THE EQUIVALENCE OF DFA'S AND NFA'S

* Since every DFA is an NFA it is clear that the class of languages accepted by NFA's includes the regular sets.

* For every NFA we can construct an equivalent DFA.

Theorem:

Let L be a set accepted by NFA, then there exists a DFA that accepts L.

Proof:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA which accepts L

Define a DFA,

$$M' = (Q', \Sigma, \delta', q_0', F')$$

* The states of M' are all the subsets of the set of states of M

i.e) $Q' = 2^Q$

* F' is the set of all states in Q' containing a final state of M.
* An element of Q' will be denoted by $[q_1, q_2, \ldots q_i]$, where $q_1, q_2, \ldots q_i$ are in Q
* $[q_1, q_2, \ldots q_i]$ is a single state of the DFA corresponding to the set of states of the NFA

* $q_0' = [q_0]$

* $\delta'([q_1], a) = [P_1]$

iff

$\delta(q_1, a) = \{P_1\}$

iff

$$\delta(\{q_1, q_2 \cdots q_i\}, a) = \{p_1, p_2, p_3 \cdots p_j\}$$

* It is easy to show by induction on the length of the input string x that

$$\delta'(q_0, x) = [q_1, q_2 \cdots q_i]$$

---

iff

$$\hat{\delta}(q_0, x) = \{q_1, q_2 \cdots q_i\}$$

Basis:

The result is trivial for $|x| = 0$, since $q_0' = [q_0]$ and x must be ϵ.

$$\delta'([q_0], \epsilon) = [q_0]$$

iff

$$\hat{\delta}(q_0, \epsilon) = q_0$$

Induction:

* Suppose that the hypothesis is true for inputs of length m or less.
* Let xa be a string of length $m+1$ with a in Σ. Then

$$\delta'(q_0', xa) = \delta'(\delta'(q_0', x), a)$$

By the inductive hypothesis

$$\delta'(q_0', x) = [P_1, P_2, \ldots P_j]$$

iff

$$\delta_N(q_0, x) = \{P_1, P_2, \ldots P_j\}$$

$$\therefore \delta'(q_0', xa) = \delta'(\delta'(q_0', x), a)$$

$$= \delta'([P_1, P_2, \ldots P_j], a)$$

$$= [r_i, r_2, \ldots r_k]$$

iff

$$\delta^*(q_0, xa) = \delta(\delta(q_0, x), a)$$

$$= \delta(\{P_1, P_2, \ldots P_j\}, a)$$

$$= \{r_1, r_2, \ldots r_k\}$$

Thus

$$\delta'(q_0', xa) = [r_1, r_2 \ldots r_k]$$

iff

$$\delta(q_0, xa) = \{r_1, r_2, \ldots r_k\}$$

which establishes the inductive hypothesis

* $\delta'(q_0', x)$ is in F' exactly when $\delta(q_0, x)$ contains a state in F.

$$\therefore L(M) = L(M')$$

1. Let $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ be an NFA, where

$\delta(q_0, 0) = \{q_0, q_1\}$

$\delta(q_0, 1) = \{q_1\}$

$\delta(q_1, 0) = \phi$

$\delta(q_1, 1) = \{q_0, q_1\}$.

	0	1
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	ϕ	$\{q_0, q_1\}$

Solution:

We can construct a DFA

$M' = (Q', \{0, 1\}, \delta', [q_0], F')$

Q' = all subsets of $\{q_0, q_1\}$.

