NFA/DFA: Closure Properties, Relation to Regular Languages

Lecture 5

Today

NFAs recap : Determinizing an NFA

Closure Properties of
class of languages accepted by NFAs/DFAs

Towards proving equivalence of regular languages and languages accepted by NFAs (and hence DFAs)

More closure Properties of
regular languages

NFA : Formally

$$N = (\Sigma, Q, \delta, s, F)$$

Σ: alphabet Q: state space s: start state F: set of accepting states

$$\delta : Q \times \{\Sigma \cup \varepsilon\} \to \mathcal{P}(Q)$$

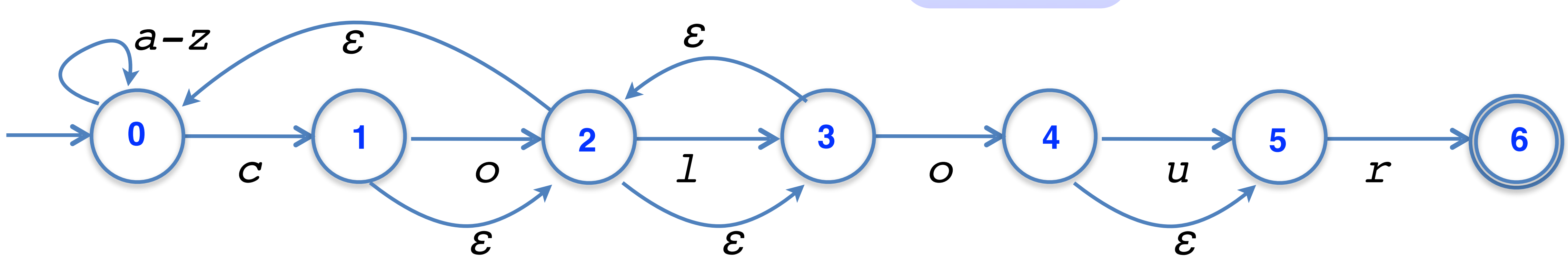
By default, NFA can have ε-moves

We say $q \overset{w}{\rightsquigarrow}_N p$ if $\exists\, a_1, \ldots, a_t \in \Sigma \cup \{\varepsilon\}$ and $q_1, \ldots, q_{t+1} \in Q$, such that $w = a_1 \ldots a_t$, $q_1 = q$, $q_{t+1} = p$, and $\forall\, i \in [\,1, t\,]$, $q_{i+1} \in \delta(q_i, a_i)$

$$L(N) = \{\, w \mid s \overset{w}{\rightsquigarrow}_N p \text{ for some } p \in F \,\}$$

e.g., $\delta(\mathbf{1}, o) = \{\mathbf{2}\}$, $\delta(\mathbf{1}, x) = \emptyset$, $\delta(\mathbf{1}, \varepsilon) = \{\mathbf{2}\}$.

ε-closure $C\varepsilon(\{\mathbf{1}\}) = \{\, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{0} \,\}$



ε-Moves is Syntactic Sugar

Can modify any NFA N, to get an NFA N_{new} without ε-moves

$a \in \Sigma$

$$N_{\text{new}} = (\Sigma, Q, \delta_{\text{new}}, s, F_{\text{new}})$$

$$\delta_{\text{new}}(q, a) = C_\varepsilon(\ \delta(\ C_\varepsilon(\{q\}), a))$$

$\{ p \mid q \overset{a}{\rightsquigarrow}_N p \}$

$q \overset{\varepsilon}{\rightsquigarrow}_N p \Leftrightarrow p \in C_\varepsilon(\{q\})$,

$q \overset{\varepsilon}{\rightsquigarrow}_{N\text{new}} p \Leftrightarrow p{=}q$,

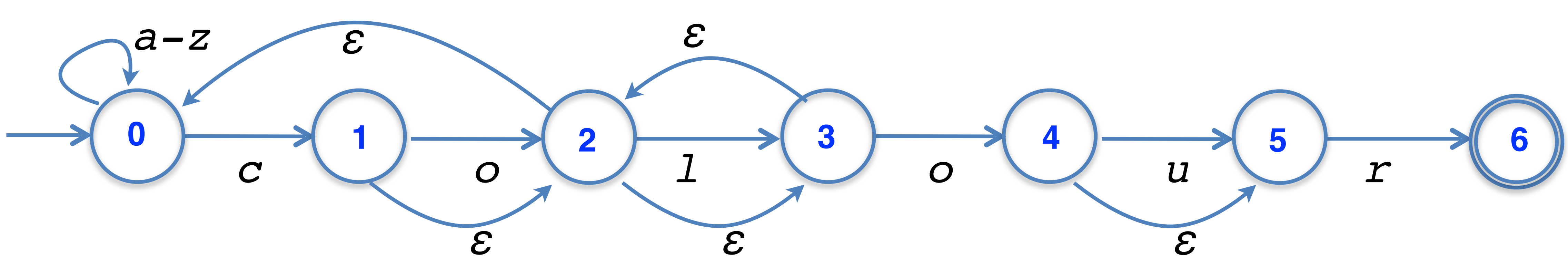
e.g.: $\delta_{\text{new}}(1, \texttt{o}) = \{0,2,3,4,5\}$

For $|w| \geq 1$, $q \overset{w}{\rightsquigarrow}_N p \Leftrightarrow q \overset{w}{\rightsquigarrow}_{N\text{new}} p$

Prove!

$$F_{\text{new}} = \begin{cases} F, & \text{if } C_\varepsilon(\{s\}) \cap F = \emptyset \\ F \cup \{s\}, & \text{otherwise.} \end{cases}$$

Theorem: $L(N) = L(N_{\text{new}})$



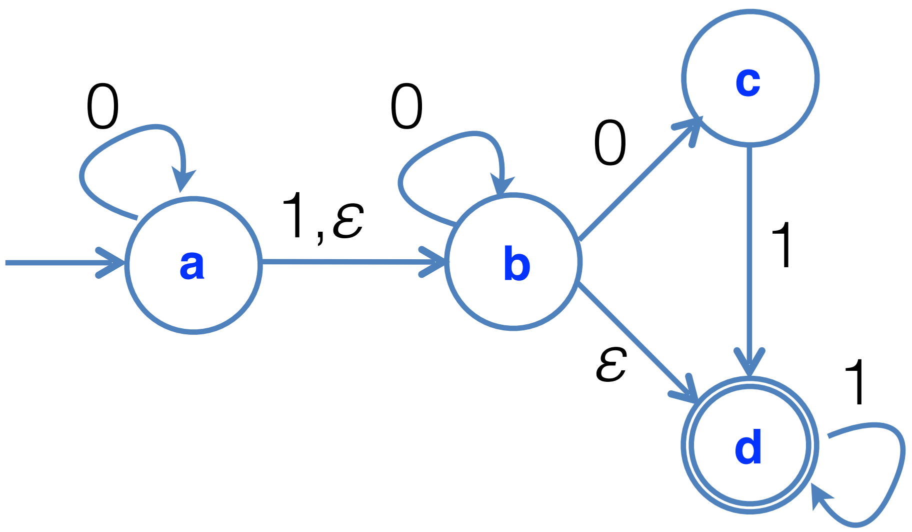
ε-Moves is Syntactic Sugar

Can modify any NFA N, to get an NFA N_{new} without ε-moves

$$N_{\text{new}} = (\Sigma, Q, \delta_{\text{new}}, s, F_{\text{new}})$$

$$\delta_{\text{new}}(q, a) = C_\varepsilon(\, \delta(\, C_\varepsilon(\{q\}), a))$$

q	C	a	δ	δ
a	{ a, b, d }	0	{ a, b, c }	{ a, b, c, d }
		1	{ b, d }	{ b, d }
b	{ b, d }	0	{ b, c }	{ b, c, d }
		1	{ d }	{ d }
c	{ c }	0	Ø	Ø
		1	{ d }	{ d }
d	{ d }	0	Ø	Ø
		1	{ d }	{ d }



