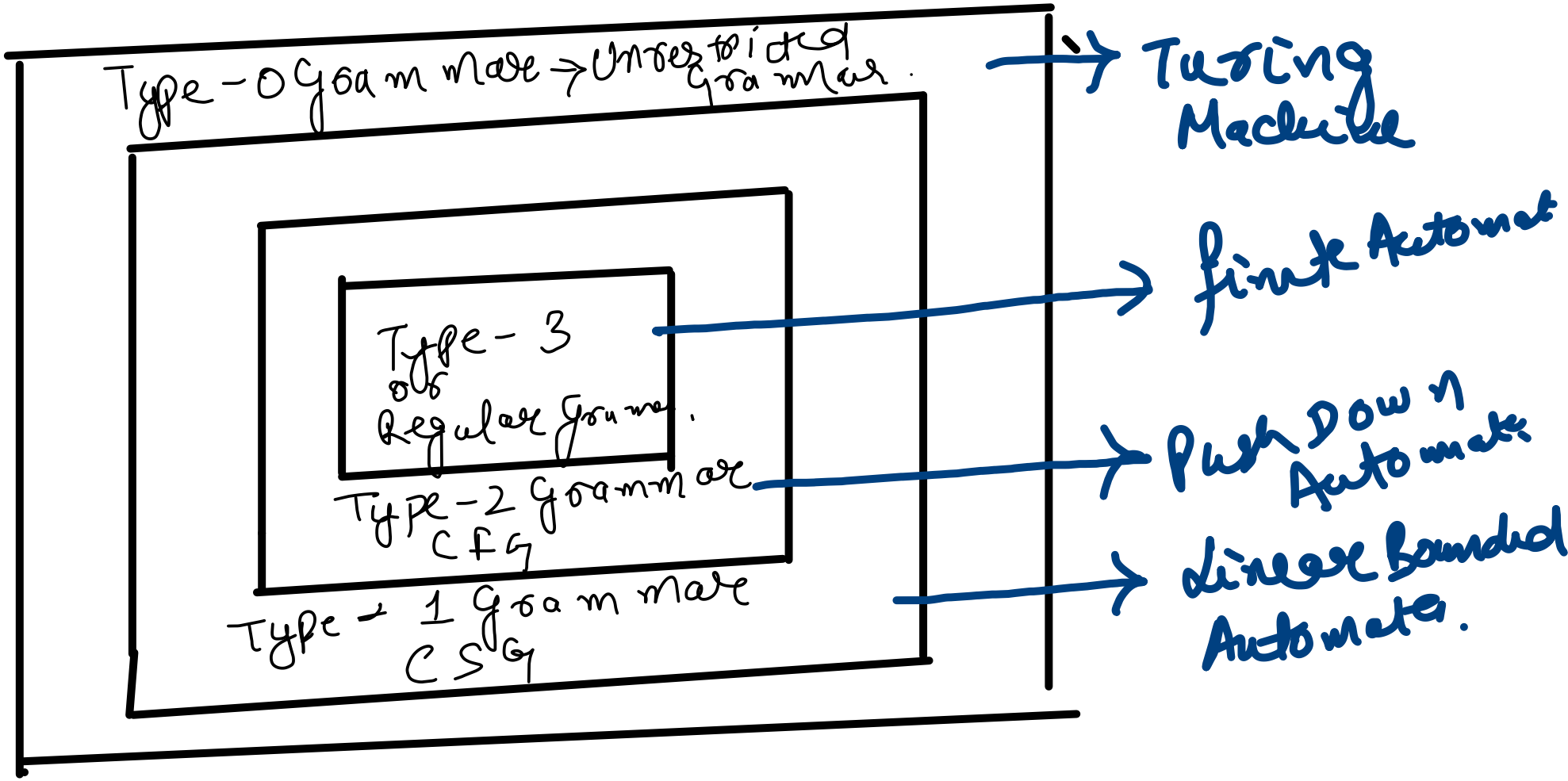
→ Terminals + o/p symbols

→ Production Rule ⇒ Generating strings

→ Starting symbol ⇒ NonTerminal

⇒ starting of a string

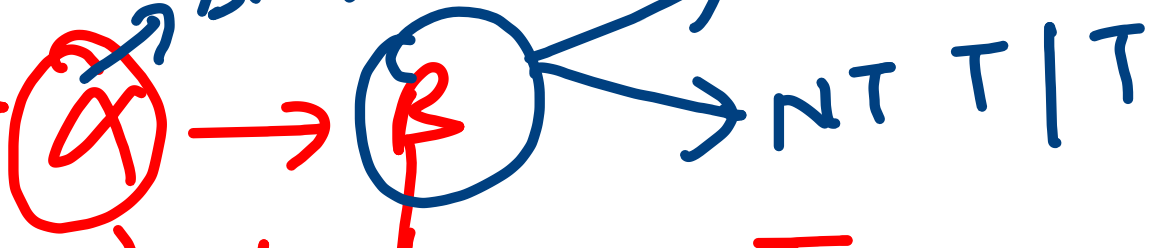
Chomskey Hierarchy of Grammar



1) Type-3 Grammar (Regular Grammar)
(Finite Automata)

