CS 302.1 - Automata Theory

Shantanav Chakraborty

Center for Quantum Science and Technology (CQST)

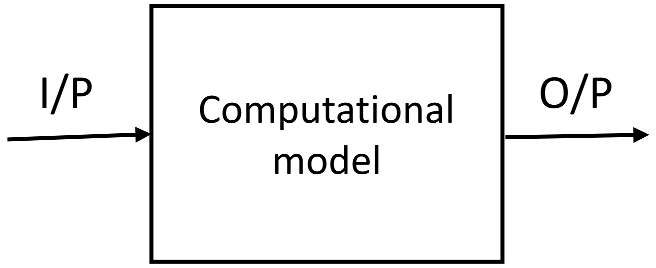
Center for Security, Theory and Algorithms (CSTAR)

IIIT Hyderabad

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY HYDERABAD

A quick recap

- Can a given problem be computed by a particular computational model?

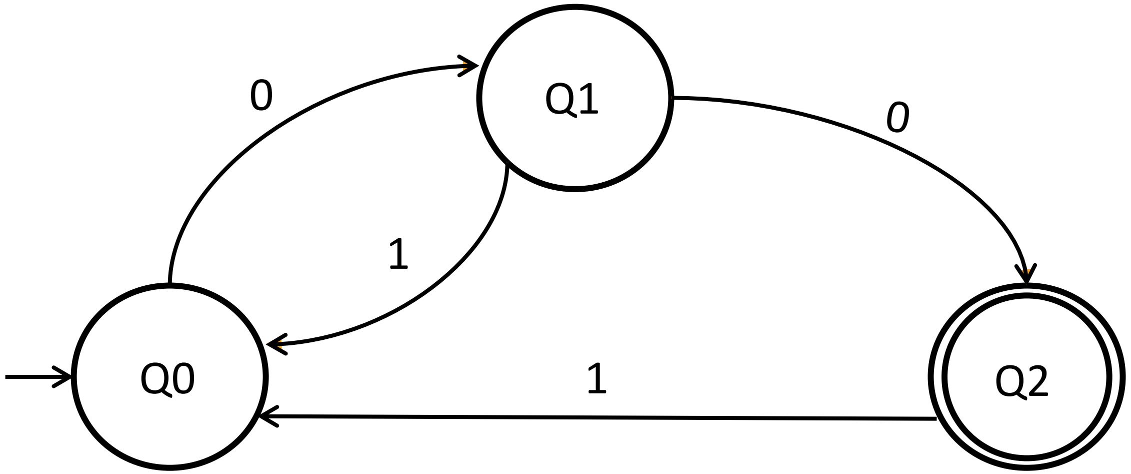


A computational model solves a problem P if,

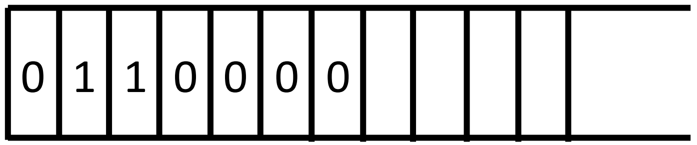
(i) For all inputs belonging to the YES instance of P, the device outputs **YES**

(ii) For all inputs belonging to the NO instance of P, the device outputs **NO**.

If (i) and (ii) hold, we say that the problem **P is computable** by this computational model.



Deterministic Finite Automata (DFA)



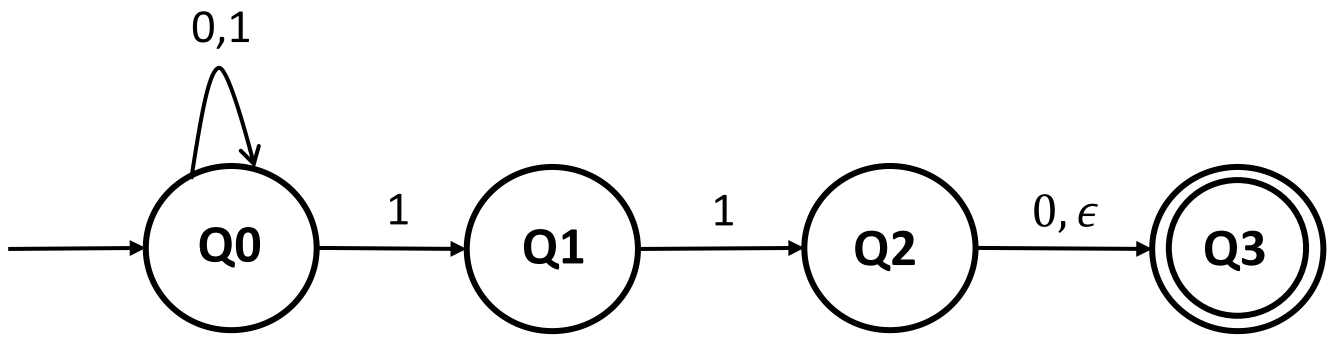
Run:

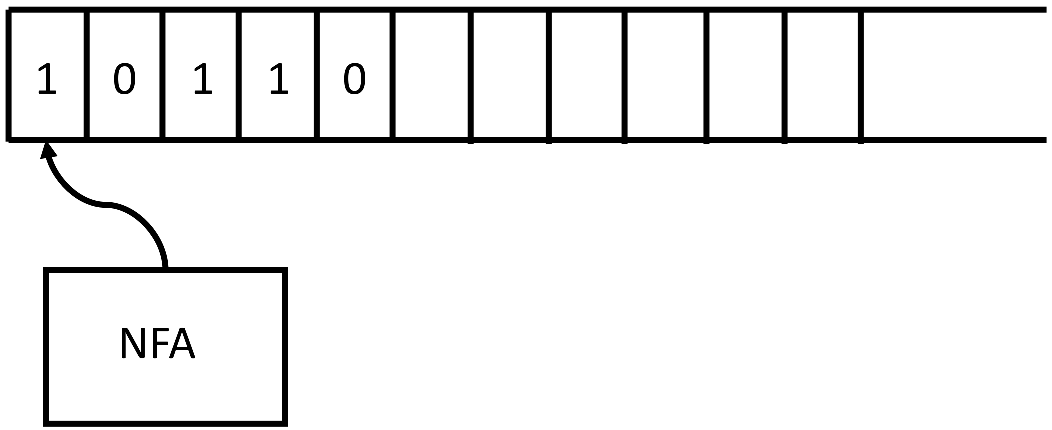
$$\boldsymbol{Q0} \xrightarrow{0} Q1 \xrightarrow{1} Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q1 \xrightarrow{0} Q2 \xrightarrow{0} Q2 \xrightarrow{0} \boldsymbol{Q2}$$

L(M) = {$\omega | \omega$ results in an accepting run}

A quick recap

Non deterministic Finite Automata (NFA)





Run 1: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0$ (**REJECT**)
Run 2: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q1 \xrightarrow{1} Q2 \xrightarrow{0} Q3$ (**ACCEPT**)
Run 3: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q0 \xrightarrow{1} Q1 \xrightarrow{0}$ CRASH **(REJECT)**
Run 4: $Q0 \xrightarrow{1} Q0 \xrightarrow{0} Q0 \xrightarrow{1} Q1 \xrightarrow{1} Q2 \xrightarrow{\epsilon} Q3 \xrightarrow{0}$ CRASH **(REJECT)**

- Multiple runs per input possible.
- The NFA "accepts" an input string, if there exists at **least one accepting *run***

	0	**1**	ϵ
Q0	**Q0**	**Q0, Q1**	
Q1		**Q2**	
Q2	**Q3**		**Q3**
Q3			

L(M) = {$\omega | \omega$ results in an accepting run}

NFA vs DFA

- Are NFAs more powerful than DFAs? Intuitively, non-determinism seems to be adding more "power".

- Let L_1 be the language accepted NFAs and L_2 be the language accepted by DFAs

- Is $L_2 \subseteq L_1$? Clearly true, because a DFA is just a special case of an NFA.

- Surprisingly, what we will show next is that $\boldsymbol{L_1 \subseteq L_2}$!

- That is, **given an NFA, we can convert it to a DFA that accepts the same language**.

- Such a DFA is called a "**Remembering DFA**".

Thus, DFAs and NFAs are completely equivalent and $\boldsymbol{L_1 = L_2}$!

Converting an NFA to a DFA

Intuitive idea for the construction of a Remembering DFA from an NFA:

- Let R be the Remembering DFA corresponding to an NFA N.
- R on an input enters a state that is labelled by all possible states that N can enter on that input.
- Note that this "trims away" the non-determinism of the NFA N without "losing" the language it accepts.
- Also note that if N has k states, then R has at most 2^k states. Why?