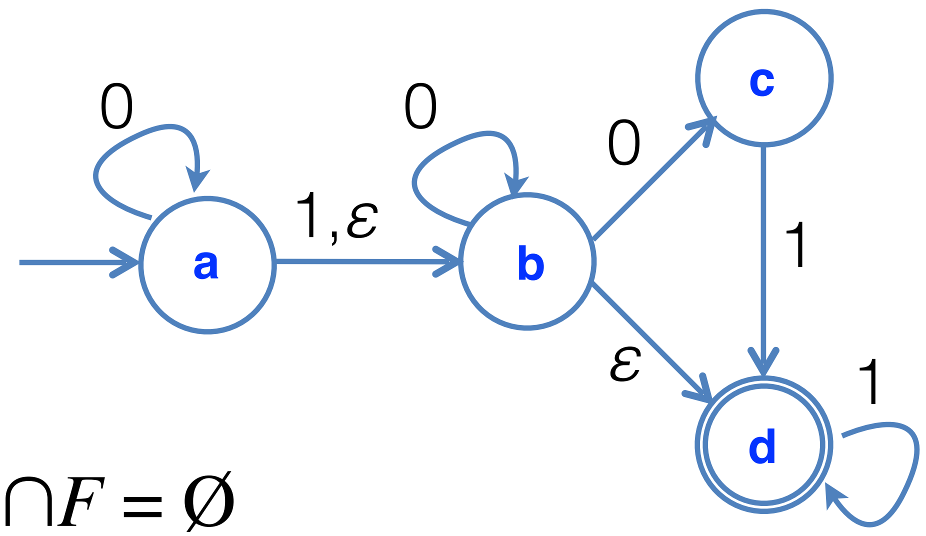
ε-Moves is Syntactic Sugar

Can modify any NFA N, to get an NFA N_{new} without ε-moves

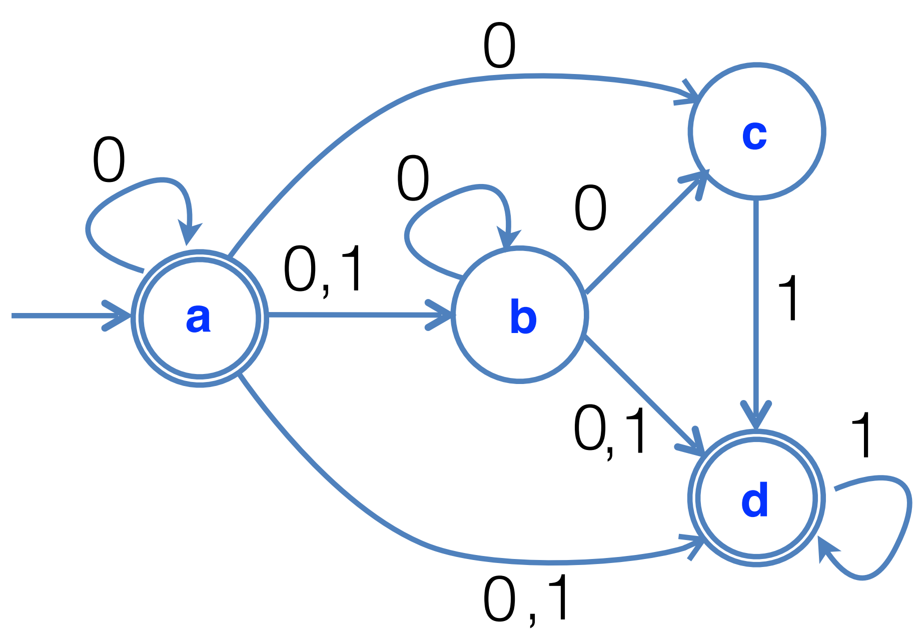
$$N_{\text{new}} = (\Sigma, Q, \delta_{\text{new}}, s, F_{\text{new}})$$

$$\delta_{\text{new}}(q, a) = C_\varepsilon(\, \delta(\, C_\varepsilon(\{q\}), a))$$

q	a	δ
a	0	**{ a, b, c, d }**
	1	**{ b, d }**
b	0	**{ b, c, d }**
	1	**{ d }**
c	0	**Ø**
	1	**{ d }**
d	0	**Ø**
	1	**{ d }**



$$F_{\text{new}} = \begin{cases} F, & \text{if } C_\varepsilon(\{s\}) \cap F = \emptyset \\ F \cup \{s\}, & \text{otherwise.} \end{cases}$$



NFA to DFA

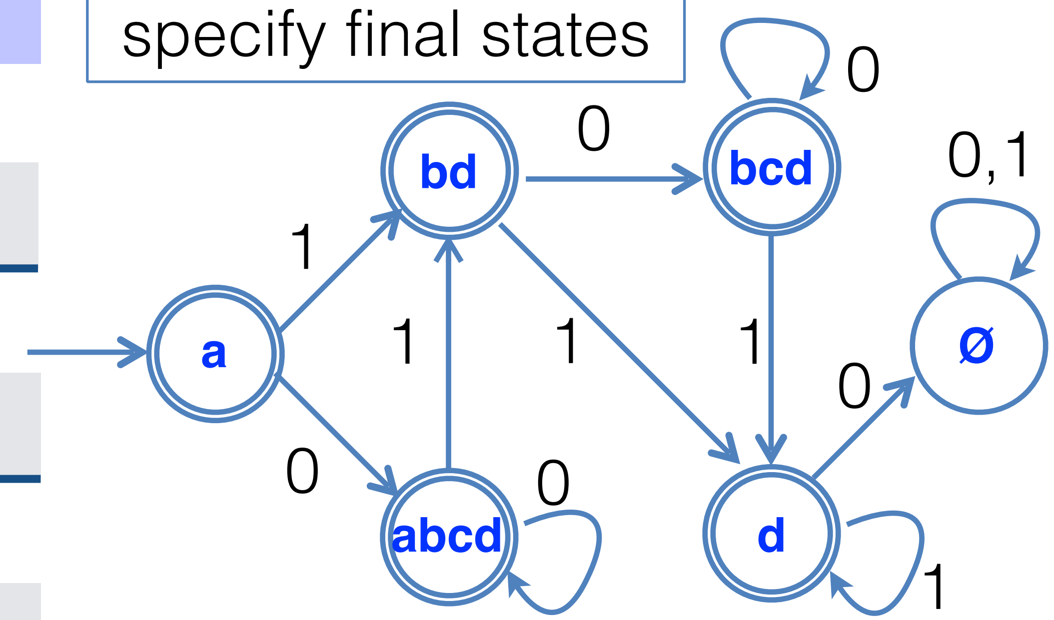
Can modify any NFA N, to get an equivalent DFA M

