Grammars and Languages

부산대학교
PUSAN NATIONAL UNIVERSITY

Review 1

	Alphabet	String	Language
Power	$\Sigma^n, \Sigma^+, \Sigma^*,$ $\Sigma^0=\{\lambda\}$	w^n, $w^0=\lambda$	$A^n, A^+, A^*,$ $A^0=\{\lambda\}$
Concatenation	(juxtaposition)	w_1w_2	AB
Substring		y from xyz	
Prefix (Suffix)		x (y) from xy	
Length, Equality		$\|x\|$, x = y	
etc		Empty String	Empty Lang. Ø

Review 2

- **Regular Expression**

  A regular expression over an alphabet $\sum$ is defined recursively as one of the following types.

  1. $\emptyset$
  2. λ ;
  3. $a \in \sum$
  4. (E_1+E_2)
  5. (E_1E_2)
  6. (E_1^*)

- **Regular Language**

  A language $L \subseteq \sum^*$ is a regular language if there is a regular expression E over $\sum$ with $L = L(E)$

  1. $\emptyset$
  2. $\{\lambda\}$
  3. $\{a\}$
  4. $L(E_1) \cup L(E_2)$
  5. $L(E_1)L(E_2)$
  6. $L(E_1)^*$

Review 3

- Deterministic Finite Automata

$$M = (Q, \sum, \delta, q_0, F)$$

where δ is a transition function.

DFA language L if $L = L(M)$ for a DFA M

- Non-deterministic Finite Automata

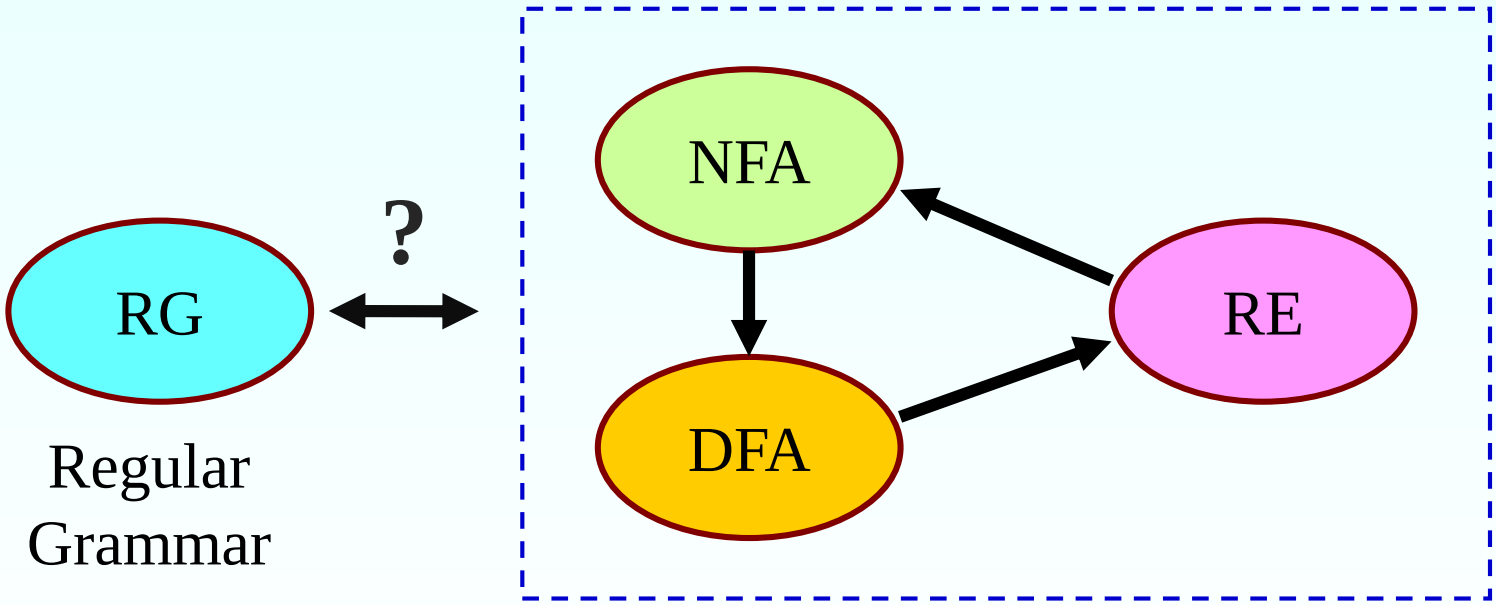
$$N = (Q, \sum, \Delta, q_0, F),$$

where Δ is a transition relation.