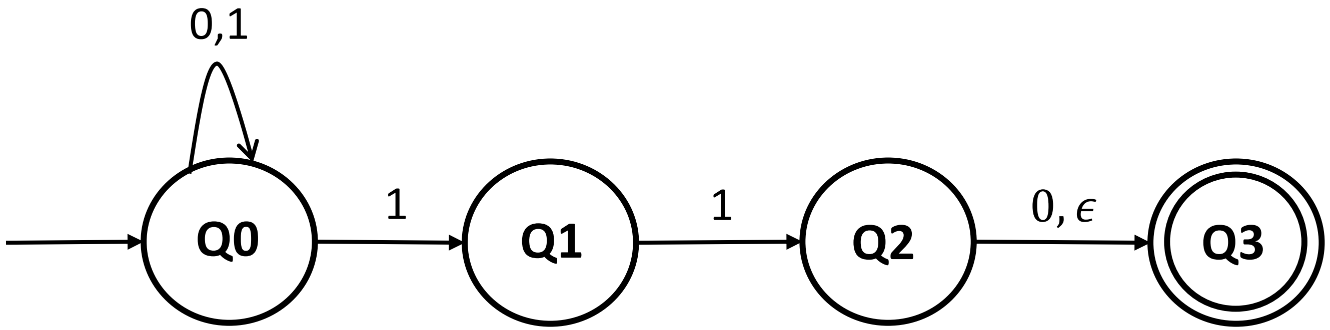
Converting an NFA to a DFA

Intuitive idea for the construction of a Remembering DFA from an NFA:

- Let R be the Remembering DFA corresponding to an NFA N.
- R on an input enters a state that is labelled by all possible states that N can enter on that input.
- Note that this “trims away” the non-determinism of the NFA N without “losing” the language it accepts.
- Also note that if N has k states, then R has at most 2^k states. Why?
- Any label in the Remembering DFA is a subset of $\{Q_0,\ Q_1, Q_2, \dots \dots, Q_{k-1}\}$, where Q_i = State of the NFA.
- There are at most 2^k labels for the DFA.

Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.

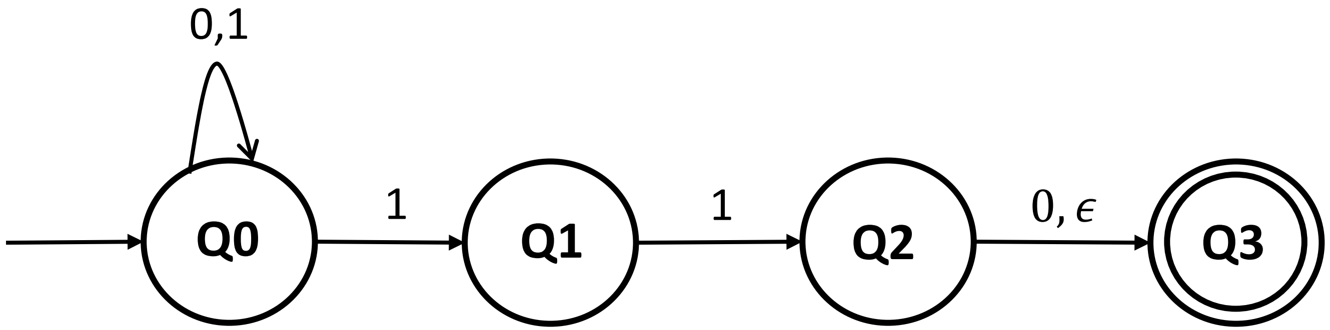


NFA N

	0	1	ϵ
Q0	**Q0**	**Q0, Q1**	
Q1		**Q2**	
Q2	**Q3**		**Q3**
Q3			

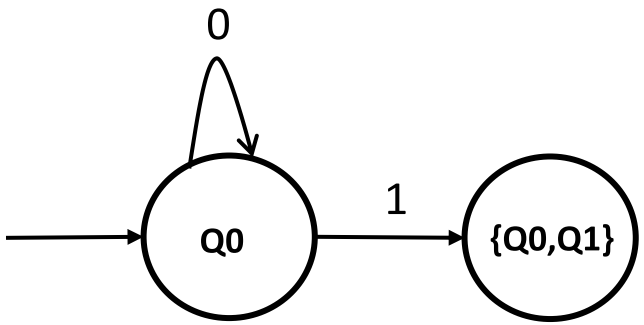
Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.



NFA N

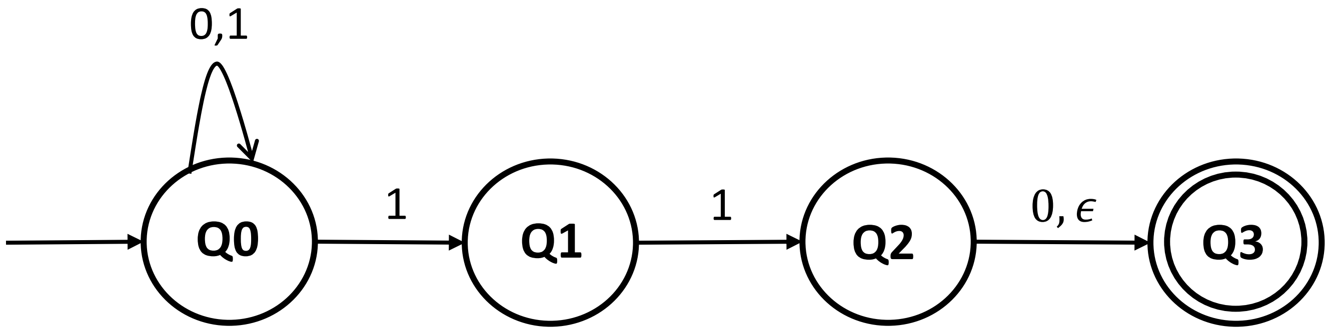
	0	1	ϵ
Q0	**Q0**	**Q0, Q1**	
Q1		**Q2**	
Q2	**Q3**		**Q3**
Q3			



Remembering DFA R

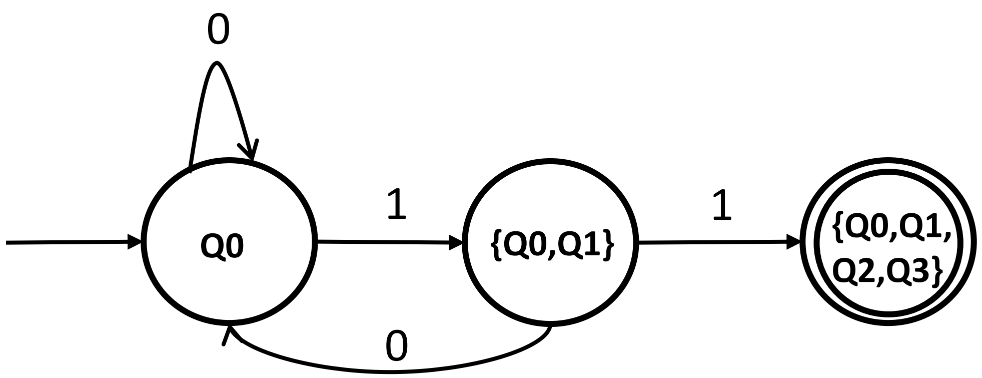
Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.



NFA N

	0	**1**	ϵ
Q0	**Q0**	**Q0, Q1**	
Q1		**Q2**	
Q2	**Q3**		**Q3**
Q3			

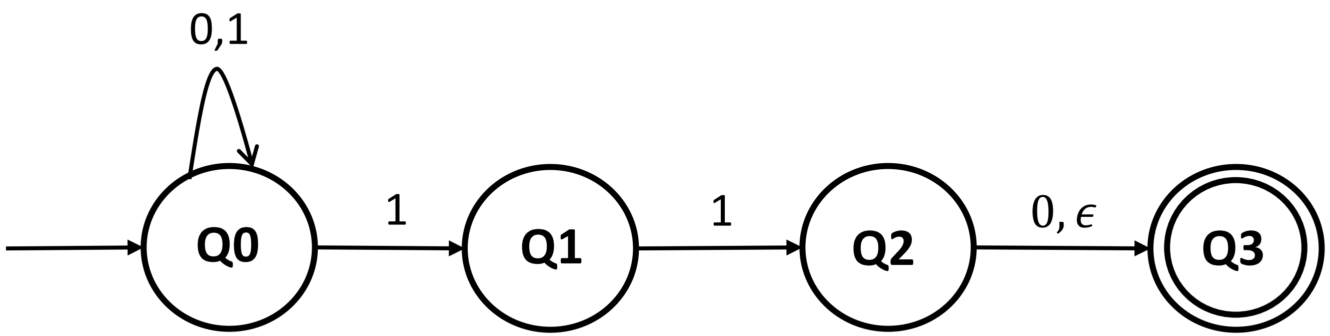


Remembering DFA R

Any state of R that contains in its label, an accepting state of N is an accepting state of R.