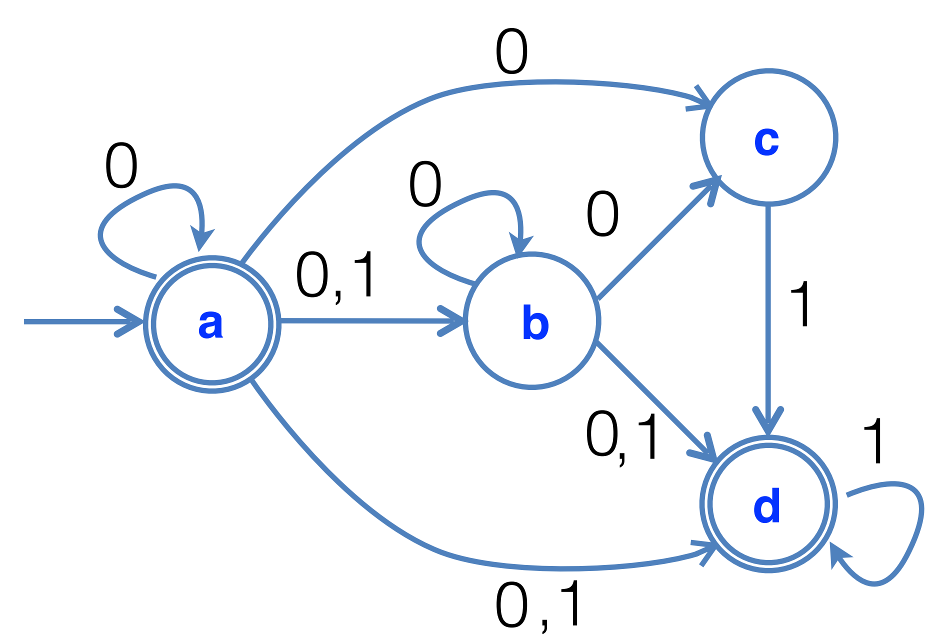
To avoid errors, first, remove ε-moves

q	a	δ
a	0	**{ a, b, c, d }**
	1	**{ b, d }**
b	0	**{ b, c, d }**
	1	**{ d }**
c	0	**Ø**
	1	**{ d }**
d	0	**Ø**
	1	**{ d }**

T	a	δ
{a}	0	**{ a,b,c,d }**
	1	**{ b,d }**
{a,b,c,d}	0	**{ a,b,c,d }**
	1	**{ b,d }**
{b,d}	0	**{ b,c,d }**
	1	**{ d }**
{b,c,d}	0	**{ b,c,d }**
	1	**{ d }**
{d}	0	**Ø**
	1	**{ d }**
Ø	0	**Ø**
	1	**Ø**

Remember to specify final states





NFA to DFA: Formally

NFA: $N = (\Sigma, Q, \delta, s, F)$

$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$$

ε-moves already removed

DFA: $M_N = (\Sigma, \mathcal{P}(Q), \delta^\dagger, s^\dagger, F^\dagger)$

$$\delta^\dagger : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$$

$$\delta^\dagger(T, a) = \cup_{q \in T} \ \delta(q, a)$$

$$s^\dagger = \{s\}, \quad F^\dagger = \{\ T \mid T \cap F \neq \emptyset\ \}$$

Theorem : $L(N) = L(M_N)$

Proof? Recall definitions of $L(\text{DFA})$, $L(\text{NFA})$

Language Accepted by a DFA

DFA: $M = (\Sigma, Q_M, \delta_M, s_M, F_M)$

Two ways to define

the state that an input w leads to starting from a state

$$q \overset{w}{\rightsquigarrow} p$$

if $w = a_1 \dots a_t$ and $\exists\ q_1, \dots, q_{t+1}$,

such that $q_1 = q,\ q_{t+1} = p,$ and

$\forall\ i \in [\ 1, t\ ],\ q_{i+1} = \delta_M(q_i, a_i)$

$$\boxed{\begin{aligned} \delta^*(q,\varepsilon) &= q \\ \delta^*(q,au) &= \delta^*(\delta_M(q,a), u) \end{aligned}}$$

Theorem : $q \overset{w}{\rightsquigarrow} p \Leftrightarrow p = \delta^*(q,w)$

Prove!

$$L(M) = \{\ w \mid \exists p \in F_M,\ s_M \overset{w}{\rightsquigarrow} p\ \} = \{\ w \mid \delta^*(s_M,w) \in F_M\ \}$$

Language Accepted by an NFA

NFA: $N = (\Sigma, Q_N, \delta_N, s_N, F_N)$

Two ways to define

the set of states that an input w leads to starting from a set of states

$$q \overset{w}{\rightsquigarrow} p$$

if $\exists\ a_1 \ldots a_t$ and $q_1, \ldots, q_{t+1}$, such that $w = a_1 \ldots a_t$, $q_1 = q$, $q_{t+1} = p$, and $\forall\ i \in [\,1, t\,]$, $q_{i+1} \in \delta_N(q_i, a_i)$

$$\delta^\dagger(T, a) = \cup_{q \in T}\ \delta_N(q, a)$$

$$\boxed{\begin{aligned} &\delta^{\dagger *}(T, \varepsilon) = T \\ &\delta^{\dagger *}(T, au) = \delta^{\dagger *}(\delta^\dagger(T, a), u) \end{aligned}}$$

$$s^\dagger = \{s_N\}, \quad F^\dagger = \{\, T \mid T \cap F_N \neq \emptyset \,\}$$

Theorem : $q \overset{w}{\rightsquigarrow} p \iff p \in \delta^{\dagger *}(\{q\}, w)$

Prove!