NFA language $L(N) = \{ w \in \Sigma^* \mid \Delta^*(q_0, w) \cap F \neq \phi \}$

Review 4

Conversions



Definition of Grammar

- Def. Grammar (문법)

$$G = (V, \Sigma, S, P)$$

V : a finite set of variables

Σ : an alphabet

S : a start variable, $S \in V$

P : a set of production rules

Production rule : $x \rightarrow y$

where $x \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$

and $y \in (V \cup \Sigma)^*$

$S, A \in V,\ a, b \in \Sigma$

$S \rightarrow aA \mid \lambda$

$aA \rightarrow aSb$

$S \rightarrow aA$

$S \rightarrow \lambda$

Language of Grammar

- Def. Language of Grammar

  For a grammar $G = (V, \Sigma, S, P)$, a language derived from G is defined as

  $$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

  where $\Rightarrow^*$ indicates k-times derivation ($k \geq 0$).

  The derivation is the process of applying a production rule $x \rightarrow y$ to $x \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$.

$S \rightarrow aA \mid \lambda$ $aA \rightarrow aSb$	$aabb$?	$S \Rightarrow aA \Rightarrow aSb \Rightarrow aaAb$ $\Rightarrow aaSbb \Rightarrow aabb$

Regular Grammar

- Def. Regular Grammar

$$G = (V, \Sigma, S, P)$$

where all the production rules have the form

$A \rightarrow wB$ or $A \rightarrow w$, $A, B \in V$ and $w \in \Sigma^*$.

The language $L(G)$ being derived from a regular grammar G is called a regular language.

An Example

(Ex.1) Find the regular language that is derived from the following regular grammar.