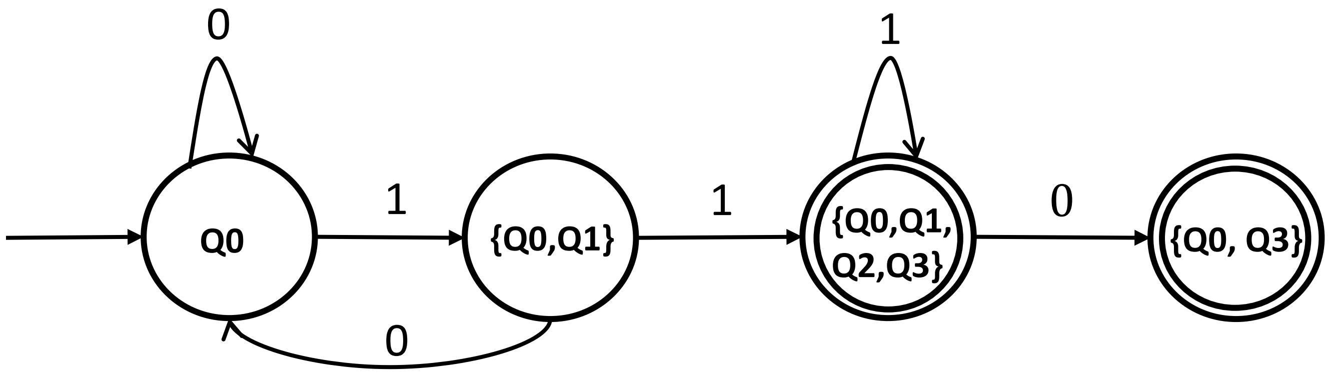
Converting an NFA to a DFA

- R on an input enters a state that is labelled by all possible states that N can enter on that input.



NFA N

	0	1	ϵ
Q0	**Q0**	**Q0, Q1**	
Q1		**Q2**	
Q2	**Q3**		**Q3**
Q3			

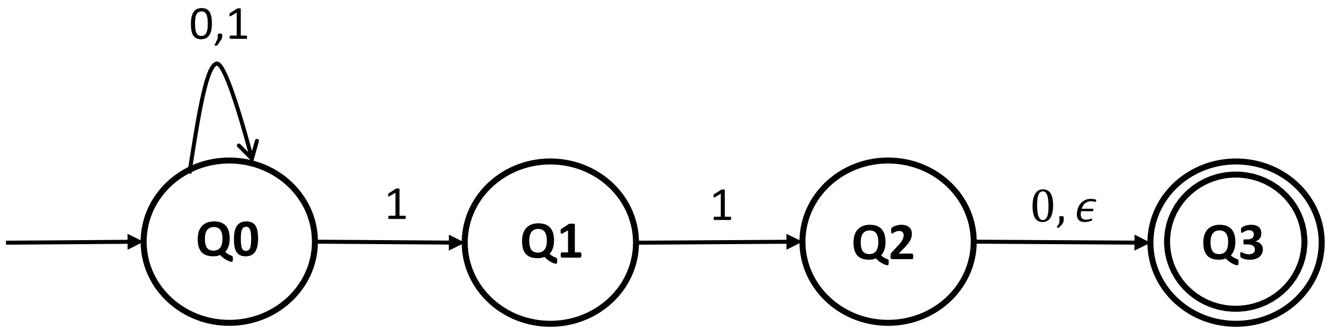


Remembering DFA R

Any state of R that contains in its label, an accepting state of N is an accepting state of R.

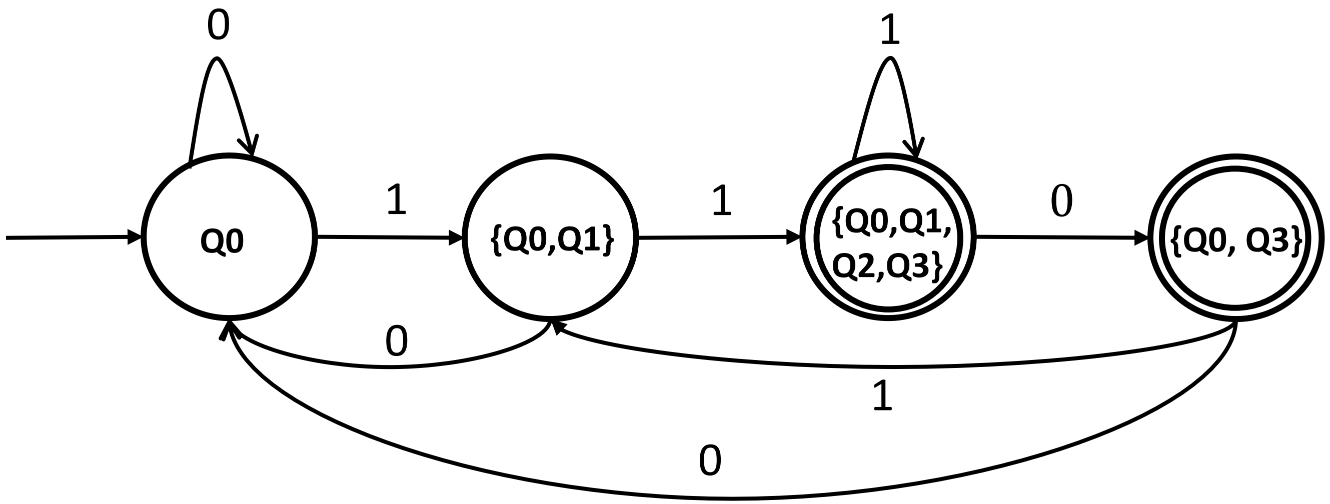
Converting an NFA to a DFA

- M_2 on an input enters a state that is labelled by all possible states that M_1 can enter on that input.



NFA N

	0	1	ϵ
Q0	**Q0**	**Q0, Q1**	
Q1		**Q2**	
Q2	**Q3**		**Q3**
Q3			

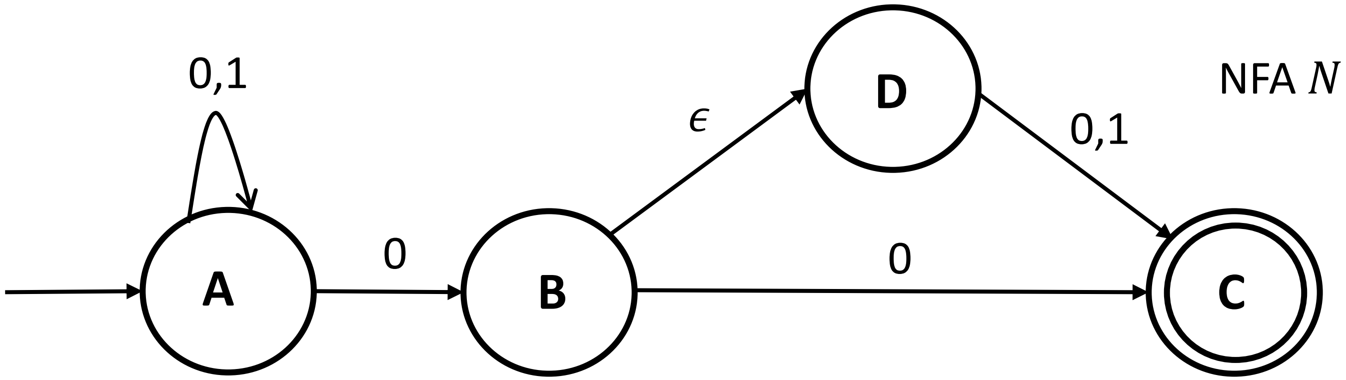


Remembering DFA R

Any state of R that contains in its label, an accepting state of N is an accepting state of R.

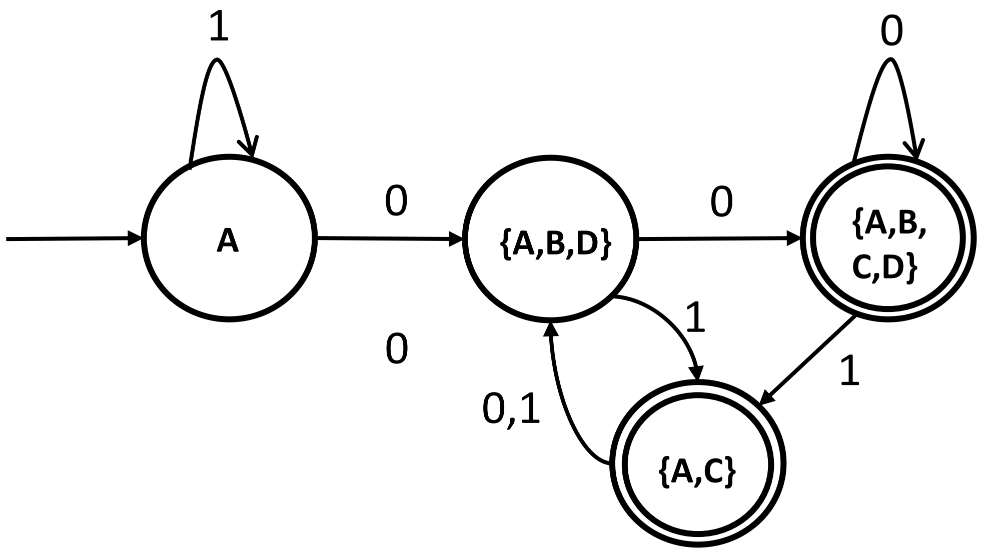
Converting an NFA to a DFA

- M_2 on an input enters a state that is labelled by all possible states that M_1 can enter on that input.