- Most Restricted form. Part of Grammar

Productions Rules



1) $NT \rightarrow \underbrace{T}_{\text{single}}\ \underbrace{NT}_{\text{single}}$

2) $NT \rightarrow \underbrace{NT}_{\text{single}}\ \underbrace{T}_{\text{single}}$

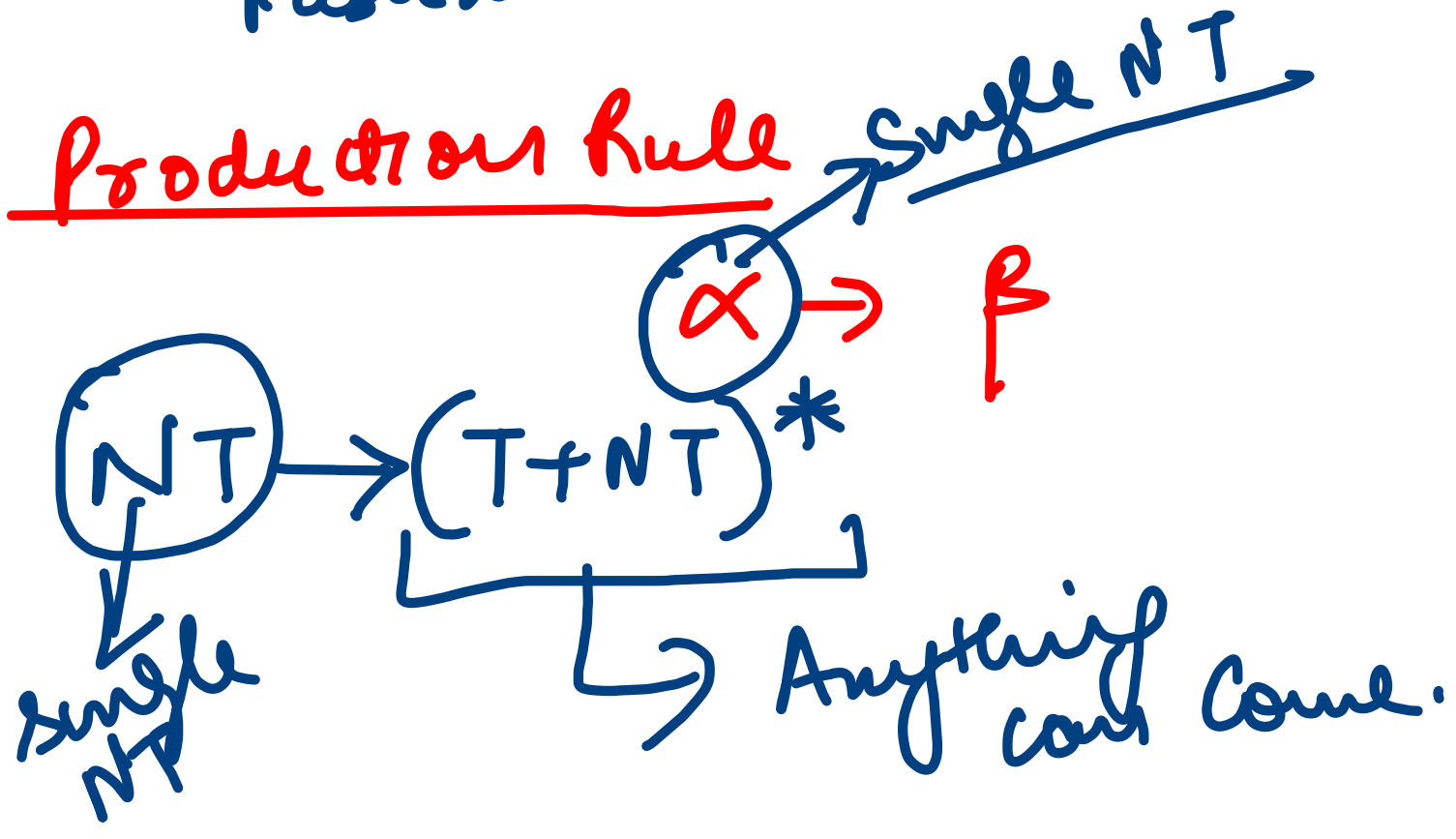
$S \rightarrow aS$ or $S \rightarrow Sa$

