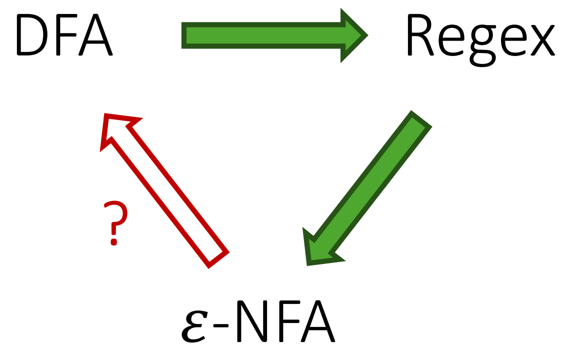
CS 483 - Introduction to the Theory of Computation

Lecture 4

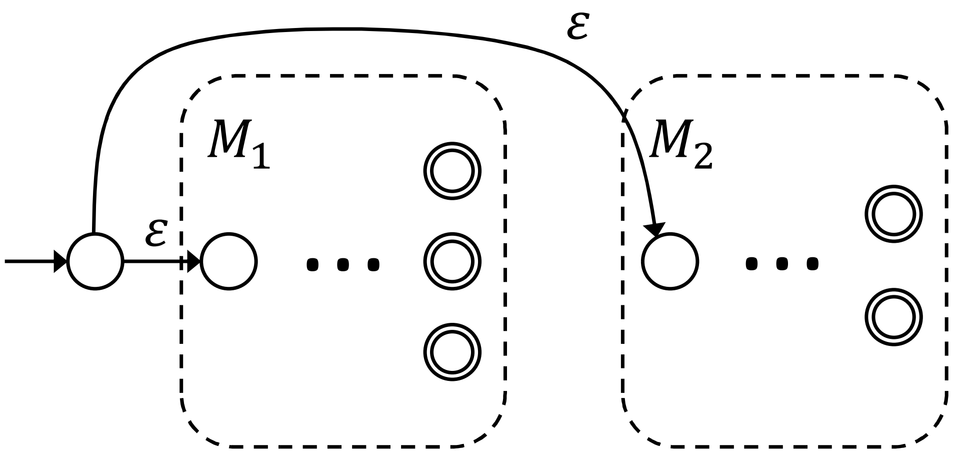
From ε-NFA to DFA

We have shown:



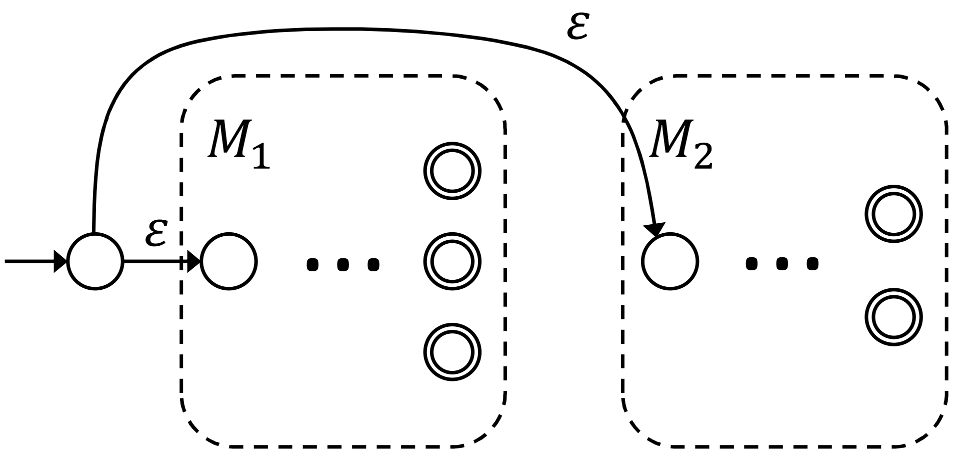
Task: Given an ε-NFA M, construct DFA M' such that $L(M') = L(M)$.

