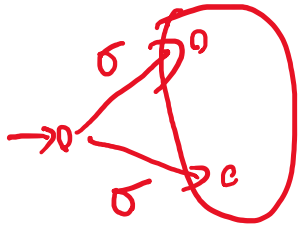
Nondeterministic Finite Automaton

Nondeterministic finite automaton

A nondeterministic finite automaton (NFA) $\mathbf{G}$ is a 5-tuple $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_a)$, where



Q : state set; a finite set of states

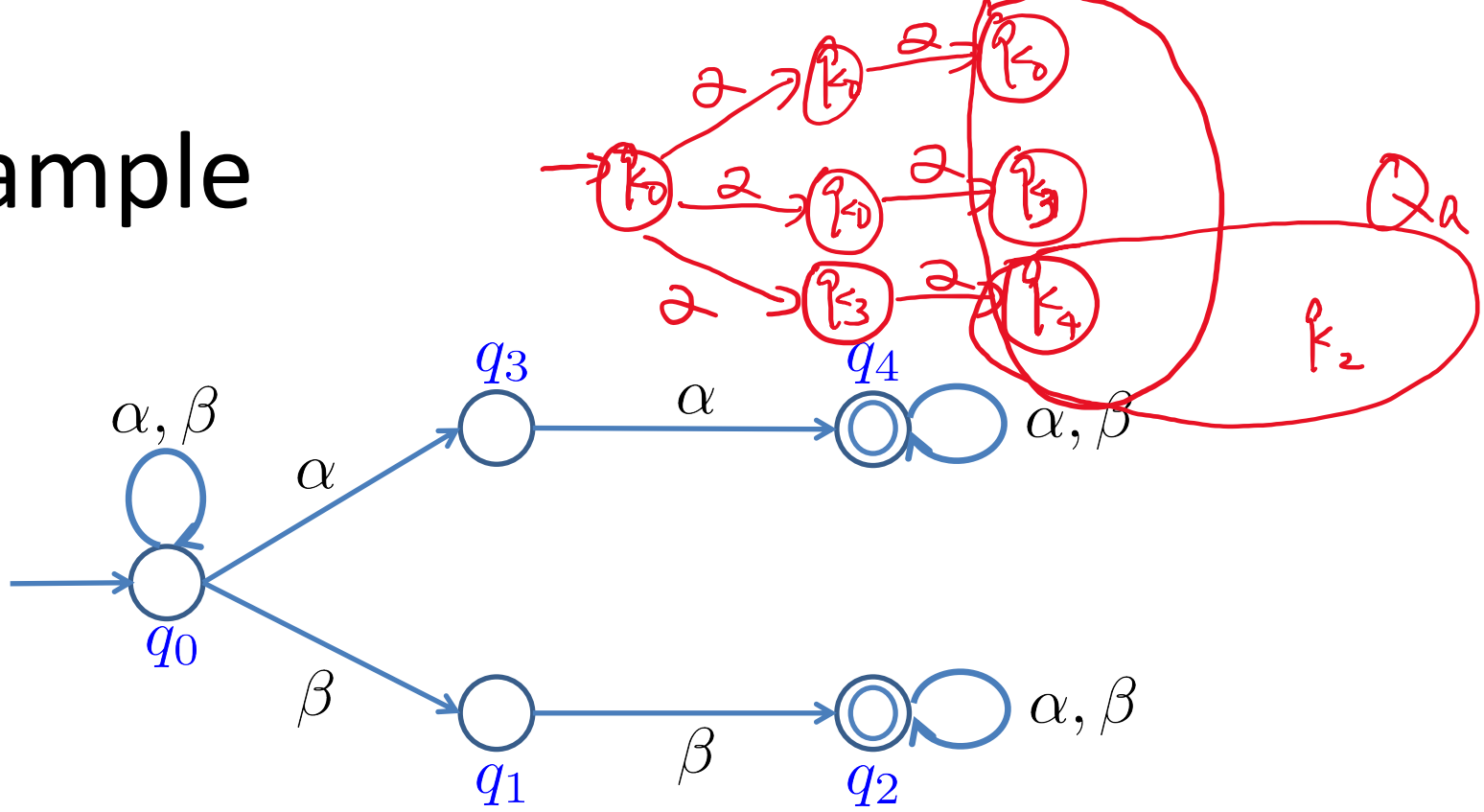
Σ : alphabet; a finite set of symbols

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow Pwr(Q)$: state transition function