$G = (\{S, A\}, \{a, b\}, S, P)$

$P : S \rightarrow aA$

$A \rightarrow abS \mid a$

S

aA

$aabS$

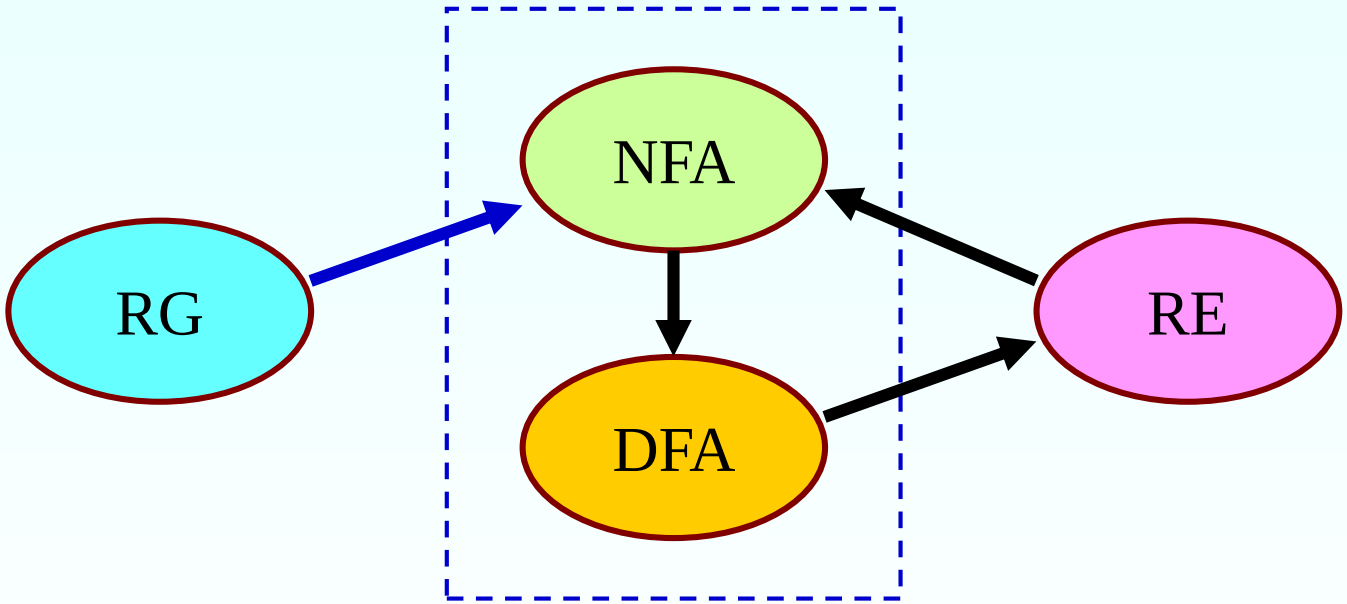
$aabaA$

$aabaabS$

$aabaabaA$

$aabaabaa$

Conversions



Finite Automata

Regular Grammar → NFA

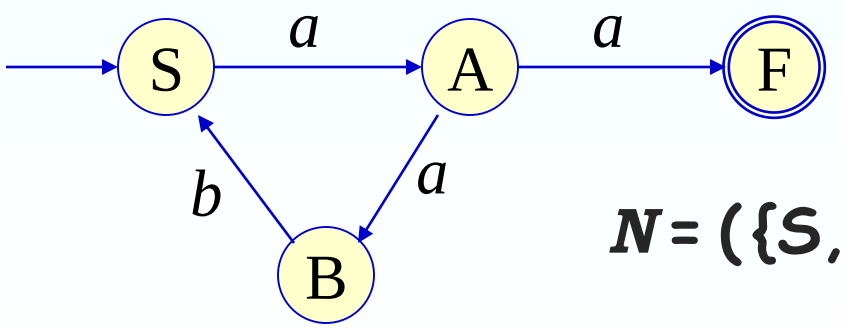
(Ex.2) Find a NFA that is equivalent to the regular grammar