From ε-NFA to DFA

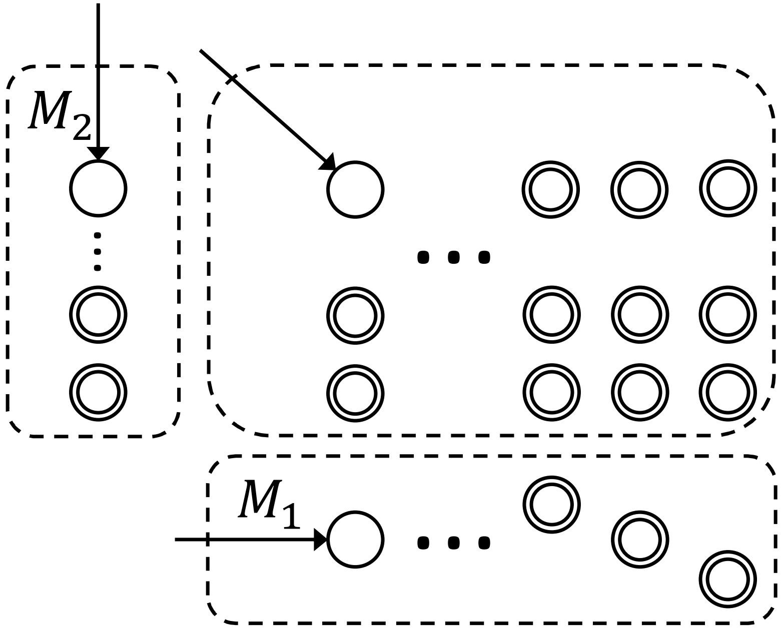


ε-NFA for union:
two possibilities

From ε-NFA to DFA



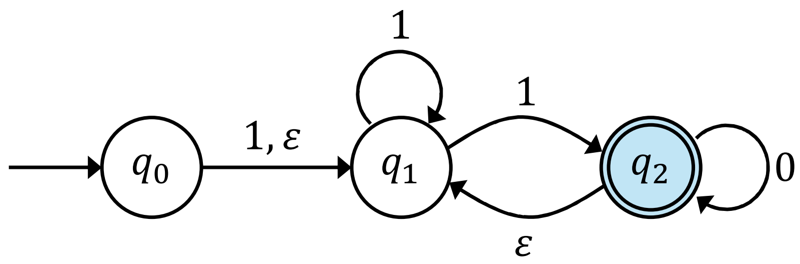
ε-NFA for union:
two possibilities



DFA for union:
two possibilities in parallel

From ε-NFA to DFA

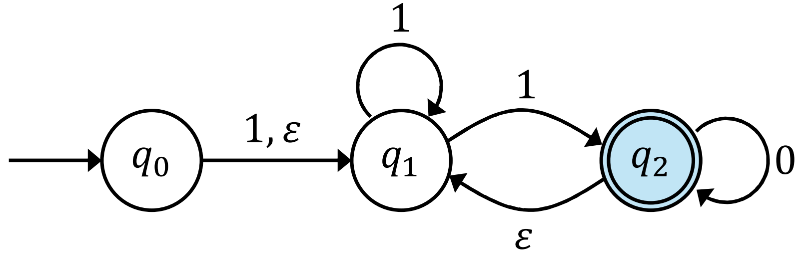
Idea: Record **all** possibilities in parallel.



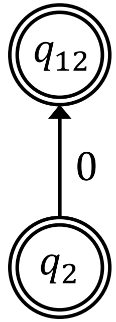
Currently at q_2, next symbol is 0:
the next possible states?

From ε-NFA to DFA

Idea: Record **all** possibilities in parallel.

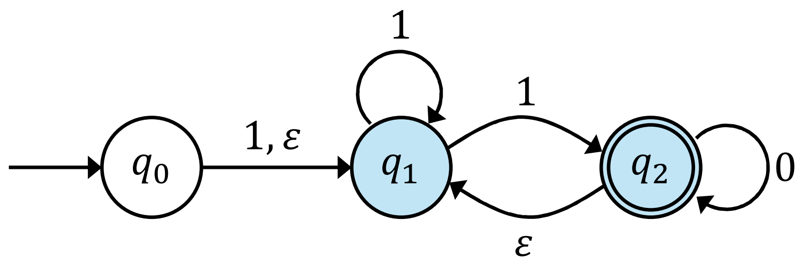


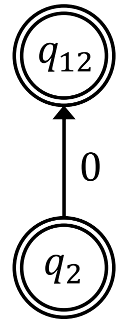
Currently at q_2, next symbol is 0:
the next possible states?



From ε-NFA to DFA

Idea: Record **all** possibilities in parallel.

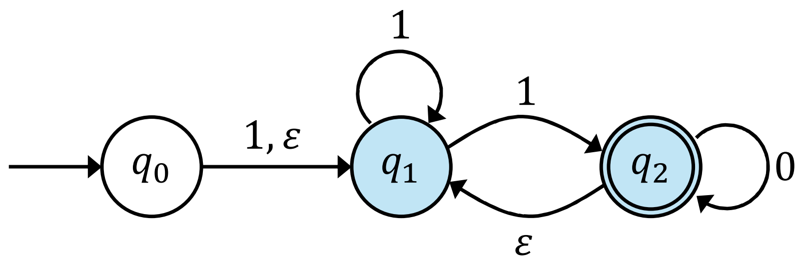




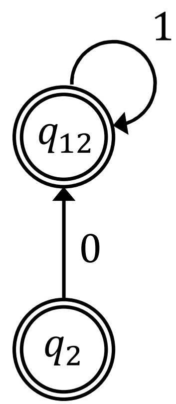
Currently at q_1 or q_2, next symbol is 1:
the next possible states?