$Q' = \{[q_0], [q_1], [q_0, q_1], \phi\}$

F' = Set of states of Q' containing a final state in F

$F' = \{[q_1], [q_0, q_1]\}$

Transition Table: δ'

	0	1
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	ϕ	$[q_0, q_1]$
$[q_0, q_1]$	?	?
ϕ	ϕ	ϕ

To find $\delta'([q_0, q_1], 0)$

$$\delta(\{q_0, q_1\}, 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

$$\delta'([q_0, q_1], 0) = [q_0, q_1]$$

To find $\delta'([q_0, q_1], 1)$

$$\delta(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$$
$$= \{q_1\} \cup \{q_0, q_1\}$$
$$= \{q_0, q_1\}$$

$$\therefore \delta'([q_0, q_1], 1) = [q_0, q_1]$$

$$\therefore M' = (Q', \Sigma', \delta', q_0', F')$$

$$Q' = \{[q_0], [q_1], [q_0, q_1], \phi\}$$

$$\Sigma = \{0, 1\}$$

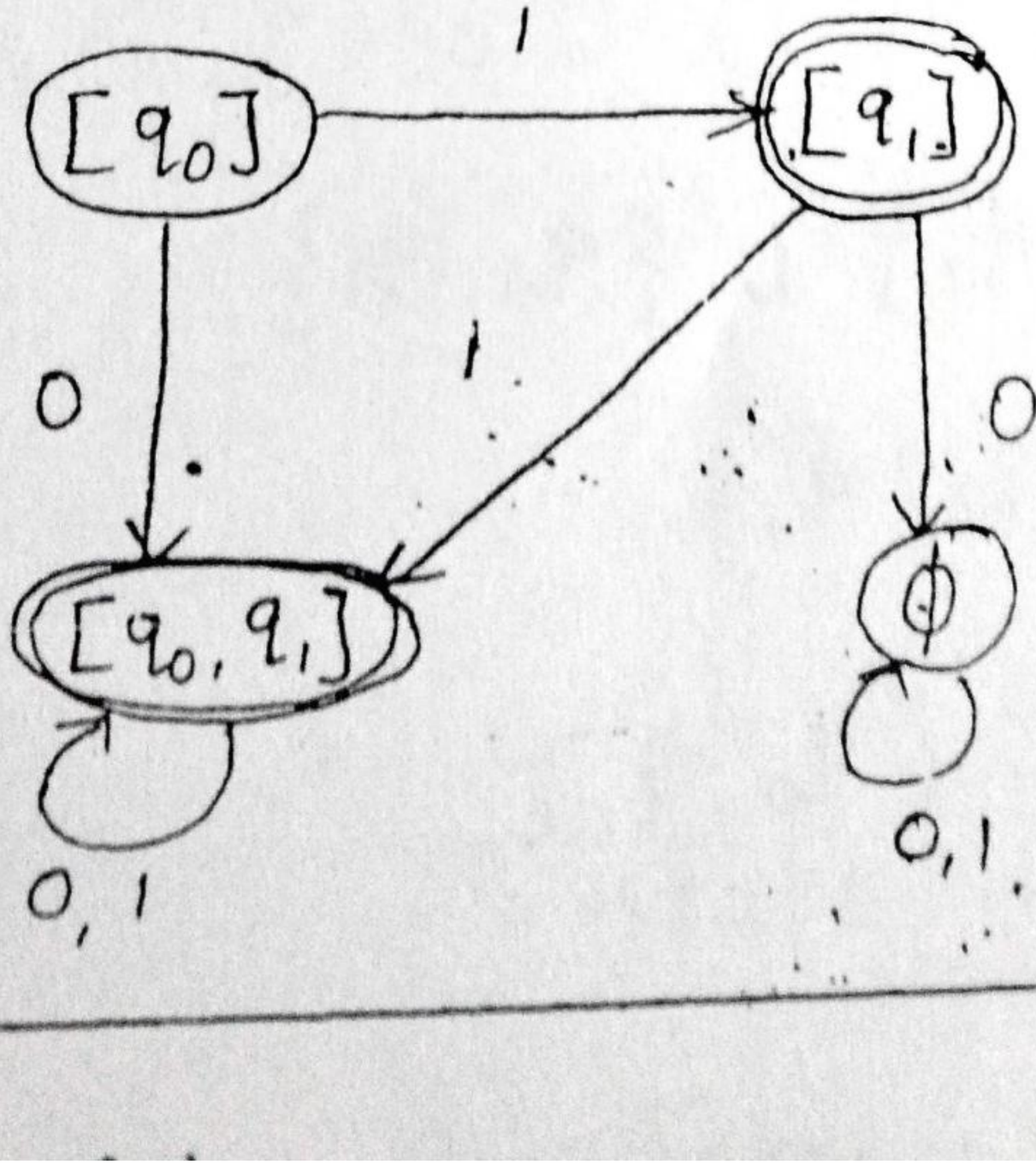
$$q_0' = [q_0]$$

$$F' = \{[q_1], [q_0, q_1]\}$$

0 :