$$G = (\{S, A\}, \{a, b\}, S, P)$$

$$P : S \rightarrow aA$$

$$A \rightarrow abS \mid a$$

$$L(G) = L((aab)^* aa)$$



$$N = (\{S, A, B, F\}, \{a, b\}, \Delta, S, \{F\})$$

Regular Grammar → NFA

(Ex.3) Find a NFA that is equivalent to the regular grammar

$G = (\{S,A\}, \{a,b\}, S, P)$

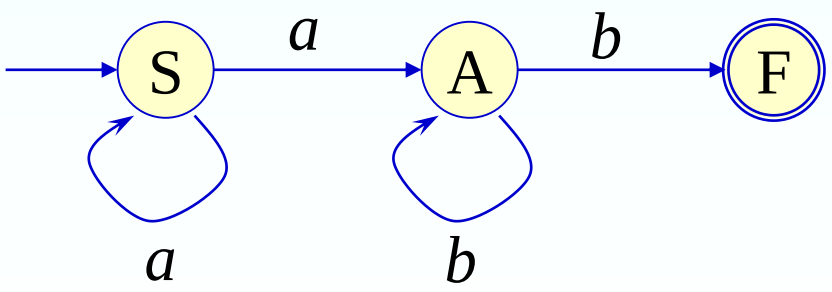
$P : S \rightarrow aS \mid aA$

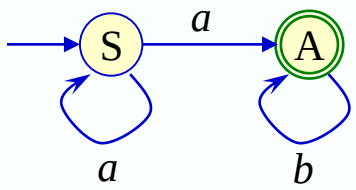
$A \rightarrow bA \mid b$

$L(G) = L(a^*ab^*b)$

$+ (A \rightarrow \lambda)$

$L(a^*ab^*)$

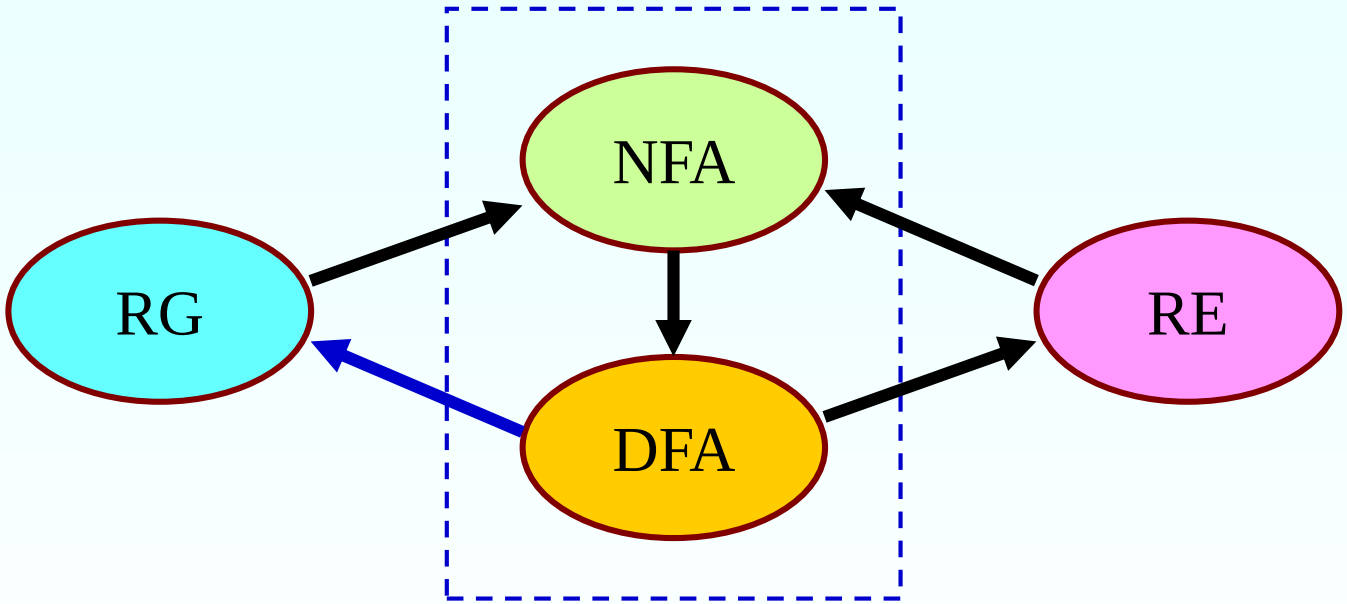




$N = (\{S,A\}, \{a,b\}, \Delta, S, \{A\})$

$N = (\{S,A,F\}, \{a,b\}, \Delta, S, \{F\})$

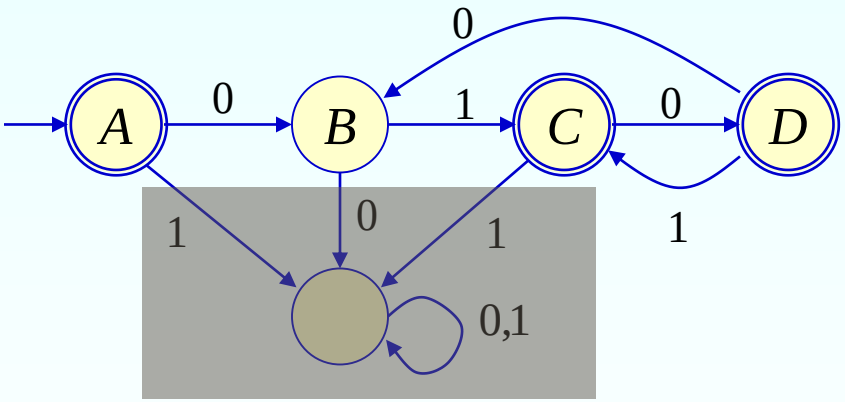
Conversions (final)



Finite Automata

DFA → Regular Grammar

(Ex.4) Find a regular grammar such that its language is equal to the language of following DFA.



We don't have to consider dead states that are not final states.