$$L(N) = \{\, w \mid \exists p \in F_N,\ s_N \overset{w}{\rightsquigarrow} p \} = \{\, w \mid \delta^{\dagger *}(\{s_N\}, w) \cap F_N \neq \emptyset \,\}$$

$$= \{\, w \mid \delta^{\dagger *}(s^\dagger, w) \in F^\dagger \,\}$$

Side-by-Side

DFA: $M = (\Sigma, Q_M, \delta_M, s_M, F_M)$

$$\delta_M : Q_M \times \Sigma \to Q_M$$

$$\boxed{\begin{aligned}\delta^*(q,\varepsilon) &= q \\ \delta^*(q,au) &= \delta^*(\delta_M(q,a), u)\end{aligned}}$$

$$L(M) = \{ w \mid \delta^*(s_M, w) \in F_M \}$$

NFA: $N = (\Sigma, Q_N, \delta_N, s_N, F_N)$

$$\delta_N : Q_N \times \Sigma \to \mathcal{P}(Q_N)$$

$$\delta^\dagger : \mathcal{P}(Q_N) \times \Sigma \to \mathcal{P}(Q_N)$$

$$\delta^\dagger(T, a) = \cup_{q \in T}\ \delta_N(q,a)$$

$$\boxed{\begin{aligned}\delta^{\dagger *}(T, \varepsilon) &= T \\ \delta^{\dagger *}(T, au) &= \delta^{\dagger *}(\delta^\dagger(T, a), u)\end{aligned}}$$

$$s^\dagger = \{s_N\}, \quad F^\dagger = \{ T \mid T \cap F_N \neq \emptyset \}$$

$$L(N) = \{ w \mid \delta^{\dagger *}(s^\dagger, w) \in F^\dagger \}$$

If $Q_M = \mathcal{P}(Q_N)$, $\delta_M = \delta^\dagger$, $s_M = s^\dagger$, $F_M = F^\dagger$, then $L(M) = L(N)$ ✓

Closure Properties for NFAs

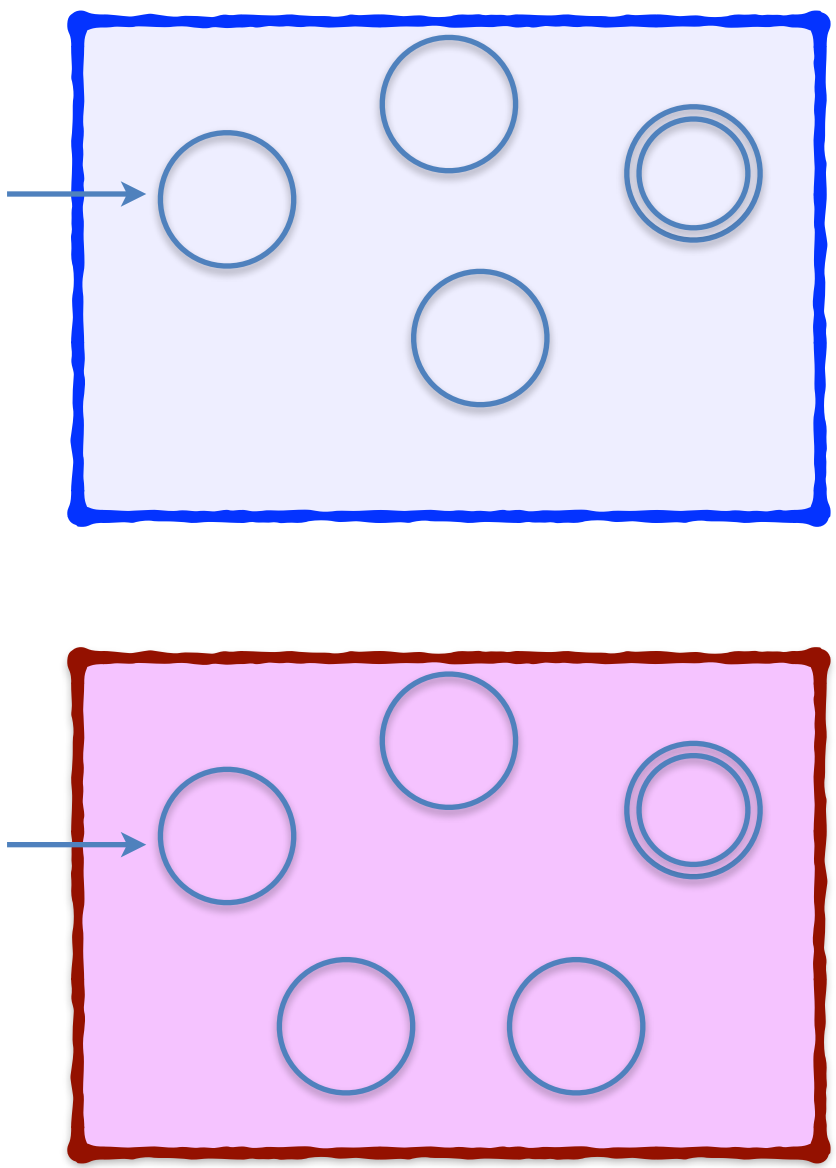
If L has an NFA, then **op**(L) has an NFA
where **op** can be **complement** or **Kleene star**

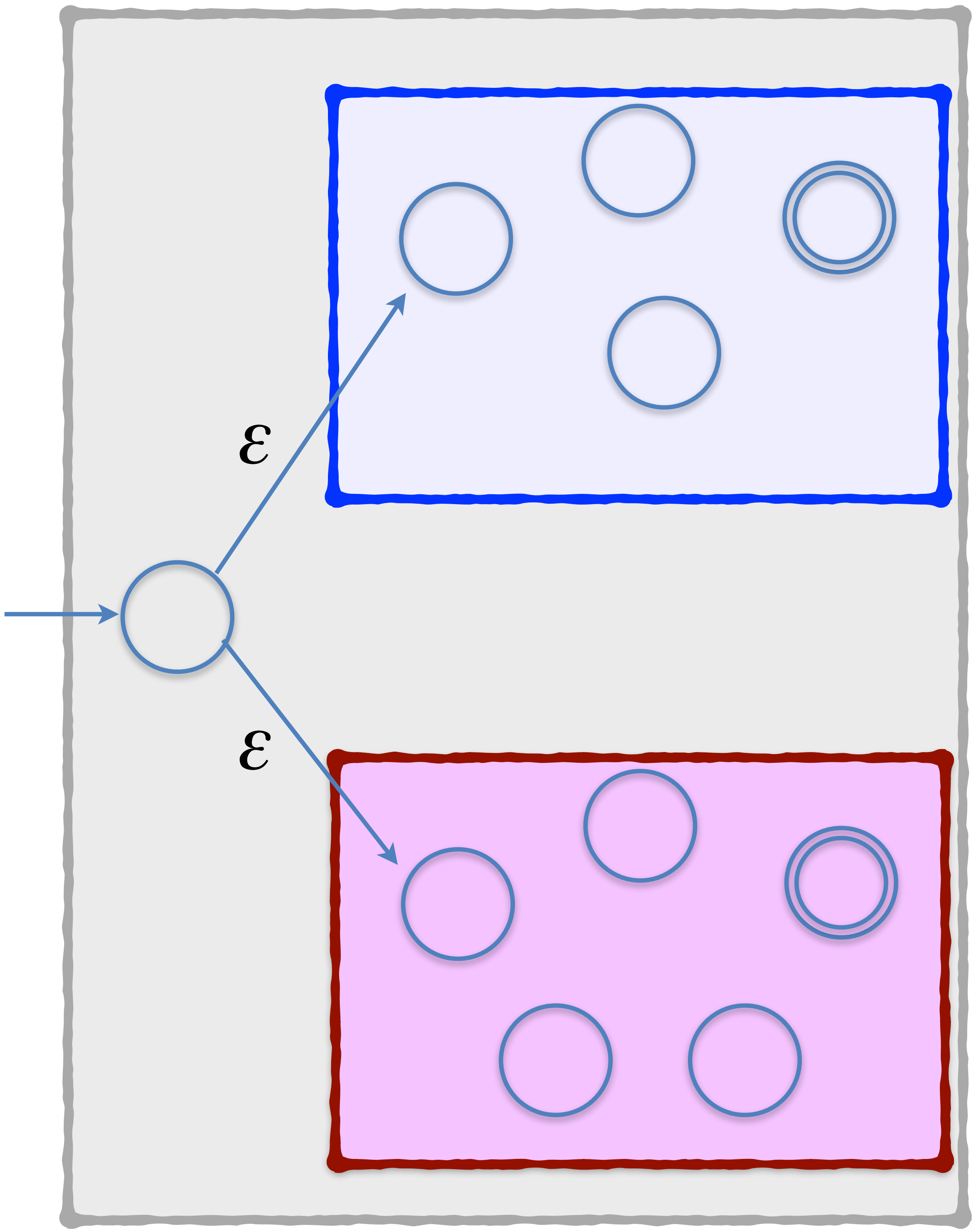
If L_1 and L_2 each has an NFA, then L_1 **op** L_2 has an NFA
where **op** can be a **binary set operation** (e.g., union, intersection, difference etc.) or **concatenation**

