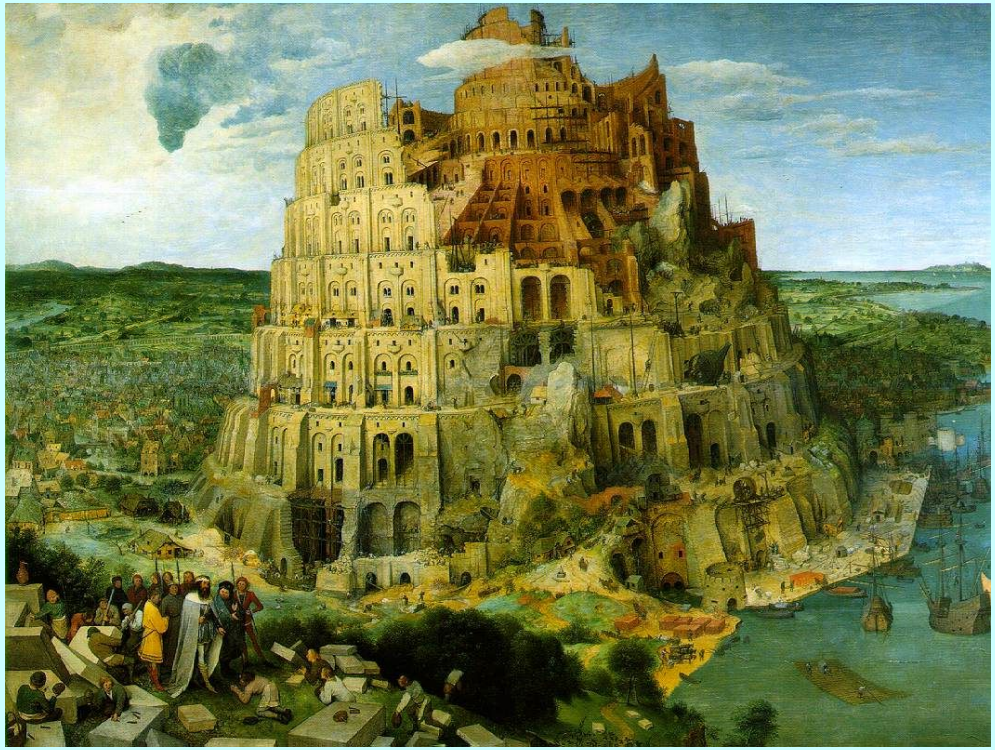
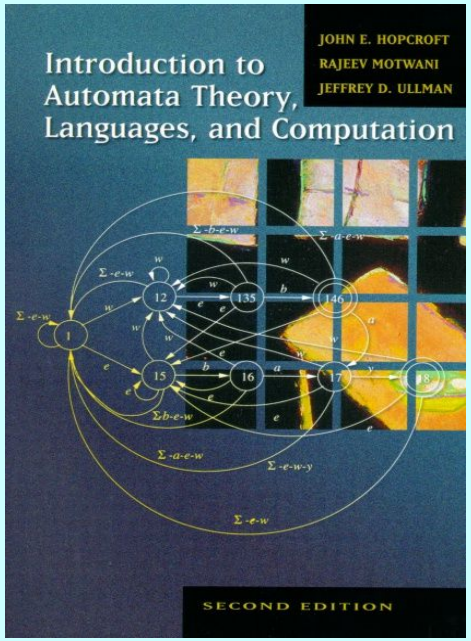
Formal Languages



Pieter Bruegel the Elder, *The Tower of Babel*, 1563



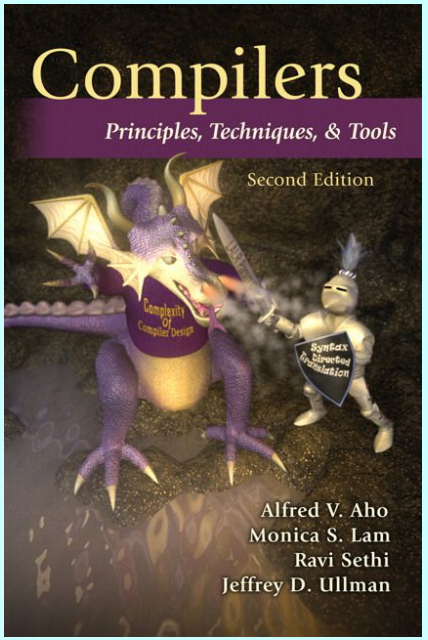


Table of Contents

References

Books

- J.E. Hopcroft, R. Motwani, J.D. Ullman : *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 2007
  - http://www-db.stanford.edu/~ullman/ialc.html

  (Italian Ed. : *Automi, linguaggi e calcolabilità,* Addison-Wesley, 2009)
  - https://www.pearson.it/opera/pearson/0-6576-automi_linguaggi_e_calcolabilita

- A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman : *Compilers: Principles, Techniques, and Tools - 2/E*, Addison-Wesley, 2007.
  - http://vig.pearsoned.co.uk/catalog/academic/product/0,1144,0321491696,00.html

  (Italian Ed. : *Compilatori: Principi, tecniche e strumenti – 2/Ed,* Addison-Wesley, 2009)
  - https://www.pearson.it/opera/pearson/0-3479-compilatori

Formal Language Classification: definitions (1)

- Alphabet
  - Finite (non-empty) set of symbols
    - $\Sigma_1 = \{0, 1\}$ the set of symbols in binary codes
    - $\Sigma_2 = \{\alpha, \beta, \gamma, \ldots, \omega\}$ the set of lower-case letters in Greek alphabet
    - $\Sigma_3 =$ the set of all ASCII characters
    - $\Sigma_4 =$ **{boy, girl, talks, the, ...}** a set of English terms
- String (word)
  - Finite sequence of symbols chosen from some alphabet
    - $s_1 = 0110001$; $s_2 = \delta\varepsilon\lambda\chi\pi\lambda$; $s_3 =$ **f7$1°Zp](*è** ; $s_4 =$ **the boy talks**
- Length of a string
  - Number of positions for symbols in the string
    - $|\ 0110001\ | = 7$
- Empty string (ε)
  - String of length zero
    - $|\ \varepsilon\ | = 0$

FLC: definitions (2)

Alphabet closure

- The set of all strings over an alphabet
  - closure operator (Kleene): *
    - $\{0, 1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
  - positive closure operator : $^+$
    - $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$
    - $\{0, 1\}^+ = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

Language

- A set of strings over a given alphabet
- $L \subseteq \Sigma^*$
  - $L_1 = \{ 0^n 1^n \mid n \geq 1\} = \{01, 0011, 000111, \ldots\}$
  - $L_2 = \{\varepsilon\}$
  - $L_3 = \varnothing$

FLC: grammars (1)

- A grammar is a 4-tuple $G = (N, T, P, S)$
  - N : alphabet of **non-terminal** symbols
  - T : alphabet of **terminal** symbols
    - $N \cap T = \varnothing$
    - $V = N \cup T$: alphabet (vocabulary) of the grammar
  - P : finite set of **rules (productions)**
    - $P = \{ \alpha \rightarrow \beta \mid \alpha \in V^+ ; \alpha \notin T^+ ; \beta \in V^* \}$
  - S : **start** (non-terminal) symbol
    - $S \in N$

FLC: grammars (2)

- Derivation

  let $\alpha \rightarrow \beta$ be a production of G

  - if $\sigma = \gamma\alpha\delta$ and $\tau = \gamma\beta\delta$

    then $\sigma \Rightarrow \tau$ (σ *produces* τ , τ *is derived from* σ)
  - if $\sigma_0 \Rightarrow \sigma_1 \Rightarrow \sigma_2 \ldots \Rightarrow \sigma_k$

    then $\sigma_0 \Rightarrow^* \sigma_k$

- Language produced by $G = (N, T, P, S)$
  - $L(G) = \{ w \mid w \in T^* \; ; \; S \Rightarrow^* w \}$
- Grammars that produce the same language are said equivalent

FLC: example of grammar (1)

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

$N = \{$ *<sentence>*, *<qualified noun>*, *<noun>*, *<pronoun>*, *<verb>*, *<adjective>* $\}$

$T = \{$ the, man, girl, boy, lecturer, he, she, talks, listens, mystifies, tall, thin, sleepy $\}$

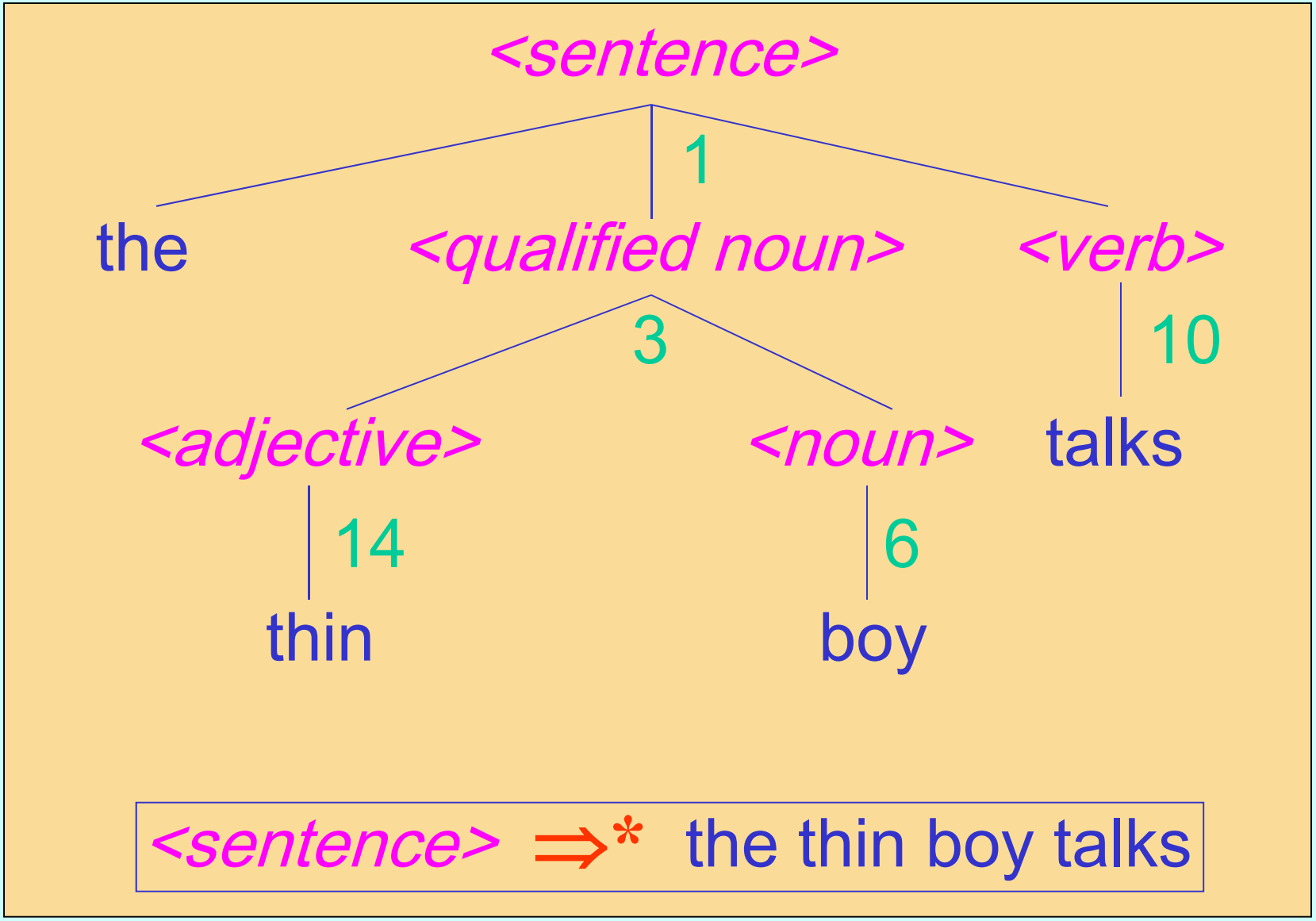
$P = \{$

<sentence>	→ the *<qualified noun> <verb>*	(1)
	| *<pronoun> <verb>*	(2)
<qualified noun>	→ *<adjective> <noun>*	(3)
<noun>	→ man | girl | boy | lecturer	(4, 5, 6, 7)
<pronoun>	→ he | she	(8, 9)
<verb>	→ talks | listens | mystifies	(10, 11, 12)
<adjective>	→ tall | thin | sleepy	(13, 14, 15)

$\}$

$S =$ *<sentence>*

FLC: example of grammar (2)



FLC: example of grammar (3)

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

$N = \{$ <goal> , <expression> , <term> , <factor> $\}$

$T = \{$ a , b , c , - , * $\}$

$P = \{$

<goal> → <expression> (1)

<expression> → <term> | <expression> - <term> (2, 3)

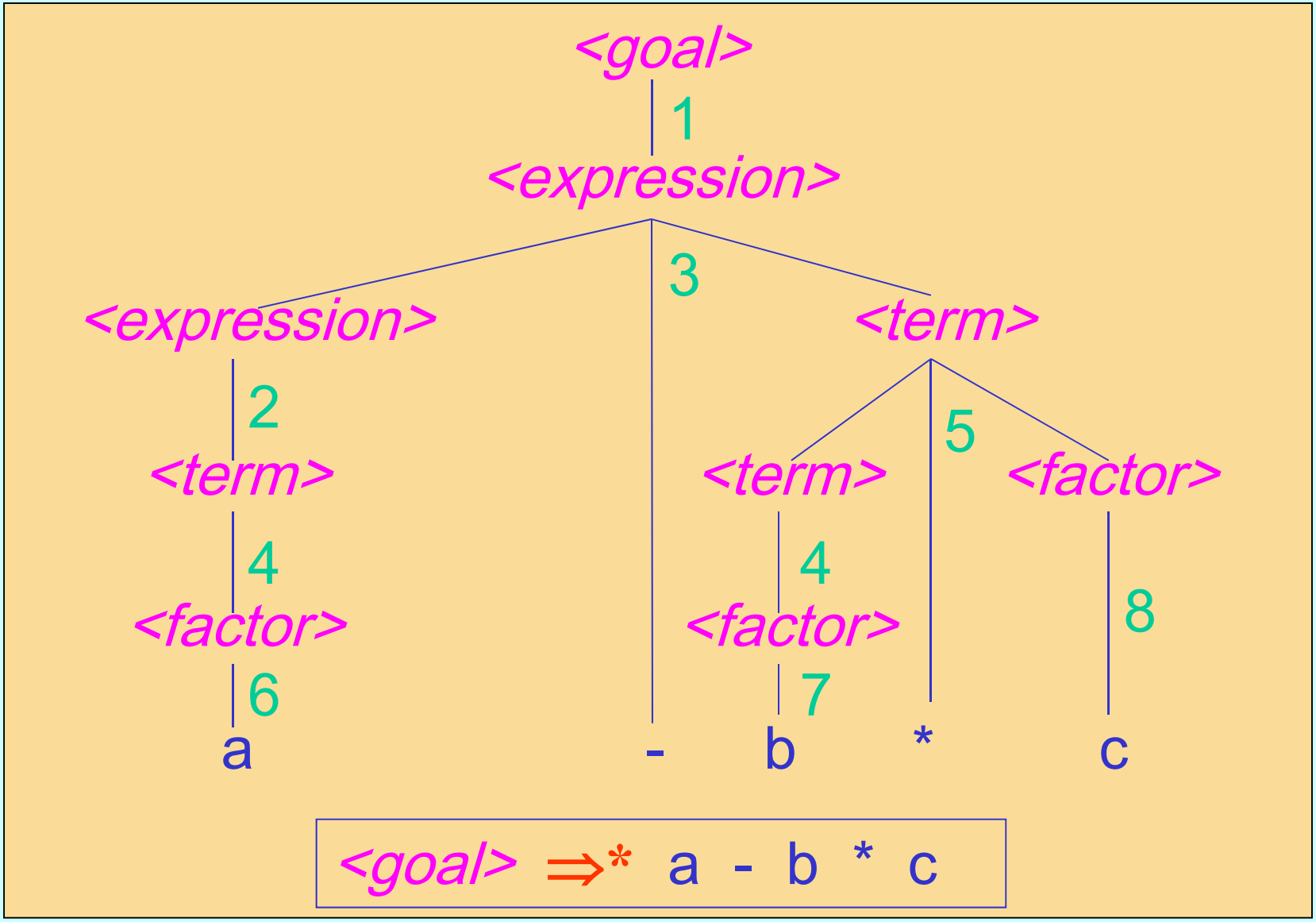
<term> → <factor> | <term> * <factor> (4, 5)

<factor> → a | b | c (6, 7, 8)

$\}$

$S =$ <goal>

FLC: example of grammar (4)



FLC: type 0 grammars (phrase-structure)

$$P = \{\ \alpha \rightarrow \beta \mid \alpha \in V^{+};\ \alpha \notin T^{+};\ \beta \in V^{*}\ \}$$

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

$P = \{$

S	$\rightarrow$	$a\,A\,b$	(1)
$a\,A$	$\rightarrow$	$a\,a\,A\,b$	(2)
A	$\rightarrow$	ε	(3)

$\}$

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

$S \rightarrow a\,A\,b \Rightarrow a\,b$

$\Rightarrow a\,a\,A\,b\,b \Rightarrow a\,a\,b\,b$

$\Rightarrow a\,a\,a\,A\,b\,b\,b \Rightarrow a\,a\,a\,b\,b\,b$

$\Rightarrow \ldots$

FLC: type 1 grammars (context-sensitive)

$$P = \{\ \alpha \rightarrow \beta \mid \alpha \in V^+;\ \alpha \notin T^+;\ \beta \in V^+;\ |\alpha| \leq |\beta|\ \}$$

$G = (\{B, C, S\}, \{a, b, c\}, P, S)$

$P = \{$

S	$\rightarrow$	$a\,S\,B\,C \mid a\,b\,C$	(1, 2)
$C\,B$	$\rightarrow$	$B\,C$	(3)
$b\,B$	$\rightarrow$	$b\,b$	(4)
$b\,C$	$\rightarrow$	$b\,c$	(5)
$c\,C$	$\rightarrow$	$c\,c$	(6)

$\}$

$$L(G) = \{\ a^n\ b^n\ c^n \mid\ n \geq 1\ \}$$

FLC: type 2 grammars (context-free) (1)

$$P = \{\ A \rightarrow \beta \mid A \in N\ ;\ \beta \in V^{+}\}$$

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

$P = \{$

S	$\rightarrow$	a B | b A	(1, 2)
A	$\rightarrow$	a S | b A A | a	(3, 4, 5)
B	$\rightarrow$	b S | a B B | b	(6, 7, 8)

$\}$

L(G) = the set of strings in { **a, b** }$^{+}$ where the number of **"a"** equals the number of **"b"**

FLC: type 2 grammars (context-free) (2)

$G = (\{O, X\}, \{a, +, -, *, /\}, P, X)$

$P = \{$

$X \rightarrow XXO \mid a$ (1, 2)

$O \rightarrow + \mid - \mid * \mid /$ (3, 4, 5, 6)

$\}$

$L(G)$ = the set of arithmetic expressions with binary operators in **postfix polish notation**

FLC: linear grammars

$$P = \{ A \rightarrow x\,B\,y\,,\ A \rightarrow x \mid A, B \in N\ ;\ x, y \in T^{+} \}$$

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

$P = \{\ S \rightarrow a\,S\,b \mid a\,b$ (1, 2)

$\}$

$$L(G) = \{ a^n\,b^n \mid n \geq 1 \}$$

FLC: type 3 grammars (right-linear and left-linear)

- Right-linear grammars

$$P = \{ A \rightarrow x\,B \;,\; A \rightarrow x \;\mid\; A, B \in N \;;\; x \in T^{+} \}$$

- Left-linear grammars

$$P = \{ A \rightarrow B\,x \;,\; A \rightarrow x \;\mid\; A, B \in N \;;\; x \in T^{+} \}$$

FLC: type 3 grammars (right-regular)

$$P = \{ A \rightarrow a\,B\ ,\ A \rightarrow a \mid A, B \in N\ ;\ a \in T \}$$

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

$P = \{$

S	$\rightarrow$	$a\,A \mid b\,C$	(1, 2)
A	$\rightarrow$	$a\,S \mid b\,B \mid a$	(3, 4, 5)
B	$\rightarrow$	$a\,C \mid b\,A$	(6, 7)
C	$\rightarrow$	$a\,B \mid b\,S \mid b$	(8, 9, 10)

$\}$

$L(G)$ = the set of strings in $\{$ **a, b** $\}^+$ where both the number of **"a"** , and the number of **"b"** are even

FLC: type 3 grammars (left-regular)

$$P = \{ A \rightarrow B\,a , A \rightarrow a \mid A, B \in N \; ; \; a \in T \}$$

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

$P = \{$

$S \rightarrow Aa \mid Sa \mid Sb$	(1, 2, 3)
$A \rightarrow Bb$	(4)
$B \rightarrow Ba \mid Ca \mid a$	(5, 6, 7)
$C \rightarrow Ab \mid Cb \mid b$	(8, 9, 10)