	0	1
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	ϕ	$[q_0, q_1]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
ϕ	ϕ	ϕ

Transition Diagram:



Equivalence

THE EQUIVALENCE OF NFA'S WITH AND WITHOUT ϵ-MOVES:

* NFA with ϵ-moves accepts regular sets

Theorem:

If L is accepted by an NFA with ϵ-transitions then L is accepted by an NFA without ϵ-transitions.

Proof:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ϵ-transitions

Now we have to define NFA without ϵ-move M'

$$M' = (Q', \Sigma, \delta', q_0, F')$$

where Q' $F' = \begin{cases} F \cup \{q_0\} & \text{if } \epsilon\text{-closure}(q_0) \text{ contains a state of } F \\ F & \text{otherwise} \end{cases}$

* It is easy to show by induction on the length of the input string.

* We have to prove that

$$\delta^1(q_0, x) = \hat{\delta}(q_0, x)$$

* This statement may not be true for $x = \epsilon$

$$\delta'(q_0, \epsilon) = \{q_0\}$$

while $\delta(q_0, \epsilon) = \epsilon\text{-closure}(q_0)$.

* Therefore we begin our induction at 1

Basis:

$|x| = 1$

Then x is a symbol a

$\delta'(q_0, a) = \hat{\delta}(q_0, a)$ by the definition of δ'

Induction:

$|x| > 1$ Let $x = wa$

$$\delta'(q_0, wa) = \delta'(\hat{\delta}(q_0, w), a)$$

$$= \delta'(P, a)$$

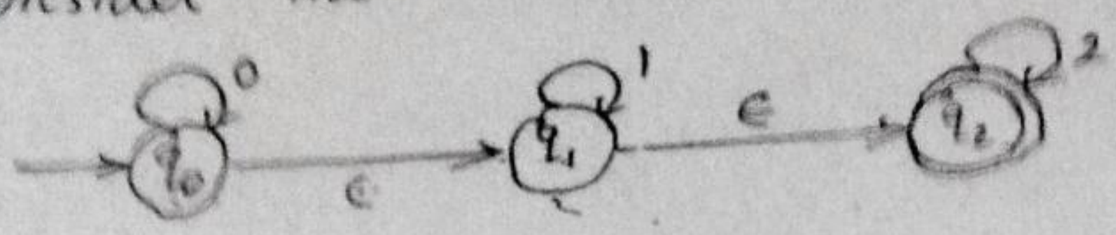
$$= \bigcup_{q \text{ in } P} \delta'(q, a)$$

$$= \bigcup_{q \text{ in } P} \delta(q, a)$$

$$= \delta(P, a)$$

Example

Consider the NFA with ϵ-moves



Find an equivalent NFA without ϵ-moves

Solution:

Given NFA with ϵ-move

$$M = (\{q_0, q_1, q_2\}, \{0, 1, 2, \epsilon\}, \delta, q_0, \{q_2\})$$

Now we have to define NFA without ϵ-move

$$M' = (Q', \Sigma, \delta', q_0, F')$$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1, 2\}$

$F' = \{q_0, q_2\}$. ∵ ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$ contains a state of F

δ':

	0	1	2
q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

$\delta(q_0, \epsilon) = \epsilon$-closure (q_0)
$= \{q_0, q_1, q_2\}$

$\delta(q_0, 0) = \epsilon\text{-closure}(\delta(q_0, 0))$

$= \epsilon\text{-closure}(\delta(\delta(q_0, \epsilon), 0)$

$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 0)$

$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$= \epsilon\text{-closure}(\{q_0\} \cup \phi \cup \phi)$