Complement and Binary set operations
Consider the equivalent DFA

Union can be seen directly too…

Closure Under Union





Closure Properties for NFAs

If L has an NFA, then **op**(L) has an NFA
where **op** can be **complement** or **Kleene star**

If L_1 and L_2 each has an NFA, then L_1 **op** L_2 has an NFA
where **op** can be a **binary set operation** (e.g., union, intersection, difference etc.) or **concatenation**

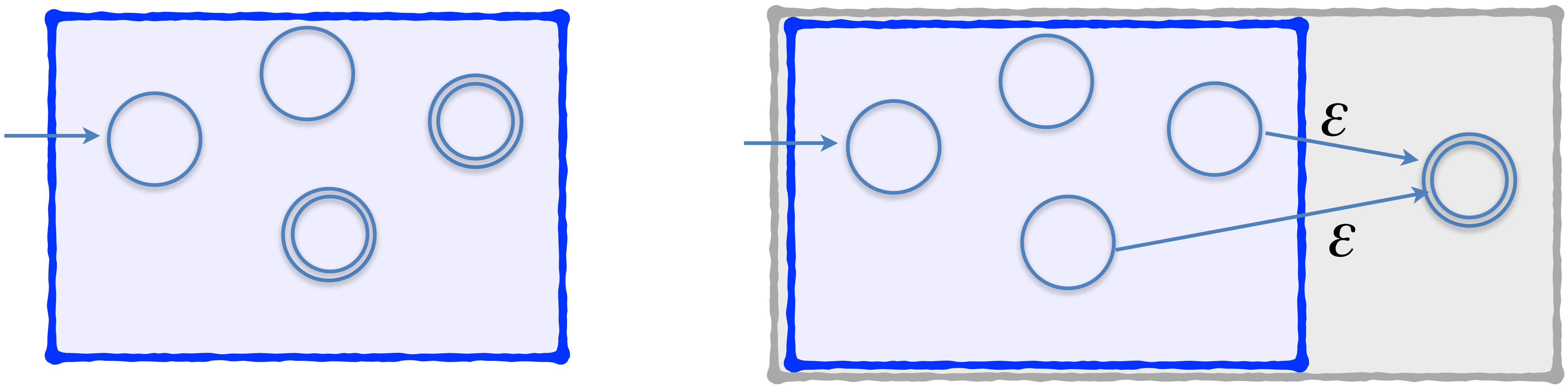
Complement and Binary set operations
Consider the equivalent DFA

(Union can be seen directly too…)

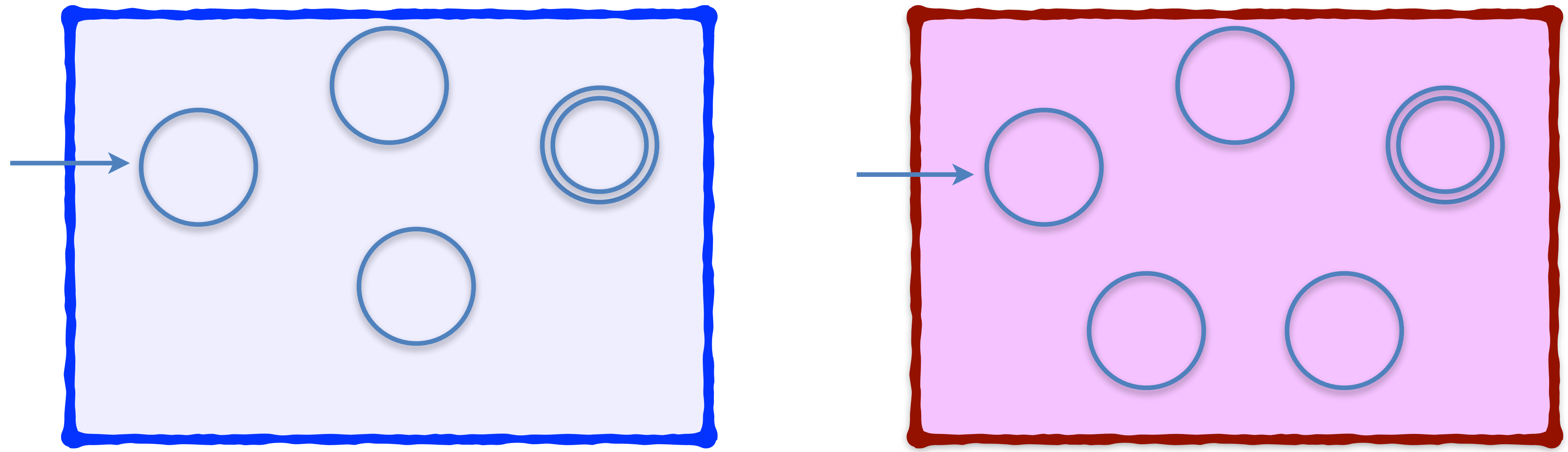
Now: **concatenation** and **Kleene star**