NFA N

	0	1	ϵ
A	A	A	
B	C		D
C			
D	C	C	



Remembering DFA R

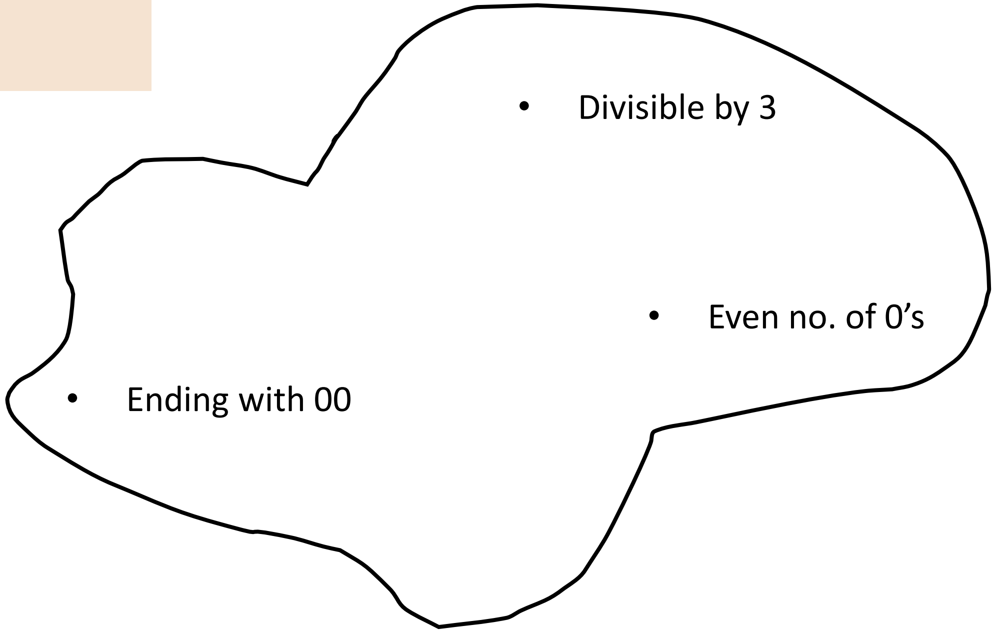
Regular Languages

A language is called a **Regular Language** if there exists some finite automata recognizing it.

If M be a finite automaton (DFA/NFA) and,

$$L(M) = \{\omega | \omega \text{ is accepted by } M\}$$

$\boldsymbol{L(M)}$ **is regular.**



Set of all regular Languages

Regular Languages

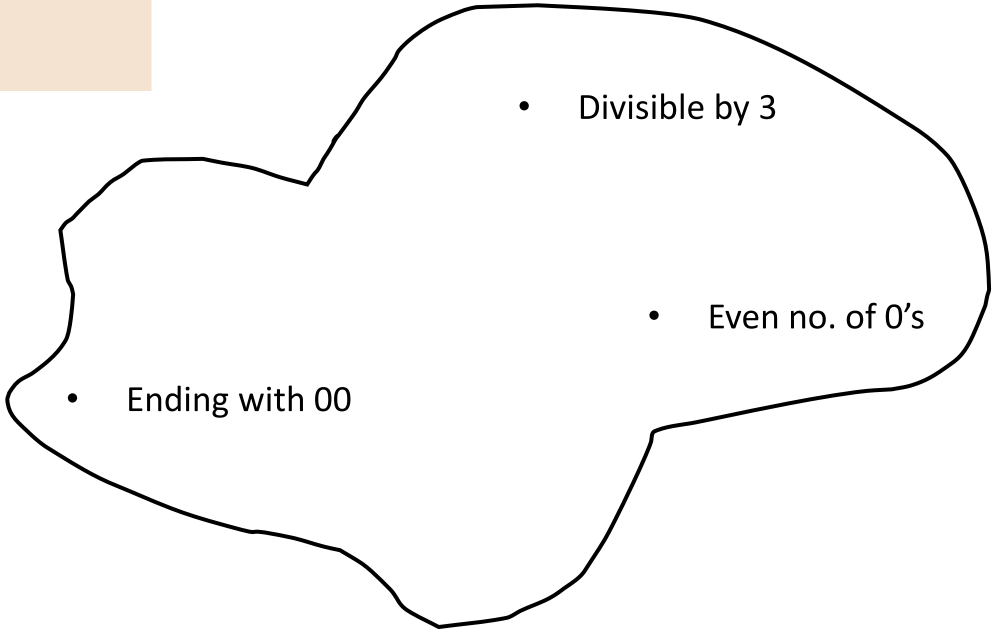
A language is called a **Regular Language** if there exists some finite automata recognizing it.

If M be a finite automaton (DFA/NFA) and,

$$L(M) = \{\omega | \omega \text{ is accepted by } M\}$$

$\boldsymbol{L(M)}$ **is regular.**

- Any language has associated with it, a set of operations that can be performed on it.
- These operations help us to understand the properties of that language, e.g. closure properties
- For regular languages, this will help us prove that certain languages are non-regular and hence we cannot hope to design a finite automaton for them



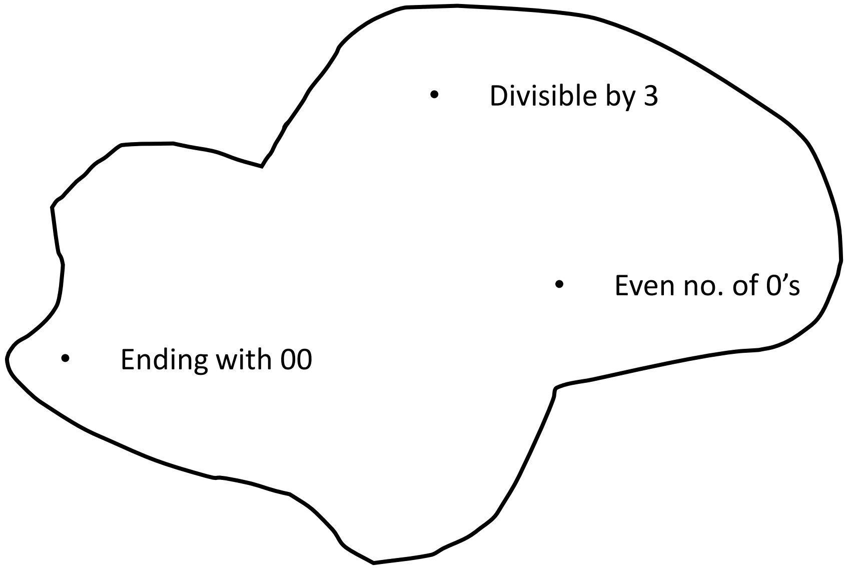
Set of all regular Languages

Regular Languages

Regular Operations:

Let L_1 and L_2 be languages. The following are the *regular operations*:

- **Union:** $L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$
- **Concatenation:** $L_1 . L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$
- **Star:** $L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0 \text{ and each } x_i \in L_1\}$



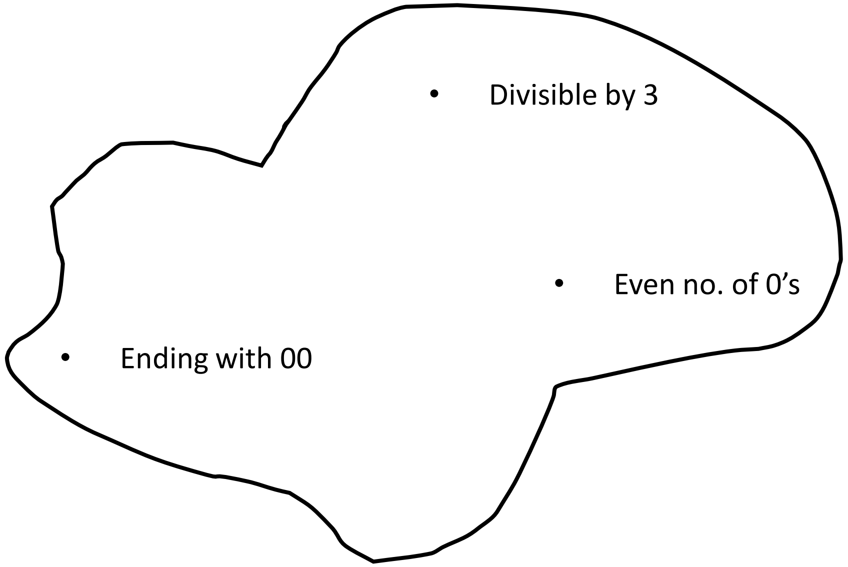
Set of all regular Languages

Regular Languages

Regular Operations:

Let L_1 and L_2 be languages. The following are the *regular operations*:

- **Union:** $L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$
- **Concatenation:** $L_1 . L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$
- **Star:** $L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0 \text{ and each } x_i \in L_1\}$



Set of all regular Languages

Star operation: It is an unary operation (unlike the other two) and involves putting together *any number of strings in* L_1 *together to obtain a new string.*

Note: Any number of strings includes "0" as a possibility and so the empty string ϵ is a member of L_1^*.