From ε-NFA to DFA

Idea: Record **all** possibilities in parallel.

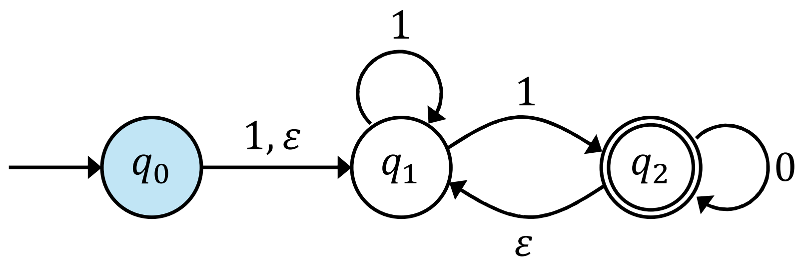


Currently at q_1 or q_2, next symbol is 1:
the next possible states?



From ε-NFA to DFA

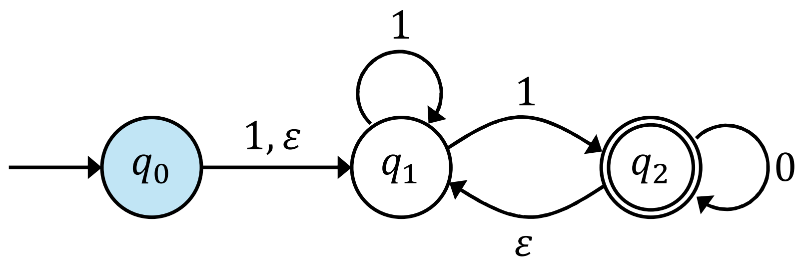
Idea: Record **all** possibilities in parallel.



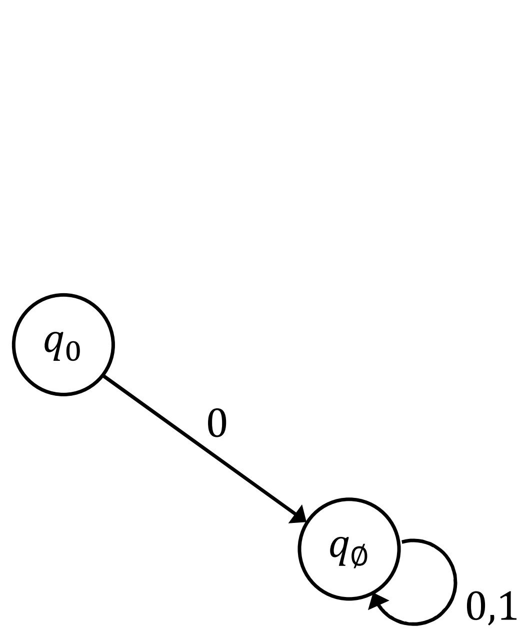
Currently at q_0, next symbol is 0:
the next possible states?

From ε-NFA to DFA

Idea: Record **all** possibilities in parallel.