$= \{q_0, q_1, q_2\}$

$\delta'(q_0, 1) = \epsilon\text{-closure}(\delta(q_0, 1))$

$= \epsilon\text{-closure}(\delta(\delta(q_0, \epsilon), 1)$

$= \epsilon\text{-closure}(\delta\{q_0, q_1, q_2\}, 1)$

$= \epsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1)$

$= \epsilon\text{-closure}(\phi \cup \{q_1\} \cup \phi)$

$= \{q_1, q_2\}$

$\delta'(q_0, 2) = \epsilon\text{-closure}(\delta(q_0, 2))$

$= \epsilon\text{-closure}(\delta(\delta(q_0, \epsilon), 2)$

$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 2)$

$= \epsilon\text{-closure}(\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2)$

$= \epsilon\text{-closure}(\phi \cup \phi \cup \{q_2\})$

$= \{q_2\}$

$\delta'(q_1, 0) = \epsilon\text{-closure}(\delta(q_1, 0))$

$= \epsilon\text{-closure}(\delta(\delta(q_1, \epsilon), 0)$

$= \epsilon\text{-closure}(\delta\{q_1, q_2\}, 0)$

$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0)$

$= \phi$

$\delta'(q_1, 1) = \epsilon\text{-closure}(\delta(q_1, 1))$

$= \epsilon\text{-closure}(\delta(\delta(q_1, \epsilon), 1)$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}, 1)$

$= \epsilon\text{-closure}(\delta(q_1, 1) \cup \delta(q_2, 1))$

$= \epsilon\text{-closure}(\{q_1\} \cup \phi)$

$= \{q_1, q_2\}$

$\delta'(q_1, 2) = \epsilon\text{-closure}(\delta(q_1, 2))$

$= \epsilon\text{-closure}(\delta(\delta(q_1, \epsilon), 2)$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}, 2)$