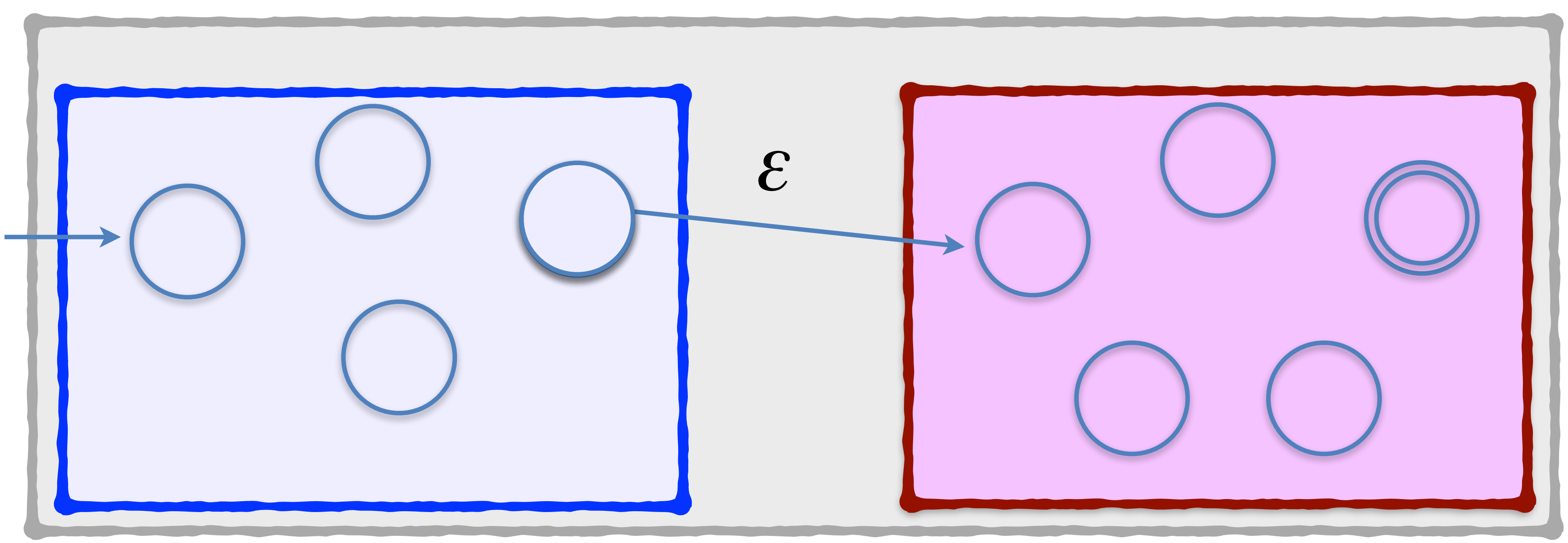
Single Final State Form

Can compile a given NFA so that there is
only one final state
(and there is no transition out of that state)

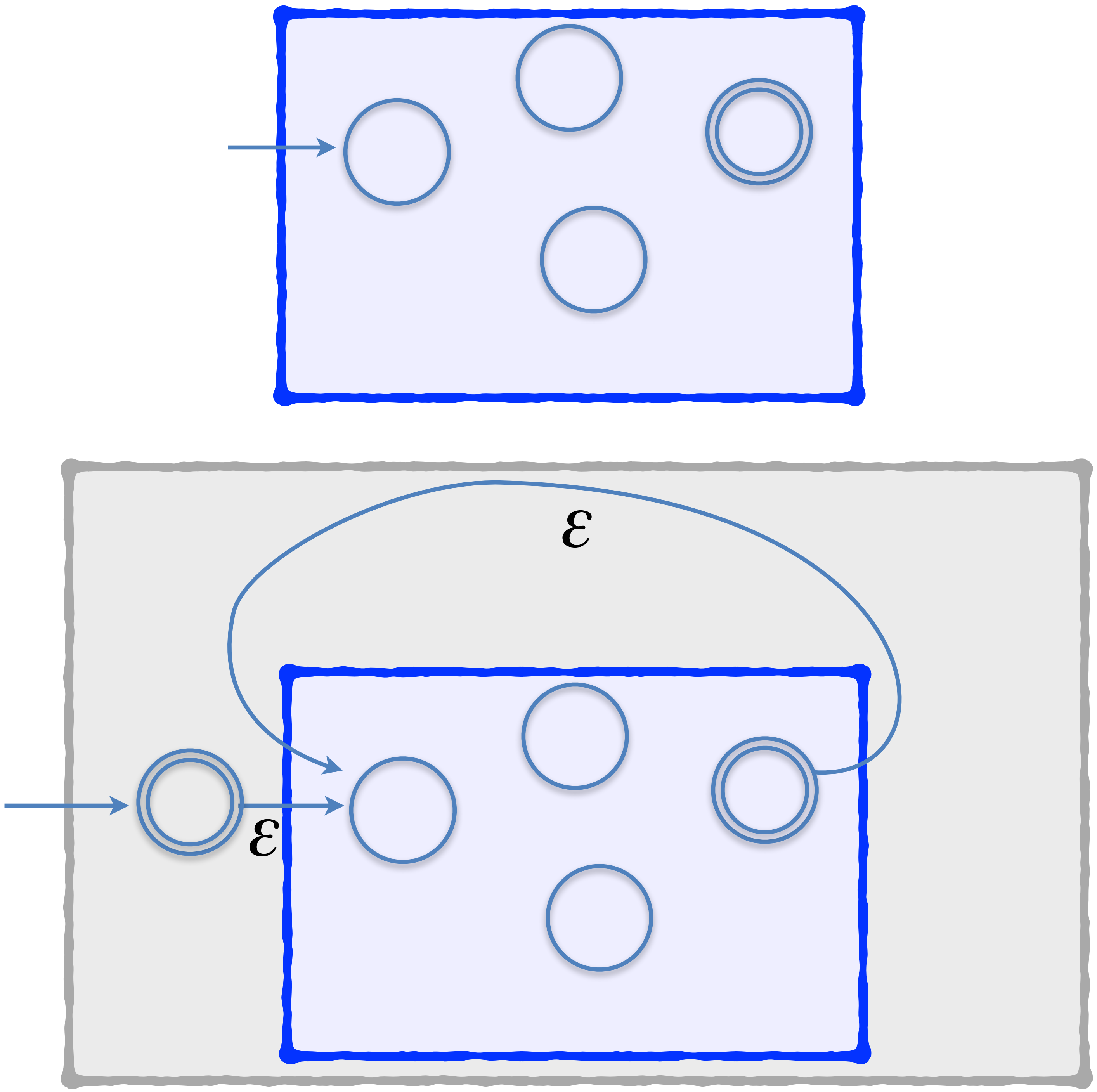


Closure Under Concatenation





Closure Under Kleene Star



NFAs & Regular Languages

Theorem : For any language L, the following are equivalent:

(a) L is accepted by an NFA
(b) L is accepted by a DFA
(c) L is regular

Saw : (a) $\Rightarrow$ (b)

Later : (b) $\Rightarrow$ (c)

Now : (c) $\Rightarrow$ (a)

Proof of (c) $\Rightarrow$ (a) : By induction on the least number of operators in a regular expression for the language

NFAs & Regular Languages

Theorem : L regular $\Rightarrow$ L is accepted by an NFA

Proof : To prove that if $L = L(r)$ for some regex r, then $L{=}L(N)$ for some NFA N. By induction on the number of operators in the regex.