$\}$

$L(G)$ = the set of strings in $\{ a, b \}^+$ containing "a b a"

FLC: equivalence among type 3 grammars

- ***right-linear*** and ***right-regular*** grammars are equivalent

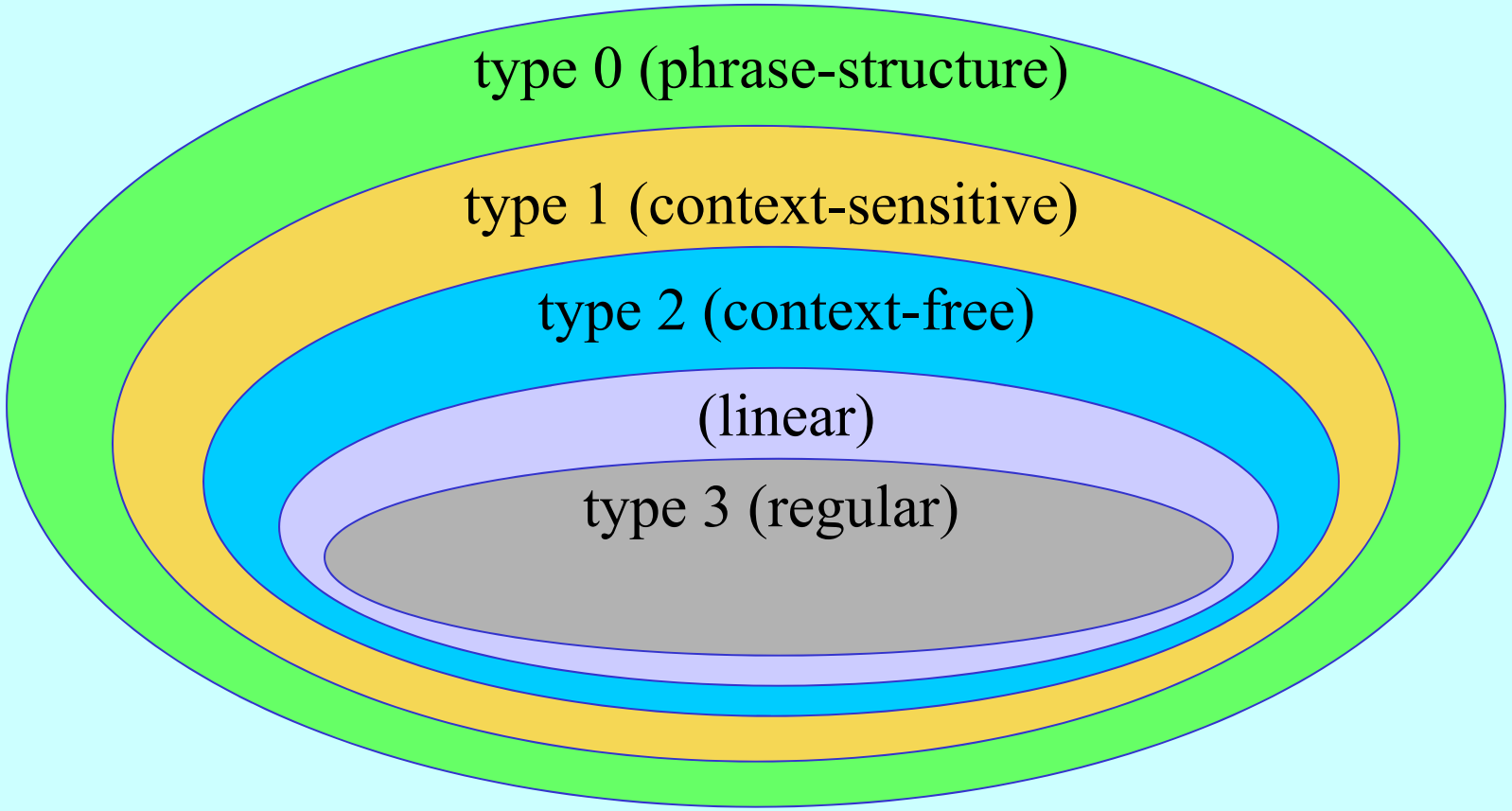
$$A \rightarrow a\,b\,c\,B \equiv \{A \rightarrow a\,C,\ C \rightarrow b\,c\,B\} \equiv \{A \rightarrow a\,C,\ C \rightarrow b\,D,\ D \rightarrow c\,B\}$$

- ***left-linear*** and ***left-regular*** grammars are equivalent

$$A \rightarrow B\,a\,b\,c \equiv \{A \rightarrow C\,c,\ C \rightarrow B\,a\,b\} \equiv \{A \rightarrow C\,c,\ C \rightarrow D\,b,\ D \rightarrow B\,a\}$$

FLC: language classification (Chomsky hierarchy)

- A *language* is *type-**n*** if it can be produced by a *type-**n** grammar*



Regular Languages: regular sets

- The following sets are ***regular sets*** over an alphabet Σ
  - the empty set $\emptyset$
  - the set $\{\varepsilon\}$ containing the empty string
  - the set $\{a\}$ containing any symbol $a \in \Sigma$
- If P and Q are regular sets over Σ , the same is true for
  - the union $P \cup Q$
  - the concatenation $P\,Q = \{ x\,y \mid x \in P\,;\, y \in Q \}$
  - the closures P^* e Q^*

RL: regular expressions

- The following expressions are ***regular expressions*** over an alphabet Σ
  - the expression φ , denoting the empty set $\varnothing$
  - the expression ε , denoting the set $\{\varepsilon\}$
  - the expression a , denoting the set $\{a\}$ where $a \in \Sigma$
- If p and q are regular expressions denoting the sets P and Q , then also the following are regular expressions
  - the expression $p \mid q$, denoting the set $P \cup Q$
  - the expression $p\ q$, denoting the set $P\,Q$
  - the expressions p^* e q^* , denoting the sets P^* e Q^*

RL: examples of regular expressions

- the set of strings over **{0,1}** containing two **1**'s
  - **0*10*10***
- the strings over **{0,1}** without consecutive equal symbols
  - **(1 | ε) (01)* (0 | ε)**
- the set of decimal characters
  - **digit = 0 | 1 | 2 | … | 9**
- the set of strings representing decimal integers
  - **digit digit***
- the set of alphabetic characters
  - **letter = A | B | … | Z | a | b | … | z**
- the set of strings representing identifiers
  - **letter (letter | digit)***

RL: regular sets / regular expressions

René Magritte, *This is not a pipe*, 1948

RL: algebraic properties of regular expressions

- two **regular expressions** are ***equivalent*** if they denote the same **regular set**

- $\alpha \mid \beta = \beta \mid \alpha$ (commutative property)
- $\alpha \mid (\beta \mid \gamma) = (\alpha \mid \beta) \mid \gamma$ (associative property)
- $\alpha (\beta \ \gamma) = (\alpha \ \beta) \gamma$ (associative property)
- $\alpha (\beta \mid \gamma) = \alpha \beta \mid \alpha \gamma$ (distributive property)
- $(\alpha \mid \beta) \gamma = \alpha \gamma \mid \beta \gamma$ (distributive property)
- $\alpha \mid \varphi = \alpha$
- $\alpha \varepsilon = \varepsilon \alpha = \alpha$
- $\alpha \varphi = \varphi \alpha = \varphi$
- $\alpha \mid \alpha = \alpha$
- $\varphi^* = \varepsilon^* = \varepsilon$
- $\alpha^* = \alpha^* \alpha^* = (\alpha^*)^* = \alpha \alpha^* \mid \varepsilon$

RL: equations of regular expressions

- if α e β are regular expressions, $X = \alpha X \mid \beta$ is an equation with unknown X
- $X = \alpha^* \beta$ is a solution of the equation
  - $\alpha X \mid \beta = \alpha \alpha^* \beta \mid \beta = (\alpha \alpha^* \mid \varepsilon) \beta = \alpha^* \beta = X$
- a set of equations with unknowns $\{X_1, X_2, \ldots, X_n\}$ is composed by n equations such as:

$$X_i = \alpha_{i0} \mid \alpha_{i1} X_1 \mid \alpha_{i2} X_2 \mid \ldots \mid \alpha_{in} X_n$$

where each α_{ij} is a regular expression over any alphabet without the unknowns

RL: solution of sets of equations

```
{
 for (int i=1 ; i<n ; i++) {
   put the equation i in the form X_i = α X_i | β;
   substitute X_i with α* β in the equations i+1...n;
 }
 for (int i=n ; i>0 ; i--) {
   // the i-th equation is in the form X_i = α X_i | β
   // where α and β do not contain unknowns
   solve the i-th equation: X_i = α* β;
   substitute X_i with α* β in the equations i-1...1;
 }
}
```

RL: example of solution of sets of equations

$$
\begin{cases} A = 1A \mid 0B \\ B = 1A \mid 0C \mid 0 \\ C = 0C \mid 1C \mid 0 \mid 1 \end{cases}
$$

$$
A = 1^*0B
$$

$$
B = 11^*0B \mid 0C \mid 0 \;\Rightarrow\; B = (11^*0)^*(0C \mid 0)
$$

$$
C = (0 \mid 1)C \mid 0 \mid 1 \;\Rightarrow\; C = (0 \mid 1)^*(0 \mid 1)
$$

$$
\begin{cases} C = (0 \mid 1)^*(0 \mid 1) \\ B = (11^*0)^*(0(0 \mid 1)^*(0 \mid 1) \mid 0) \\ A = 1^*0(11^*0)^*(0(0 \mid 1)^*(0 \mid 1) \mid 0) \end{cases}
$$

RL: right-linear languages ⊆ regular sets

- let $G = (\{A_1, A_2, \ldots, A_n\}, T, P, A_1)$ be a right-linear grammar
- let us transform each rule of the grammar:

$$A_i \rightarrow \alpha_{i0} \mid \alpha_{i1} A_1 \mid \alpha_{i2} A_2 \mid \ldots \mid \alpha_{in} A_n$$

into an equation of regular expressions :

$$A_i = \alpha_{i0} \mid \alpha_{i1} A_1 \mid \alpha_{i2} A_2 \mid \ldots \mid \alpha_{in} A_n$$

- let us solve the resulting set of equations
- the language $L(G)$ generated by the grammar is denoted by the regular expression corresponding to the symbol A_1

RL: regular expression of a right-linear language

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

$$P = \begin{cases} S \to 0A \mid 1S \mid 0 \\ A \to 0B \mid 1A \\ B \to 0S \mid 1B \end{cases} \quad \Rightarrow \quad \begin{cases} S = 0A \mid 1S \mid 0 \\ A = 0B \mid 1A \\ B = 0S \mid 1B \end{cases}$$

$B = 1^*0S$

$A = 1^*0B = 1^*01^*0S$

$S = 01^*01^*0S \mid 1S \mid 0 = (01^*01^*0 \mid 1)S \mid 0$

$S = (01^*01^*0 \mid 1)^*0$ **denotes** $L(G)$

RL: regular sets $\subseteq$ right-linear languages (1)

- the ***regular sets*** : $\varnothing$, $\{\varepsilon\}$, $\{ \mathtt{a} \mid \mathtt{a} \in \Sigma \}$ can be generated by right-linear grammars
  - $G_1 = (\{S\}, \Sigma, \varnothing, S) \quad \Rightarrow \quad L(G_1) = \varnothing$
  - $G_2 = (\{S\}, \Sigma, \{S \to \varepsilon\}, S) \quad \Rightarrow \quad L(G_2) = \{\varepsilon\}$
  - $G_3 = (\{S\}, \Sigma, \{S \to \mathtt{a} \mid \mathtt{a} \in \Sigma\}, S)$

    $\Rightarrow \quad L(G_3) = \{\mathtt{a}\}$ where $\mathtt{a} \in \Sigma$

RL: regular sets $\subseteq$ right-linear languages (2)

- let $G_1 = (N_1, \Sigma, P_1, S_1)$ and $G_2 = (N_2, \Sigma, P_2, S_2)$ be right-linear grammars where $N_1 \cap N_2 = \varnothing$
- the language $L(G_1) \cup L(G_2) = L(G_4)$ is a right-linear language
  - $G_4 = (N_1 \cup N_2 \cup \{S_4\}, \Sigma, P_1 \cup P_2 \cup \{S_4 \to S_1 \mid S_2\}, S_4)$
- the language $L(G_1)\,L(G_2) = L(G_5)$ is a right-linear language
  - $G_5 = (N_1 \cup N_2, \Sigma, P_2 \cup P_5, S_1)$; $P_5 = \{ A \to x\,B$ if $A \to x\,B \in P_1$
    $A \to x\,S_2$ if $A \to x \in P_1 \}$
- the language $L(G_1)^* = L(G_6)$ is a right-linear language
  - $G_6 = (N_1 \cup \{S_6\}, \Sigma, \{S_6 \to S_1 \mid \varepsilon\} \cup P_6, S_6)$;
    $P_6 = \{ A \to x\,B$ if $A \to x\,B \in P_1$
    $A \to x\,S_6$ if $A \to x \in P_1 \}$

RL: regular sets ≡ right/left-linear languages

- *right-linear languages* ⊆ *regular sets*
- *regular sets* ⊆ *right-linear languages*
- *right-linear languages* ≡ *regular sets*
- *left-linear languages* ≡ *regular sets*
  - the equation of regular expressions $X = X\,\alpha \mid \beta$ has the solution $X = \beta\,\alpha^*$
  - it is possible to solve sets of equations corresponding to the rules of a left-linear grammar
  - it is possible to define left-linear grammars that generate any regular set
- *right-linear languages* ≡ *left-linear languages*

RL: regular expression of a left-linear language

$G = (\{U, V, Z\}, \{0,1\}, P, Z)$

$$P = \{\ Z \to U\,0 \mid V\,1 \quad U \to Z\,1 \mid 0 \quad V \to Z\,0 \mid 1\ \} \Rightarrow \{\ Z = U\,0 \mid V\,1 \quad U = Z\,1 \mid 0 \quad V = Z\,0 \mid 1\ \}$$

$$Z = (Z\,1 \mid 0)0 \mid (Z\,0 \mid 1)1 =$$
$$= Z\,10 \mid 00 \mid Z\,01 \mid 11 =$$
$$= Z\,(10 \mid 01) \mid 00 \mid 11$$

$Z = (00 \mid 11)\,(10 \mid 01)^*$ **denotes** $L(G)$

RL: deterministic finite automata (DFA)

- A **DFA** is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$
  - Q : finite (non empty) set of **states**
  - Σ : alphabet of **input** symbols
  - δ : **transition** function
    - δ: $Q \times \Sigma \rightarrow Q$
  - q_0 : **start** state
    - $q_0 \in Q$
  - F : set of **final states**
    - $F \subseteq Q$

RL: an example of DFA

$A = (Q, \Sigma, \delta, q_0, F)$

$Q = \{ q_0 , q_1 , q_2 , q_3 \}$

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

$\delta(q_0 , 0) = q_2$	$\delta(q_0 , 1) = q_1$
$\delta(q_1 , 0) = q_3$	$\delta(q_1 , 1) = q_0$
$\delta(q_2 , 0) = q_0$	$\delta(q_2 , 1) = q_3$
$\delta(q_3 , 0) = q_1$	$\delta(q_3 , 1) = q_2$

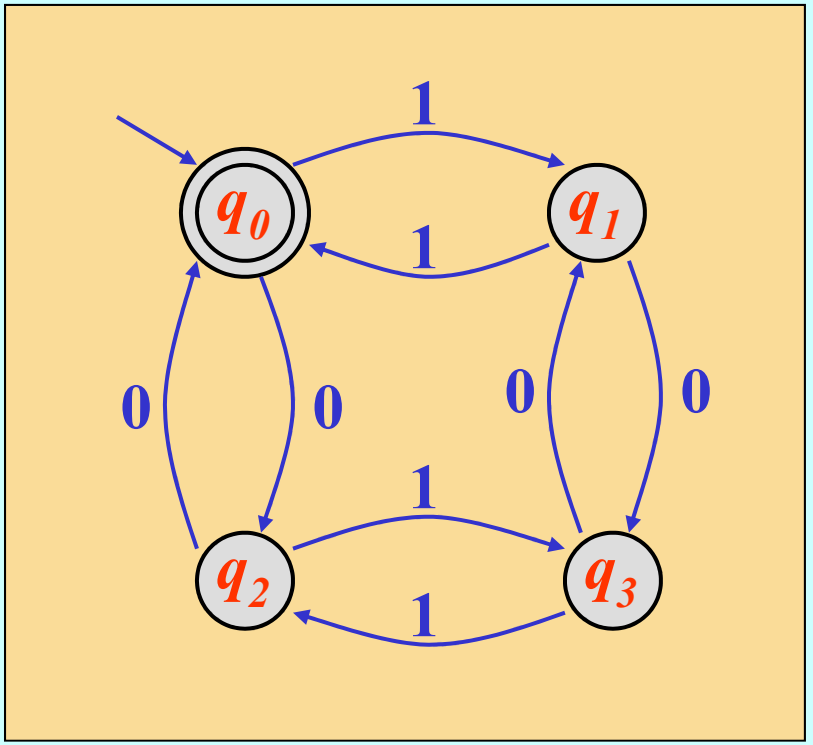
$F = \{ q_0 \}$

RL: transition table / diagram of a DFA

- transition table
  - tabular representation of the transition function
- transition diagram: a *graph* where
  - for each state in the automaton there is a node
  - for each transition $\delta(p, a) = q$ there is an arc from p to q labeled a
  - the start state has an entering non labeled arc
  - the final states are marked by a double circle

RL: representations of a DFA

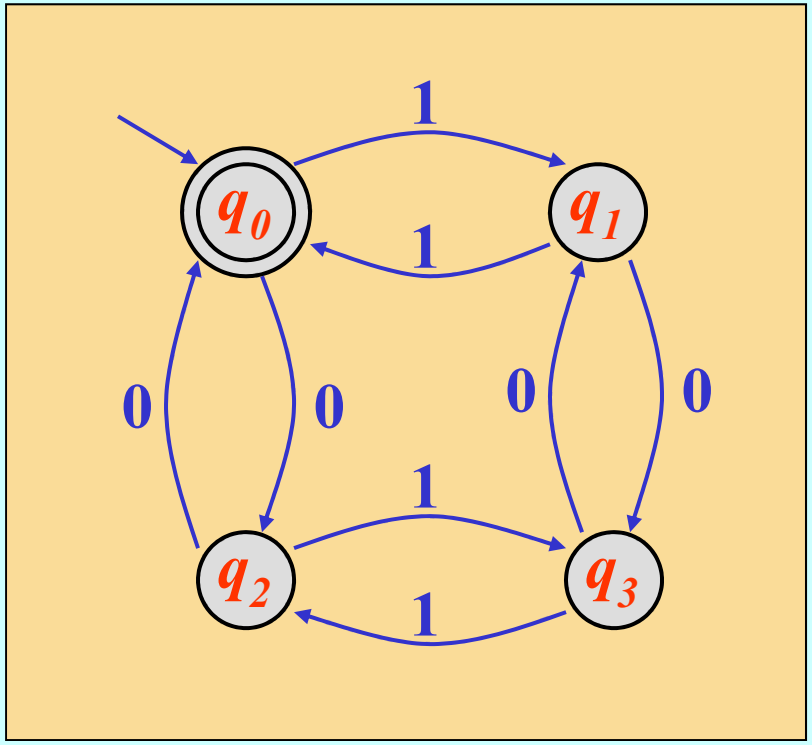
	0	1
$\rightarrow {}^{*} q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2



RL: the language accepted by a DFA

- The domain of function δ can be extended from $Q \times \Sigma$ to $Q \times \Sigma^*$
  - $\delta(q, \varepsilon) = q$
  - $\delta(q, aw) = \delta(\delta(q, a), w)$ where $a \in \Sigma$; $w \in \Sigma^*$
- Language accepted by $A = (Q, \Sigma, \delta, q_0, F)$
  - $L(A) = \{ w \mid w \in \Sigma^* \ ; \ \delta(q_0, w) \in F \}$