2) Context free Grammar (Type -2)

↓

Push Down Automata.

Production Rule



3) Type-1 Grammar or Context Sensitive Grammar → linear Bounded Automata

Restriction (PR)

$\alpha \Rightarrow \beta$

$3 < 2$

$NTTNT \rightarrow NTT$

1 2 3

$\alpha \rightarrow (T+NT)^* \underline{NT} (T+NT)^*$

$\beta = (T+NT)^*$ (Anything on R.H.S)

Restriction: length of $\alpha \leq \beta$ (length of)

4) Unrestricted Grammar (Type-0 grammar)

Turing Machine

$\alpha \rightarrow \beta$

Production Rule

α must contain a single NT

$\alpha \neq \wedge$

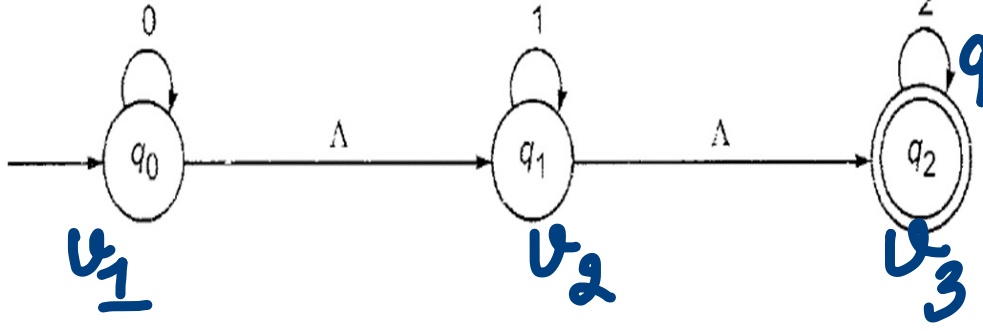
$\alpha \Rightarrow (T+NT)^* NT (T+NT)^*$

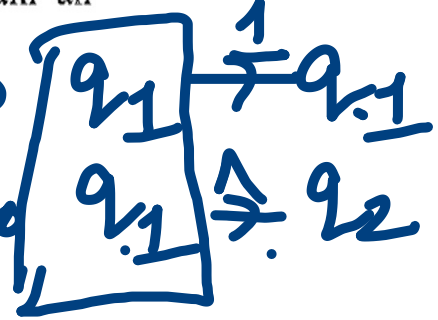
$\beta = (T+NT)^*$

Single NT must be present

$v_1 \overset{\wedge}{\Rightarrow} v_2 \quad v_2 \overset{\wedge}{\Rightarrow} v_3$

Consider a finite automaton, with Λ-moves, given in Fig. 5.1. Obtain an equivalent automaton without Λ-moves.





4 Steps

a. Remove the null move ($v_1 \overset{\wedge}{\Rightarrow} v_2$)

b. Copy what v_2 is doing (outgoing transitions of v_2) to v_1

c. If v_1 is initial state, make v_2 initial

d. If v_2 " final state " v_1 final.