$= \epsilon\text{-closure}(\delta(q_1, 2) \cup \delta(q_2, 2))$

$= \epsilon\text{-closure}(\phi \cup \{q_2\})$

$= \{q_2\}$

$\delta'(q_2, 0) = \epsilon\text{-closure}(\delta(\delta(q_2, \epsilon), 0)$

$= \epsilon\text{-closure}(\delta(\{q_2\}, 0))$

$= \epsilon\text{-closure}(\phi)$

$= \phi$

$\delta'(q_2, 1) = \epsilon\text{-closure}(\delta(\delta(q_2, \epsilon), 1)$

$= \epsilon\text{-closure}(\delta(\{q_2\}, 1)$

$= \epsilon\text{-closure}(\phi)$

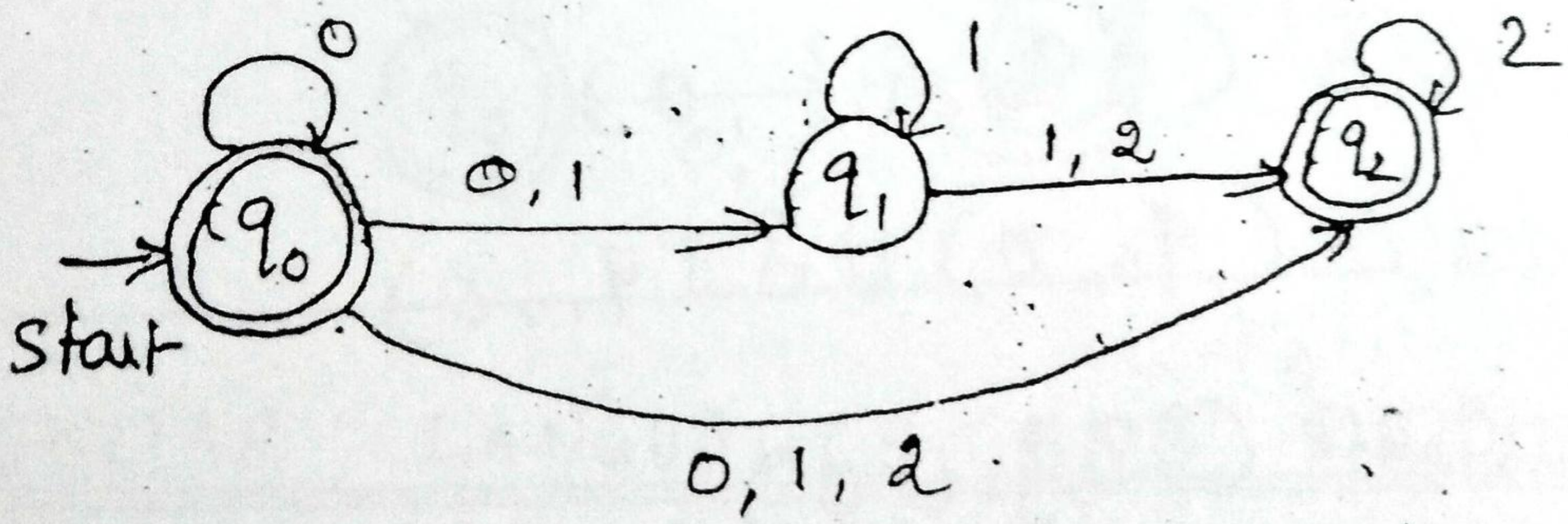
$= \phi$

ϵ-closure $(q_2, -) = \epsilon$-closure

$= \epsilon$-closure $(\delta(\{q_2\}, 2)$

$= \epsilon$-closure (q_2)

$= \{q_2\}$.



* A tree is a derivation tree for G if

(1) Every vertex has a label, which is a symbol of $V \cup T \cup \{\epsilon\}$

(2) The label of the root is S

(3) If a vertex is interior and has label A, then A must be in V

(4) If vertex has label A and the sons of vertex A are labeled from left as, $X_1, X_2 \ldots X_k$ then $A \to X_1 X_2 \ldots X_k$ must be a production in P

(5) If a vertex has a label ϵ, then it is a leaf and is the only son of its father.

AMBIGUITY:

* A context free Gramma G such that some word has two parse trees is said to be ambiguous

* An equivalent definition of ambiguity is that some word has more than one left most derivation or more than one right most derivation.

* example.

Show that the Grammar G

$E \rightarrow E+E \mid E*E \mid (E) \mid id$ is ambiguous

Solution:

* Deriving a string id + id * id

LMD 1:

$E \underset{lm}{\Rightarrow} E+E$

$\underset{lm}{\Rightarrow} id+E$

$\underset{lm}{\Rightarrow} id+E*E$

$\underset{lm}{\Rightarrow} id+id*E$

$\underset{lm}{\Rightarrow} id+id*id$

LMD 2:

$E \underset{lm}{\Rightarrow} E*E$

$\underset{lm}{\Rightarrow} E+E*E$

$\underset{lm}{\Rightarrow} id+E*E$

$\underset{lm}{\Rightarrow} id+id*E$

$\underset{lm}{\Rightarrow} id+id*id$.

* For the word id+id*id, there exists two left most derivation.

RMD 1:

$E \underset{rm}{\Rightarrow} E+E$

$\underset{rm}{\Rightarrow} E+E*E$

$\underset{rm}{\Rightarrow} E+E*id$

$\underset{rm}{\Rightarrow} E+id*id$

$\underset{rm}{\Rightarrow} id+id*id$

RMD 2:

$E \underset{rm}{\Rightarrow} E*E$

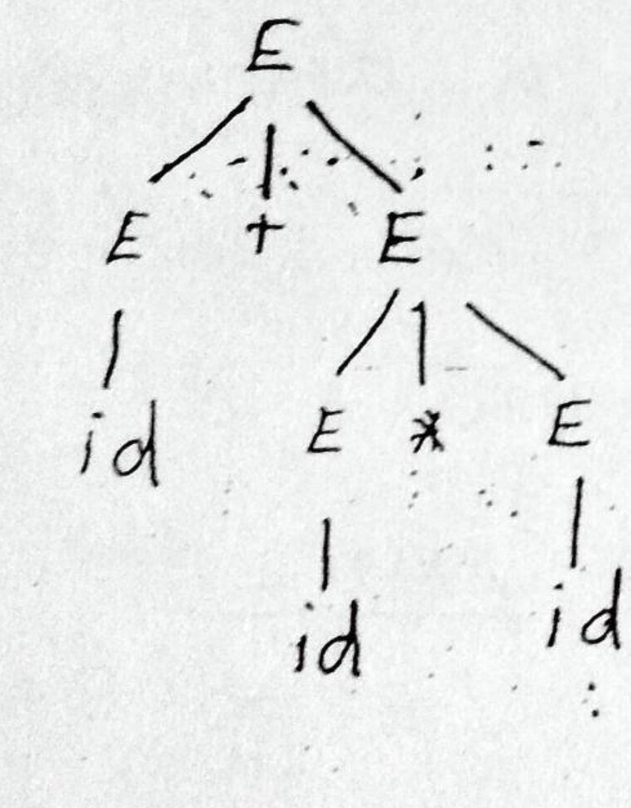
$\underset{rm}{\Rightarrow} E*id$

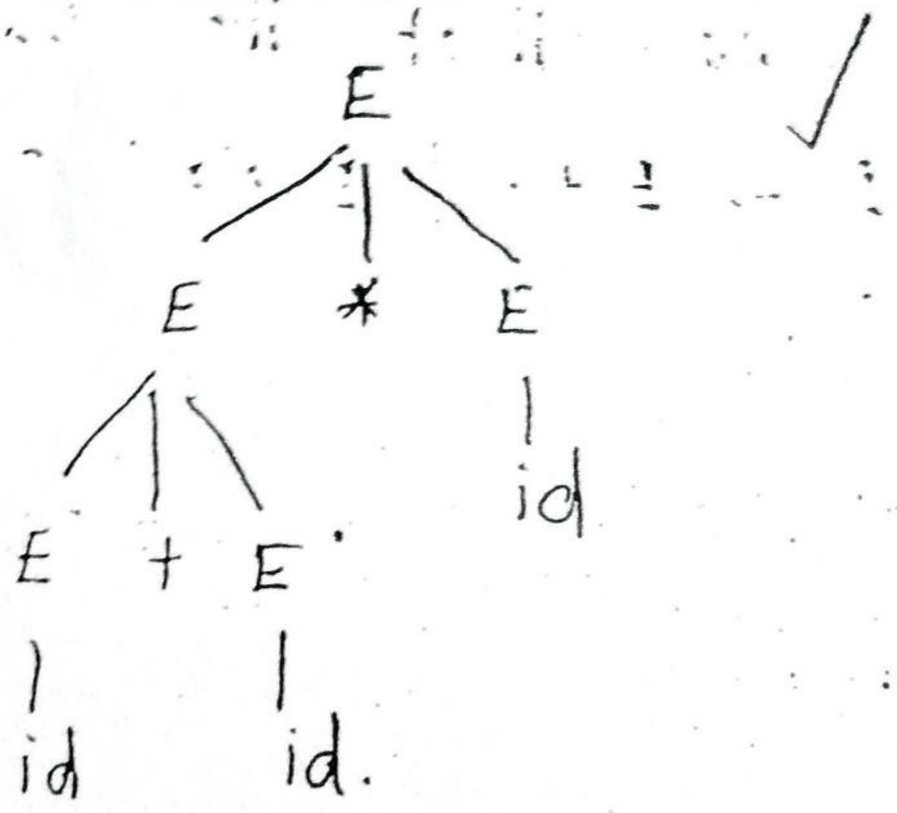
$\underset{rm}{\Rightarrow} E+E*id$

$\underset{rm}{\Rightarrow} E+id*id$

$\underset{rm}{\Rightarrow} id+id*id$.