RL: how a DFA accepts strings

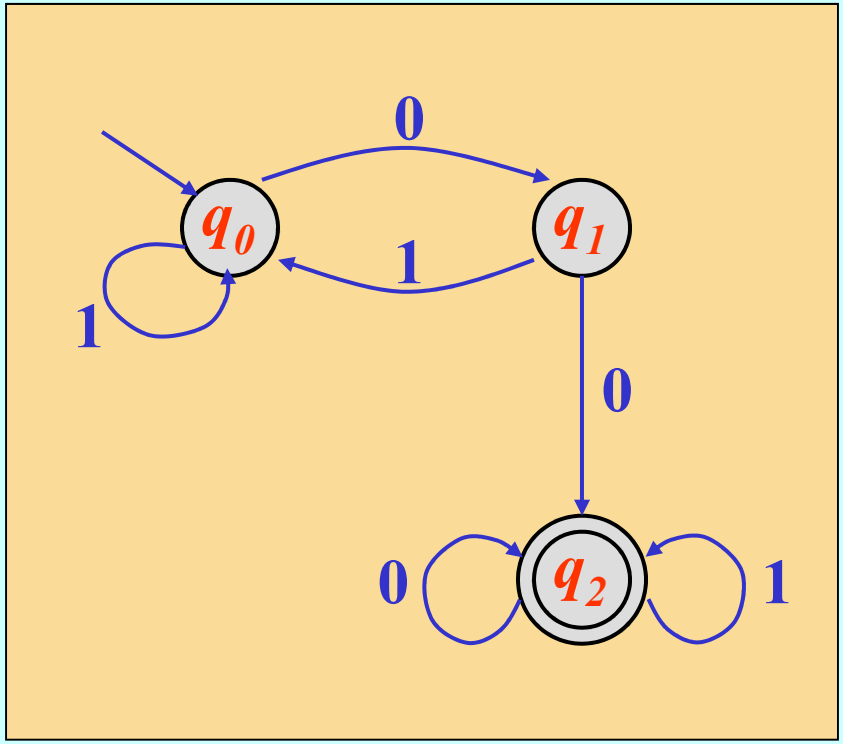


$011101 \in L(A)$

$01101 \notin L(A)$

RL: examples of DFA (1)

- $L(A)$ = the set of all strings over **{0,1}** with at least two consecutive **0**'s

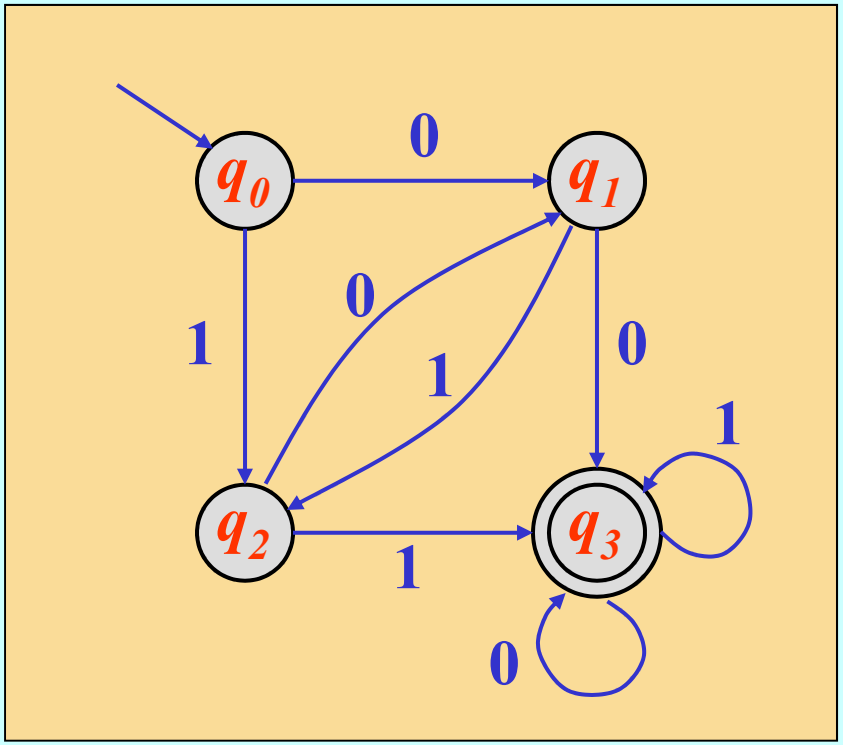


q_0 : strings that do not end in **0**

q_1 : strings that end with only one **0**

RL: examples of DFA (2)

- $L(A)$ = the set of all strings over **{0,1}** with at least two consecutive **0**'s or two consecutive **1**'s



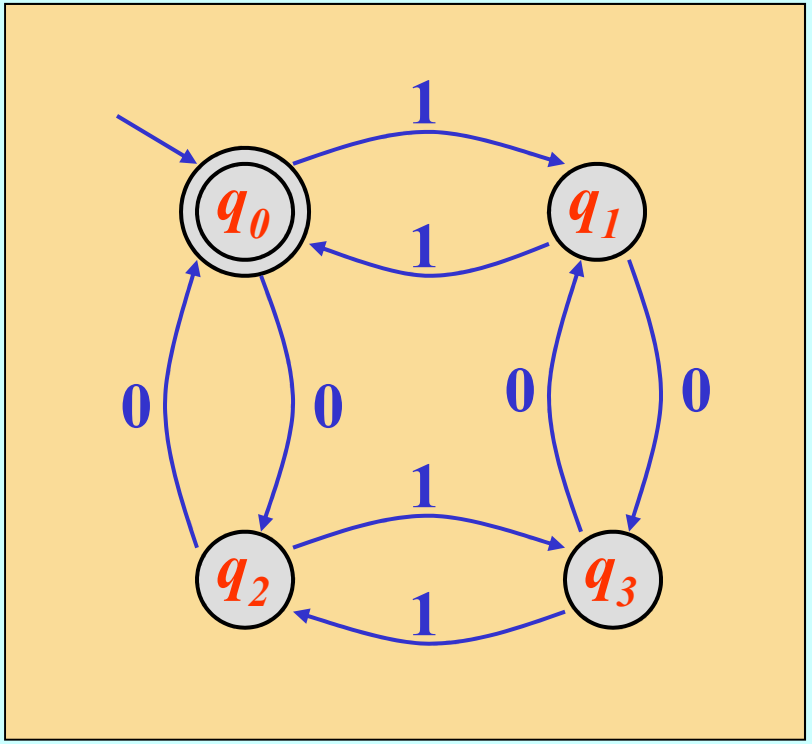
q_0 : strings that do not end in **0** or in **1**

q_1 : strings that end with only one **0**

q_2 : strings that end with only one **1**

RL: examples of DFA (3)

- $L(A)$ = the set of all strings over **{0,1}** having both an even number of **0**'s and an even number of **1**'s



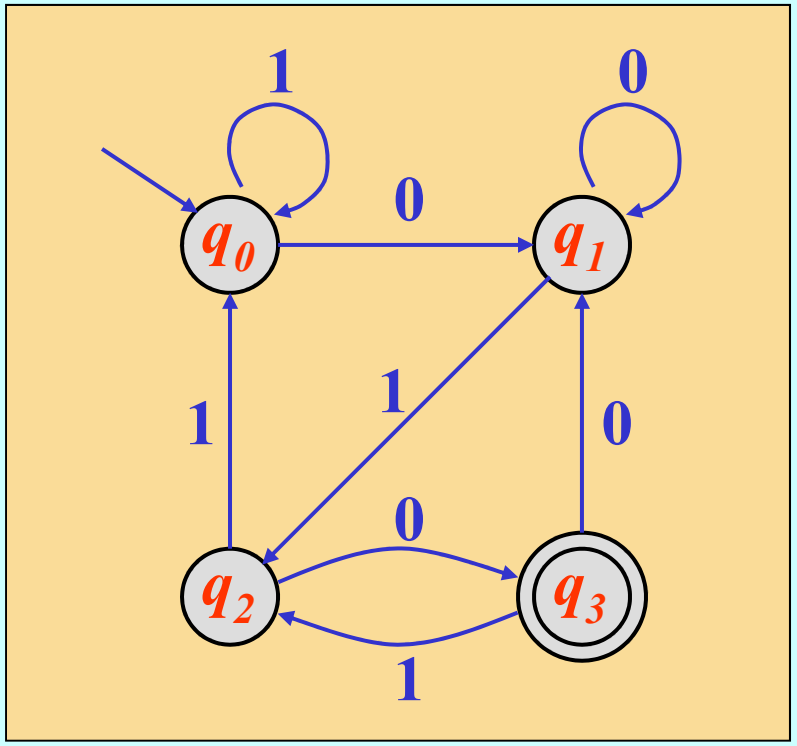
q_1 : strings with even # of **0**'s and odd # of **1**'s

q_2 : strings with odd # of **0**'s and even # of **1**'s

q_3 : strings with odd # of **0**'s and odd # of **1**'s

RL: examples of DFA (4)

- $L(A)$ = the set of all strings over **{0,1}** ending in **" 010 "**



q_0 : strings not ending in **0** or in **01**

q_1 : strings ending in **0** but not in **010**

q_2 : strings ending in **01**

RL: examples of DFA (5)

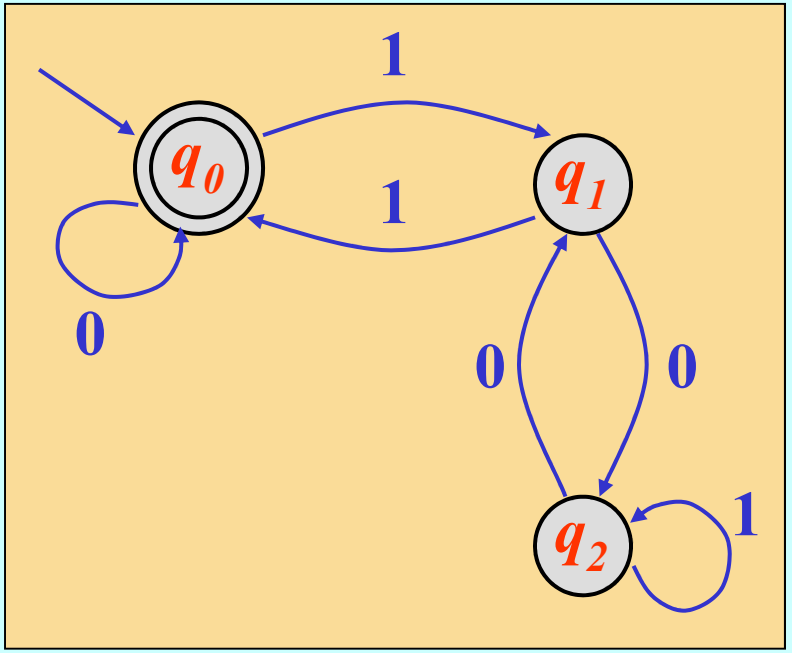
- $L(A)$ = the set of all strings that represent positive binary integers (*pbi*) multiple of 3

$pbi \in \{0,1\}^*$

$\underline{pbi} = (\underline{pbi} \text{ div } 3) \times 3 + (\underline{pbi} \bmod 3)$

$pbi\ 0 := 2 \times \underline{pbi}$

$pbi\ 1 := 2 \times \underline{pbi} + 1$



q_0 : integers that give remainder **0** when divided by **3**

q_1 : integers that give remainder **1** when divided by **3**

q_2 : integers that give remainder **2** when divided by **3**

RL: non-deterministic finite automata (NFA)

➢ An **NFA** is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$

- Q : finite (non empty) set of **states**
- Σ : alphabet of **input** symbols
- δ : **transition** function
  - $\delta: \; Q \times \Sigma \rightarrow \wp(Q)$
- q_0 : **start** state
  - $q_0 \in Q$
- F : set of **final states**
  - $F \subseteq Q$

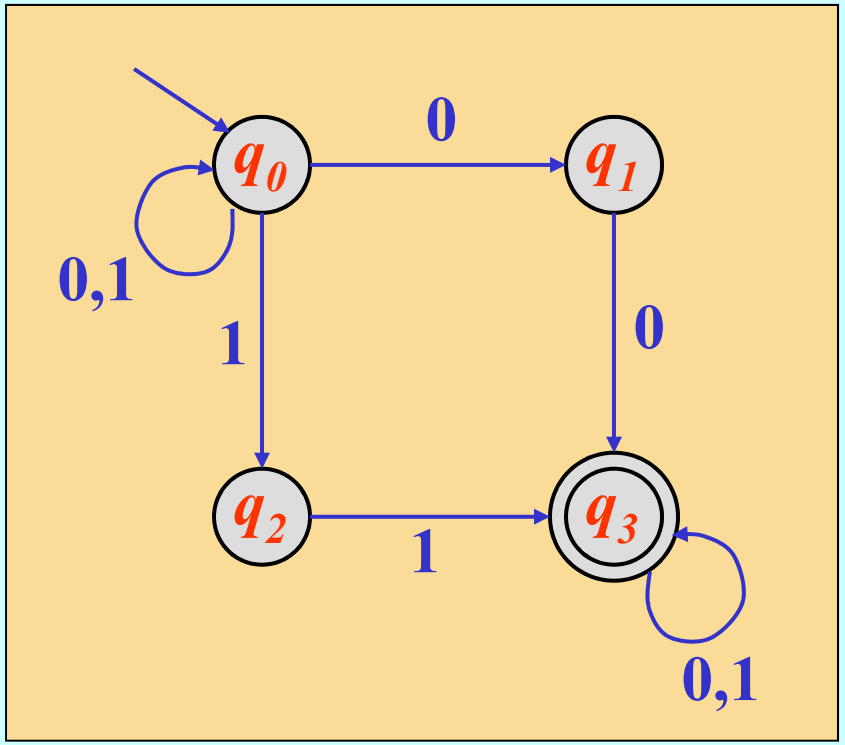
$\wp(Q)$: *power set* of Q
(the set of all subsets)
$\| \wp(Q) \| = 2^{\|Q\|}$

RL: the language accepted by an NFA

- The domain of function δ can be extended from $Q \times \Sigma$ to $Q \times \Sigma^*$ to $\wp(Q) \times \Sigma^*$
  - $\delta(q, \varepsilon) = \{q\}$
  - $\delta(q, aw) = \cup_i \delta(p_i, w)$ where $p_i \in \delta(q, a)$
  - $\delta(\{q_1, q_2, \ldots, q_n\}, w) = \cup_j \delta(q_j, w)$
- Language accepted by $A = (Q, \Sigma, \delta, q_0, F)$
  - $L(A) = \{ w \mid w \in \Sigma^* \; ; \; \delta(q_0, w) \cap F \neq \varnothing \}$
- a **DFA** is a special case of **NFA**

RL: examples of NFA (1)

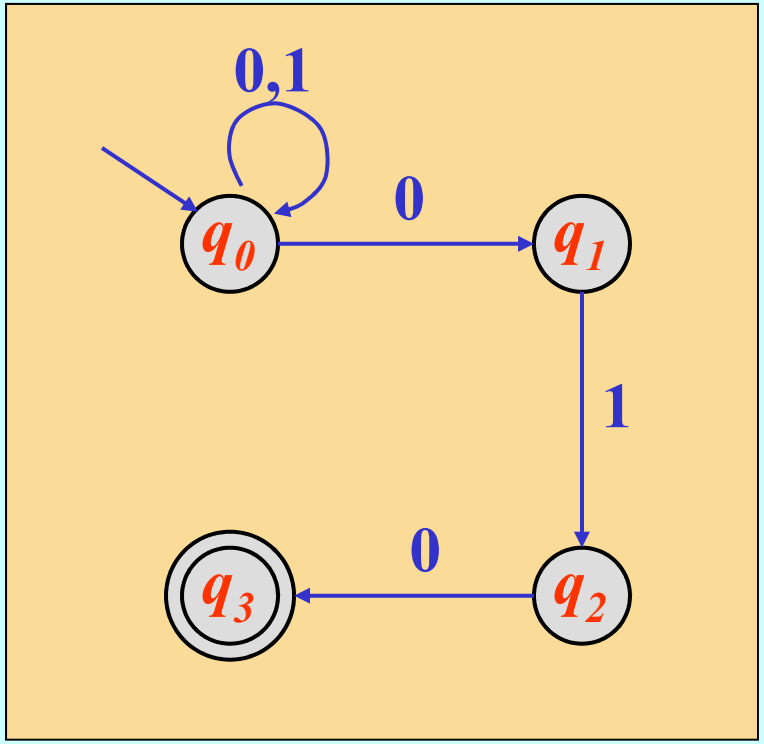
- ***L(A)*** = the set of all strings over **{0,1}** with at least two consecutive **0**'s or two consecutive **1**'s



(0 | 1)* (00 | 11) (0 | 1)*

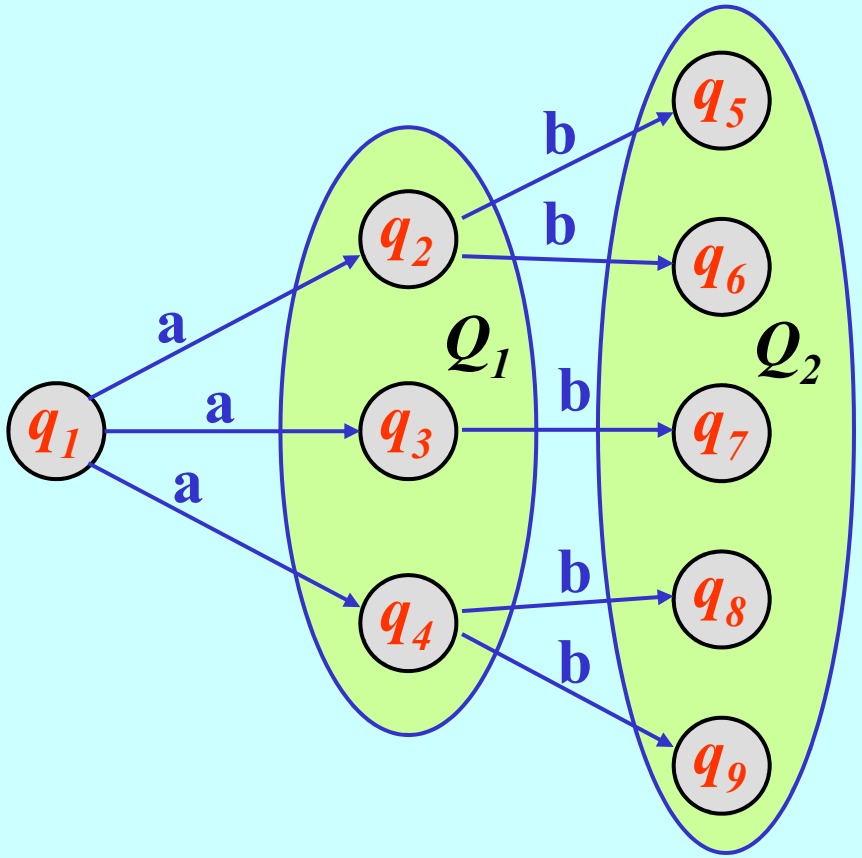
RL: examples of NFA (2)

- $L(A)$ = the set of all strings over **{0,1}** ending in **" 010 "**

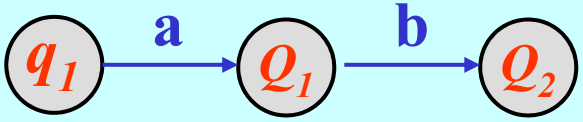


(0 | 1)* 010

RL: equivalence of NFA and DFA (1)



≡



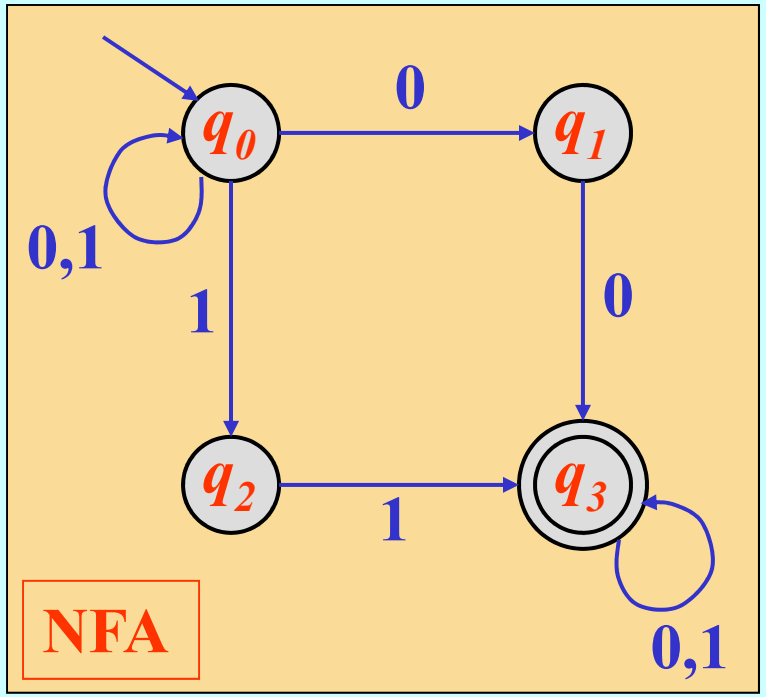
$$Q_1 = \{ q_2, q_3, q_4 \}$$

$$Q_2 = \{ q_5, q_6, q_7, q_8, q_9 \}$$

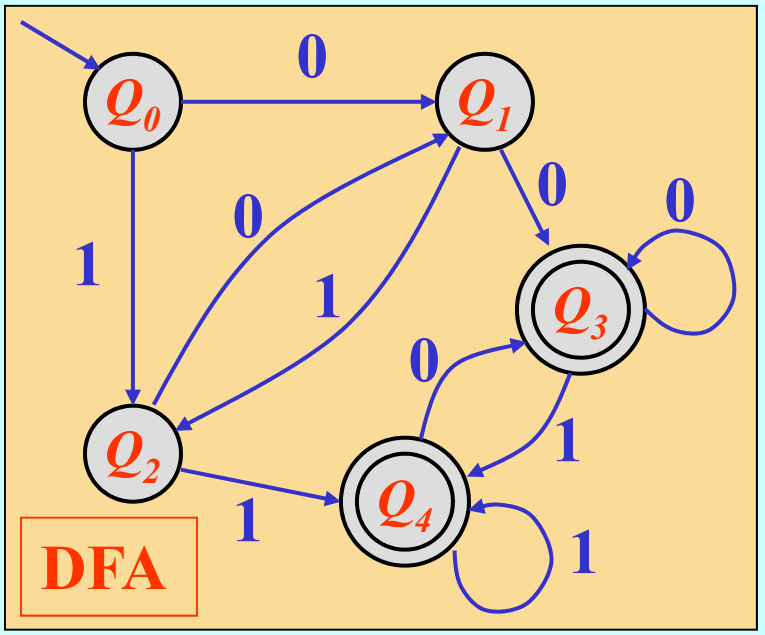
RL: equivalence of NFA and DFA (2)