$G = (\{A, B, C, D\}, \{0, 1\}, A, P)$

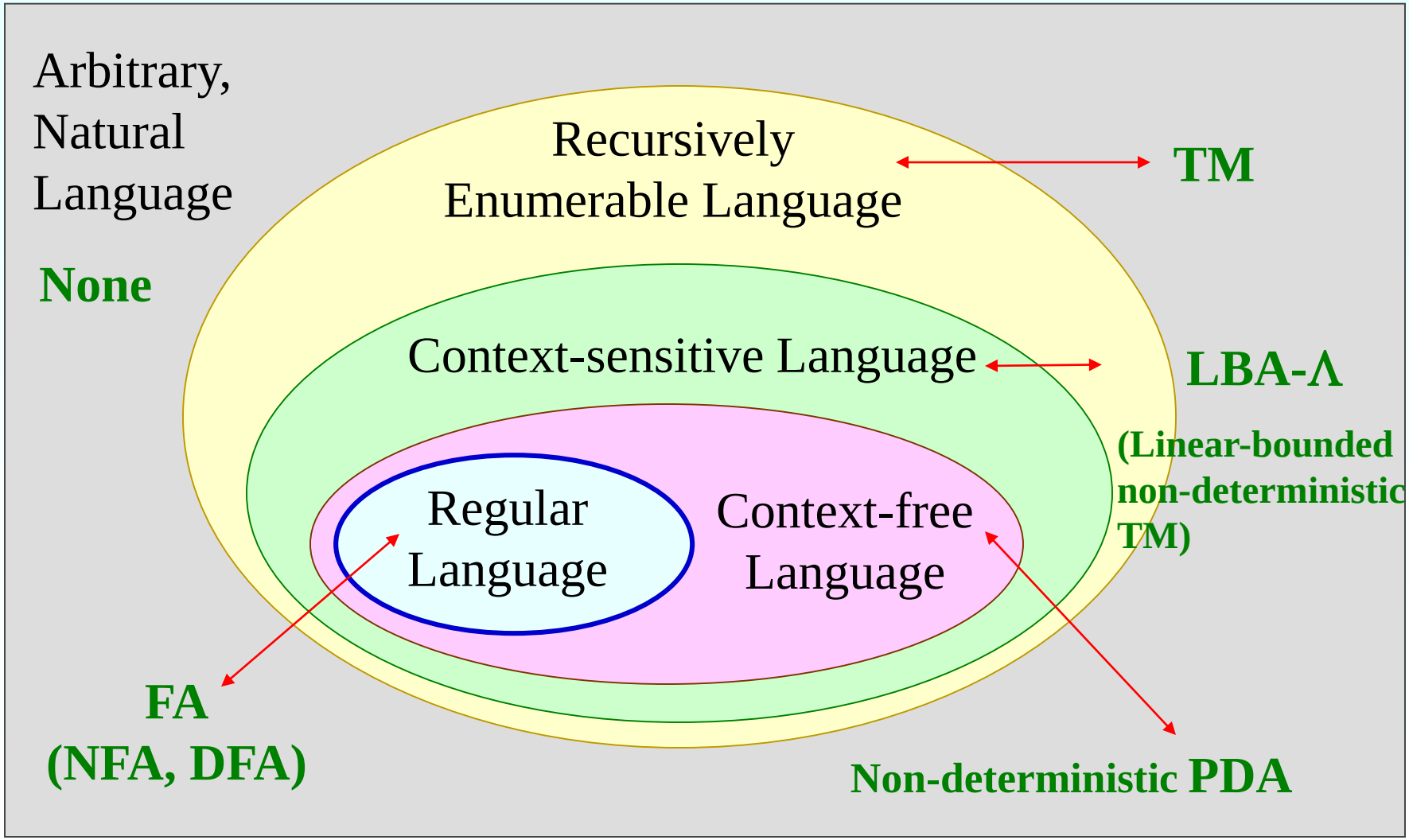
P : $A \rightarrow 0B \mid \lambda$

$B \rightarrow 1C$

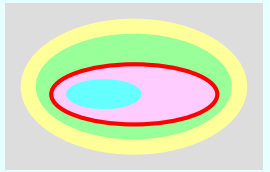
$C \rightarrow 0D \mid \lambda$

$D \rightarrow 0B \mid 1C \mid \lambda$

Language Types by Chomsky



Context-Free Grammar



- Def. Context-Free Grammar (문맥무관 문법)

$$G = (V, \Sigma, S, P)$$

Where all production rules have the form

$A \rightarrow x$, $A \in V$ and $x \in (V \cup \Sigma)^*$.

(cf.)

In regular grammar, $A \rightarrow wB$ or $A \rightarrow w$ where $w \in \Sigma^*$.

$A \rightarrow Aa \mid bB \mid \lambda$

$bB \rightarrow bBc \mid bA$

Is it context-free ?

Examples

(Ex.5) $G = (\{S\}, \{0,1\}, S, P)$

$P : S \rightarrow 0S1 \mid \lambda$

$L(G) = \{ 0^n 1^n \mid n \geq 0 \}$

(Ex.6) $G = (\{S,A\}, \{a,b,c\}, S, P)$

$P : S \rightarrow aSc \mid A$

$A \rightarrow bAc \mid \lambda$

$L(G) = \{ a^m b^n c^k \mid m + n = k, \; k \geq 0 \}$

Pushdown Automata

- Def. Pushdown Automata

or Stack Automata

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

Q : a set of finite states

Σ : an alphabet

Γ : a stack alphabet (start symbol = #)

Δ : a subset of the transition relation

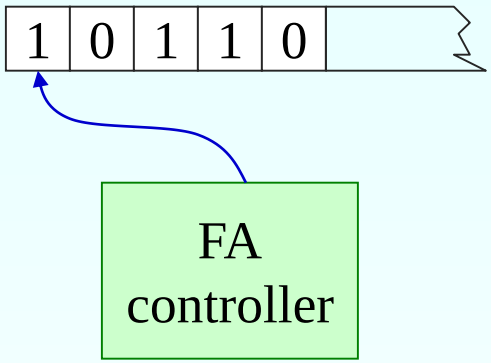
$$(Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\})) \times (Q \times \Gamma^*)$$

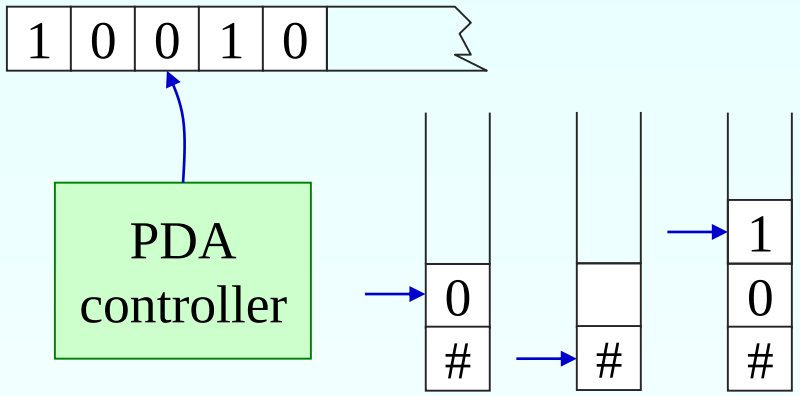
q_0 : a start (initial) state

F : a set of final states, $F \subseteq Q$

(cf.) FA $M = (Q, \sum, \delta \text{ or } \Delta, q_0, F)$

FA vs. PDA





- No memory
- Read-only head moves to the right
- (current state, tape symbol) → (next state)
- Initially at the start state & at the leftmost position