$q_0 \in Q$: initial state

$Q_a \subseteq Q$: subset of accept states

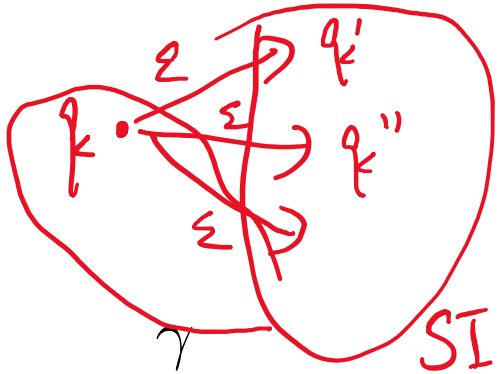
Example



Accepted/recognized language

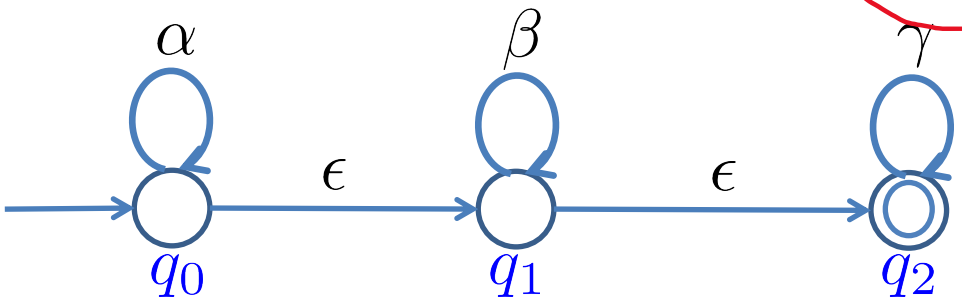
$L_a(\mathbf{G}) =$ {22, 222, ...
ββ, βββ, ...
β22, β222, ...
2ββ, 2βββ, ...}

Example



SILENT(q)

$\mathbf{G}$



SILENT(q_0) = {q_0, q_1, q_2}

SILENT(q_1) = {q_1, q_2}

SILENT(q_2) = {q_2}

Accepted/recognized language

$L_a(\mathbf{G}) =$ {ε, α, αα, ...
β, αβ, ααβ, ...
γ, γγ ...}

Equivalence of NFA and DFA

NFA appears to be more powerful than DFA
(indeed, DFA is a special NFA)

In fact: NFA and DFA recognize the same class of languages:
i.e. *regular languages*

(Recall: Defn. A language $L \subseteq \Sigma^*$ is a *regular language*
if L can be recognized by a DFA)

Why need NFA then?
NFA turns out to be much simpler for language representation

Equivalence of NFA and DFA

Let's prove the equivalence.

First DFA $\rightarrow$ NFA: this is trivial.
If L is recognized by a DFA $\mathbf{D}$,
then L is recognized by an NFA $\mathbf{G} = \mathbf{D}$

Next NFA $\rightarrow$ DFA.
If L is recognized by an NFA $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_a)$,
then we are going to construct a DFA $\mathbf{D}$ that recognizes L

Equivalence of NFA and DFA

Given an NFA $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_a)$ that recognizes L
Construct a DFA $\mathbf{D} = (X, \Sigma, \xi, x_0, X_a)$ where

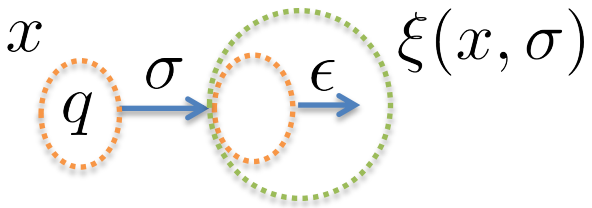
$X = Pwr(Q)$

$x_0 = \text{SILENT}(q_0) =$

$X_a = \{x \in X \mid x \cap Q_a \neq \emptyset\}$

$\xi : X \times \Sigma \to X$

$\xi(x, \sigma) = \text{SILENT}(\bigcup_{q \in x} \delta(q, \sigma))$



Let $s = \sigma_1 \cdots \sigma_k (\sigma_i \in \Sigma \cup \{\epsilon\})$ be a string recognized by $\mathbf{G}$
i.e. there exists states $q_1, \ldots, q_k \in Q$ s.t.

$$q_1 \in \delta(q_0, \sigma_1), \ldots, q_k \in \delta(q_{k-1}, \sigma_k) \;\&\; q_k \in Q_a$$

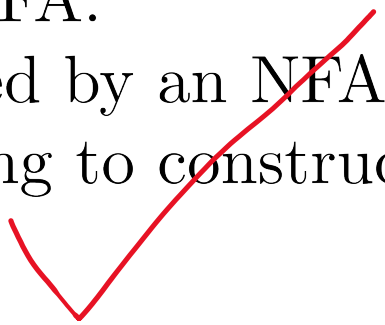
Then s travels from x_0 to a state in X_a, i.e. is recognized by $\mathbf{D}$

Equivalence of NFA and DFA

Let's prove the equivalence.