- let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ be an **NFA**
- let us construct a **DFA** $D = (Q_D, \Sigma, \delta_D, \{ q_0 \}, F_D)$
  - $Q_D \subseteq \wp(Q_N)$
  - $\delta_D(S, a) = \cup_i \delta_N(p_i, a)$ where $p_i \in S \in Q_D$
  - $F_D = \{ S \mid S \in Q_D ; \; S \cap F_N \neq \varnothing \}$
- by construction $L(D) = L(N)$
- **NFA ≡ DFA** (**FA** : finite automata)

RL: constructing a DFA from an NFA (1)

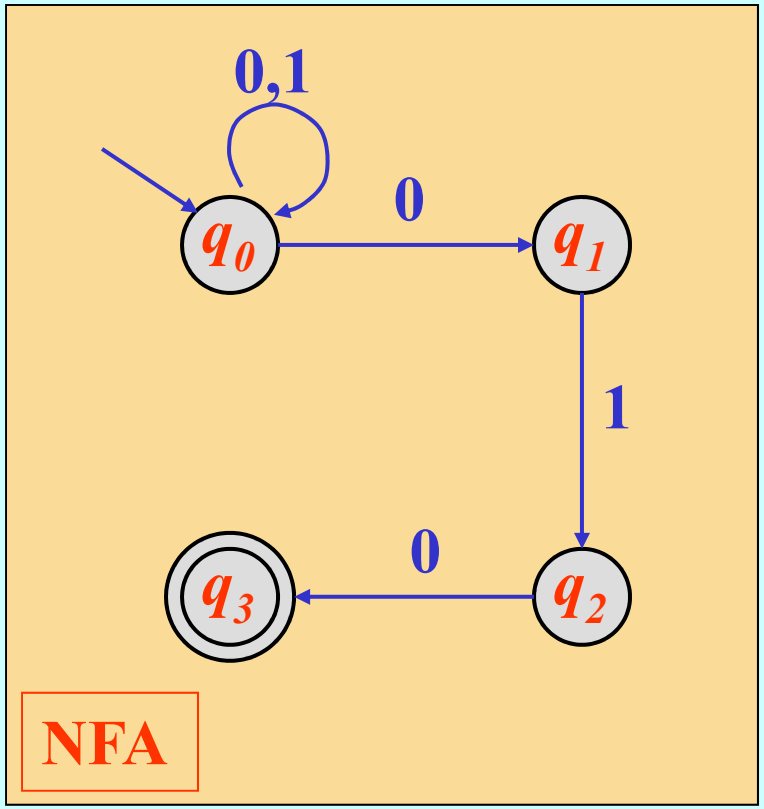


		0	1
Q_0	$\rightarrow\{q_0\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
Q_1	$\{q_0,q_1\}$	$\{q_0,q_1,q_3\}$	$\{q_0,q_2\}$
Q_2	$\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0,q_2,q_3\}$
Q_3	$*\{q_0,q_1,q_3\}$	$\{q_0,q_1,q_3\}$	$\{q_0,q_2,q_3\}$
Q_4	$*\{q_0,q_2,q_3\}$	$\{q_0,q_1,q_3\}$	$\{q_0,q_2,q_3\}$

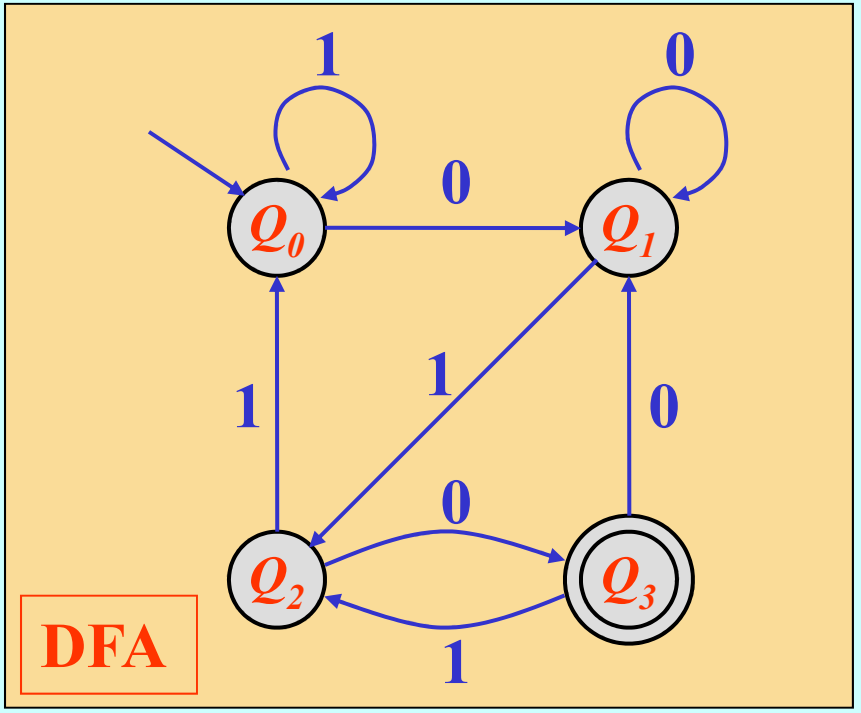


(0 | 1)* (00 | 11) (0 | 1)*

RL: constructing a DFA from an NFA (2)



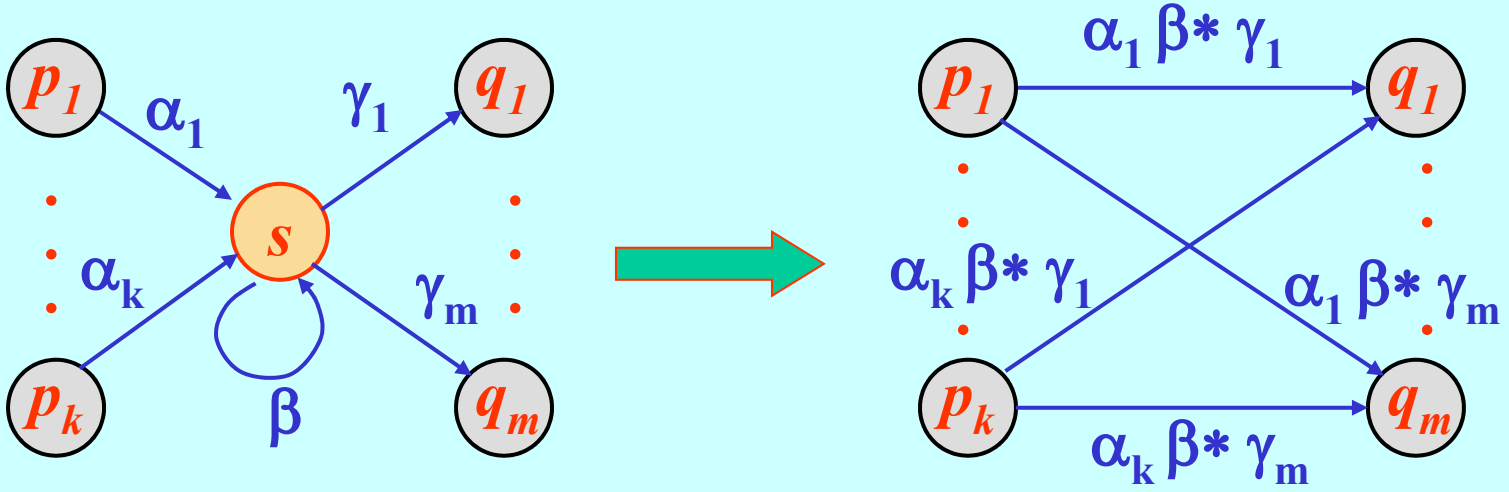
		0	1
Q_0	$\rightarrow\{q_0\}$	$\{q_0,q_1\}$	$\{q_0\}$
Q_1	$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
Q_2	$\{q_0,q_2\}$	$\{q_0,q_1,q_3\}$	$\{q_0\}$
Q_3	$*\{q_0,q_1,q_3\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$



(0 | 1)* 010

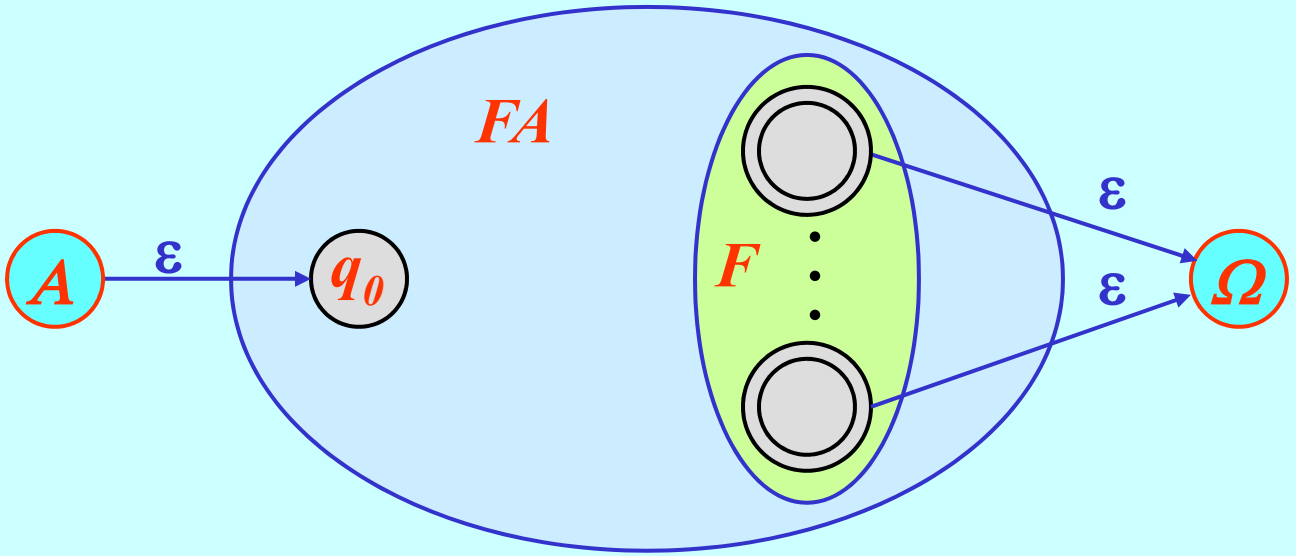
RL: FA languages ⊆ regular sets (1)

- it is possible to ***eliminate*** states in an **FA**
  - maintaining all the paths
  - labeling the transitions with regular expressions



RL: FA languages ⊆ regular sets (2)

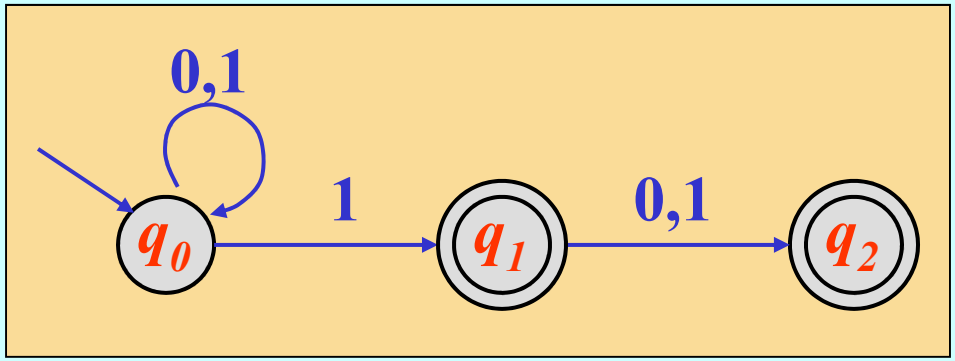
- given a finite state automaton $FA = (Q, \Sigma, \delta, q_0, F)$, add an initial state A and a final state Ω



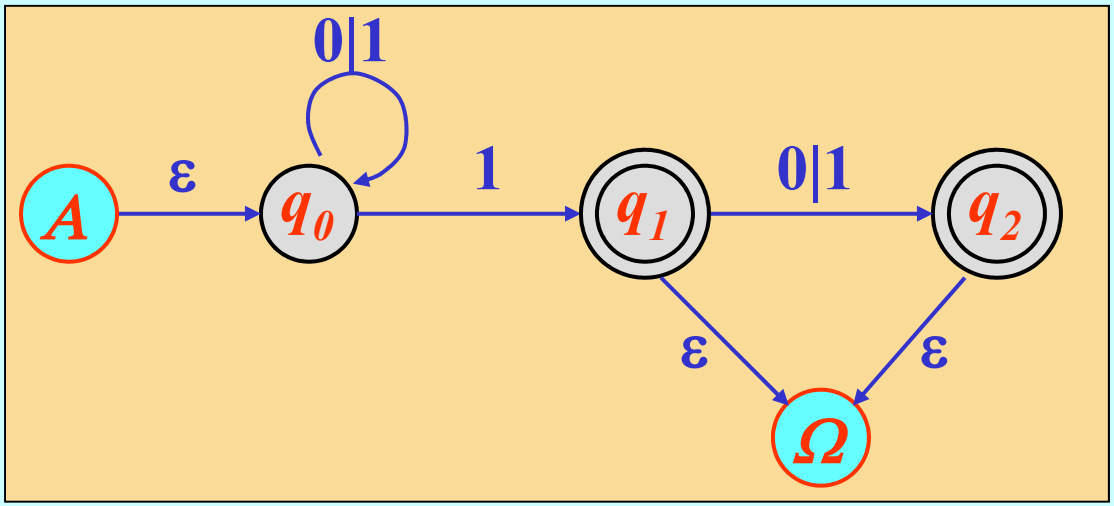
- eliminate all the states in FA
- the union of the labels on the transitions from A to Ω gives the regular expression of the language $L(FA)$

RL: from FA to regular expressions (1)

- ***L(A)*** = the set of all strings over **{0,1}** containing a "**1**" in the first or second position from the end

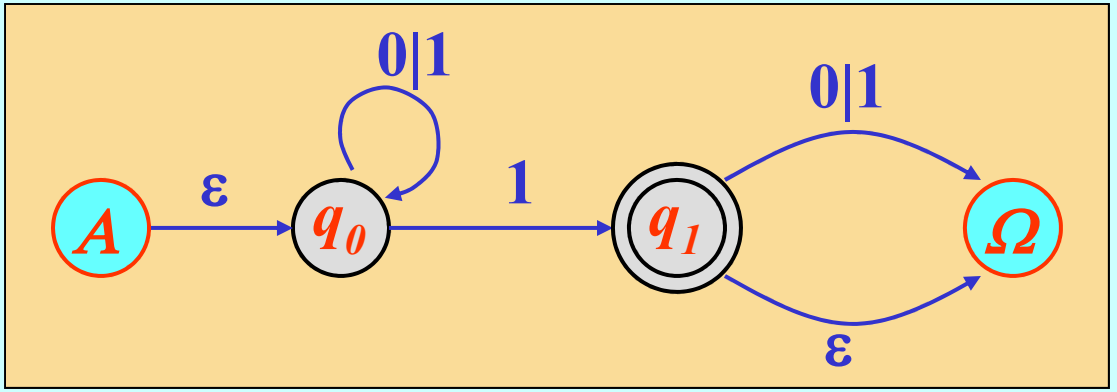


- adding the states ***A*** and ***Ω***

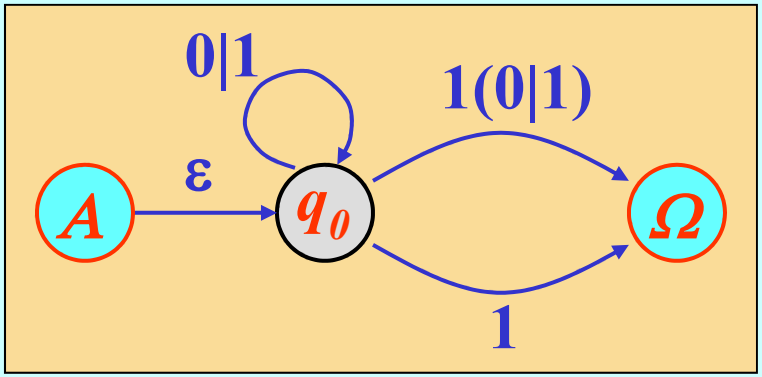


RL: from FA to regular expressions (2)

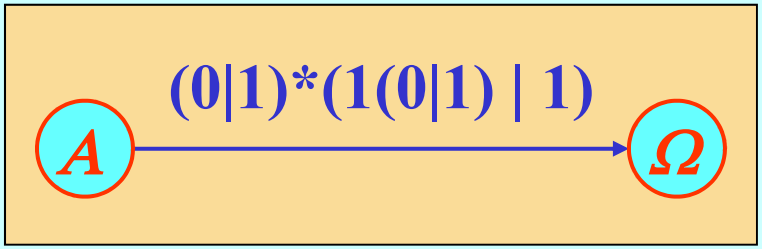
- eliminating q_2



- eliminating q_1

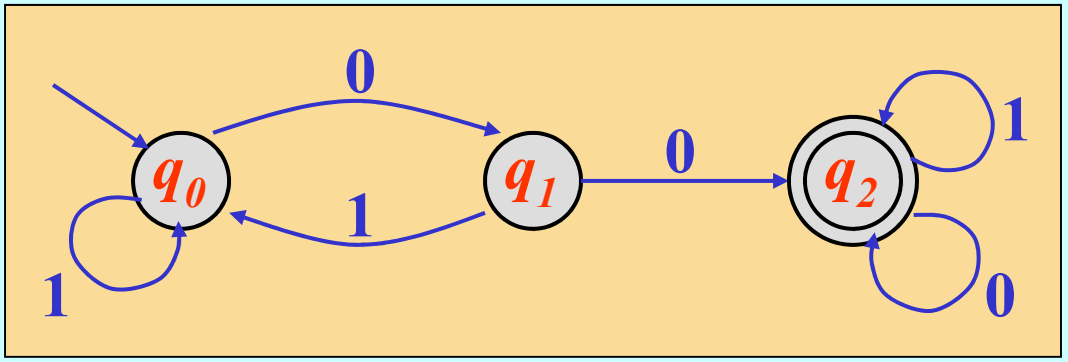


- eliminating q_0

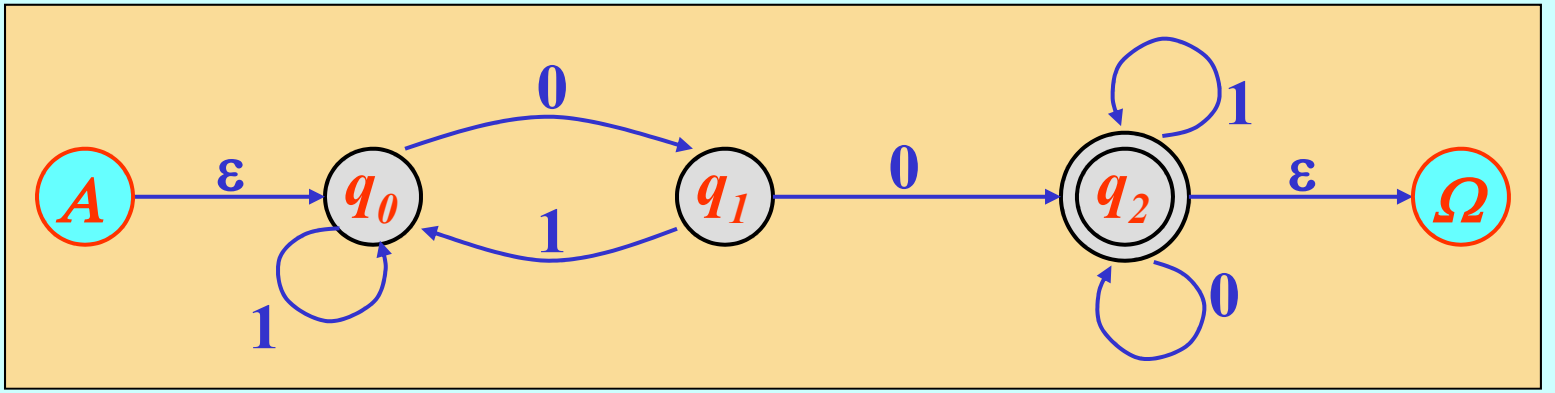


RL: from FA to regular expressions (3)