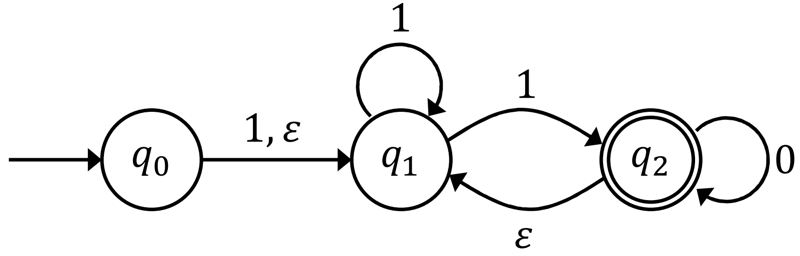


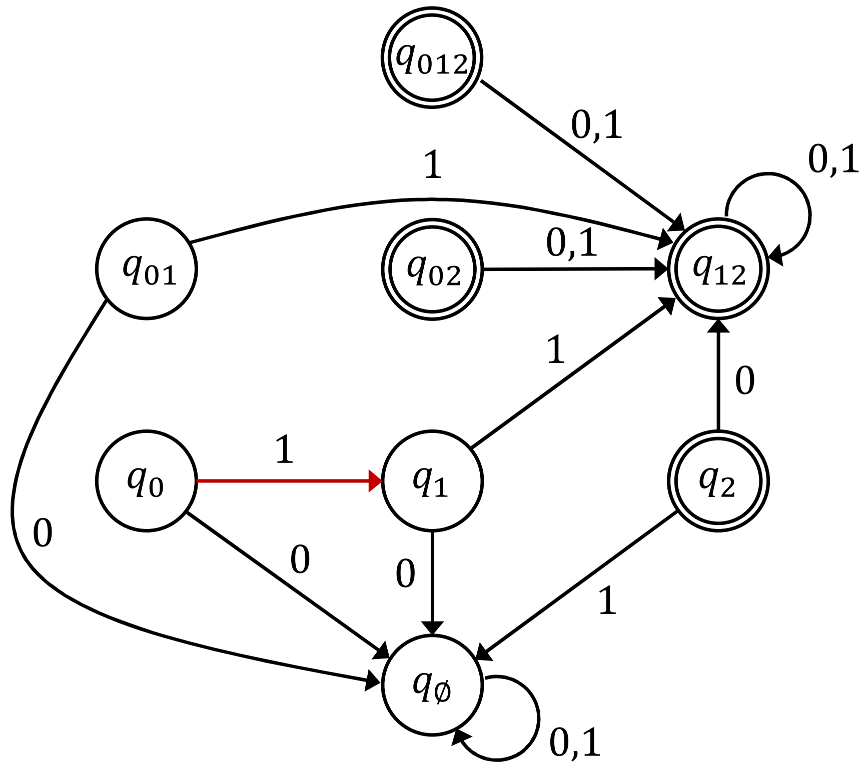
Currently at q_0, next symbol is 0:
the next possible states?



From ε-NFA to DFA

Idea: Record **all** possibilities in parallel.

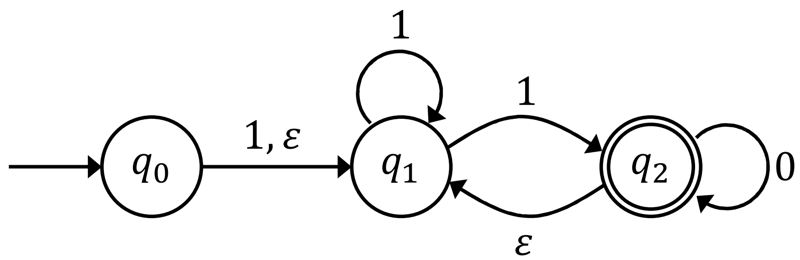


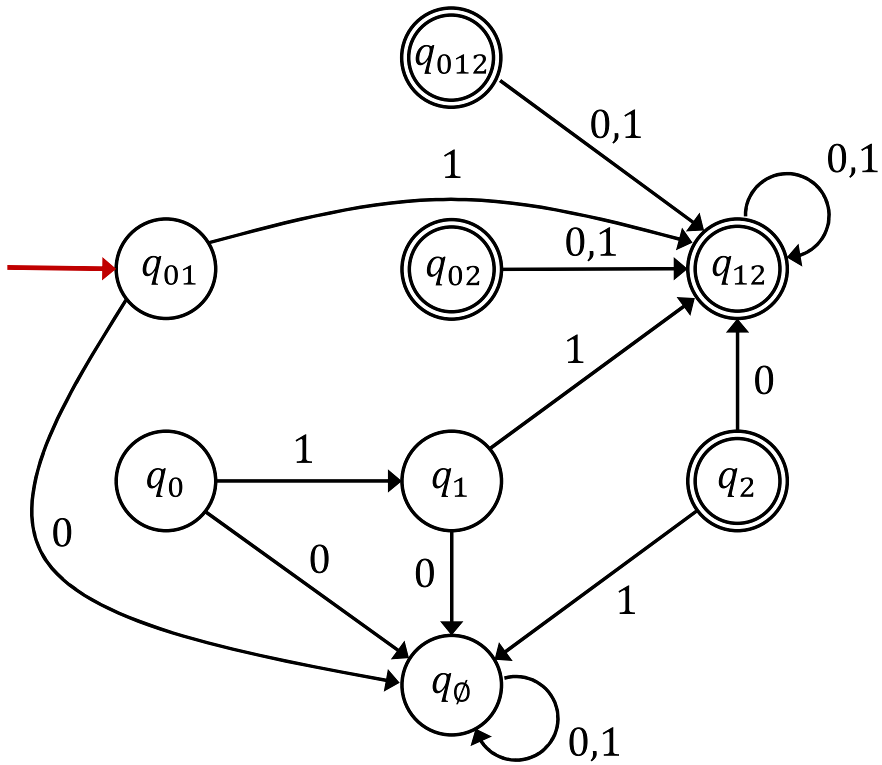


Transitions with $\Sigma = \{0,1\}$,
followed by possible ε's.