First DFA $\rightarrow$ NFA: this is trivial.
If L is recognized by a DFA $\mathbf{D}$,
then L is recognized by an DFA $\mathbf{G} = \mathbf{D}$

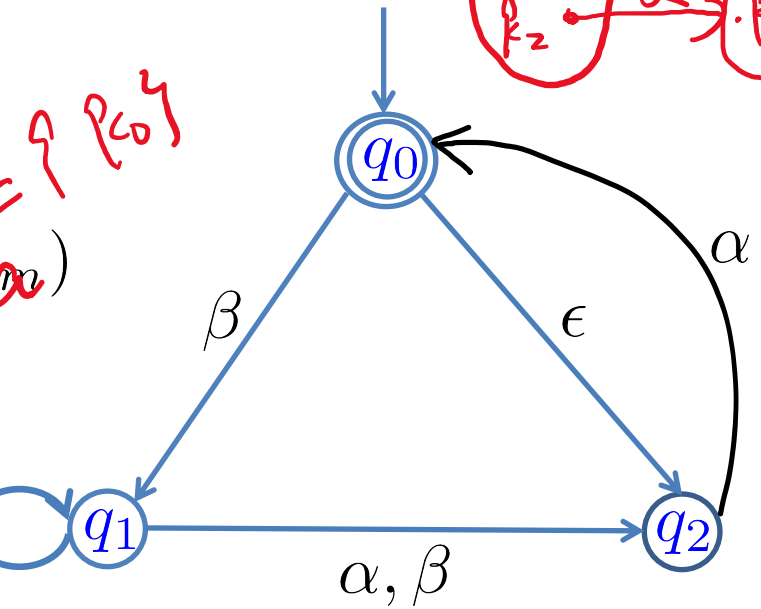
Next NFA $\rightarrow$ DFA.
If L is recognized by an NFA $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_a)$,
then we are going to construct a DFA $\mathbf{D}$ that recognizes L



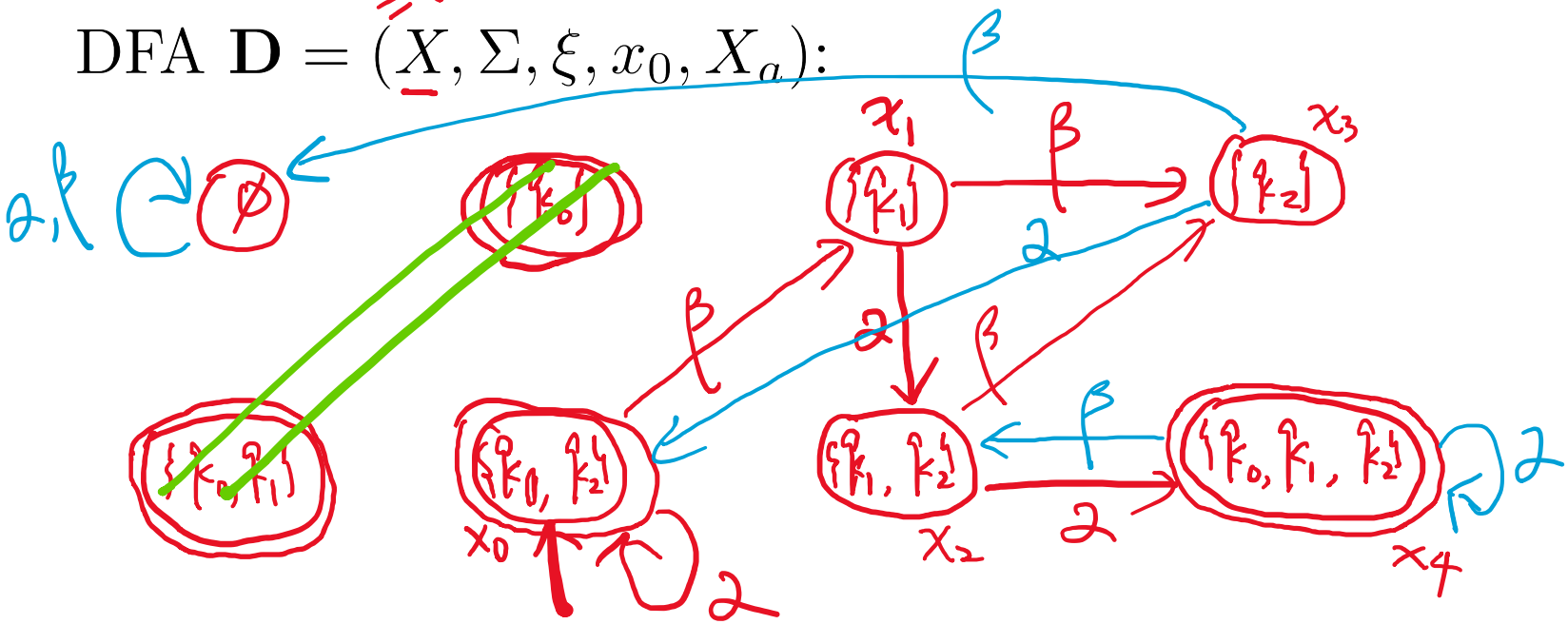
Conclusion: NFA and DFA are equivalent.