- $L(A)$ = the set of all strings over {0,1} containing at least two consecutive " **0** "

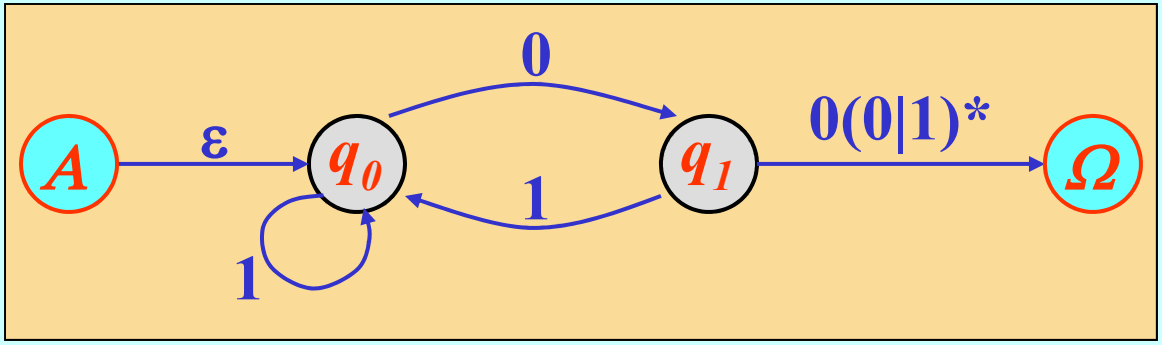


- adding the states A and Ω

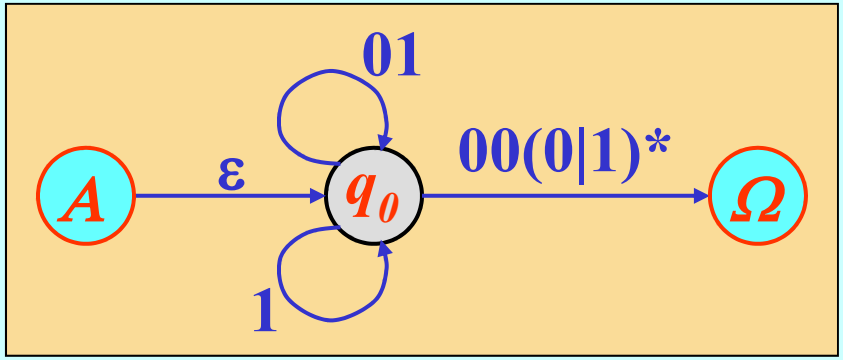


RL: from FA to regular expressions (4)

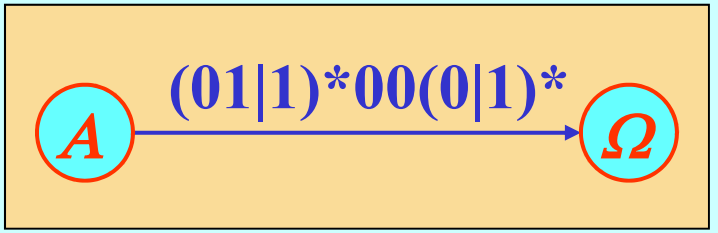
- eliminating q_2



- eliminating q_1

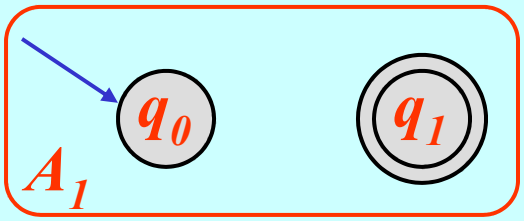


- eliminating q_0

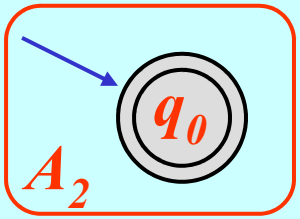


RL: regular sets ⊆ FA languages (1)

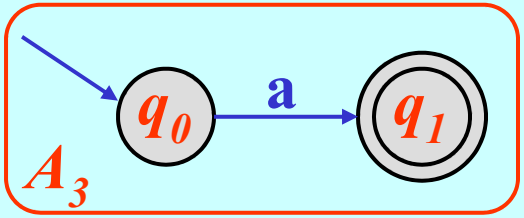
- the ***regular sets*** : $\varnothing$, $\{\varepsilon\}$, $\{ \mathtt{a} \}$, $\mathtt{a} \in \Sigma$ are accepted by finite state automata



$\Rightarrow \quad L(A_1) = \varnothing$



$\Rightarrow \quad L(A_2) = \{\varepsilon\}$

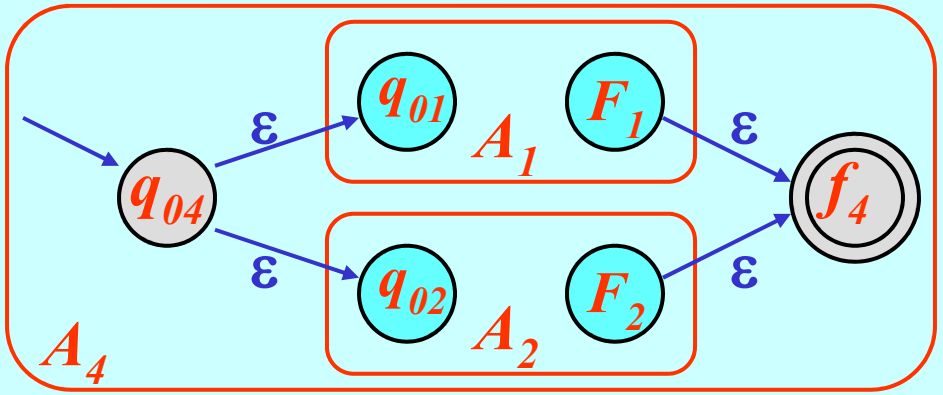


$\Rightarrow \quad L(A_3) = \{ \mathtt{a} \}$, $\mathtt{a} \in \Sigma$

RL: regular sets $\subseteq$ FA languages (2)

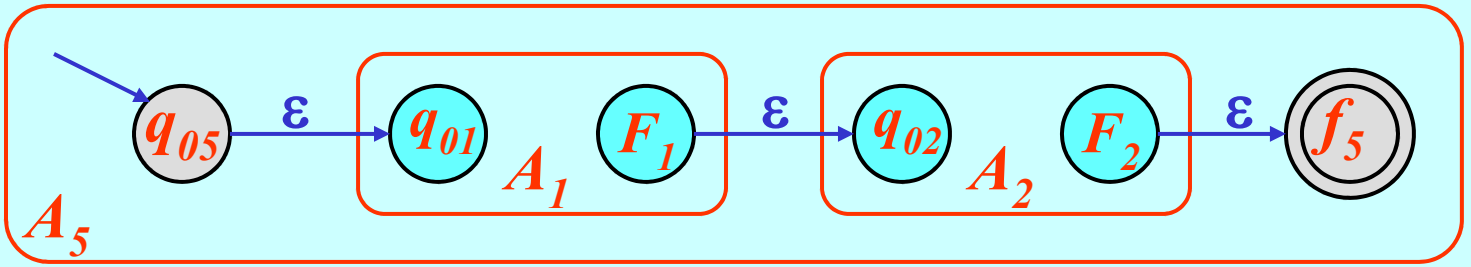
- let $A_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ be finite state automata

- the language $L(A_1) \cup L(A_2)$ is accepted by a finite state automaton A_4

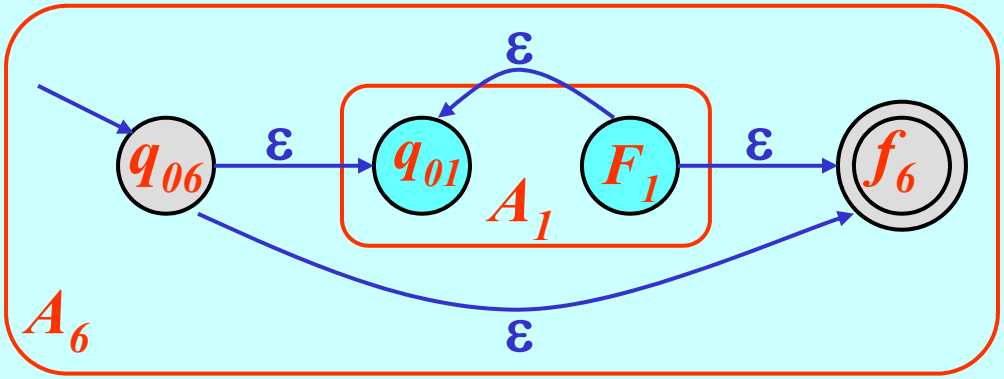


RL: regular sets ⊆ FA languages (3)

- the language $L(A_1)\,L(A_2)$ is accepted by a finite state automaton A_5

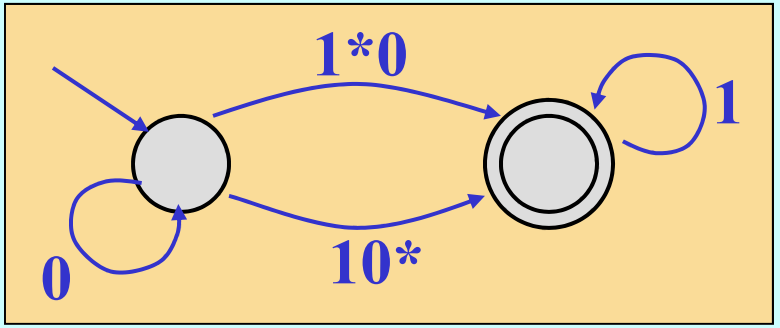


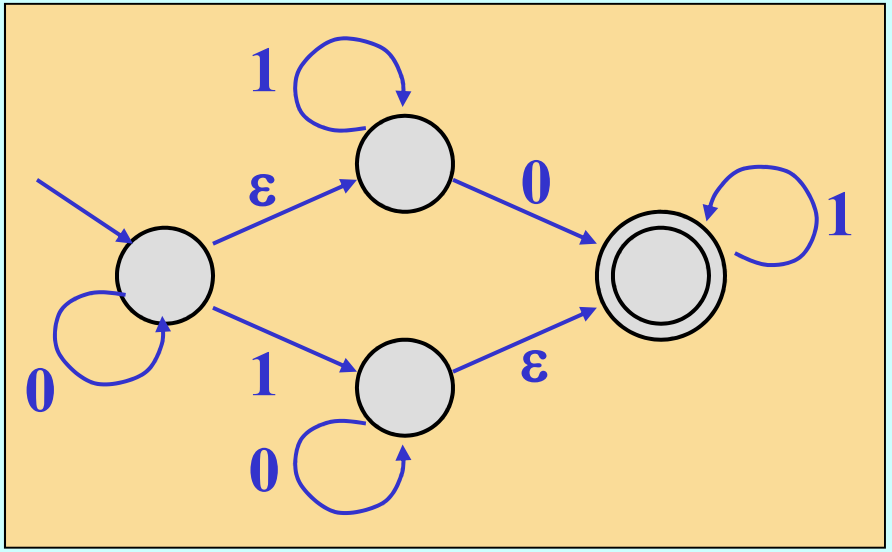
- the language $L(A_1)^*$ is accepted by a finite state automaton A_6



RL: from regular expressions to FA

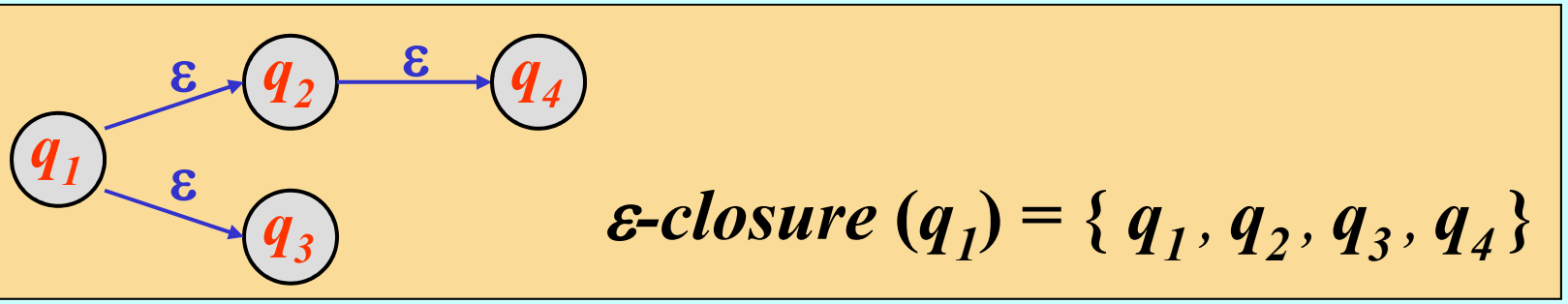
0*(1*0 | 10*)1*





RL: NFA with ε-transitions (ε-NFA)

- in the construction of **FA** from regular expressions, the ***ε-transitions*** make the automata non-deterministic
- the function ***ε-closure*** ***(q)*** gives the set of states that can be reached (recursively) from state ***q*** with the empty string

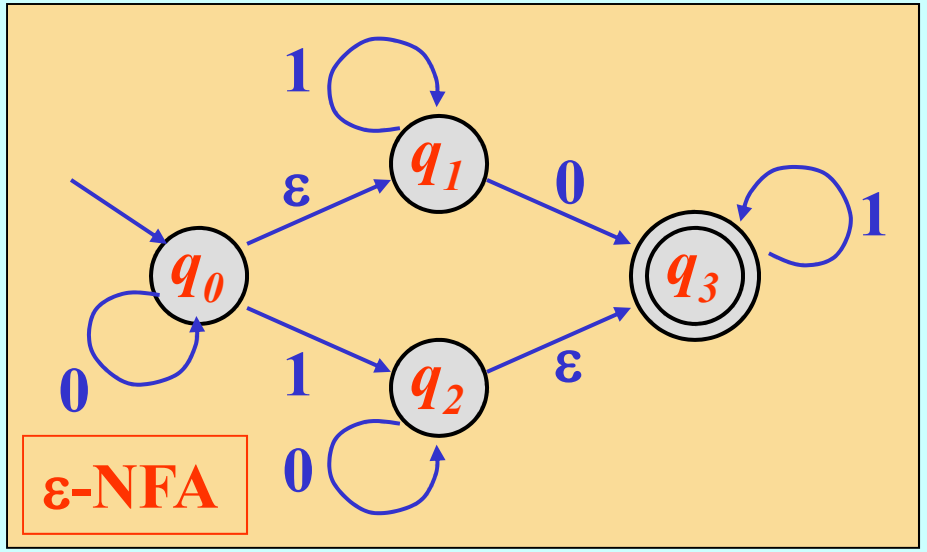


- $\varepsilon\text{-closure}(\{ q_1, q_2, \ldots, q_n \}) = \cup_i \ \varepsilon\text{-closure}(q_i)$

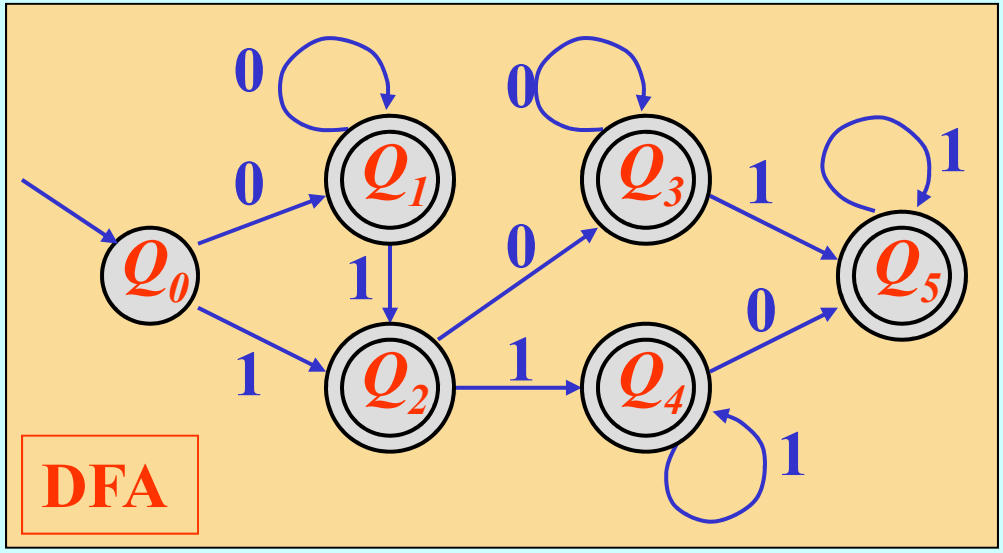
RL: equivalence of ε-NFA and DFA

- let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ be an **ε-NFA**
- let us construct a **DFA** $D = (Q_D, \Sigma, \delta_D, \varepsilon\text{-closure}(q_0), F_D)$
  - $Q_D \subseteq \wp(Q_N)$
  - $\delta_D(S, a) = \varepsilon\text{-closure}(\cup_i \delta_N(p_i, a))$ where $p_i \in S \in Q_D$
  - $F_D = \{ S \mid S \in Q_D \;;\; S \cap F_N \neq \varnothing \}$
- by construction $L(D) = L(N)$

RL: constructing a DFA from an ε-NFA

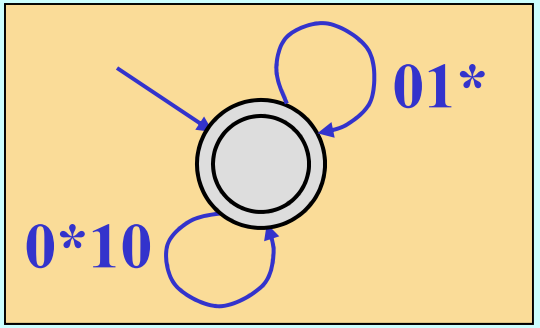


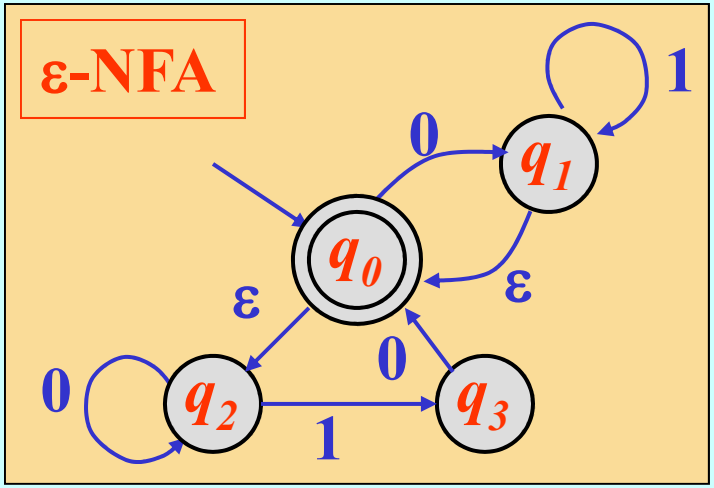
		0	1
Q_0	$\rightarrow\{q_0,q_1\}$	$\{q_0,q_1,q_3\}$	$\{q_1,q_2,q_3\}$
Q_1	$*\{q_0,q_1,q_3\}$	$\{q_0,q_1,q_3\}$	$\{q_1,q_2,q_3\}$
Q_2	$*\{q_1,q_2,q_3\}$	$\{q_2,q_3\}$	$\{q_1,q_3\}$
Q_3	$*\{q_2,q_3\}$	$\{q_2,q_3\}$	$\{q_3\}$
Q_4	$*\{q_1,q_3\}$	$\{q_3\}$	$\{q_1,q_3\}$
Q_5	$*\{q_3\}$	-	$\{q_3\}$