From ε-NFA to DFA

Idea: Record **all** possibilities in parallel.





The initial states contains every state reachable by initial ε's.

From ε-NFA to DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the ε-NFA, define DFA $M' = (Q', \Sigma, \delta', q_0', F')$ where:

- $Q' = 2^Q = \{P \mid P \subseteq Q\}$

From ε-NFA to DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the ε-NFA, define DFA $M' = (Q', \Sigma, \delta', q_0', F')$ where:

- $Q' = 2^Q = \{P \mid P \subseteq Q\}$

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

From ε-NFA to DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the ε-NFA, define DFA $M' = (Q', \Sigma, \delta', q_0', F')$ where:

- $Q' = 2^Q = \{P \mid P \subseteq Q\}$

- $\delta'(P, a) = R\left(\bigcup_{q \in P} \delta(q, a)\right)$

  Here $R(S) = \{r \in Q \mid \exists s \in S \text{ that can reach } r \text{ with only } \varepsilon\text{-transitions}\}$

From ε-NFA to DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the ε-NFA, define DFA $M' = (Q', \Sigma, \delta', q_0', F')$ where:

- $Q' = 2^Q = \{P \mid P \subseteq Q\}$

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

  Here $R(S) = \{r \in Q \mid \exists s \in S \text{ that can reach } r \text{ with only } \varepsilon\text{-transitions}\}$

- $q_0' = R(\{q_0\})$

From ε-NFA to DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the ε-NFA, define DFA $M' = (Q', \Sigma, \delta', q_0', F')$ where:

- $Q' = 2^Q = \{P \mid P \subseteq Q\}$

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

  Here $R(S) = \{r \in Q \mid \exists s \in S \text{ that can reach } r \text{ with only } \varepsilon\text{-transitions}\}$

- $q_0' = R(\{q_0\})$

- $F' = \{P \mid P \subseteq Q, P \cap F \neq \emptyset\}$

From ε-NFA to DFA

To prove $L(M') = L(M)$, consider $w = w_1 \dots w_n$ and the sequence of states $P_0, P_1, \dots, P_n$ on M' with input w.

By definition, P_i is the set of states in M that can be reached from states in P_{i-1} with transitions $w_i \varepsilon^*$.

From ε-NFA to DFA

To prove $L(M') = L(M)$, consider $w = w_1 \dots w_n$ and the sequence of states $P_0, P_1, \dots, P_n$ on M' with input w.

By definition, P_i is the set of states in M that can be reached from states in P_{i-1} with transitions $w_i\varepsilon^*$.

Use induction, we can show that P_i is exactly the set of states in M that can be reached from q_0 with transitions $\varepsilon^* w_1 \varepsilon^* \dots w_i \varepsilon^*$.

From ε-NFA to DFA

To prove $L(M') = L(M)$, consider $w = w_1 \dots w_n$ and the sequence of states $P_0, P_1, \dots, P_n$ on M' with input w.

By definition, P_i is the set of states in M that can be reached from states in P_{i-1} with transitions $w_i \varepsilon^*$.

Use induction, we can show that P_i is exactly the set of states in M that can be reached from q_0 with transitions $\varepsilon^* w_1 \varepsilon^* \dots w_i \varepsilon^*$.

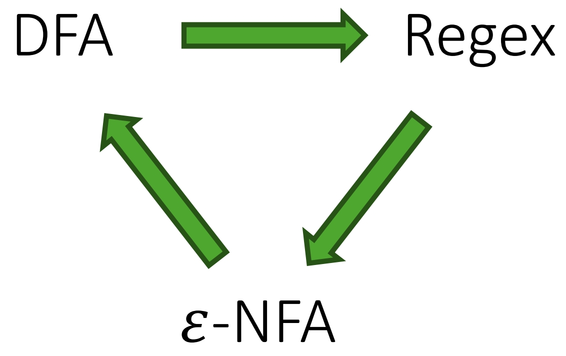
Therefore $w \in L(M') \Leftrightarrow \exists q \in P_n \cap F$