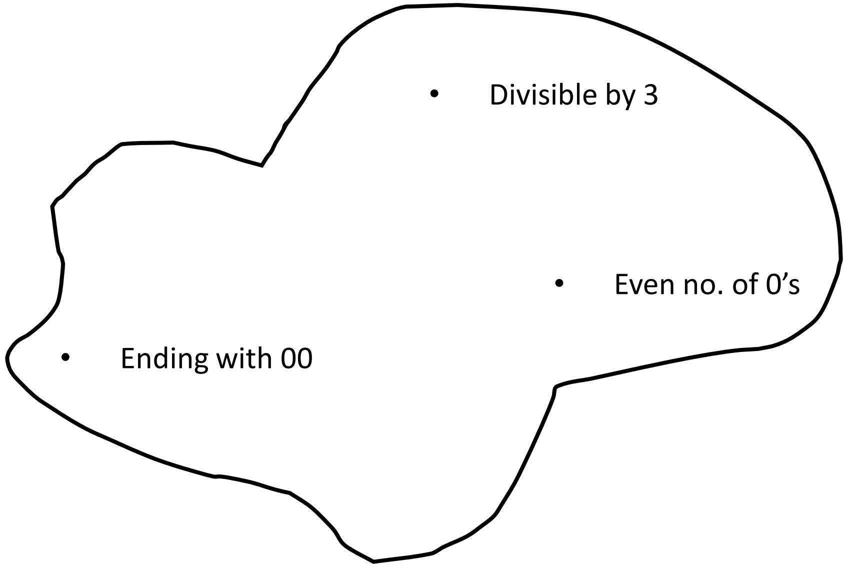
If $\Sigma = \{a\}$, $\Sigma^* = \{\epsilon, a, aa, aaa, \ldots \ldots\}$; If $\Sigma = \{\Phi\}, \Sigma^* = \{\}$

Regular Languages

Regular Operations:

Let L_1 and L_2 be languages. The following are the *regular operations*:

- **Union:** $L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$
- **Concatenation:** $L_1 . L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$
- **Star:** $L_1^* = \{x_1 x_2 \cdots x_k | \, k \geq 0 \text{ and each } x_i \in L_1\}$



Set of all regular Languages

Star operation: It is an unary operation (unlike the other two) and involves putting together *any number of strings in L_1 together to obtain a new string.*

Note: Any number of strings includes "0" as a possibility and so the empty string ϵ is a member of L_1^***.**

If $\Sigma = \{0,1\}$, we have that $\Sigma^* = \{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots \ldots\}$

Regular Languages

Regular Operations: Let L_1 and L_2 be languages.

- **Union:** $L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$
- **Concatenation:** $L_1 . L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$
- **Star:** $L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0 \text{ and each } x_i \in L_1\}$

Example: Let the alphabet $\Sigma = \{a, b, \cdots, z\}$. If $L_1 = \{$*social, economic*$\}$ and $L_2 = \{$*justice, reform*$\}$, then

- $L_1 \cup L_2 = \{$*social, economic, justice, reform*$\}$

Regular Languages

Regular Operations: Let L_1 and L_2 be languages.

- **Union:** $L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$
- **Concatenation:** $L_1 . L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$
- **Star:** $L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0 \text{ and each } x_i \in L\}$

Example: Let the alphabet $\Sigma = \{a, b, \cdots, z\}$. If $L_1 = \{\textit{social, economic}\}$ and $L_2 = \{\textit{justice, reform}\}$, then

- $L_1 \cup L_2 = \{\textit{social, economic, justice, reform}\}$
- $L_1 . L_2 = \{\textit{socialjustice, socialreform, economicjustice, economicreform}\}$

Regular Languages

Regular Operations: Let L_1 and L_2 be languages.

- **Union:** $L_1 \cup L_2 = \{x | x \in L_1$ or $x \in L_2\}$
- **Concatenation:** $L_1 . L_2 = \{xy | x \in L_1$ and $y \in L_2\}$
- **Star:** $L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0$ and each $x_i \in L\}$

Example: Let the alphabet $\Sigma = \{a, b, \cdots, z\}$. If $L_1 = \{$*social, economic*$\}$ and $L_2 = \{$*justice, reform*$\}$, then

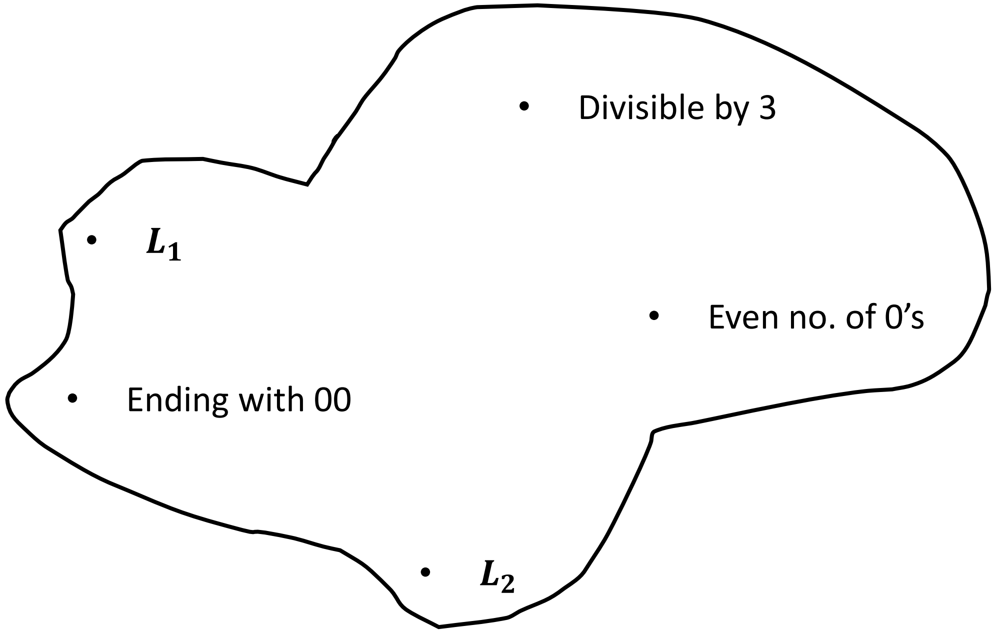
- $L_1 \cup L_2 = \{$*social, economic, justice, reform*$\}$
- $L_1 . L_2 = \{$*socialjustice, socialreform, economicjustice, economicreform*$\}$
- $L_1^* = \{\epsilon$, *social, economic, socialsocial, socialeconomic, economicsocial, economiceconomic, socialsocialsocial, socialsocialeconomic, socialeconomiceconomic,……*$\}$
- $L_2^* = \{\epsilon$, *justice, reform, justicejustice, justicereform, reformjustice, reformreform, justicejusticejustice, ……*$\}$

Closure of Regular Languages

We want to check whether the set of regular languages are **closed** under some operations.

What does this mean?

- We pick up points within the set of all regular languages (say L_1 and L_2)
- Perform *set operations* such as Union, concatenation, Star, intersection, reversal, compliment etc on them.
- Observe whether the resulting language still belongs to the set of all regular languages.
- If so, we say, regular languages are **closed** under that operation.