1- Remove Null move

2. Copy what a_2 is doing to a_1

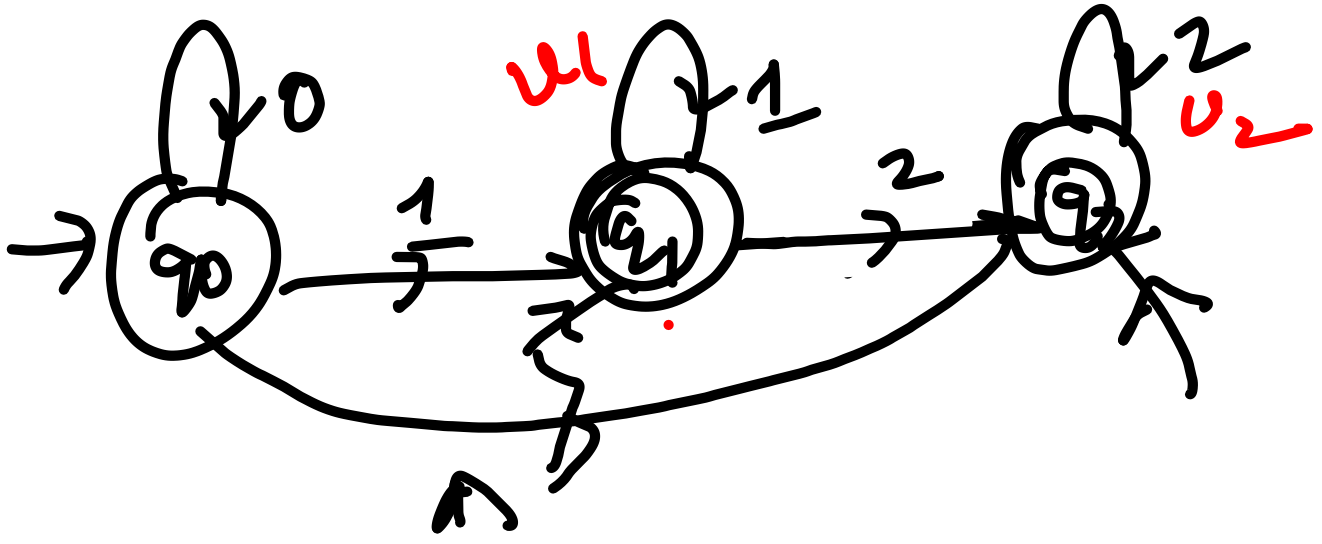
$q_0(1) = q_1$

$q_0(\wedge) = q_2$

	1	^
q_0	q_1	q_2

3- If a_1 is initial, make a_2 also initial

4- If a_2 is final, " a_1 — final.