Example

NFA $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$



DFA $\mathbf{D} = (X, \Sigma, \xi, x_0, X_a)$:



Recap

Fact: NFA and DFA recognize the same class of languages

Proposition: A language $L \subseteq \Sigma^*$ is regular iff
L can be recognized by an NFA

Regular languages $\Leftrightarrow$ NFA $\Leftrightarrow$ DFA

Regular Operations

Union

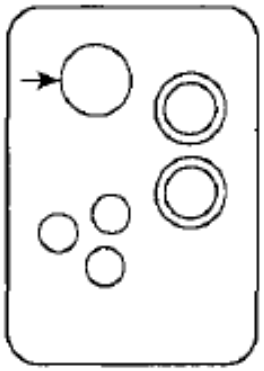
Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages.

Then their union is $L_1 \cup L_2 = \{s \in \Sigma^* \mid s \in L_1 \text{ or } s \in L_2\}$

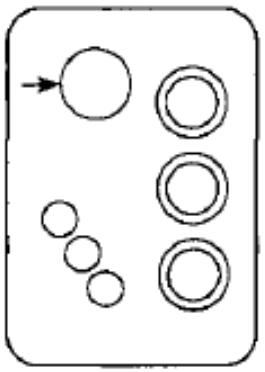
Claim: $L_1 \cup L_2$ is again a regular language.