$\Leftrightarrow \exists q \in F$, q_0 can reach q with padded w

$\Leftrightarrow w \in L(M)$

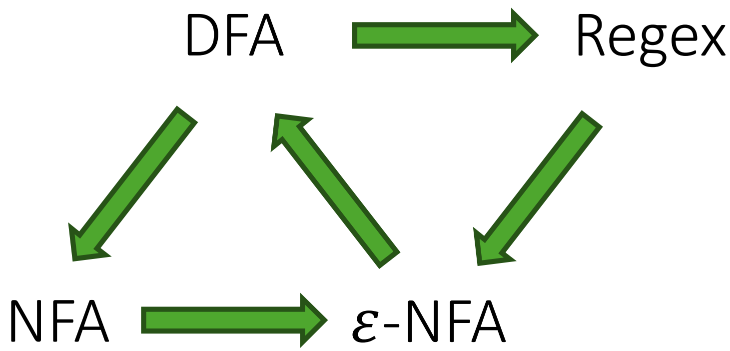
Equivalent Definitions of Regular Languages

What we have proved:



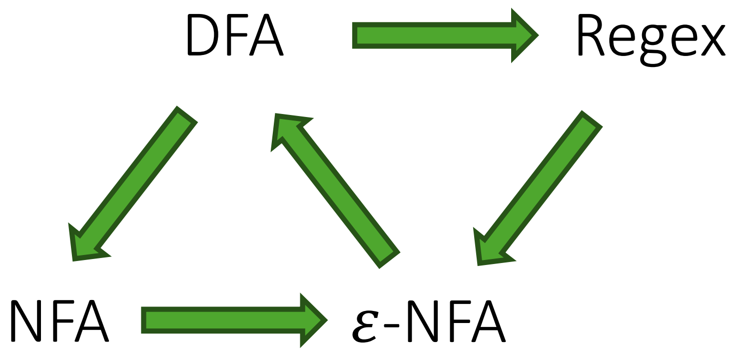
Equivalent Definitions of Regular Languages

What we have proved:



Equivalent Definitions of Regular Languages

What we have proved:



Regular language can be equivalently defined using any one of these models.

Equivalent Definitions of Regular Languages

Example: Prove that every finite language is regular.

Equivalent Definitions of Regular Languages

Example: Prove that every finite language is regular.

Proof: Assume $L = \{w_1, w_2 \ldots, w_N\}$, construct a regex for L:

$$w_1 \cup w_2 \cup \cdots \cup w_N$$

Regular Grammar

A grammar is a 4-tuple $G = (N, \Sigma, P, S)$ where:

- N is a finite set of nonterminal symbols

Regular Grammar

A grammar is a 4-tuple $G = (N, \Sigma, P, S)$ where:

- N is a finite set of nonterminal symbols
- Σ is a finite set of terminal symbols (alphabet) that is disjoint from N

Regular Grammar

A grammar is a 4-tuple $G = (N, \Sigma, P, S)$ where:

- N is a finite set of nonterminal symbols
- Σ is a finite set of terminal symbols (alphabet) that is disjoint from N
- P is a finite set of production rules. Each rule is of form $\alpha \rightarrow \beta$, where
  - $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ (i.e., w contains at least one nonterminal symbol)
  - $\beta \in (N \cup \Sigma)^*$

Regular Grammar

A grammar is a 4-tuple $G = (N, \Sigma, P, S)$ where:

- N is a finite set of nonterminal symbols
- Σ is a finite set of terminal symbols (alphabet) that is disjoint from N
- P is a finite set of production rules. Each rule is of form $\alpha \rightarrow \beta$, where
  - $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ (i.e., w contains at least one nonterminal symbol)
  - $\beta \in (N \cup \Sigma)^*$
- $S \in N$ is the starting symbol.

Regular Grammar

A grammar is a 4-tuple $G = (N, \Sigma, P, S)$ where:

- N is a finite set of nonterminal symbols
- Σ is a finite set of terminal symbols (alphabet) that is disjoint from N
- P is a finite set of production rules. Each rule is of form $\alpha \rightarrow \beta$, where
  - $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ (i.e., w contains at least one nonterminal symbol)
  - $\beta \in (N \cup \Sigma)^*$
- $S \in N$ is the starting symbol.

The grammar G defines a language $L(G)$. Every $w \in L(G)$ can be produced by starting with S and using the rules in P to replace consecutive parts of the string until it contains only terminal symbols.