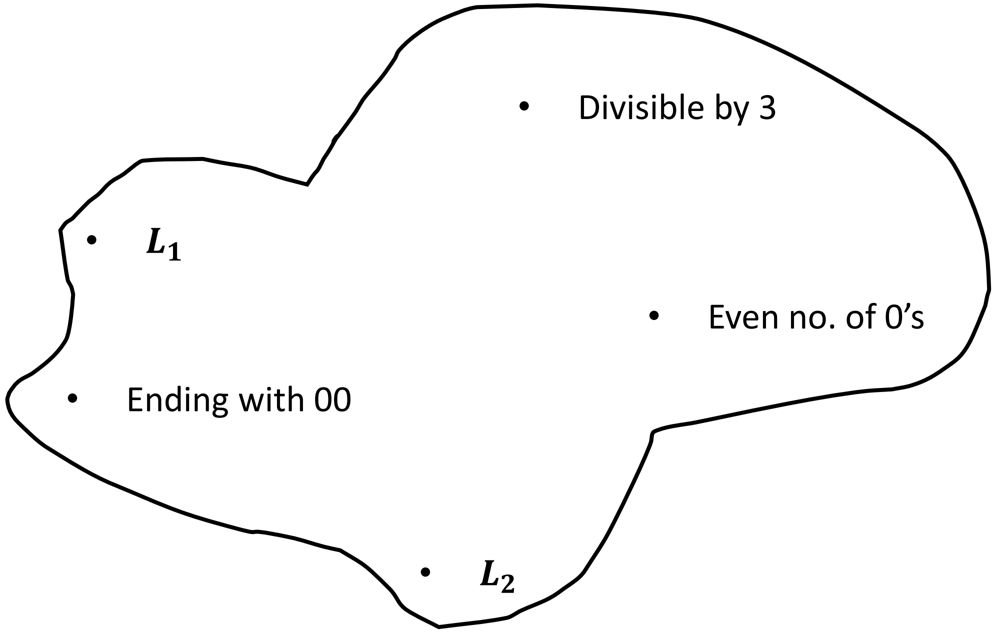
Set of all regular Languages

Closure of Regular Languages

We want to check whether the set of regular languages are **closed** under some operations.

What does this mean?

- We pick up points within the set of all regular languages (say L_1 and L_2)
- Perform *set operations* such as Union, concatenation, Star, intersection, reversal, compliment etc on them.
- Observe whether the resulting language still belongs to the set of all regular languages.
- If so, we say, regular languages are **closed** under that operation.



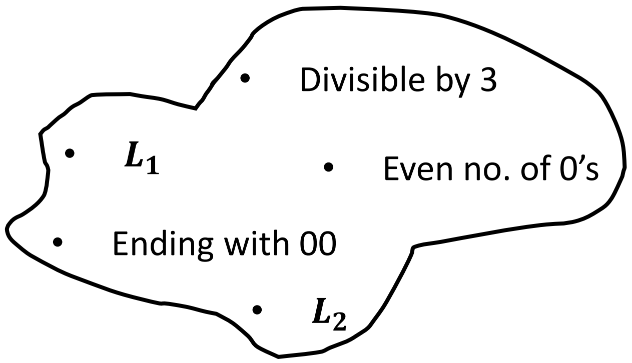
Set of all regular Languages

For example, the **natural numbers are closed under addition/multiplication** and **not under subtraction/division**.

Closure of Regular Languages

Q: Is the set of all regular languages **closed under union**?

Suppose L_1 and L_2 are regular languages. Is $L = L_1 \cup L_2$ also regular?



Set of all regular Languages

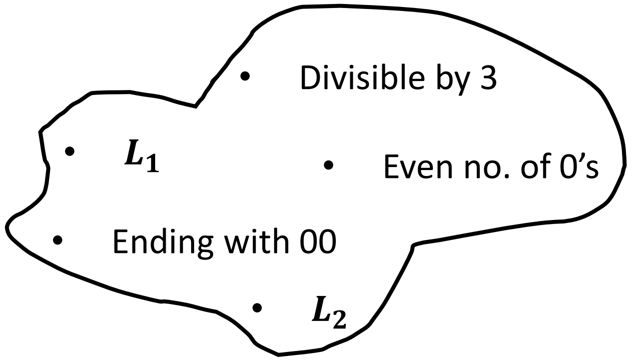
Closure of Regular Languages

Q: Is the set of all regular languages **closed under union**?

Suppose L_1 and L_2 are regular languages. Is $L = L_1 \cup L_2$ also regular?

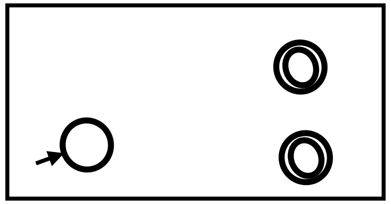
Proof: Since L_1 and L_2 are regular, there must be a DFA M_1 that accepts L_1, i.e. $L(M_1) = L_1$ and a DFA M_2 that accepts L_2, i.e. $L(M_2) = L_2$**.**

Using $M_1$and M_2, we will show how to construct an NFA M that accepts $L = L_1 \cup L_2$, i.e. $L(M) = L_1 \cup L_2$.

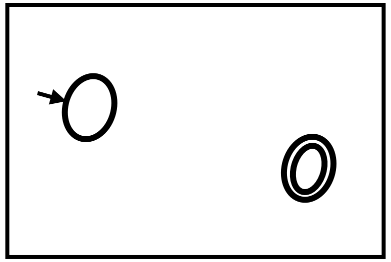


Set of all regular Languages

Suppose the DFA for M_1 is



And the DFA for M_2 is



Closure of Regular Languages

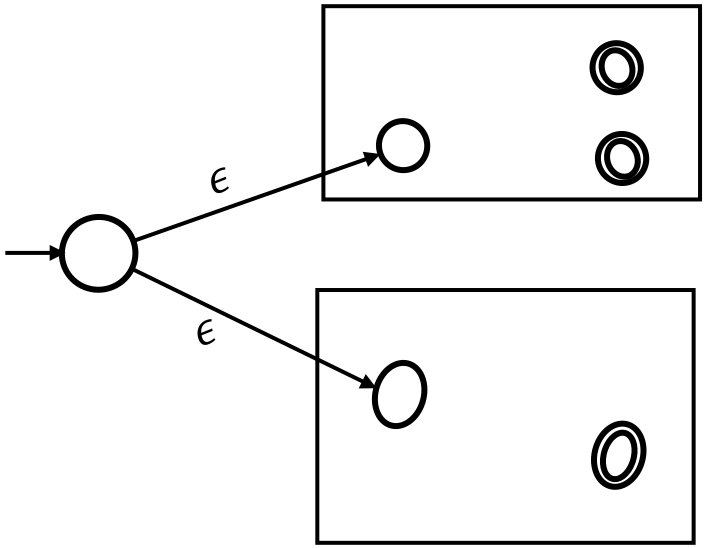
Q: Is the set of all regular languages **closed under union**?

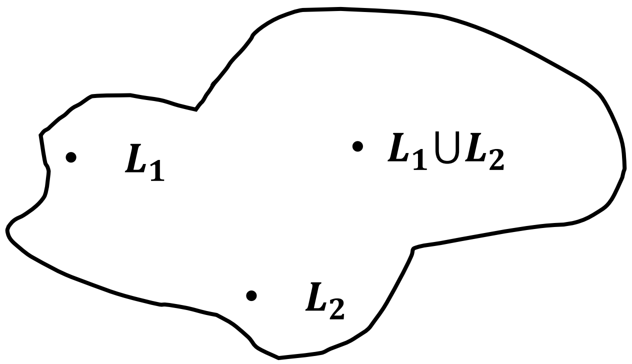
Suppose L_1 and L_2 are regular languages. Is $L = L_1 \cup L_2$ also regular?

Proof: Since L_1 and L_2 are regular, there must be a DFA M_1 that accepts L_1, i.e. $L(M_1) = L_1$ and a DFA M_2 that accepts L_2, i.e. $L(M_2) = L_2$**.**

Using $M_1$and M_2, we will show how to construct an NFA M that accepts $L = L_1 \cup L_2$, i.e. $L(M) = L_1 \cup L_2$.