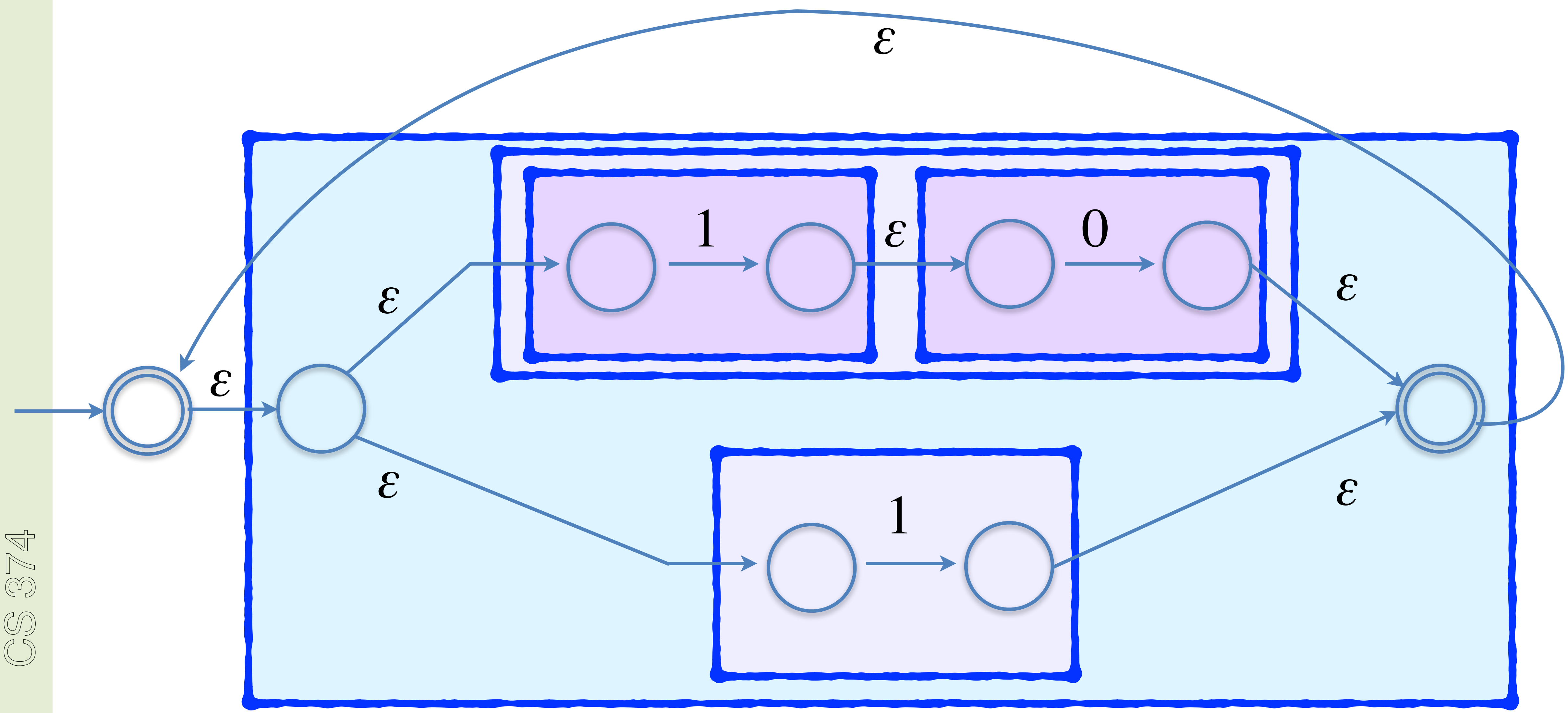
Base case: L has a regular expression with 0 operators. Then the regex should be one of $\emptyset$, ε, $a \in \Sigma$. In each case, $\exists N$ s.t. $L{=}L(N)$. ✓

Inductive step: Let $n > 0$. Assume that every language which has a regex with k operators has an NFA, where $0 \leq k < n$.

If L has a regex with n operators, it must be of the form r_1r_2, $r_1{+}r_2$, or $r_1{*}$, and hence $L = L_1L_2$, or $L_1 \cup L_2$ or $(L_1)^*$, where $L_1{=}L(r_1)$ and $L_2{=}L(r_2)$. Since r_1 and r_2 must have $< n$ operators, by IH L_1, L_2 have NFAs. *By closure of NFAs under these operations*, so does L. ✓

NFAs & Regular Languages

Example : L given by regular expression $(10+1)^*$



Closure Properties for Regular Languages

Theorem : If L_i are regular then, so is:

- $L_1 \cup L_2, \;\; L_1{}^*, \;\; L_1L_2$

  From the definition of regular languages (or from NFA closure properties)

- $\bar{L}_1$

  By considering DFAs for the languages and using the complement construction for DFAs

- $L_1 \cap L_2$

  By De Morgan's Law (or by the cross-product construction for DFAs)

- formula$(L_1, L_2, \ldots, L_k)$
- suffix(L_1)
- $h(L_1)$ and $h^{-1}(L_1)$, where h is a homomorphism

  Skipped from this course

- …

More Closure Properties

$$\text{formula}_f(L_1, \ldots, L_k) = \{ w \mid f(b_1, \ldots, b_k) \text{ holds, where } b_i \equiv (w \in L_i) \}$$

$$\text{e.g., } f(b_1, b_2, b_3) = \text{majority}(b_1, b_2, b_3)$$

Theorem: If $L_1, \ldots, L_k$ are regular, then for any boolean formula f, $\text{formula}_f(L_1, \ldots, L_k)$ is regular

Proof: Any boolean formula can be written using operators $\wedge$, $\vee$ and $\neg$ (AND, OR, NOT).

$$\text{formula}_{f \wedge g}(L_1, \ldots, L_k) = \text{formula}_f(L_1, \ldots, L_k) \cap \text{formula}_g(L_1, \ldots, L_k)$$
$$\text{formula}_{f \vee g}(L_1, \ldots, L_k) = \text{formula}_f(L_1, \ldots, L_k) \cup \text{formula}_g(L_1, \ldots, L_k)$$
$$\text{formula}_{\neg f}(L_1, \ldots, L_k) = \Sigma^* - \text{formula}_f(L_1, \ldots, L_k)$$

Complete the proof by induction on the number of operators in f.

More Closure Properties

suffix(L) = { w | w is a suffix of a string in L } = { w | $\exists\, x \in \Sigma^*\ \ xw \in L$ }

Theorem: If L is regular, then suffix(L) is regular

Proof: Let M be a DFA for L.
We shall construct an NFA N s.t. $L(N) = \text{suffix}(L(M))$.

Idea: N will guess the state that M will be in after seeing a "correct" x and directly jump to that state. Then starts behaving like M.

Need to ensure that (some thread of) N accepts w iff $w \in \text{suffix}(L)$.

If $w \in \text{suffix}(L)$, $\exists x,\ xw \in L$. Hence $\exists q$ s.t. $s \overset{x}{\rightsquigarrow}_M q$ and $q \overset{w}{\rightsquigarrow}_M p,\ p \in F$.
So some thread of N will jump to q ($s \overset{\varepsilon}{\rightsquigarrow}_N q$) and accept w ($q \overset{w}{\rightsquigarrow}_N p$).

Converse? Trouble if N jumps to q and accepts w from there, but no x could take M to q (i.e., q unreachable)!