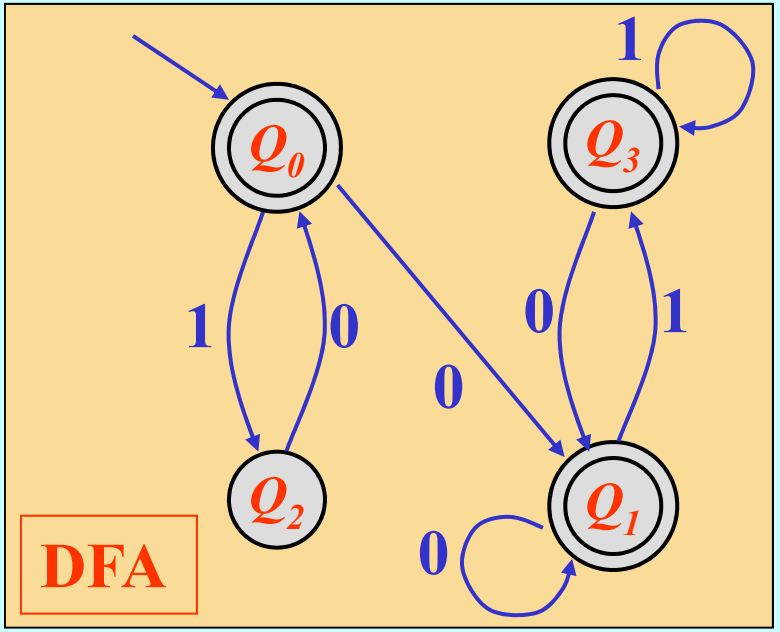
RL: from regular expressions to DFA

(0*10 | 01*)*





		0	1
Q_0	$\rightarrow^*\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_3\}$
Q_1	$^*\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2,q_3\}$
Q_2	$\{q_3\}$	$\{q_0,q_2\}$	-
Q_3	$^*\{q_0,q_1,q_2,q_3\}$	$\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2,q_3\}$



RL: FA languages ≡ regular languages (1)

- let $G = (N, T, P, S)$ be a right-regular grammar
- let us construct an **FA** $A = (Q, T, \delta, S, F)$
  - $Q = N \cup \{\Omega\}$ with $\Omega \notin N$
  - $F = \{\Omega\}$
  - $\delta = \{ \delta(A, a) = B$ if $A \rightarrow a\,B \in P$

    $\delta(A, a) = \Omega$ if $A \rightarrow a \in P \}$
- by construction $L(G) = L(A)$

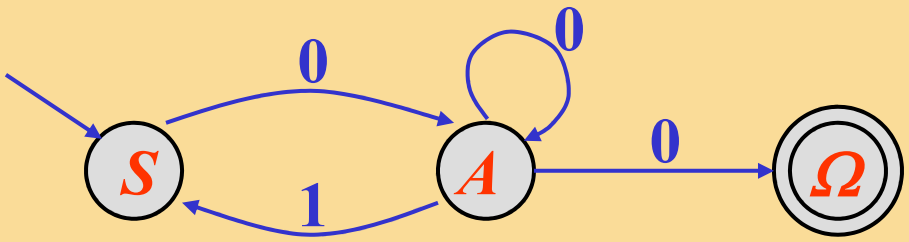
RL: from right regular grammars to FA

$G = (\{ A, S \}, \{0,1\}, P, S)$

$P = \{ S \rightarrow 0\,A$

$A \rightarrow 0\,A \mid 1\,S \mid 0$

$\}$



$S \rightarrow 0\,A \Rightarrow 0\,0\,A \Rightarrow 0\,0\,1\,S \Rightarrow 0\,0\,1\,0\,A \Rightarrow 0\,0\,1\,0\,0$

RL: FA languages ≡ regular languages (2)

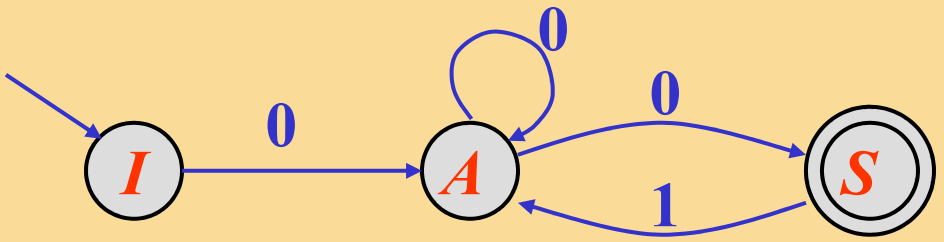
- let $G = (N, T, P, S)$ be a left-regular grammar
- let us construct an **FA** $A = (Q, T, \delta, I, \{S\})$
  - $Q = N \cup \{ I \}$ with $I \notin N$
  - $F = \{S\}$
  - $\delta = \{ \delta(B, a) = A$ if $A \rightarrow B\,a \in P$

    $\delta(I, a) = A$ if $A \rightarrow a \in P \}$
- by construction $L(G) = L(A)$

RL: from left regular grammars to FA

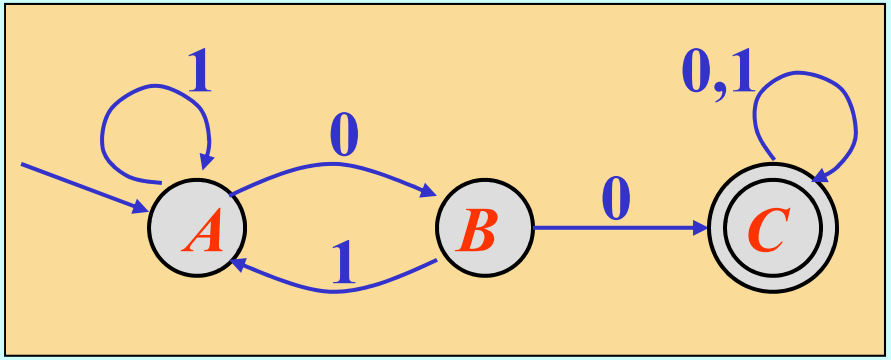
$G = (\{ A, S \}, \{0,1\}, P, S)$

$P = \{ S \rightarrow A\,0$
$A \rightarrow A\,0 \mid S\,1 \mid 0$
$\}$



$S \rightarrow A0 \Rightarrow A00 \Rightarrow S100 \Rightarrow A0100 \Rightarrow 00100$

RL: from FA to regular grammars



$G_1 = (\{ A, B, C \}, \{0,1\}, P_1, A)$

$P_1 = \{ A \rightarrow 1\,A \mid 0\,B$
$B \rightarrow 1\,A \mid 0\,C \mid 0$
$C \rightarrow 0\,C \mid 1\,C \mid 0 \mid 1$
$\}$

$G_2 = (\{ A, B, C \}, \{0,1\}, P_2, C)$

$P_2 = \{ C \rightarrow C\,0 \mid C\,1 \mid B\,0$
$B \rightarrow A\,0 \mid 0$
$A \rightarrow A\,1 \mid B\,1 \mid 1$
$\}$

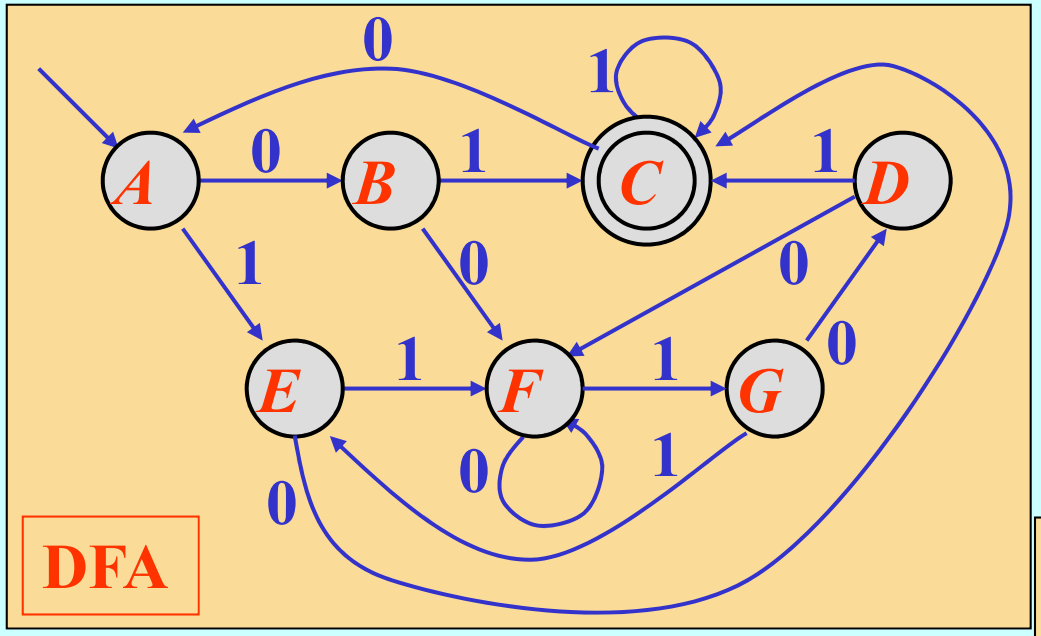
RL: minimum-state DFA

- let $DFA = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite state automaton
- two states p and q of DFA are ***distinguishable*** if there is a string $w \in \Sigma^*$ such that $\delta(p, w) \in F$ e $\delta(q, w) \notin F$
- two states p and q of DFA are ***equivalent*** $(p \equiv q)$ if they are ***non- distinguishable*** for any string $w \in \Sigma^*$
- a DFA is ***minimum-state*** if it does not contain equivalent states

RL: minimization of DFA (1)

- two states p and q of ***DFA*** are ***m-equivalent*** ($p \equiv_m q$) if they are ***non-distinguishable*** for all the strings $w \in \Sigma^*$ with $|w| \leq m$
  - $p \equiv_0 q$ if $p \in F$; $q \in F$ or $p \notin F$; $q \notin F$
  - if $p \equiv_m q$ and for any $a \in \Sigma$, $\delta(p, a) \equiv_m \delta(q, a)$ then $p \equiv_{m+1} q$
  - if $p \equiv_m q$ and $m = ||Q|| - 2$ then $p \equiv q$
- the equivalent states can be determined by partitioning the set Q in classes of ***m-equivalent*** states, for $m = 0, 1, \ldots, ||Q|| - 2$

RL: minimization of DFA (2)

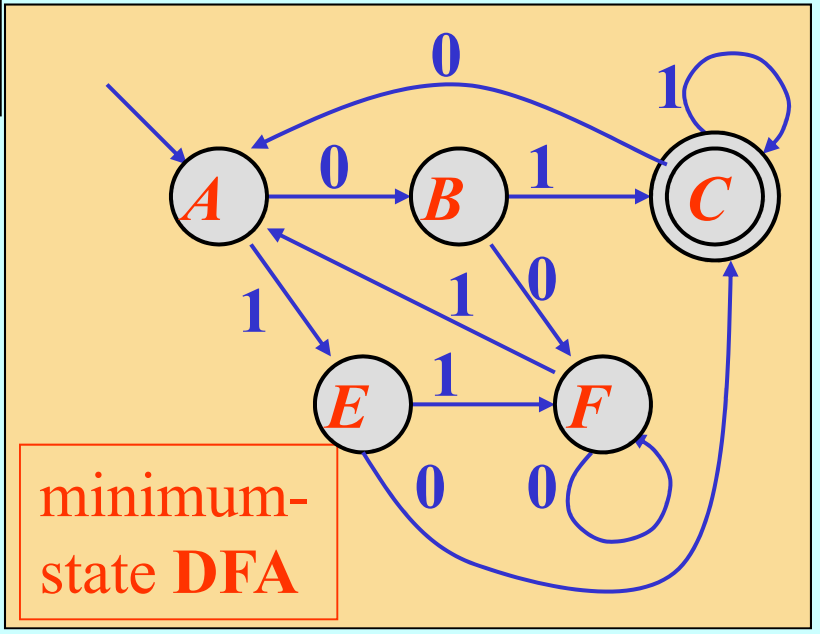


Π_0 : {C}, {A, B, D, E, F, G}

Π_1 : {C}, {A, F, G}, {B, D}, {E}

Π_2 : {C}, {A, G }, {F}, {B, D}, {E}

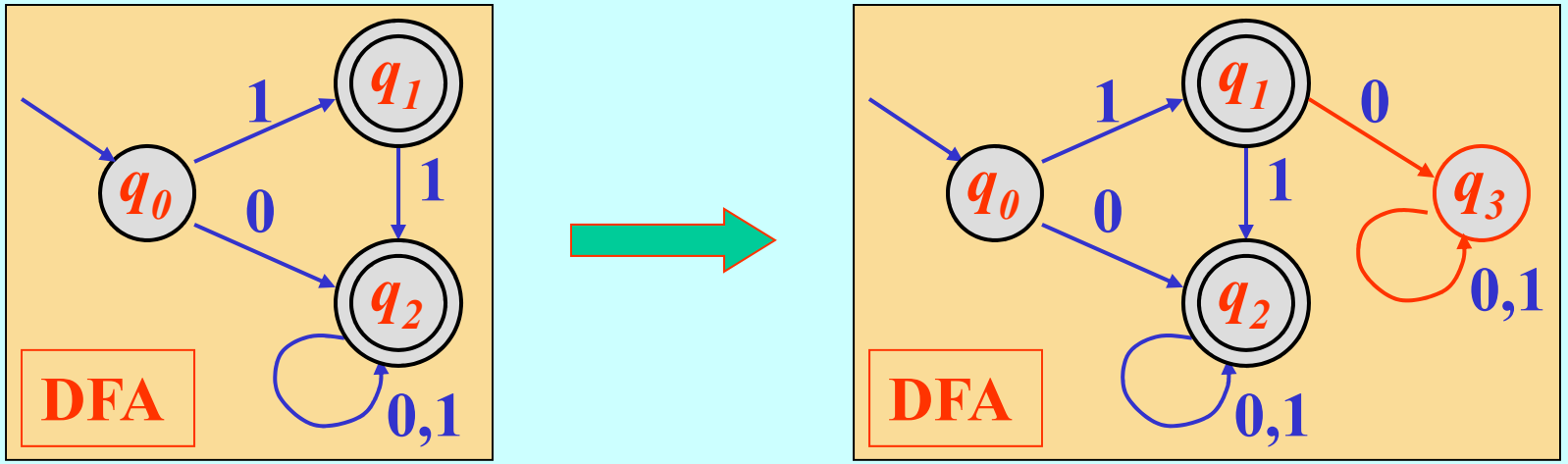
Π_3 : {C}, {A, G }, {F}, {B, D}, {E}

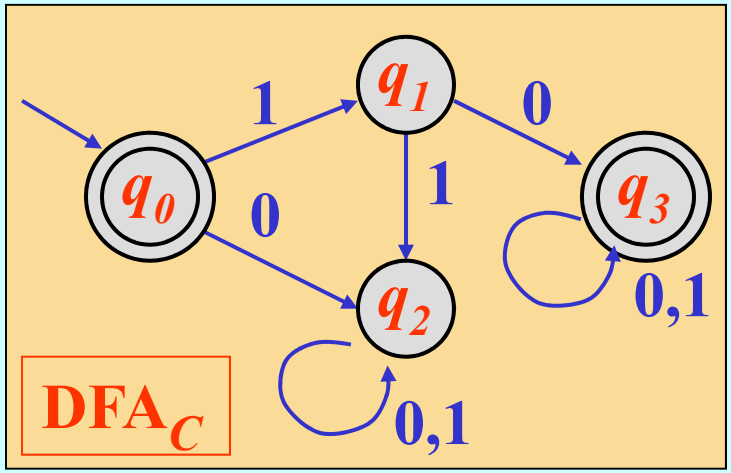


RL: complement of regular languages (1)

- the *complement* of a regular language is a regular language
  - let $DFA = (Q, \Sigma, \delta, q_0, F)$ be a ***completely specified deterministic*** finite state automaton
    - there is a transition on every symbol of Σ from every state
  - the automaton $DFA_C = (Q, \Sigma, \delta, q_0, Q - F)$ accepts the language $L(DFA_C) = \Sigma^* - L(DFA) = \neg L(DFA)$

RL: complement of regular languages (2)





RL: intersection of regular languages (1)

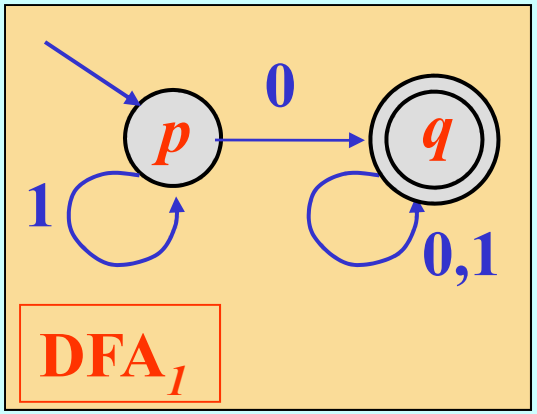
- the *intersection* of two regular languages is a regular language
  - $L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$
- let $DFA_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$
  $DFA_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
  - the automaton

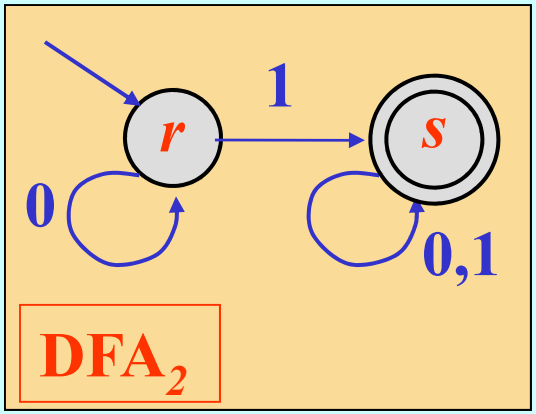
$DFA_I = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F_1 \times F_2)$,
where : $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a)))$,

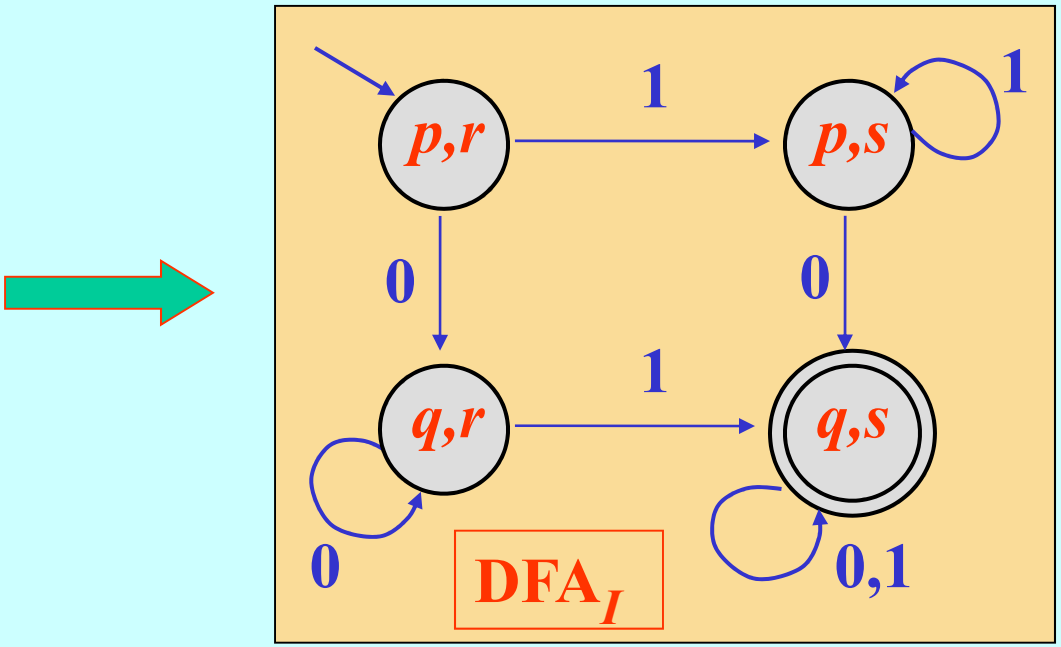
accepts the language :

$L(DFA_I) = L(DFA_1) \cap L(DFA_2)$

RL: intersection of regular languages (2)

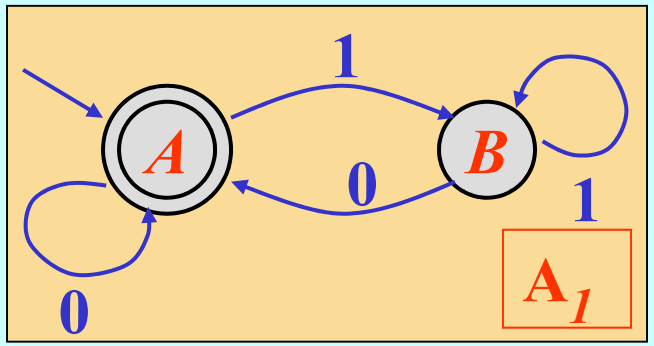


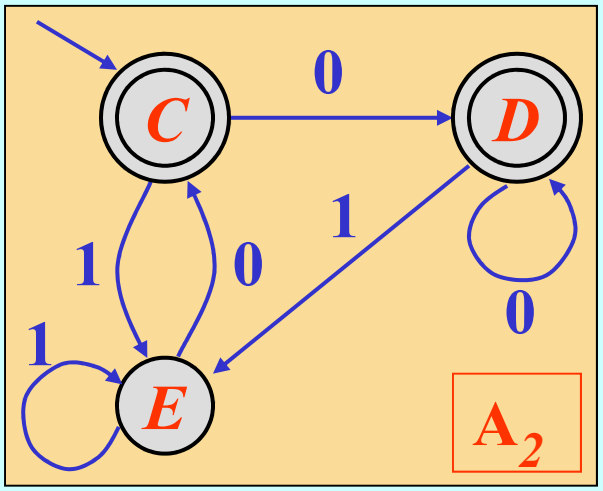




RL: equivalence of regular languages

- it is possible to test if two regular languages are the same
  - $DFA_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$; $DFA_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$
  - let us find the equivalence states in the set $Q_1 \cup Q_2$
  - if $q_{01} \equiv q_{02}$ then $L(DFA_1) = L(DFA_2)$