$q_1 \hat{\rightarrow} q_2$ ✓

$q_0 \hat{\rightarrow} q_2$

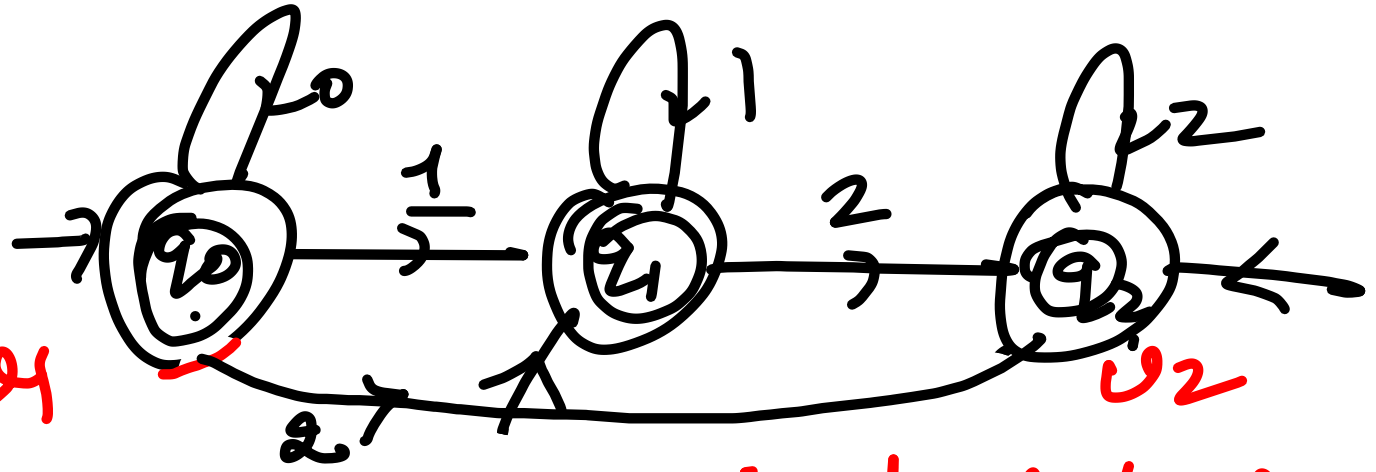
Copy what u_2 is doing to u_1

$q_1 \boxed{q_2}(2) = q_2$

$q_1(2) = q_2$

If q_1 is initial, q_2 becomes initial

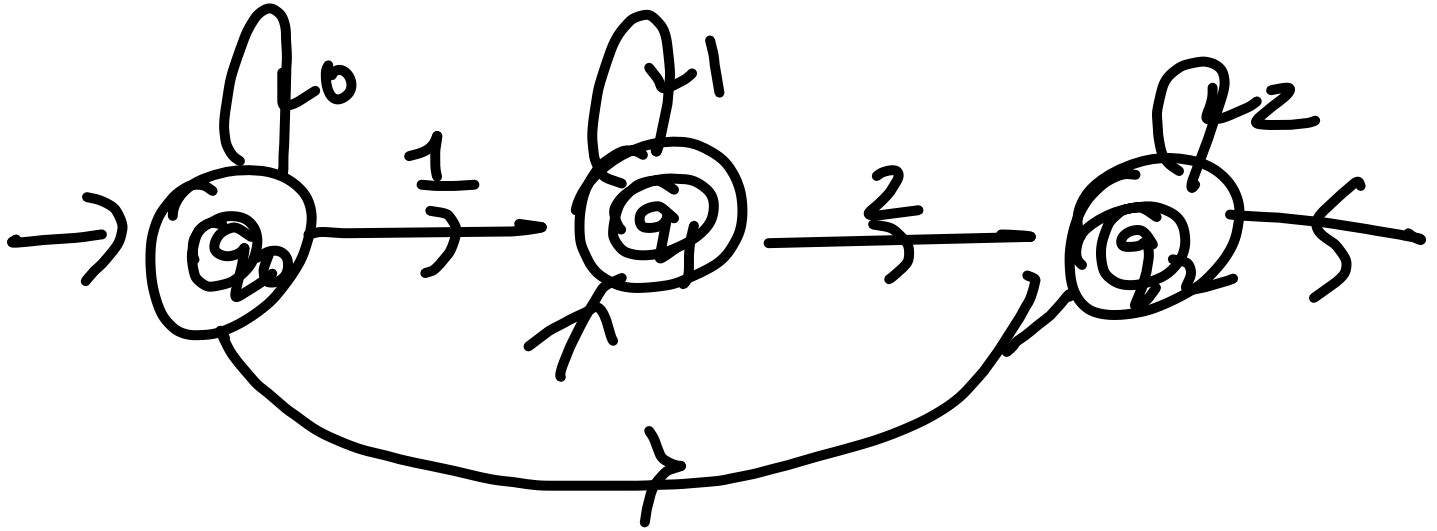
q_2 — final q_1 — final.