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \to a$
- $A \to aB$
- $A \to \varepsilon$

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \to a$
- $A \to aB$
- $A \to \varepsilon$

Example: $N = \{S, T\}, \Sigma = \{0,1\}$

$$\mathrm{S} \to 1T$$
$$T \to 0T$$
$$T \to 1T$$
$$T \to 1$$

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \to a$
- $A \to aB$
- $A \to \varepsilon$

Example: $N = \{S, T\}$, $\Sigma = \{0,1\}$

$S \to 1T$

$T \to 0T$

$T \to 1T$

$T \to 1$

S

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \to a$
- $A \to aB$
- $A \to \varepsilon$

Example: $N = \{S, T\}$, $\Sigma = \{0,1\}$

$\mathrm{S} \to 1T$

$T \to 0T$

$T \to 1T$

$T \to 1$

$1T$

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \rightarrow a$
- $A \rightarrow aB$
- $A \rightarrow \varepsilon$

Example: $N = \{S, T\}$, $\Sigma = \{0,1\}$

$S \rightarrow 1T$

$T \rightarrow 0T$

$T \rightarrow 1T$

$T \rightarrow 1$

$11T$

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \rightarrow a$
- $A \rightarrow aB$
- $A \rightarrow \varepsilon$

Example: $N = \{S, T\}$, $\Sigma = \{0,1\}$

$\mathrm{S} \rightarrow 1T$

$T \rightarrow 0T$

$T \rightarrow 1T$

$T \rightarrow 1$

$110T$

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \to a$
- $A \to aB$
- $A \to \varepsilon$

Example: $N = \{S, T\}$, $\Sigma = \{0,1\}$

$\mathrm{S} \to 1T$

$T \to 0T$

$T \to 1T$

$T \to 1$

$1101T$

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \to a$
- $A \to aB$
- $A \to \varepsilon$

Example: $N = \{S, T\}$, $\Sigma = \{0,1\}$

$$\mathrm{S} \to 1T$$
$$T \to 0T$$
$$T \to 1T$$
$$T \to 1$$

11011

Regular Grammar

A regular grammar $G = (N, \Sigma, P, S)$ is a grammar where every rule in P is one of the following (here $A, B \in N$, $a \in \Sigma$):

- $A \to a$
- $A \to aB$
- $A \to \varepsilon$

Example: $N = \{S, T\}, \Sigma = \{0,1\}$ $\quad$ $\mathrm{S} \to 1T$

simplified form: $\quad$ $T \to 0T \mid 1T \mid 1$

Regular Grammar

Theorem. A language is regular if and only if it can be produced by a regular grammar.

Regular Grammar

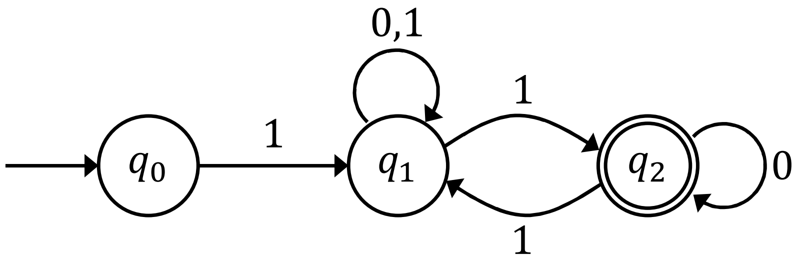
Theorem. A language is regular if and only if it can be produced by a regular grammar.

Here we use NFAs to characterize regular languages:

- For every NFA M, construct a regular grammar G with $L(G) = L(M)$;
- For every regular grammar G, construct an NFA M with $L(M) = L(G)$.

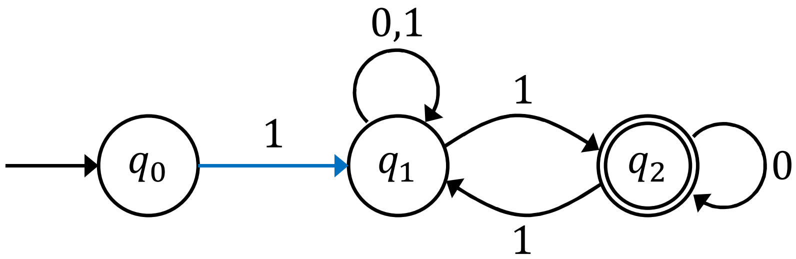
From NFA to Regular Grammar

NFA “reads” the string while regular grammar “writes” the string.