Π_0 : {A ,C, D}, {B, E}

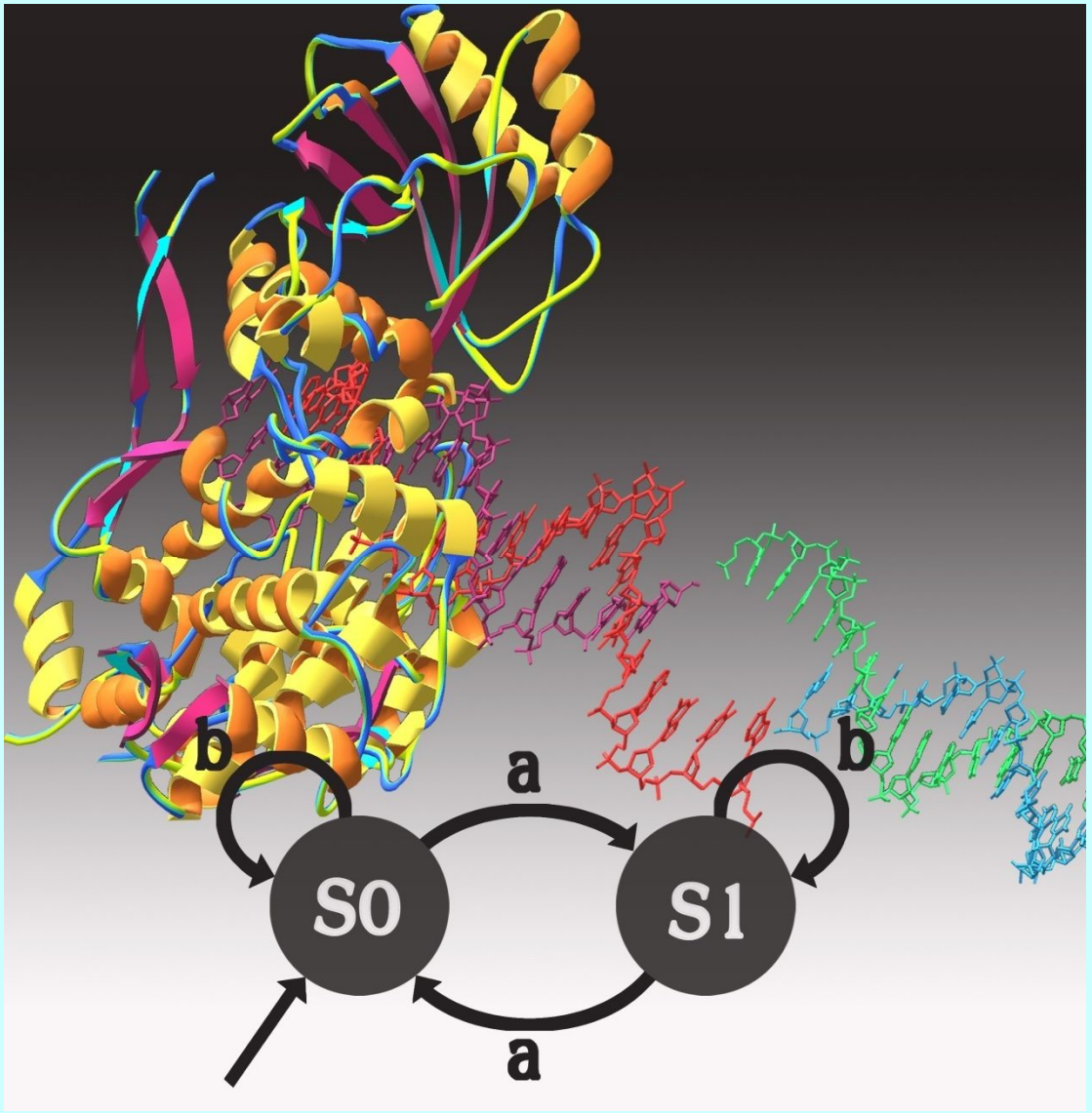
Π_1 : {A ,C, D}, {B, E}

$L(A_1) = L(A_2)$

RL: applications of regular languages

- Finding occurrences of words, phrases, *patterns* in a text
  - software for *editing* , *word processing* , …
- Constructing lexical analyzers (*scanners*)
  - compiler components that break the source text into lexical elements
    - identifiers, keywords, numeric or alphabetic constants, operators, punctuation, …
- Designing and verifying systems that have a finite number of distinct states
  - digital circuits, communication protocols, programmable controllers, …

RL: Molecular realization of an automaton



An input DNA molecule (green/blue) provides both data and fuel for the computation. Software DNA molecules (red/purple) encode program rules, and the restriction enzyme FokI (colored ribbons) functions as the automaton's hardware.

Ehud.Shapiro@weizmann.ac.il

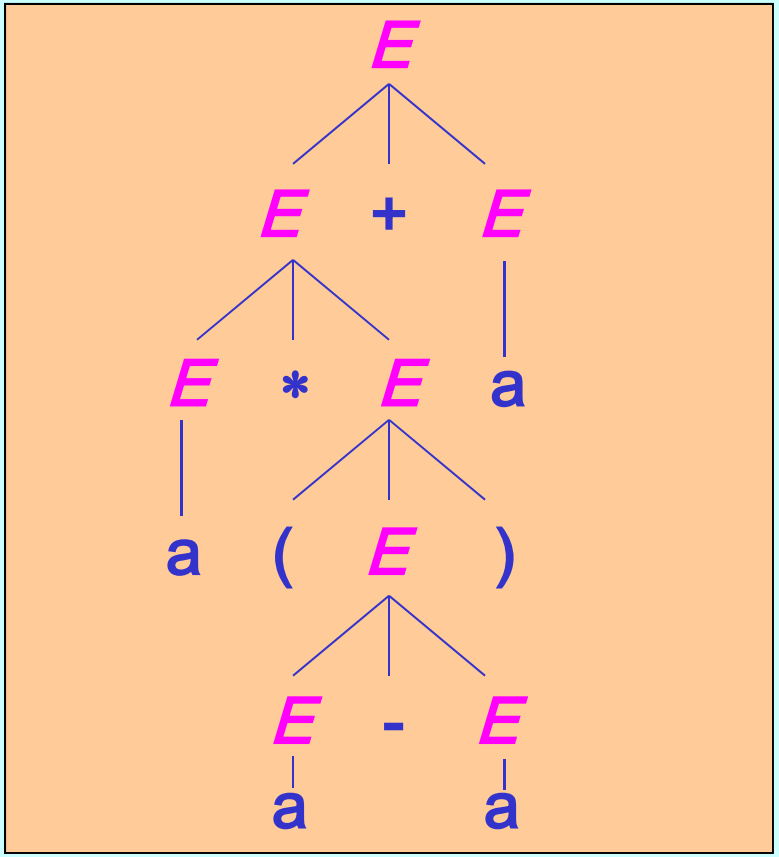
Context-Free Languages: parse trees (1)

- a *parse tree* for a context-free grammar (*CFG*) $G = (N, T, P, S)$ is a tree where
  - the root is labeled by the start symbol S
  - each interior node is labeled by a symbol in N
  - each leaf is labeled by a symbol in $N \cup T \cup \{\varepsilon\}$
  - an interior node labeled by A has children (from left to right) labeled by $X_1, X_2, \ldots, X_k$ only if $A \rightarrow X_1 X_2 \ldots X_k$ is a production in P
- yield of a parse tree
  - string obtained by concatenating (from left to right) the labels of the leaves

CFL: parse trees (2)

$G = (\{E\}, \{a, +, -, *, /, (,)\}, P, E)$

$P = \{ E \rightarrow E+E \mid E-E \mid E*E \mid E/E \mid (E) \mid a \}$



$E \Rightarrow^* a*(a-a)+a$

yield

CFL: leftmost / rightmost derivations

- leftmost derivation
  - the leftmost non-terminal symbol is replaced at each derivation step
    - $E \Rightarrow \underline{E} + E \Rightarrow \underline{E} * E + E \Rightarrow a * \underline{E} + E \Rightarrow a * (\underline{E}) + E \Rightarrow a * (\underline{E} - E) + E \Rightarrow a * (a - \underline{E}) + E \Rightarrow a * (a - a) + \underline{E} \Rightarrow a * (a - a) + a$
- rightmost derivation
  - the rightmost non-terminal symbol is replaced at each derivation step
    - $E \Rightarrow E + \underline{E} \Rightarrow \underline{E} + a \Rightarrow E * \underline{E} + a \Rightarrow E * (\underline{E}) + a \Rightarrow E * (E - \underline{E}) + a \Rightarrow E * (\underline{E} - a) + a \Rightarrow \underline{E} * (a - a) + a \Rightarrow a * (a - a) + a$

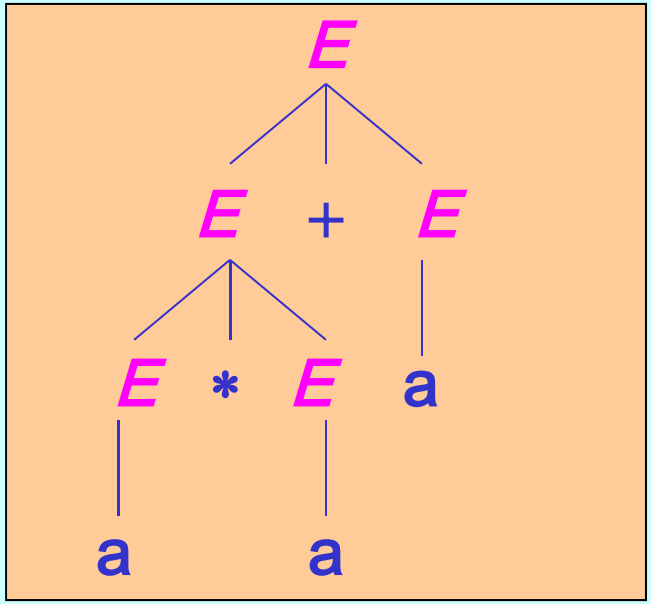
CFL: ambiguity

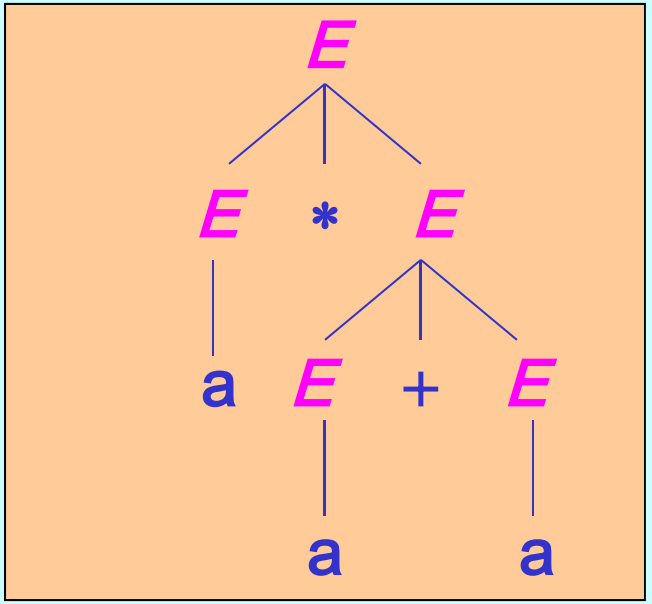
- every string in a CFL has at least one parse tree
- each parse tree has just one leftmost derivation and just one rightmost derivation
- a ***CFG*** is ***ambiguous*** if there is at least one string in its language having two different parse trees
- a ***CFL*** is ***inherently ambiguous*** if all its grammars are ambiguous

CFL: ambiguous grammars (1)

$G_1 = (\{E\}, \{a, +, -, *, /, (,)\}, P_1, E)$

$P_1 = \{E \rightarrow E+E \mid E-E \mid E*E \mid E/E \mid (E) \mid a\}$



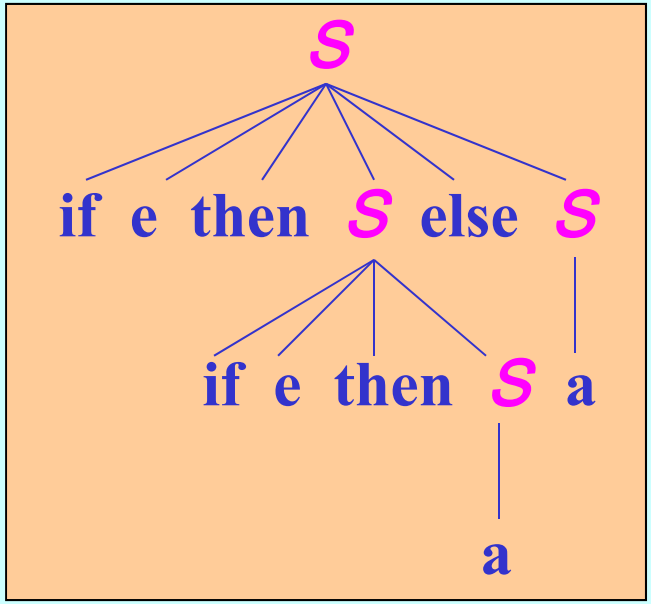


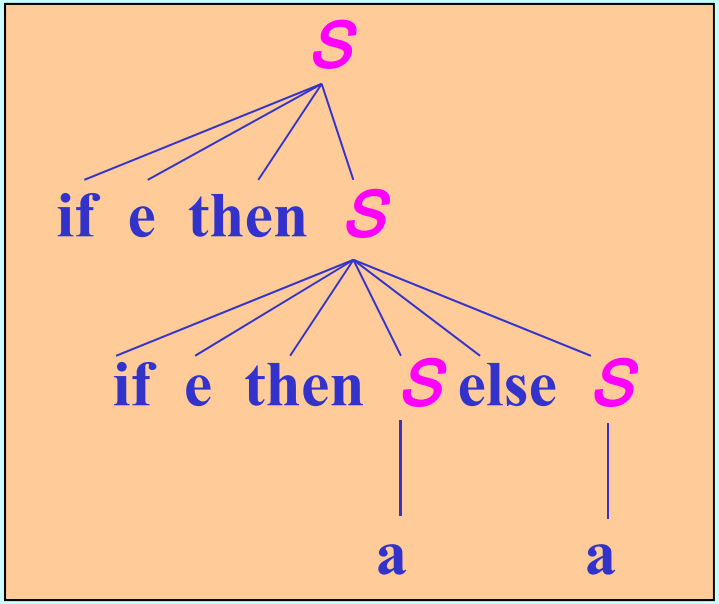
$E \Rightarrow^* a*a+a$

CFL: ambiguous grammars (2)

$G_2 = (\{ S \}, \{ \text{if}, \text{then}, \text{else}, \text{e}, \text{a} \}, P_2, S)$

$P_2 = \{ S \rightarrow \text{if e then } S \text{ else } S \mid \text{if e then } S \mid \text{a} \}$





$S \Rightarrow^*$ if e then if e then a else a

CFL: equivalent non-ambiguous grammars

$G_3 = (\{ E, T, F \}, \{ a, +, -, *, /, (,) \}, P_3, E)$

$P_3 = \{ E \rightarrow E + T \mid E - T \mid T$

$T \rightarrow T * F \mid T / F \mid F$

$F \rightarrow (E) \mid a$

$\}$

$L(G_1) = L(G_3)$

$G_4 = (\{ S, M, U \}, \{ \text{if}, \text{then}, \text{else}, e, a \}, P_4, S)$

$P_4 = \{ S \rightarrow M \mid U$

$M \rightarrow \text{if e then } M \text{ else } M \mid a$

$U \rightarrow \text{if e then } M \text{ else } U \mid \text{if e then } S$

$\}$

$L(G_2) = L(G_4)$

CFL: eliminating useless symbols in CFG

- a symbol X is useful for a $CFG = (N, T, P, S)$ if there is some derivation $S \Rightarrow^* \alpha X \beta \Rightarrow^* w \in T^*$
  - a useful symbol X generates a *non-empty language*:
    $X \Rightarrow^* x \in T^*$
  - a useful symbol X is *reachable*:
    $S \Rightarrow^* \alpha X \beta$
- eliminating useless symbols from a grammar will not change the generated language

1. eliminate symbols generating an empty language
2. eliminate unreachable symbols

CFL: finding the useful symbols in CFG

- finding symbols generating *non-empty languages*
  - every symbol of T generates a non-empty language
  - if $A \rightarrow \alpha$ and all symbols in α generate a non-empty language, then A generates a non-empty language
- finding *reachable* symbols
  - the start symbol S is reachable
  - if $A \rightarrow \alpha$ and A is reachable, all symbols in α are reachable

CFL: symbols generating an empty language

$G_1 = (\{ S, A, B, C \}, \{ a, b \}, P_1, S)$

$P_1 = \{ \; S \rightarrow A\,a \mid b\,C\,b$

$A \rightarrow a\,B\,A \mid b\,A\,S$

$B \rightarrow a\,S \mid b\,A \mid b$

$C \rightarrow a\,S\,a \mid a \; \}$

symbols generating a ***non-empty language*** :

$\{ a, b \} \cup \{ B, C \} \cup \{ S \}$

symbols generating an ***empty language*** :

$\{ A \}$

$G_2 = (\{ S, B, C \}, \{ a, b \}, P_2, S)$

$P_2 = \{ \; S \rightarrow b\,C\,b$

$B \rightarrow a\,S \mid b$

$C \rightarrow a\,S\,a \mid a \; \}$

$L(G_1) = L(G_2)$

CFL: unreachable symbols

$G_2 = (\{ S, B, C \}, \{ a, b \}, P_2, S)$

$P_2 = \{ \; S \rightarrow b\,C\,b$

$B \rightarrow a\,S \mid b$

$C \rightarrow a\,S\,a \mid a \; \}$

reachable symbols :

$\{ S \} \cup \{ b, C \} \cup \{ a \}$

unreachable symbols :

$\{ B \}$

$G_3 = (\{ S, C \}, \{ a, b \}, P_3, S)$

$P_3 = \{ \; S \rightarrow b\,C\,b$

$C \rightarrow a\,S\,a \mid a \; \}$

$L(G_1) = L(G_2) = L(G_3)$

CFL: ε-productions in CFG

- according to the Chomsky classification, only type 0 grammars can have *ε-productions*
- anyway the languages generated by CFG's that contain *ε-productions* are CFL
  - a CFG G_1 with *ε-productions* can be transformed into an equivalent CFG G_2 without *ε-productions* :

    $$L(G_2) = L(G_1) - \{\varepsilon\}$$

  - if $A \rightarrow X_1 \dots X_i \dots X_n$ is in P_1 and $X_i \Rightarrow^* \varepsilon$, then P_2 will contain $A \rightarrow X_1 \dots X_i \dots X_n$ and $A \rightarrow X_1 \dots X_{i-1} X_{i+1} \dots X_n$

CFL: eliminating ε-productions in CFG

$G_1 = (\{S, A, B\}, \{a, b\}, P_1, S)$

$P_1 = \{ S \rightarrow a\,A \mid b$
$A \rightarrow B\,S\,B \mid B\,B \mid a$
$B \rightarrow a\,A\,b \mid b \mid \varepsilon$
$\}$

symbols that generate ε : $\{B, A\}$

$G_2 = (\{S, A, B\}, \{a, b\}, P_2, S)$

$P_2 = \{ S \rightarrow a\,A \mid b \mid a$
$A \rightarrow B\,S\,B \mid B\,B \mid a \mid S\,B \mid B\,S \mid S \mid B$
$B \rightarrow a\,A\,b \mid b \mid a\,b$
$\}$

$L(G_1) = L(G_2)$

CFL: pushdown automata (PDA)

- A **PDA** is a 7-tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
  - Q : finite (non empty) set of **states**
  - Σ : alphabet of **input** symbols
  - Γ : alphabet of **stack** symbols
  - δ : **transition** function
    - $\delta: \; Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \{(p, \gamma) \mid p \in Q \; ; \; \gamma \in \Gamma^*\}$
  - q_0 : **start** state ($q_0 \in Q$)
  - Z_0 : **start** stack symbol ($Z_0 \in \Gamma$)
  - F : set of **final states** ($F \subseteq Q$)

CFL: transitions of PDA (1)

- $\delta(q, a, X) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$
  - from state q, with a in input and X on top of the stack:
    - consumes a from the input string
    - goes to a state p_i and replaces X with γ_i
      - the first symbol of γ_i goes on top of the stack
- $\delta(q, \varepsilon, X) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$
  - from state q, with X on top of the stack:
    - no input symbol is consumed
    - goes to a state p_i and replaces X with γ_i
      - the first symbol of γ_i goes on top of the stack

CFL: transitions of PDA (2)