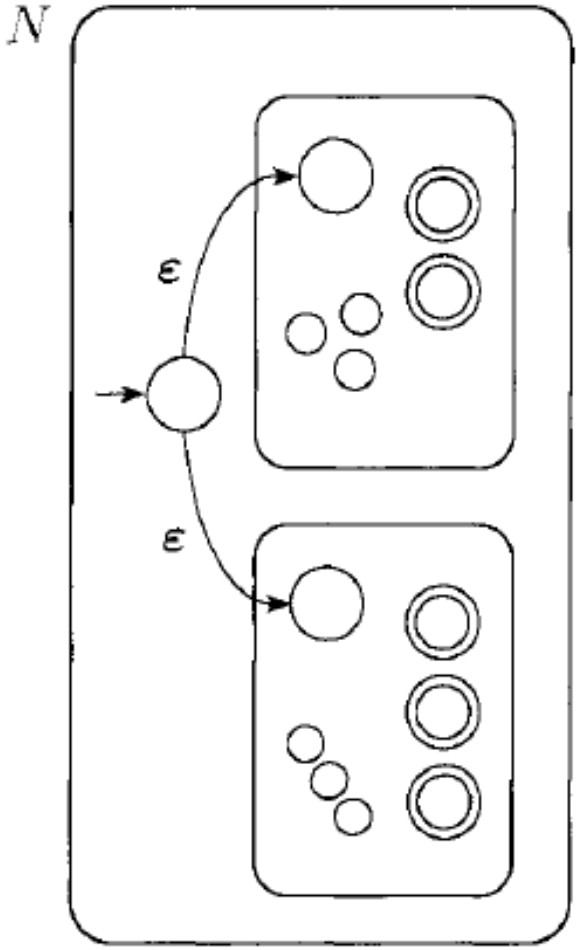
Union

L_1



L_2

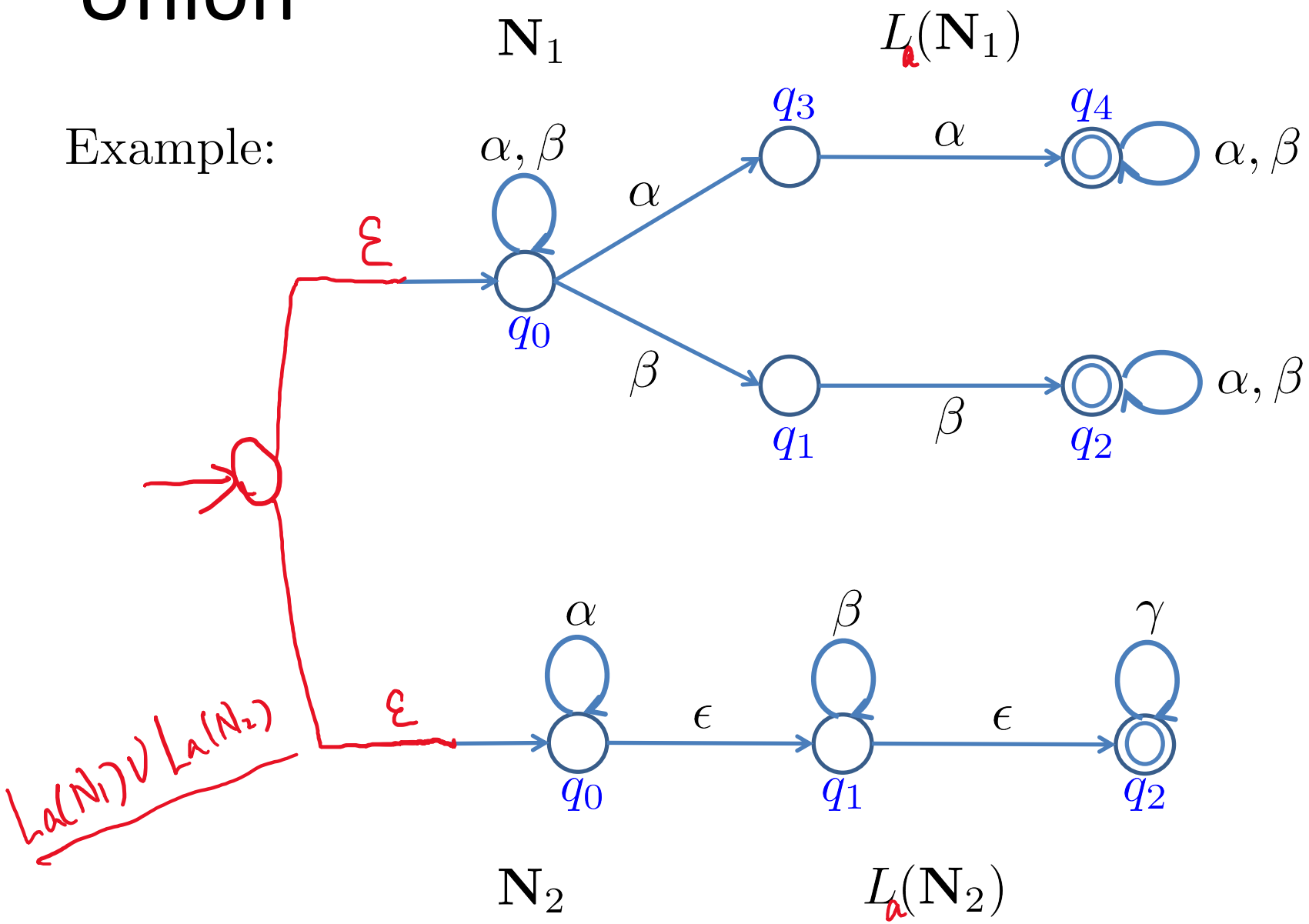




$L_1 \cup L_2$

Union

Example:



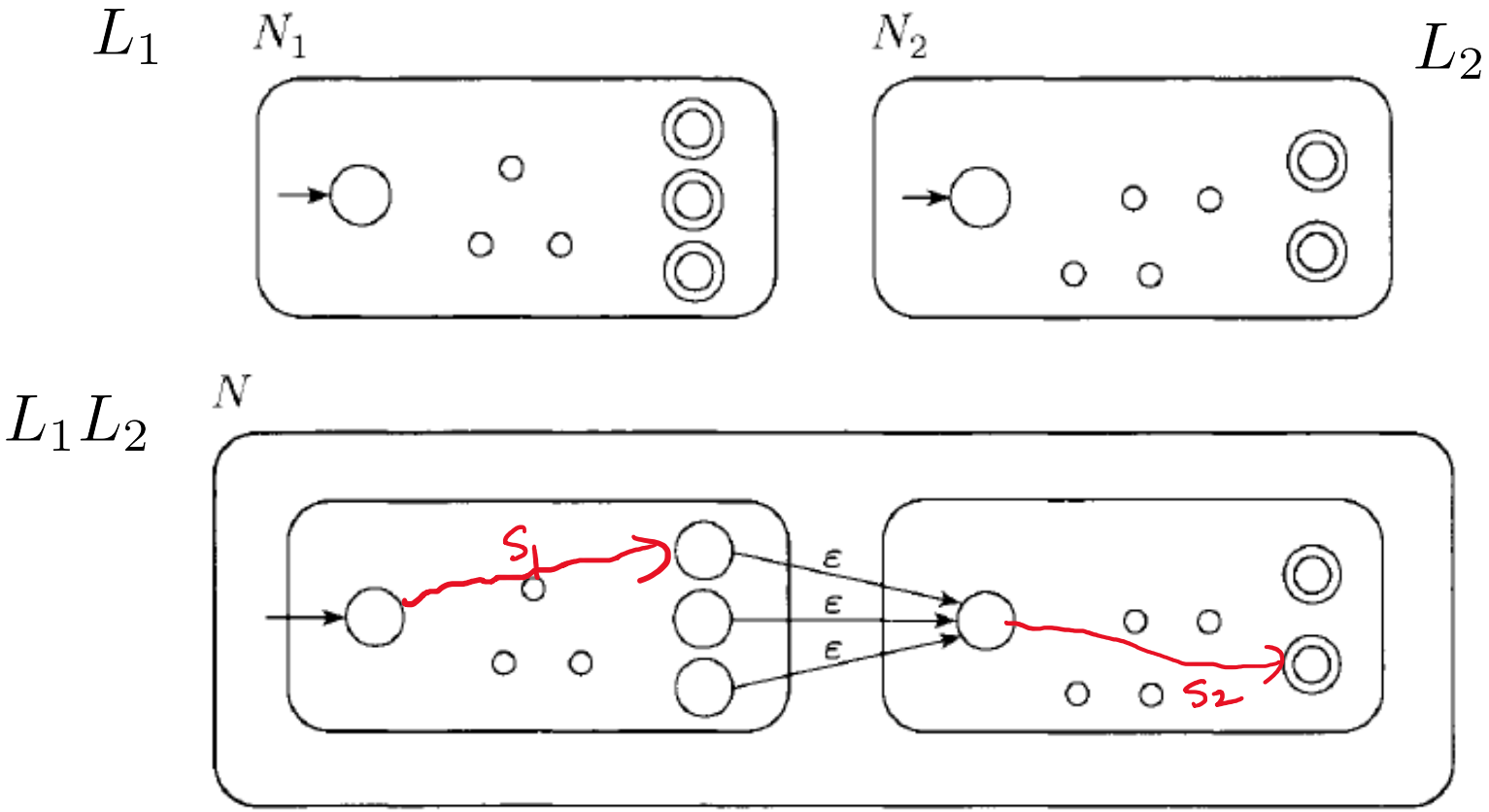
Catenation

Let $L_1, L_2 \subseteq \Sigma^*$ be two regular languages.

Then their catenation is $L_1 L_2 = \{s = s_1 s_2 \in \Sigma^* \mid s_1 \in L_1 \text{ and } s_2 \in L_2\}$

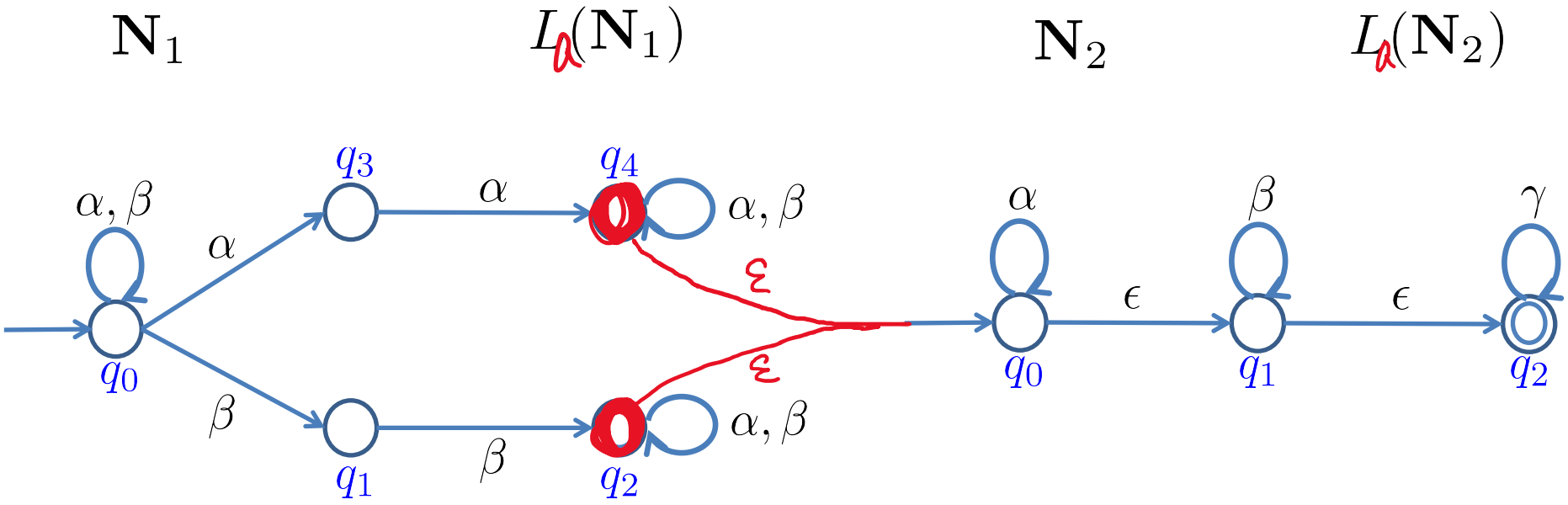
Claim: $L_1 L_2$ is again a regular language.