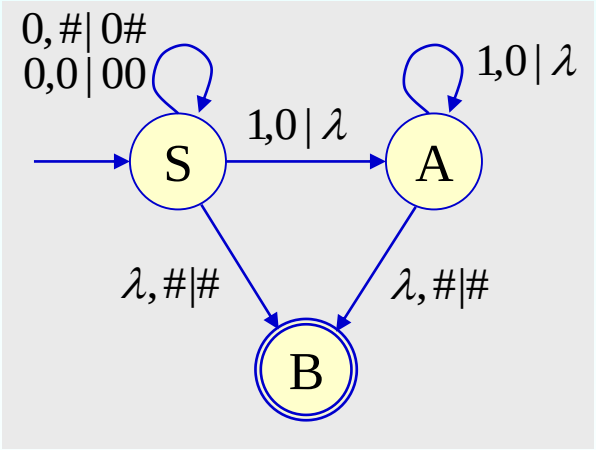
- Same structure except the stack
- (current state, tape symbol, stack symbol) → (next state, stack string)
- Initially (start state, leftmost position of input string w, #)
- Accept w when the state of PDA after reading w is a final state
- (*Ex.*) **((A,0,0), (B, λ))** ; pop 0
  ((A,0,0), (B, 10)) ; push 1

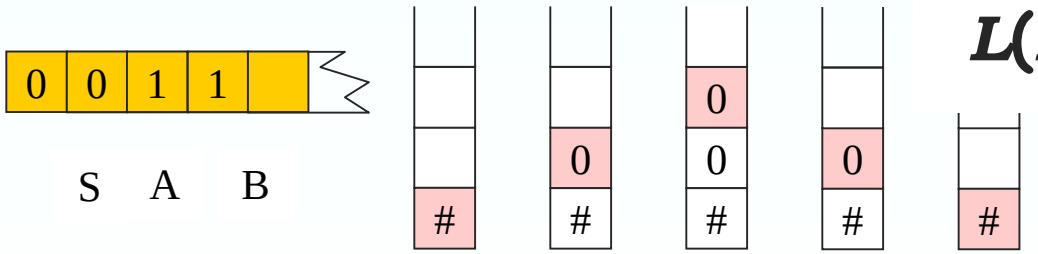
An Example of PDA

- Find $L(M)$ for the following PDA.

$M = (\{S,A,B\}, \{0,1\}, \Gamma=\{0,\#\}, \Delta, S, \{B\})$

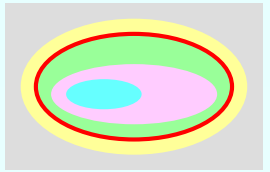
Δ : ((S,0,#), (S,0#)) ; push 0
((S,0,0), (S,00)) ; push 0
((S,λ,#), (B,#))
((S,1,0), (A,λ)) ; pop 0
((A,1,0), (A,λ)) ; pop 0
((A,λ,#), (B,#))





$L(M) = \{ 0^n 1^n \mid n \geq 0 \}$

Context-Sensitive Language



- Def. Context-Sensitive Grammar

$$G = (V, \Sigma, S, P)$$

Each production rule $x \rightarrow y$ satisfies

$$\| x \| \leq \| y \|$$

where $x \in (V \cup \Sigma)^* \; V \; (V \cup \Sigma)^*$ and $y \in (V \cup \Sigma)^*$.

(Note)