More Closure Properties

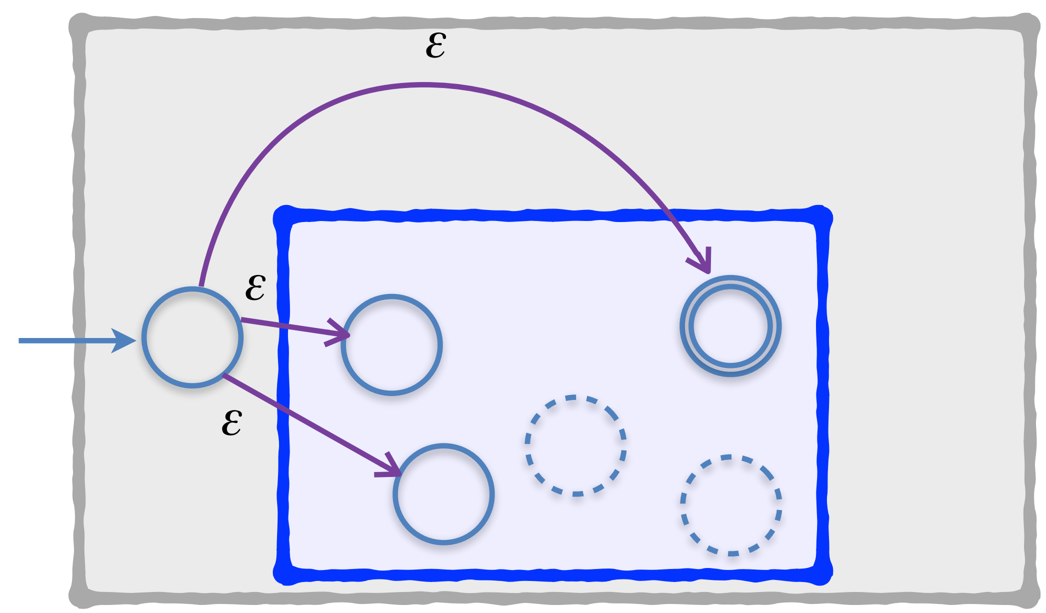
$\text{suffix}(L) = \{\, w \mid w$ is a suffix of a string in $L\,\} = \{\, w \mid \exists\, x \in \Sigma^* \quad xw \in L\,\}$

Theorem: If L is regular, then suffix(L) is regular

Proof: Let M be a DFA for L.
We shall construct an NFA N s.t. $L(N) = \text{suffix}(L(M))$.

Idea: N will guess the state that M will be in after seeing a "correct" x and directly jump to that state. Then starts behaving like M.

$Q_N = Q_M \cup \{s_N\}$. $F_N = F_M$.
$\delta_N(q,a) = \{\delta_M(q,a)\}$ for $q \in Q_M$.
$\delta_N(s_N,\varepsilon) = \{q \in Q_M \mid q$ reachable from $s_M\}$

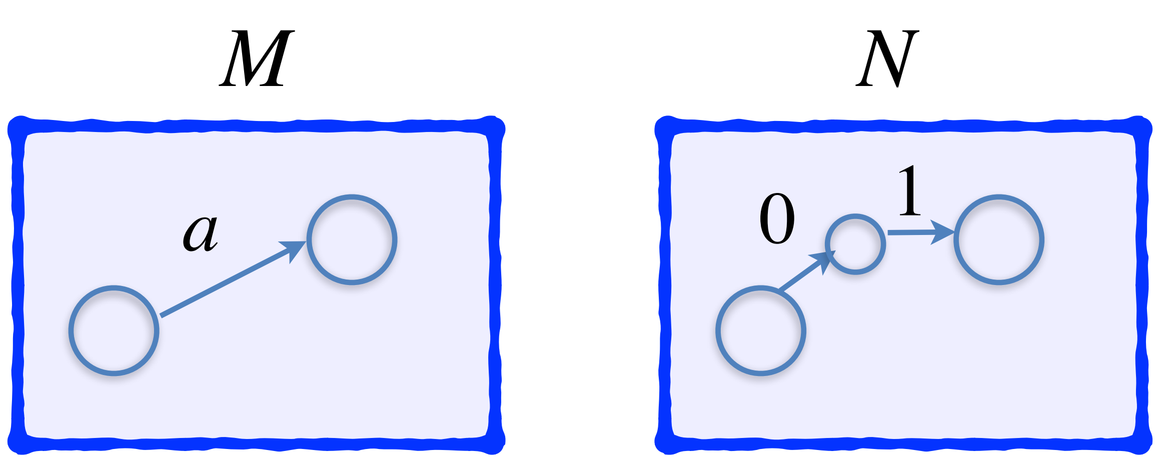


Exercise: Verify "corner cases": e.g., $L = \emptyset$, $\varepsilon \notin L$ etc.

More Closure Properties (FYI): Homomorphism/Inverse Homomorphism

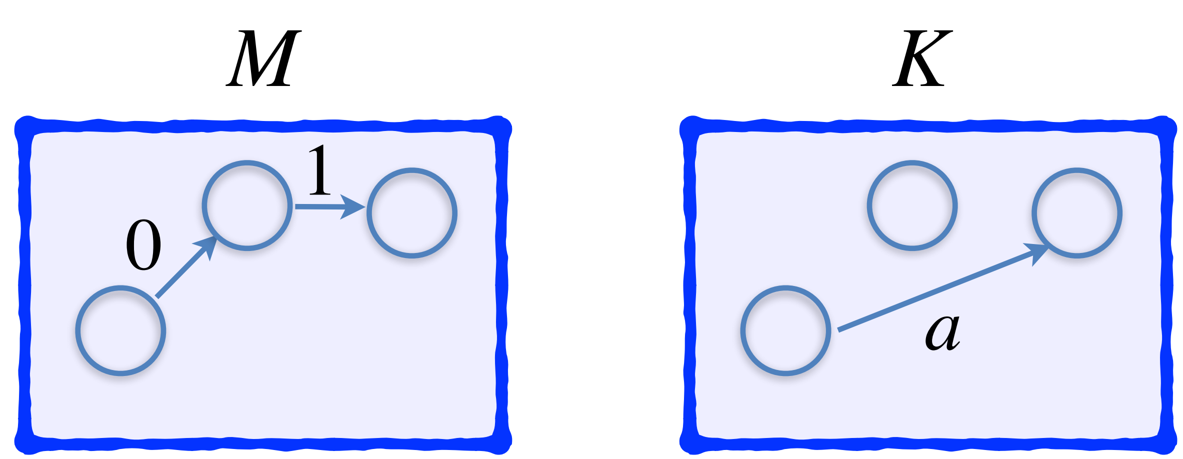
Suppose given a mapping $h : \Sigma \rightarrow \Delta^*$.

Given DFA M over Σ, consider NFA N over Δ (with additional states) s.t. for any two of the original states, p,q, if $p \xrightarrow{a}_M q$ then $p \overset{h(a)}{\rightsquigarrow}_N q$ via a path of new states



$L(N) = h(L(M))$

Given DFA M over Δ, consider DFA K over Σ and the same set of states, s.t. $p \xrightarrow{a}_K q$ iff $p \overset{h(a)}{\rightsquigarrow}_M q$



$L(K) = h^{-1}(L(M))$

e.g., for $h(a) = 01$