* For the word id+id*id, there exists two right most derivation.





* For the word id + id * id has two parse trees

* So the Grammar is ambiguous.

PUSH DOWN AUTOMATA

* The PDA will have an input tape, a finite control, and a stack
* The stack holds a string of symbols from some alphabet.
* The device will be non-deterministic having some finite number of choices of moves in each situations

Formal Definition:

* A push Down Automata M is a system

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

where

$Q \rightarrow$ is a finite set of states

$\Sigma \rightarrow$ is an alphabet called the input alphabe

$\Gamma \rightarrow$ is an alphabet called the stack alphabet.

$q_0 \rightarrow$ in Q is the initial state.

$Z_0 \rightarrow$ is Γ is a particular stack symbol called the start symbol

$F \rightarrow F \subseteq Q$, is the set of final states.

$\delta \rightarrow$ is a transition function mapping from

$Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$

The moves will be of two types

* The first type of move,
* An input symbol is used. Depending on the input symbol the top symbol on the stack and the state of a finite control, a number of choices are possible
* Each choice consists of next state for the finite control and a string of symbols to replace the top stack symbol
* After selecting a choice, the input head is advanced one symbol

$$\delta(q, a, Z) = \{(P_1, \gamma_1), (P_2, \gamma_2), \ldots (P_m, \gamma_m)\}$$

where

q and P_i $1 \leq i \leq m$ are states

a is in Σ

$Z \rightarrow$ is a stack symbol

$\gamma_1 \rightarrow$ is in Γ^*

i.e) The PDA in state q with the input symbol a and Z top symbol on the stack can for any i enter state P_i, replace symbol Z by string γ_i and advance the input head one symbol.

is similar to the first, except that the input symbol is not used and the input head is not advaced after the move

$$\delta(q, \epsilon, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots (p_m, \gamma_1)\}$$

(c) The PDA in state q, independent of the input symbol being scanned and with Z, the top symbol on stack, can enter state p_i, for any i and replace Z by γ_i. The input head is not advan

INSTANTANEOUS DESCRIPTIONS (ID)

* The ID records the state, input and stack contents
* ID is defined as a triple.

$$(q, w, \gamma)$$

where $q \rightarrow$ is a state in the finite control

$w \rightarrow$ is a string of input

$\gamma \rightarrow$ is a string of stack symbols.

* If $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA

we say

$$(q, aw, Z\alpha) \vdash (p, w, \beta\alpha)$$

if $\delta(q, a, Z) = (p, \beta)$

1. Language accepted by empty stack

* To define the language accepted to be the set of all inputs for which some sequence of moves causes the PDA to empty its stack.

* For PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, we define $N(M)$, the language accepted by empty stack to be,

$$N(M) = \{ w \mid (q_0, w, Z_0) \vdash (p, \epsilon, \epsilon), \text{ for some } p \text{ in } Q \}$$

2. Language accepted by final state:

* To define the language accepted to be the set of all inputs for which some choices of moves causes the PDA to enter a final state

* We define $L(M)$, the language accepted by PDA M by final state to be.

$$L(M) = \{ w \mid (q_0, w, Z_0) \vdash^* (p, \epsilon, \alpha) \text{ for some } p \text{ in } F \text{ and } \alpha \text{ in } \Gamma^* \}$$

* If a set can be accepted by empty stack by some PDA, it can be accepted by final state by some other PDA and vice versa.

* The PDA is deterministic, if atmost one move is possible from any ID.

* The PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if

i) for each q in Q and Z in Γ, whenever $\delta(q, \epsilon, Z)$ is not empty then $\delta(q, a, Z)$ is empty for all a in Σ.

2) for no q in Q, Z in Γ and a in $\Sigma \cup \{\epsilon\}$ does $\delta(q, a, Z)$ contain more than one element

* The deterministic and non-deterministic models of PDA, are not equivalent with respect to the language accepted

* WW^R is accepted by Non-deterministic PDA but not by any Deterministic PDA