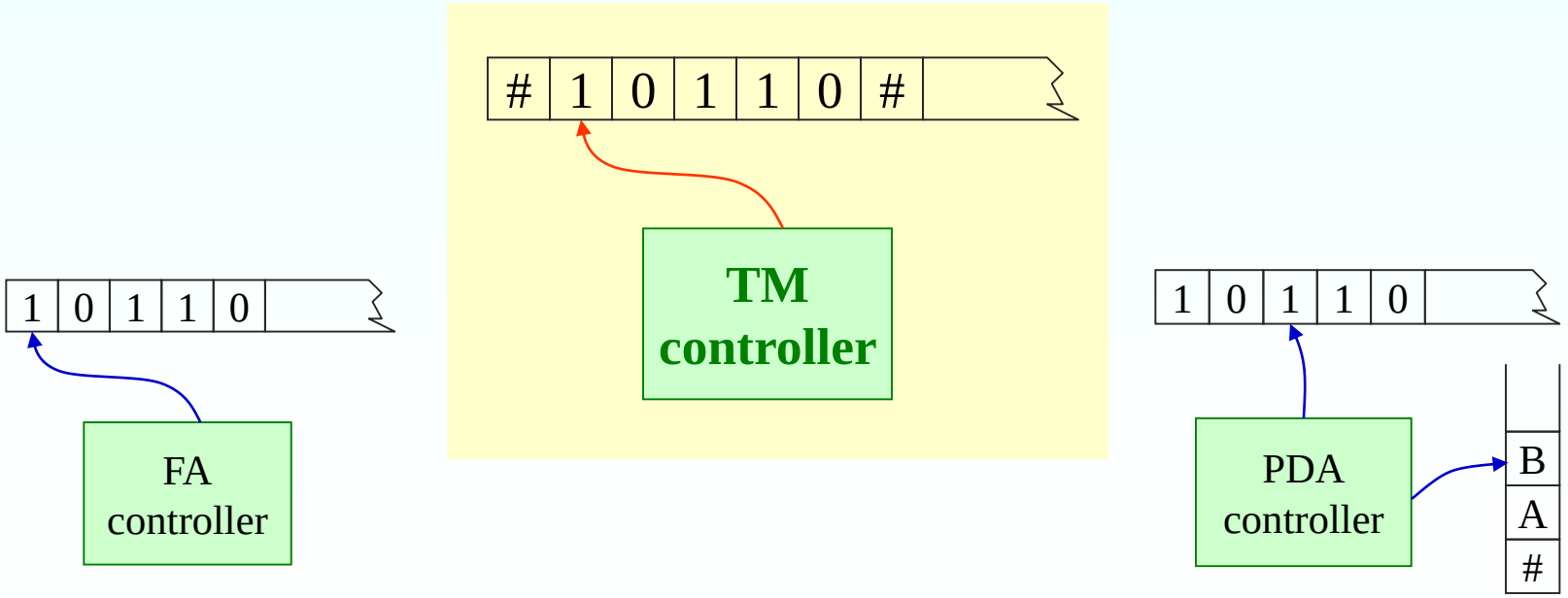
Each production rule can be converted into

$$u \; A \; v \rightarrow u \; z \; v \,, \quad u, v, z \in (V \cup \Sigma)^+ .$$

A is sensitive to $u, v \in (V \cup \Sigma)^*$.

Turing Machine

- Turing Machine (TM)
  - It consists of a tape and a controller
  - The read/write tape-head can move to the left or the right. Furthermore, it may stay



Turing Machine

- Def. Turing Machine

$$T = (Q, \Sigma, \Gamma, \delta, q_0, H)$$

Q : a finite set of states

Σ : an input alphabet

Γ : a tape alphabet (#: blank)

δ : a transition function

$$(Q-H)\times\Gamma \rightarrow Q\times\Gamma\times\{L,R,S\}$$

q_0 : a start state, $q_0 \in Q$

H : a set of halt states, $H \subseteq Q$

TM Operation

1. TM is a deterministic machine because a transition function δ is used
2. L and R in the definition of δ means moving the tape-head to the left and the right, respectively. S means staying the tape-head
3. TM can operate after reading all the input symbols, while a DFA and a PDA should stop. Thus, a set of halt states should be defined in TM
4. We assume that the tape-head cannot move to the left at leftmost position of the tape

TM Configuration

- TM Configuration (상황)

$$(q , u \underline{a} v)$$

q : current state
$\underline{a}$: symbol at position of tape-head
u : left string of $\underline{a}$ in tape
v : right string of $\underline{a}$ in tape

Thus, it is determined by the current state q, the tape contents, and the position of tape-head

Start configuration : $(q_0 , \underline{\#} w)$
where w is an input string

An Example

- Find a TM that defines $L = \{0^n 1^n \mid n \geq 1\}$.