Catenation



Catenation

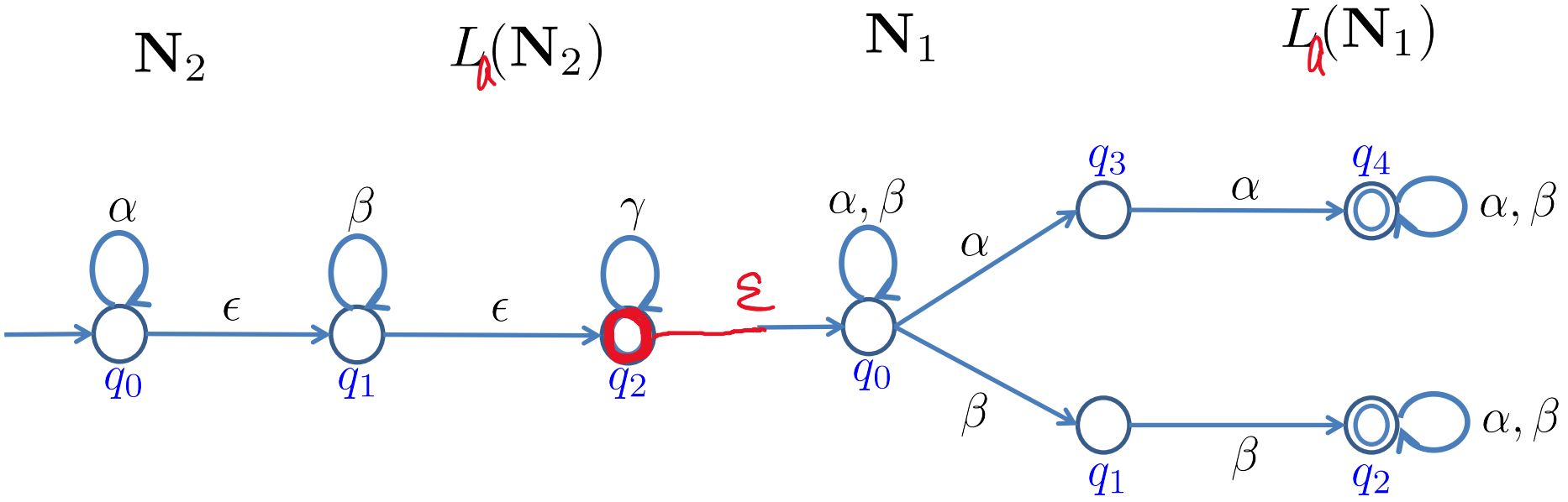
Example:



$L_a(N_1)L_a(N_2)$

Catenation

Example:



$L_a(N_2)\,L_a(N_1)$

Star

$L = \{a, a\beta\}$

$L^* = \{\varepsilon, a, a\beta, aa, a\beta a\beta, aa\beta, a\beta a, aaa, a\beta a\beta a\beta, aaa\beta, a a\beta a\beta, a\beta a\beta a, \ldots\}$

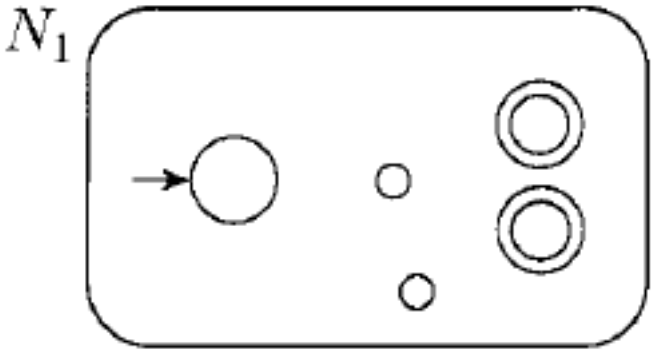
Let $L \subseteq \Sigma^*$ be a regular language.

Then its star is $L^* = \{\epsilon\} \cup \{s = s_1 \cdots s_k \in \Sigma^* \mid k \geq 1 \text{ and } s_i \in L\}$

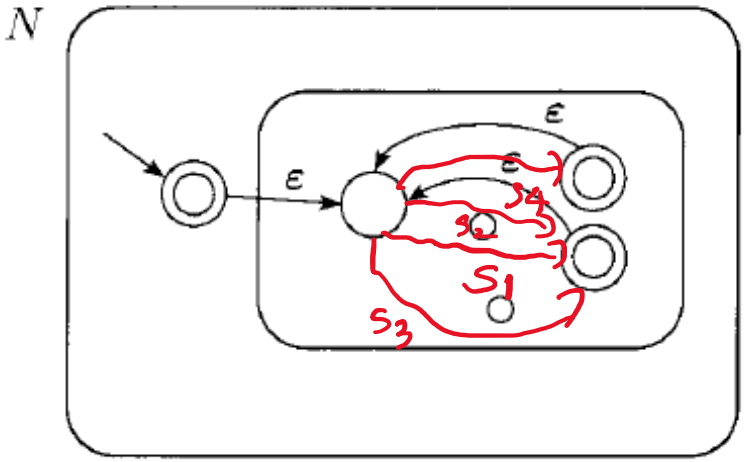
Claim: L^* is again a regular language.