Copy what q_2 is doing to q_1

$$q_2(2) = q_2$$
$$q_0(2) = q_2$$

If q_2 is final make q_0 final



&

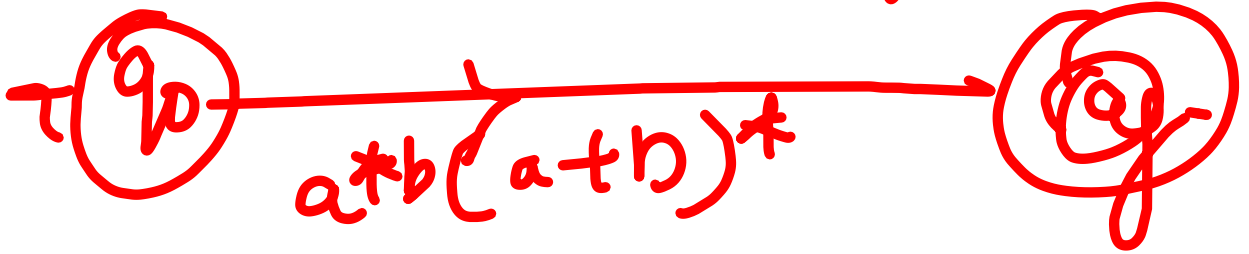
NFA

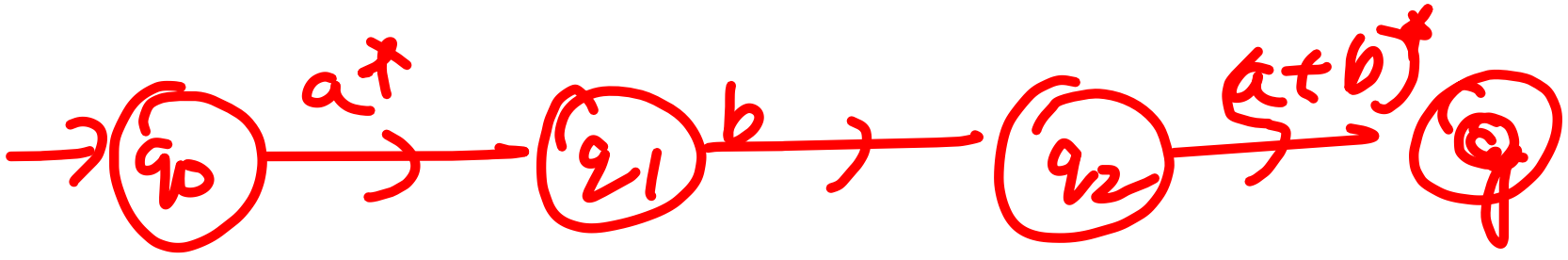
3 Intial States

Regular Expressions to Regular Grammar

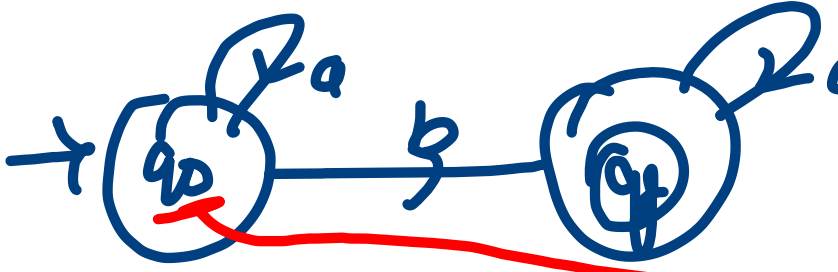
RE = $a^* b(a+b)^*$

Convert R-E to finite Automata





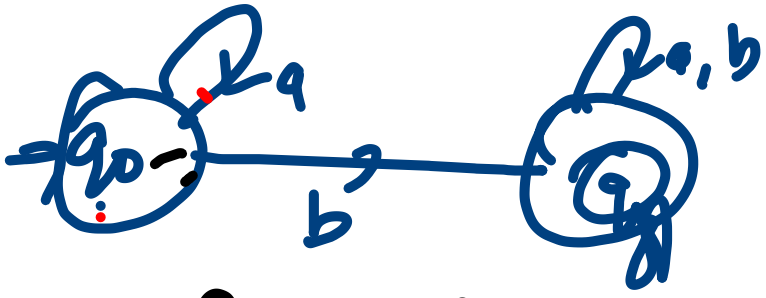
→ finite Automata



Non Terminals ⇒ States

R-G has 4 types

1) NonTerminals (variables) ⇒ {q_0, q_f} ⇒ States
2) Terminals (I/P symbols) ⇒ a, b
3) Production Rules
4) Starting Symbol ⇒ Non Terminal
(Starting state) ⇒ q_0 (Starting symbol)



Production Rules

- Transitive Rules: $q_0 \to a q_0$, $NT \to T\ NT$
- Terminating Rules

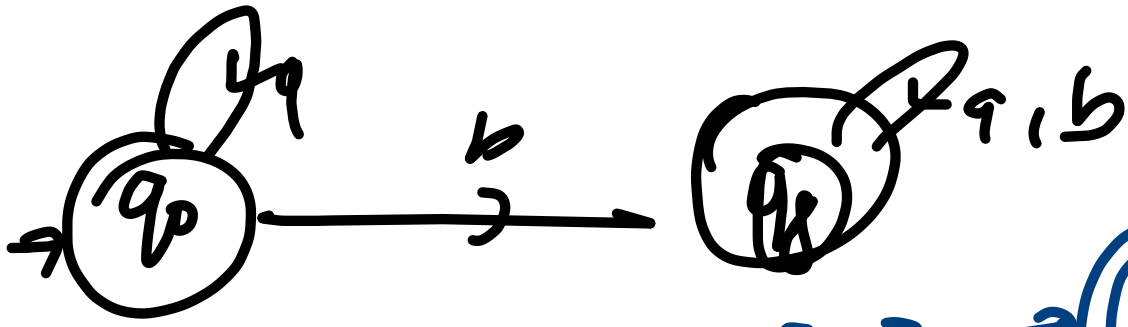
$q_0 \to a q_0$
$q_0 \to b q_f$
$q_f \to a q_f$
$q_f \to b q_f$