  # 0 0 1 1 # ; Move R until finding a 0

  # a 0 1 1 # ; Replace 0 with a & move R to first 1

  # a 0 b 1 # ; Replace 1 with b & move L until finding the leftmost 0

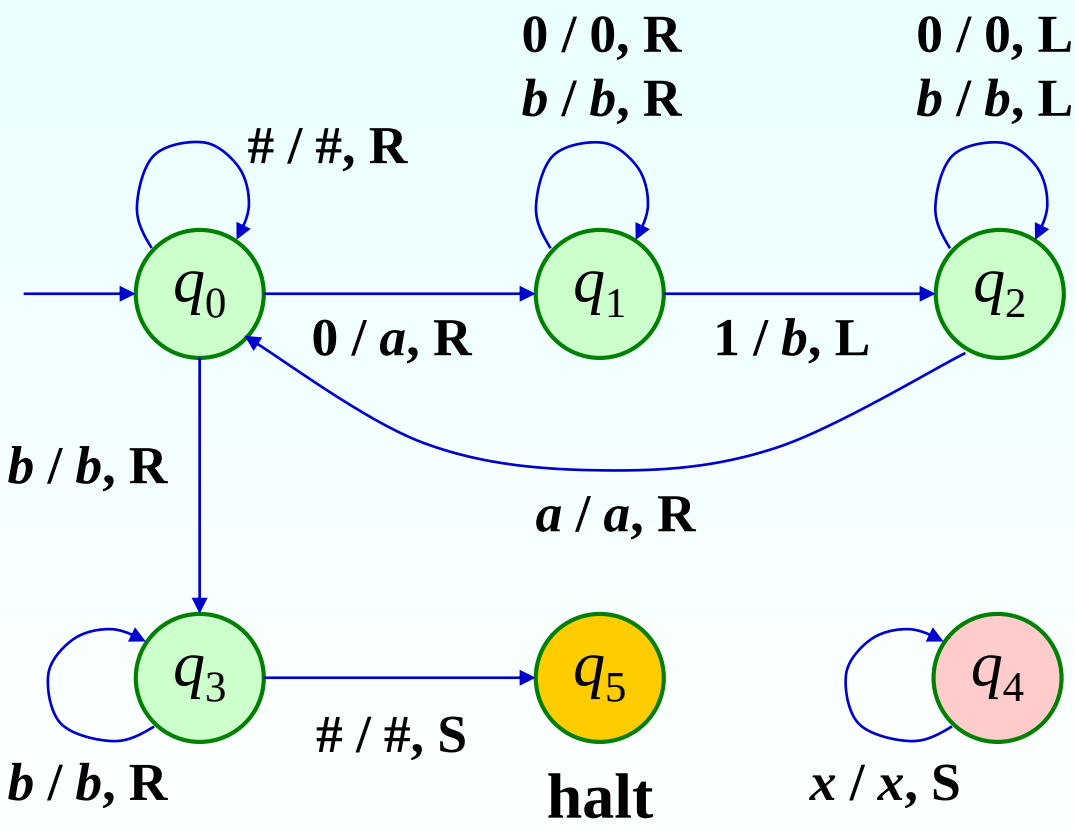
  # a a b 1 #

  # a a b b # ; No 0 & no 1 → halt

continue

$T = (\{q_0, \ldots, q_5\}, \{0, 1\}, \{0, 1, a, b, \#\}, \delta, q_0, \{q_5\})$

δ	0	1	a	b	#
q_0	(q_1, a, R)			(q_3, b, R)	$(q_0, \#, R)$
q_1	$(q_1, 0, R)$	(q_2, b, L)	(q_4, a, S)	(q_1, b, R)	
q_2	$(q_2, 0, L)$		(q_0, a, R)	(q_2, b, L)	$(q_4, \#, S)$
q_3				(q_3, b, R)	$(q_5, \#, S)$
q_4		$\delta(q, x) = (q_4, x, S)$			



q_0 # 0 0 1 1 #

q_0 # 0 0 1 1 #

q_1 # a 0 1 1 #

q_1 # a 0 1 1 #

q_2 # a 0 b 1 #

q_2 # a 0 b 1 #

q_0 # a 0 b 1 #

q_1 # a a b 1 #

q_1 # a a b 1 #

q_2 # a a b b #

q_2 # a a b b #

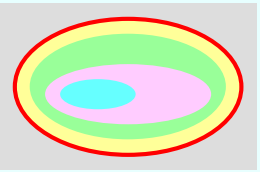
q_0 # a a b b #

q_3 # a a b b #

q_3 # a a b b #

Halt q_5 # a a b b #

Recursively Enumerable Language



- Def. Language for a TM

  TM $T = (Q, \Sigma, \Gamma, \delta, q_0, H)$

  $L(T) = \{ w \in \Sigma^* \mid (q_0, \#w) \vdash^* (h, *) \}, h \in H$

  Thus, $L(T)$ is the set of all the strings that halt the TM

- Recursively Enumerable Language L

  There exists a TM T such that $L = L(T)$

Non-deterministic TM

- Def. Non-deterministic TM

$$N = (Q, \Sigma, \Gamma, \Delta, q_0, H)$$

Q : a finite set of states

Σ : an input alphabet

Γ : a tape alphabet (#: blank)

Δ : a subset of $((Q-H)\times\Gamma) \times (Q\times\Gamma\times\{L,R,S\})$

q_0 : a start state, $q_0 \in Q$

H : a set of halt states, $H \subseteq Q$

(cf.) δ : a function $(Q-H)\times\Gamma \rightarrow Q\times\Gamma\times\{L,R,S\}$

Language of N-TM

- Def.

  N-TM $N = (Q, \Sigma, \Gamma, \Delta, q_0, H)$

  $L(N) = \{ w \in \Sigma^* \mid (q_0, \underline{\#}w) \vdash^*_N (h, *) \}, h \in H$

  Because the N-TM is non-deterministic, $L(N)$ is the set of all the strings that have at least one transition path to the H