- *instantaneous configuration* of a PDA: (q, w, γ)
  - q : current state
  - w : remaining input string
  - γ : current stack contents
- transition:
  - if $\delta(q, a, X) = \{\ldots, (p, \alpha), \ldots\}$, then

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

CFL: languages accepted by PDA

- Language accepted by *final state* by the PDA

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

  - $L(P) = \{ w \mid w \in \Sigma^* ; (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha) ; q \in F \}$

- Language accepted by *empty stack* by the PDA

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing)$$

  - $N(P) = \{ w \mid w \in \Sigma^* ; (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon) \}$

CFL: example of PDA

$P = (\{ q_0 , q_1 \} , \{ 0 , 1 \} , \{ 0 , 1 , Z \} , \delta , q_0 , Z , \varnothing)$

$\delta(q_0 , 0 , Z) = \{(q_0 , 0Z)\}$ $\quad$ $\delta(q_0 , 1 , Z) = \{(q_0 , 1Z)\}$

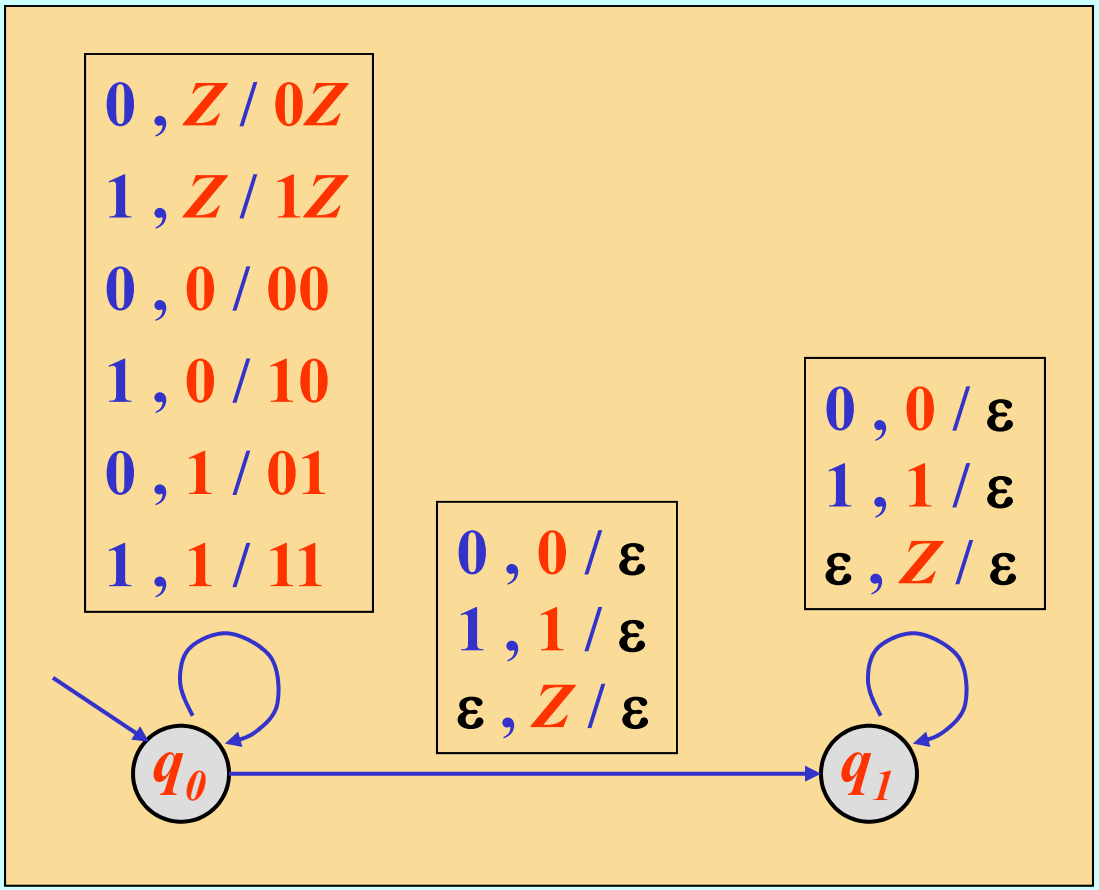
$\delta(q_0 , 0 , 0) = \{(q_0 , 00),(q_1 , \varepsilon)\}$ $\quad$ $\delta(q_0 , 1 , 0) = \{(q_0 , 10)\}$

$\delta(q_0 , 0 , 1) = \{(q_0 , 01)\}$ $\quad$ $\delta(q_0 , 1 , 1) = \{(q_0 , 11),(q_1 , \varepsilon)\}$

$\delta(q_1 , 0 , 0) = \{(q_1 , \varepsilon)\}$ $\quad$ $\delta(q_1 , 1 , 1) = \{(q_1 , \varepsilon)\}$

$\delta(q_0 , \varepsilon , Z) = \{(q_1 , \varepsilon)\}$ $\quad$ $\delta(q_1 , \varepsilon , Z) = \{(q_1 , \varepsilon)\}$

CFL: graphical notation for PDA



$$N(P) = \{\, w\, w^R \mid w \in \{0, 1\}^* \,\}$$

CFL: configuration sequences of PDA

$(q_0, 001100, Z)$ ← *initial configuration*

$\top$

$(q_0, 01100, 0Z) \vdash (q_1, 1100, Z) \vdash (q_1, 1100, \varepsilon)$

$\top$

$(q_0, 1100, 00Z)$

$\top$

$(q_0, 100, 100Z) \vdash (q_0, 00, 1100Z) \vdash (q_0, 0, 01100Z) \vdash (q_0, \varepsilon, 001100Z)$

$\top$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\top$

$(q_1, 00, 00Z) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (q_1, \varepsilon, 1100Z)$

$\top$

$(q_1, 0, 0Z)$

$\top$

(q_1, ε, Z)

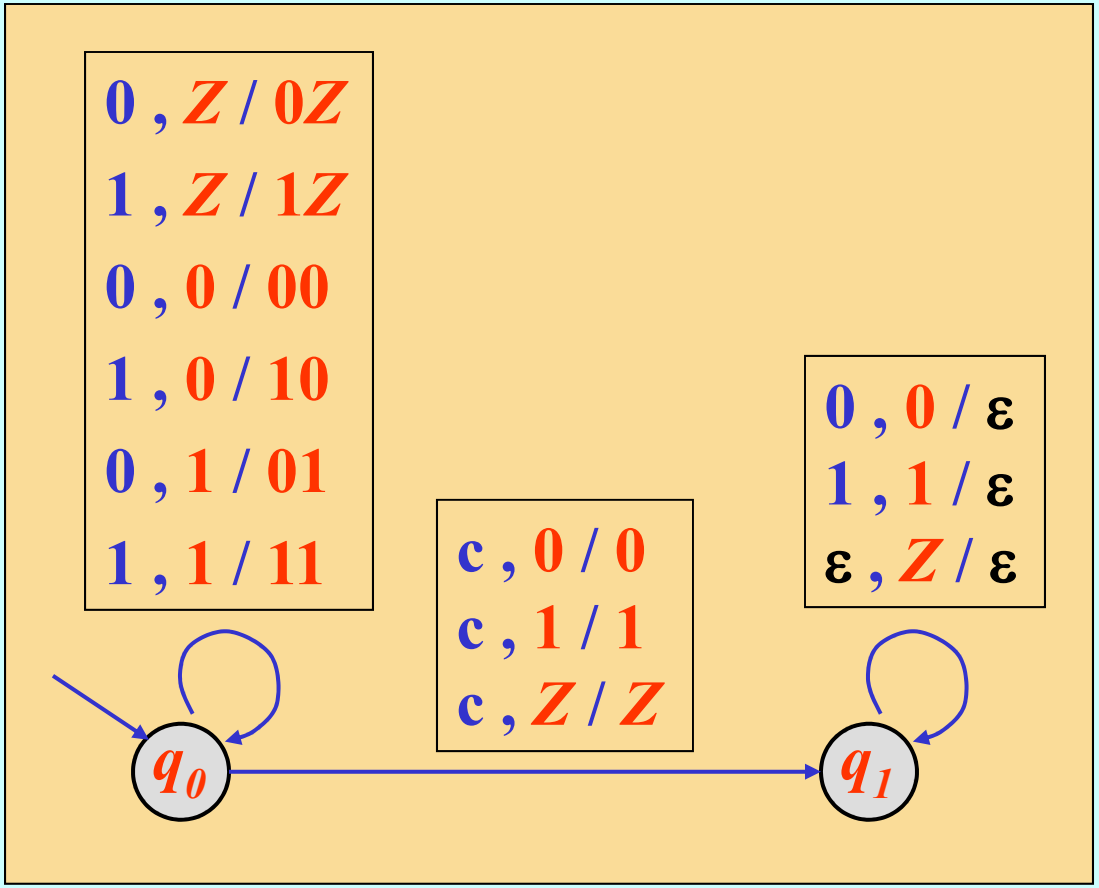
$\top$

$(q_1, \varepsilon, \varepsilon)$ ← *accepting configuration*

CFL: deterministic pushdown automata (DPDA)

- A **PDA** $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is *deterministic (DPDA)* if:
  - $\delta(q, a, X)$ has at most one member for any $q \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$, $X \in \Gamma$
  - if $\delta(q, a, X) \neq \varnothing$ for some $a \in \Sigma$, then $\delta(q, \varepsilon, X) = \varnothing$
- the languages accepted by DPDA are *properly included* ($\subset$) in the languages accepted by PDA
  - the language $\{ w\, w^R \mid w \in \{0, 1\}^* \}$ is not accepted by DPDA

CFL: example of DPDA



$$N(P) = \{\, w\, c\, w^R \mid w \in \{0, 1\}^* \,\}$$

CFL: PDA languages ≡ CFL

- let $G = (N, T, P, S)$ be a context-free grammar
- let us construct a $PDA = (\{q\}, T, \Gamma, \delta, q, S, \varnothing)$
  - $\Gamma = N \cup T$
  - $\delta = \{\ \delta(q, \varepsilon, A) = \{(q, \alpha)$ for each $A \to \alpha \in P\}$

    $\delta(q, a, a) = \{(q, \varepsilon)\}$ for each $a \in T$

    $\}$
- PDA accepts $L(G)$ by *empty stack*, making a sequence of transitions corresponding to a *leftmost derivation*

CFL: from CFL to PDA (1)

$L(G) = \{ w\, w^R \mid w \in \{0, 1\}^* \}$

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

$P = \{ S \rightarrow 0S0 \mid 1S1 \mid \varepsilon \}$

$S \rightarrow 0S0$

$\Rightarrow 00S00$

$\Rightarrow 001S100$

$\Rightarrow 001100$

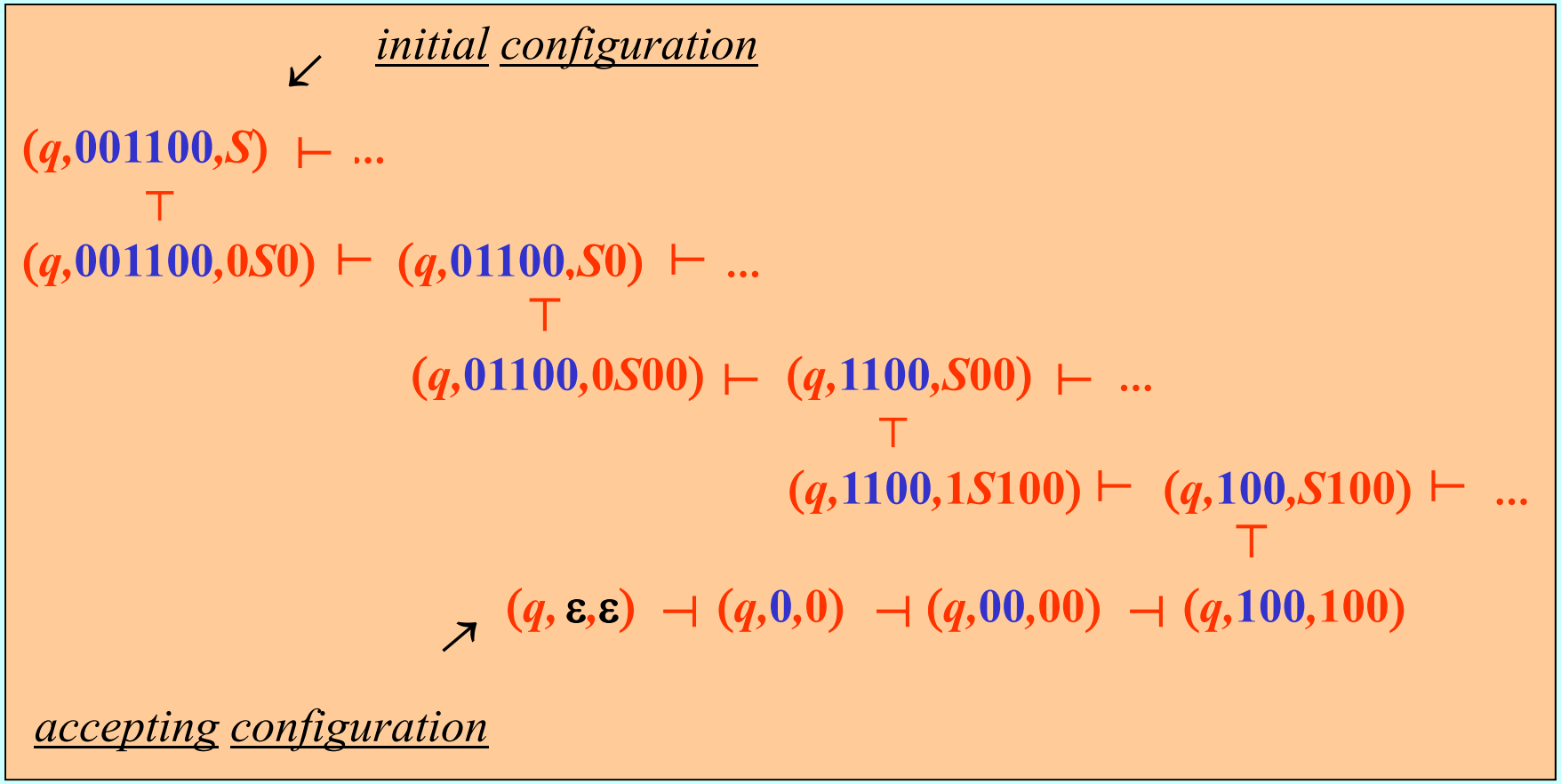
$P = (\{q\}, \{0, 1\}, \{0, 1, S\}, \delta, q, S, \varnothing)$

$\delta(q, \varepsilon, S) = \{ (q, 0S0), (q, 1S1), (q, \varepsilon) \}$

$\delta(q, 0, 0) = \{ (q, \varepsilon) \}$

$\delta(q, 1, 1) = \{ (q, \varepsilon) \}$

CFL: from CFL to PDA (2)



CFL: properties of CFL

- the CFL's are *closed* under the operations:
  - ***union***
  - ***concatenation***
  - ***Kleene closure***
- the CFL's are *not closed* under the operations:
  - ***complement***
  - ***intersection***
- it is possible to decide membership of a string w in a CFL by algorithms (Cocke-Younger-Kasamy, Earley, …) with complexity $O(n^3)$, where $n = |w|$

CFL: deterministic context free languages

- the *deterministic* CFL's (the languages accepted by ***DPDA***) are *closed* under the operations:
  - ***complement***
- the *deterministic* CFL's are *not closed* under the operations:
  - ***union***
  - ***intersection***
  - ***concatenation***
  - ***Kleene closure***
- it is possible to decide membership of a string w in a *deterministic* CFL by algorithms with complexity $O(n)$, where $n = |w|$

CFL: applications of context free languages

- Representation of programming languages
  - grammars for Algol, Pascal, C, Java, …
- Construction of syntax analyzers (*parsers*)
  - compiler components that analyze the structure of a source program and represent it by means of a parse tree
- Description of the structure and the semantic contents of documents (*Semantic Web*) by means of *Markup Languages*
  - XML (*Extensible Markup Language*) , RDF (*Resource Description Framework*) , OWL (*Web Ontology Language*) , …

Turing Machines

- A **TM** is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$
  - Q : finite (non empty) set of **states**
  - Σ : alphabet of **input** symbols ($\mathbf{B} \notin \Sigma$)
  - Γ : alphabet of **tape** symbols ($\mathbf{B} \in \Gamma$; $\Sigma \subset \Gamma$)
    - the tape extends infinitely to the left and the right
    - the tape initially holds the input string, preceded and followed by an infinite number of **B** symbols
  - δ : **transition** function
    - δ : $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ ($L = left$, $R = right$)
  - q_0 : **start** state ($q_0 \in Q$)
  - F : set of **final states** ($F \subseteq Q$)

TM: transitions of a TM (1)

- $\delta(q, X) = (p, Y, L)$
  - from state q, having X as the current tape symbol:
    - goes to state p and replaces X with Y
    - moves the tape head one position ***left***
- $\delta(q, X) = (p, Y, R)$
  - from state q, having X as the current tape symbol:
    - goes to state p and replaces X with Y
    - moves the tape head one position ***right***

TM: transitions of a TM (2)

- *instantaneous configuration* of a TM:

$$(X_1 \ldots X_{i-1}\ q\ X_i \ldots X_n)$$

  - q : current state
  - $X_1 \ldots X_{i-1}\ X_i \ldots X_n$: current string on tape
  - X_i : current tape symbol

TM: transitions of a TM (3)

- transition:
  - if $\delta(q, X_i) = (p, Y, L)$, then

    $(X_1 \ldots X_{i-1}\ q\ X_i \ldots X_n) \vdash (X_1 \ldots X_{i-2}\ p\ X_{i-1}\ Y\ X_{i+1} \ldots X_n)$

    - if $i = 1$, then $(q\ X_1 \ldots X_n) \vdash (p\ \mathsf{B}\ Y\ X_2 \ldots X_n)$
    - if $i = n$ and $Y = \mathsf{B}$, then $(X_1 \ldots X_{n-1}\ q\ X_n) \vdash (X_1 \ldots X_{n-2}\ p\ X_{n-1})$
  - if $\delta(q, X_i) = (p, Y, R)$, then

    $(X_1 \ldots X_{i-1}\ q\ X_i \ldots X_n) \vdash (X_1 \ldots X_{i-1}\ Y\ p\ X_{i+1} \ldots X_n)$

    - if $i = 1$ and $Y = \mathsf{B}$, then $(q\ X_1 \ldots X_n) \vdash (p\ X_2 \ldots X_n)$
    - if $i = n$, then $(X_1 \ldots X_{n-1}\ q\ X_n) \vdash (X_1 \ldots X_{n-1}\ Y\ p\ \mathsf{B})$

TM: language accepted by a TM

- Language accepted by a TM

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

  - $L(M) = \{ w \mid w \in \Sigma^* ; (q_0 w) \vdash^* (\alpha\, q\, \beta) ; q \in F \}$

TM: example of TM

$$M = (\{ q_0 , q_1 , q_2 , q_3 , q_4 \} , \{ 0 , 1 \} , \{ 0 , 1 , X , Y , \mathbf{B} \} , \delta , q_0 , \{q_4\})$$

δ	0	1	X	Y	B
$\rightarrow q_0$	(q_1, X, R)			(q_3, Y, R)	
q_1	$(q_1, 0, R)$	(q_2, Y, L)		(q_1, Y, R)	
q_2	$(q_2, 0, L)$		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	$(q_4, \mathbf{B}, R)$
$^*q_4$					

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

$(q_0 0011) \vdash (X q_1 011) \vdash (X0 q_1 11) \vdash (X q_2 0Y1) \vdash (q_2 X0Y1) \vdash (X q_0 0Y1) \vdash$
$(XX q_1 Y1) \vdash (XXY q_1 1) \vdash (XX q_2 YY) \vdash (X q_2 XYY) \vdash (XX q_0 YY) \vdash (XXY q_3 Y)$
$(XXYY q_3 \mathbf{B} \vdash \quad (XXYY\mathbf{B} q_4 \vdash \quad \leftarrow$ *accepting configuration*

TM: recursive sets

- the languages accepted by *TM's* are called ***recursively enumerable sets*** and are equivalent to the ***type 0 languages*** (*phrase structure*)
- Halting problem
  - a TM always ***halts*** when it is in an accepting state
  - it is not always possible to require that a TM ***halts*** if it does not accept
- the *membership* of a string in a *recursively enumerable set* is ***undecidable***
- the languages accepted by TM's that always ***halt*** are called ***recursive sets***
- the *membership* of a string in a *recursive set* is ***decidable***

TM: decidability / computability

- the TM defines the most general model of computation
  - any computable function can be computed by a TM (*Church-Turing thesis*)
- the TM can be used to classify languages / problems / functions
  - non recursively enumerable
    - cannot be represented by any TM
  - recursively enumerable / undecidable / uncomputable
    - represented by a TM that not always halts
  - recursive / decidable / computable
    - represented by a TM that always halts (*algorithm*)