$NT \to T\ NT$

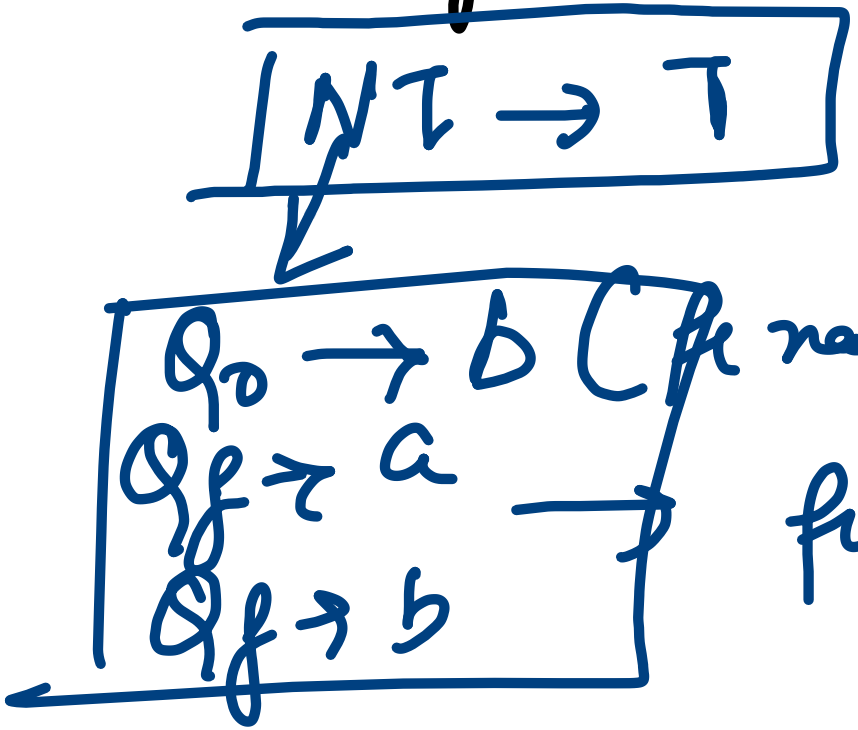
$q_0(a) \to q_0$
$q_0(b) \to q_f$
$q_f(a) \to q_f$
$q_f(b) \to q_f$

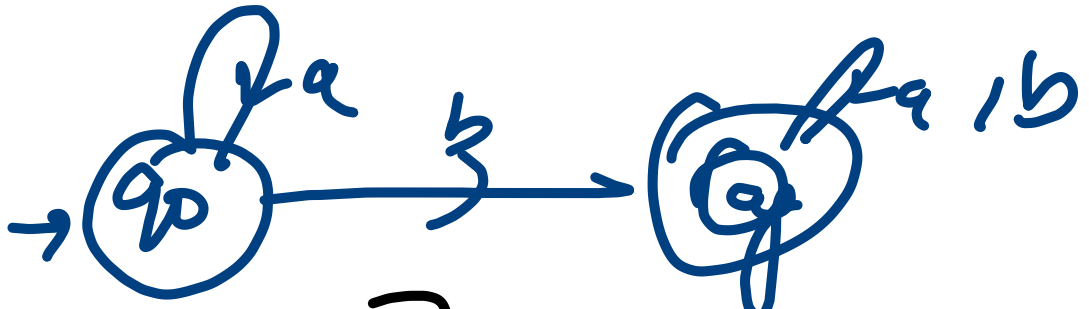
$q_0 \to b q_f$ (T, NT)
$q_f \to a q_f$
$q_f \to b q_f$

NT



Terminatory Rule





1) q_0, q_f
2) a / b.
3)

$q_0 \to a\,q_0$
$q_0 \to b\,q_f$
$q_f \to a\,q_f$
$q_f \to b\,q_f$

→ Transition Rules

Terminating Rules

$q_0 \to b$
$q_f \to a$
$q_f \to b$

→ final state Rules

4) q_0