From NFA to Regular Grammar

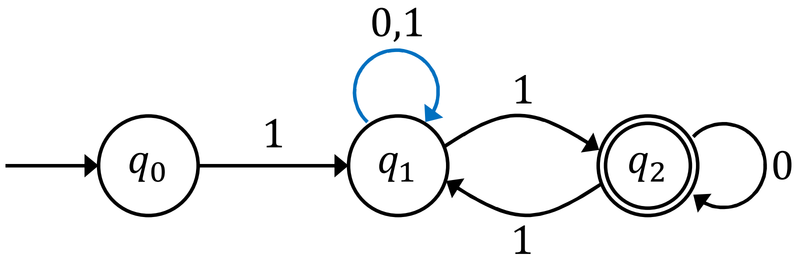
NFA “reads” the string while regular grammar “writes” the string.



$$q_0 \to 1q_1$$

From NFA to Regular Grammar

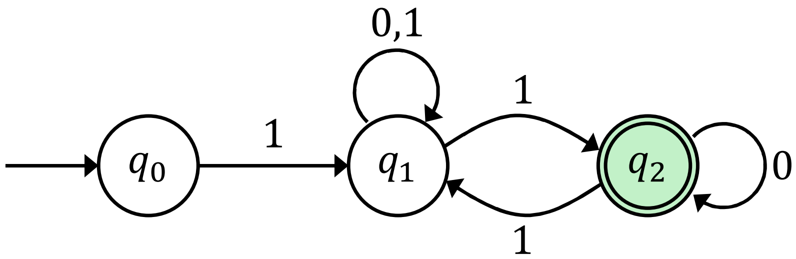
NFA “reads” the string while regular grammar “writes” the string.



$$q_0 \to 1q_1$$
$$q_1 \to 0q_1 \mid 1q_1$$

From NFA to Regular Grammar

NFA “reads” the string while regular grammar “writes” the string.



$$q_0 \to 1q_1$$

$$q_1 \to 0q_1 \mid 1q_1$$

…

$$q_2 \to \varepsilon$$

From NFA to Regular Grammar

Given NFA $M = (Q, \Sigma, \delta, q_0, F)$, define $G = (N, \Sigma, P, S)$ where:

- $N = Q$
- For every possible transition $q' \in \delta(q, a)$, P contains a rule $q \rightarrow aq'$
- For every $q \in F$, P also contains a rule $q \rightarrow \varepsilon$
- $S = q_0$

From NFA to Regular Grammar

Given NFA $M = (Q, \Sigma, \delta, q_0, F)$, define $G = (N, \Sigma, P, S)$ where:

- $N = Q$
- For every possible transition $q' \in \delta(q, a)$, P contains a rule $q \rightarrow aq'$
- For every $q \in F$, P also contains a rule $q \rightarrow \varepsilon$
- $S = q_0$

The production of $w \in L(G)$ corresponds exactly to the accepting path of $w \in L(M)$.

From Regular Grammar to NFA

Given regular grammar $G = (N, \Sigma, P, S)$, we first remove the rules of form $A \rightarrow a$ by replacing them with $A \rightarrow aE$, $E \rightarrow \varepsilon$.

From Regular Grammar to NFA

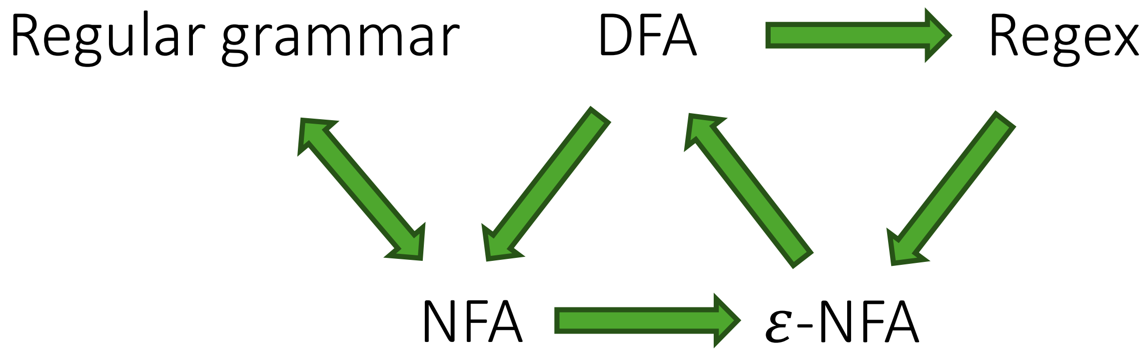
Given regular grammar $G = (N, \Sigma, P, S)$, we first remove the rules of form $A \to a$ by replacing them with $A \to aE$, $E \to \varepsilon$.

Now construct NFA $M = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = N \sqcup \{E\}$
- $\delta(q, a) = \{q' \in Q \mid P \text{ contains rule } q \to aq'\}$
- $q_0 = S$
- $F = \{q \in Q \mid P \text{ contains rule } q \to \varepsilon\}$

Equivalent Definitions of Regular Languages

What we have proved:



Regular language can be equivalently defined using any one of these models.