Star

L



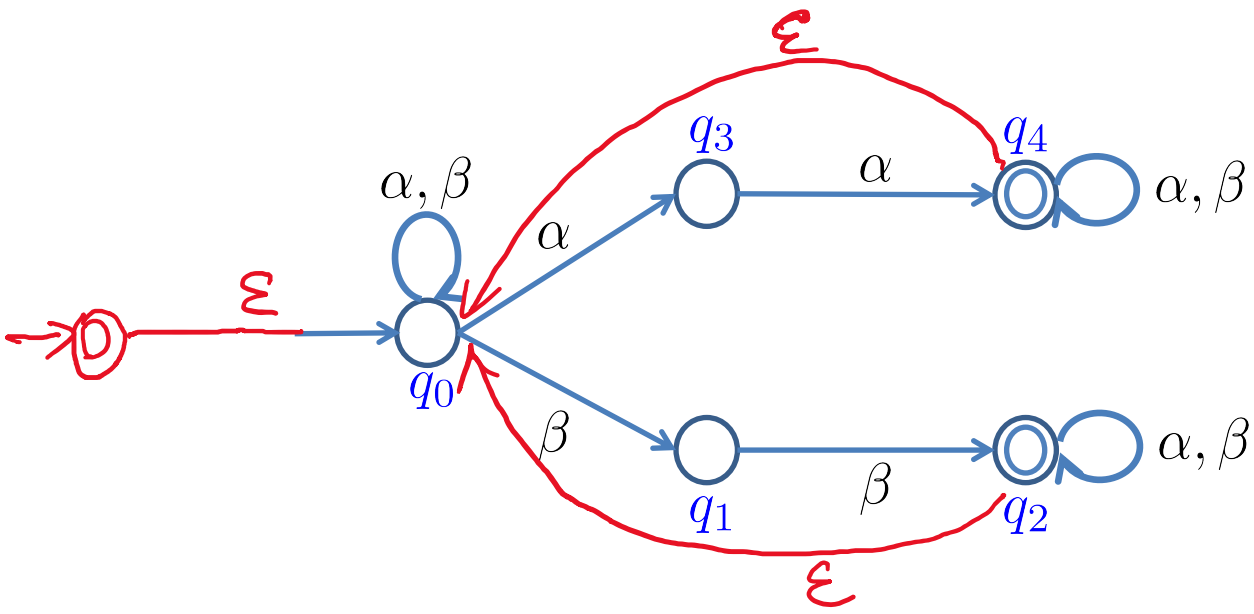
L^*



Star

Example:

$\mathbf{N}_1$ $L_a(\mathbf{N}_1)$

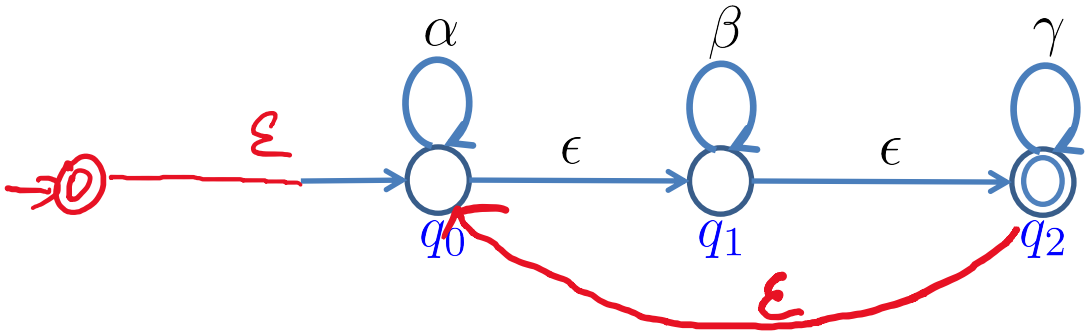


$L_a(N_1)^*$

Star

Example:

$\mathbf{N}_2$ $L_a(\mathbf{N}_2)$



$L_a^*(N_2)$

Recap

Regular operations: union, catenation, star

Fact: regular languages are closed under regular operations.

Regular expressions

In arithmetic, we use $+$, $\times$ to build up expressions e.g.

$$(5+3)\times 4$$

Similarly, we can use regular operations to build up regular expressions: e.g. $(\{\alpha\}\cup\{\beta\})\{\alpha\}^*$

L_1 L_2 L_3

For simplicity: $(\alpha\cup\beta)\alpha^*$