NFA M accepting $\boldsymbol{L = L_1 \cup L_2}$





Set of all regular Languages

Closure of Regular Languages

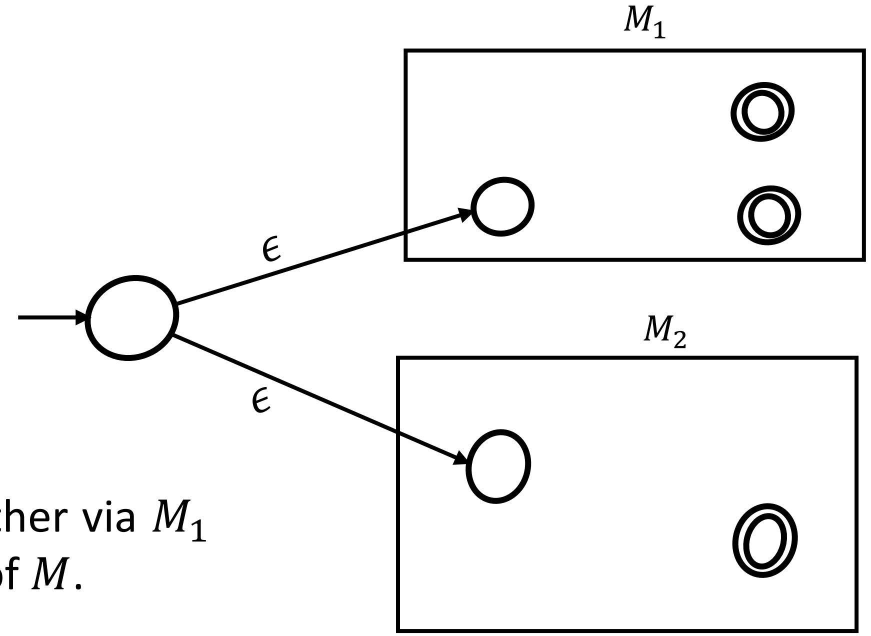
Q: Is the set of all regular languages **closed under union**?

Suppose L_1 and L_2 are regular languages. Is $L = L_1 \cup L_2$ also regular?

Proof: In order to prove that $L(M) = L_1 \cup L_2$, we show two things:

(i) $L \subseteq L_1 \cup L_2$

Let $\omega \in L$, i.e. ω is accepted by M. The final state for L can be reached either via M_1 or M_2. Thus ω must be accepted by either of them to reach the final state of M.



Closure of Regular Languages

Q: Is the set of all regular languages **closed under union**?

Suppose L_1 and L_2 are regular languages. Is $L = L_1 \cup L_2$ also regular?

Proof: In order to prove that $L(M) = L_1 \cup L_2$, we show two things:

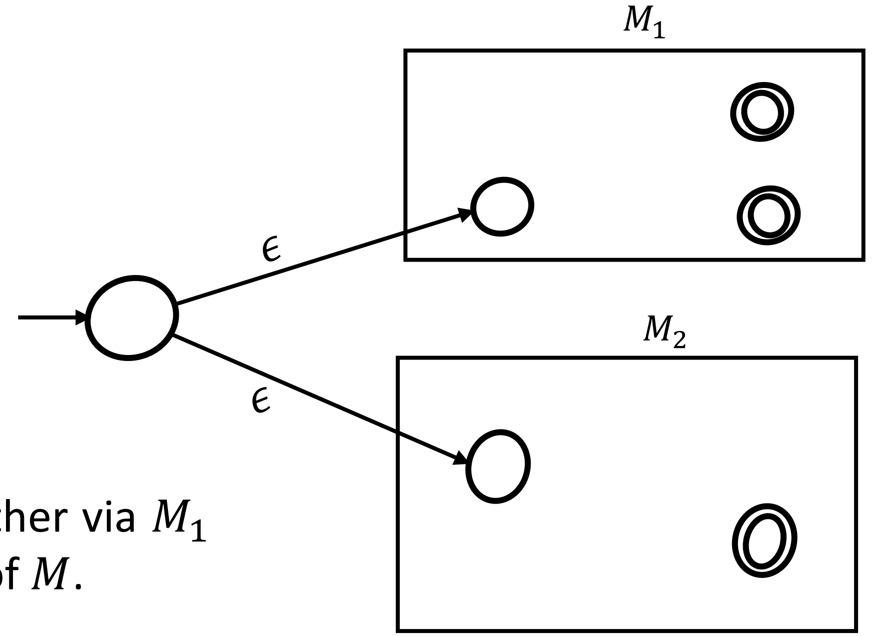
(i) $L \subseteq L_1 \cup L_2$

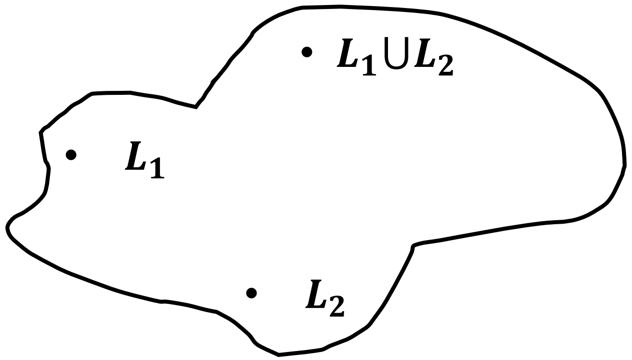
Let $\omega \in L$, i.e. ω is accepted by M. The final state for L can be reached either via M_1 or M_2. Thus ω must be accepted by either of them to reach the final state of M.

(ii) $L_1 \cup L_2 \subseteq L$

Let $\omega \in L_1 \cup L_2$. Then, $\omega \in L_1$ or $\omega \in L_2$.

Thus, ω must reach the final state of M_1 or M_2. But since the start state of M_1 or M_2 can be reached from the start state of M by taking an ϵ-transition, $\omega \in L$.





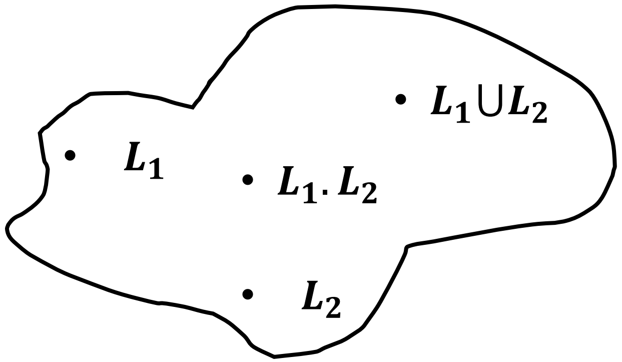
Set of all regular Languages

Closure of Regular Languages

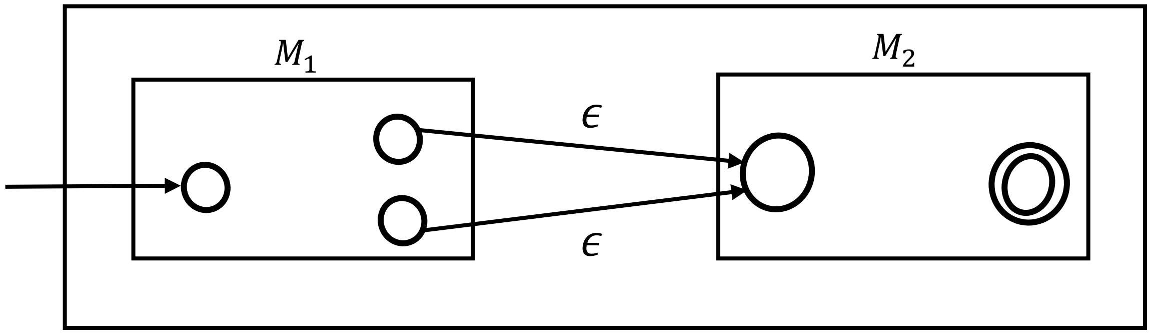
Q: Is the set of all regular languages **closed under concatenation**? Suppose L_1 and L_2 are regular languages. Is $L = L_1.L_2$ also regular?

Proof: Since L_1 and L_2 are regular, there must be a DFA M_1 that accepts L_1, i.e. $L(M_1) = L_1$ and a DFA M_2 that accepts L_2, i.e. $L(M_2) = L_2$**.**

Using $M_1$and M_2, we will show how to construct an NFA M that accepts $L = L_1.L_2$.



Set of all regular Languages



$L_1.L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$

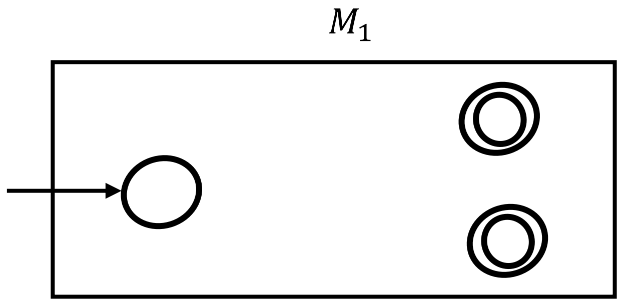
NFA M accepting $\boldsymbol{L = L_1.L_2}$

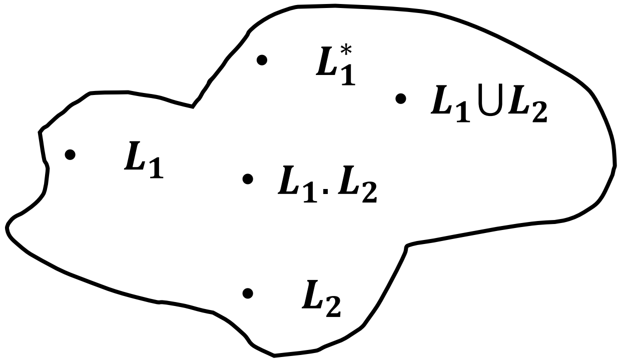
Closure of Regular Languages

Q: Is the set of all regular languages **closed under star**? Suppose L_1 is a regular language. Is L_1^* also regular?

Proof: Since L_1 is regular, there must be a DFA M_1 that accepts L_1, i.e. $L(M_1) = L_1$**.** Using M_1, we will show how to construct an NFA M that accepts $L = L_1^*$.

$L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0$ and each $x_i \in L_1\}$



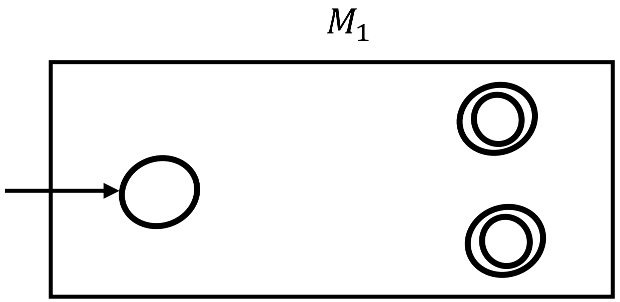


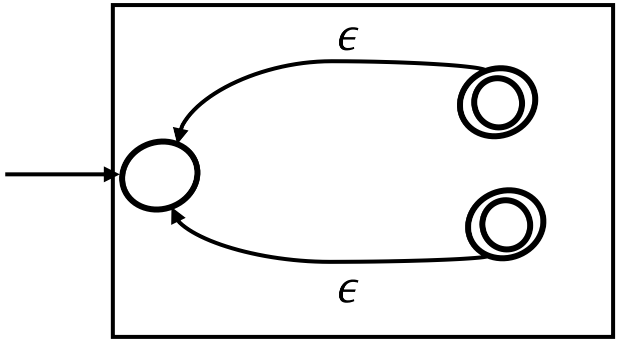
Set of all regular Languages

Closure of Regular Languages

Q: Is the set of all regular languages **closed under star**? Suppose L_1 is a regular language. Is L_1^* also regular?

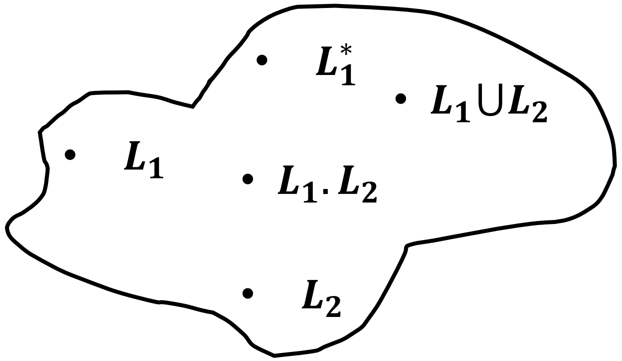
Proof: Since L_1 is regular, there must be a DFA M_1 that accepts L_1, i.e. $L(M_1) = L_1$**.** Using M_1, we will show how to construct an NFA M that accepts $L = L_1^*$.





NFA accepting $\boldsymbol{L = L_1^*}$

$L_1^* = \{x_1 x_2 \cdots x_k |\ k \geq 0$ and each $x_i \in L_1\}$



Set of all regular Languages

Steps:

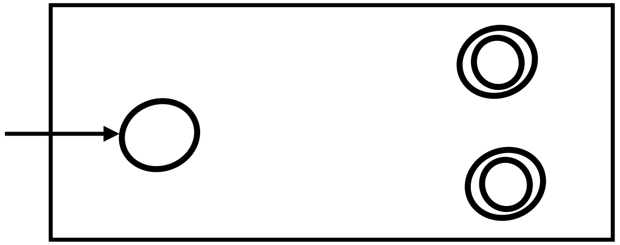
- Make ϵ-transitions from the final states of L_1 to the initial state of L_1.

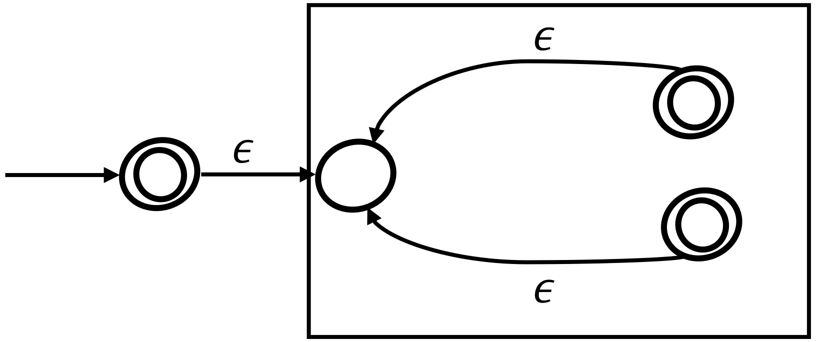
Closure of Regular Languages

Q: Is the set of all regular languages **closed under star**? Suppose L_1 is a regular language. Is L_1^* also regular?

Proof: Since L_1 is regular, there must be a DFA M_1 that accepts L_1, i.e. $L(M_1) = L_1$**.** Using M_1, we will show how to construct an NFA M that accepts $L = L_1^*$.

M_1



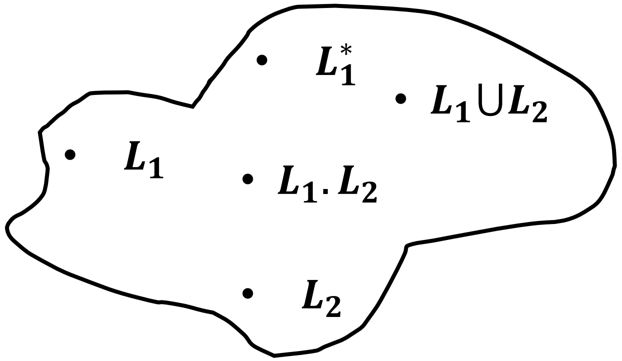


NFA accepting $\boldsymbol{L = L_1^*}$

$L_1^* = \{x_1 x_2 \cdots x_k \mid k \geq 0$ and each $x_i \in L_1\}$

Steps:

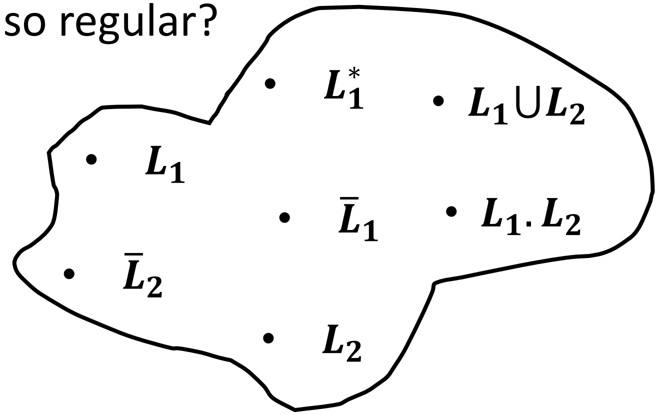
- Make ϵ-transitions from the final states of L_1 to the initial state of L_1.
- Make a new final state as the start state and make an ϵ-transition from this state to the previous start state of L_1.



Set of all regular Languages

Closure of Regular Languages

Q: Is the set of all regular languages **closed under complement**? If L is regular, then is $\overline{L}$ also regular?



Set of all regular Languages

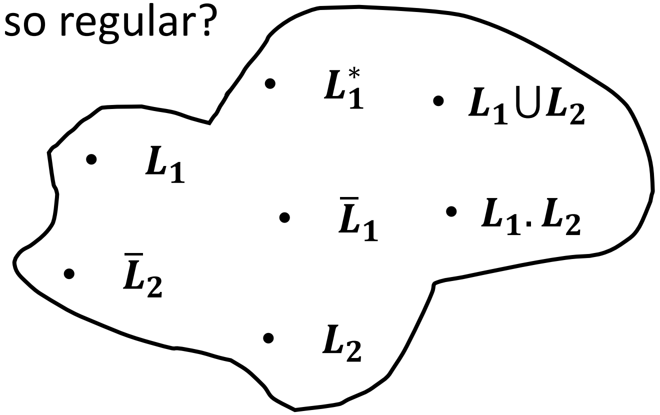
Proof: Given a DFA M, such that $L(M) = L$, construct the **toggled DFA** M' from M, by

(i) changing all the non-final states of M to be the final states of M' and
(ii) changing all the final states M to be the non-final states of M'.

$$\boldsymbol{L(M') = \overline{L}}$$

Closure of Regular Languages

Q: Is the set of all regular languages **closed under complement**? If L is regular, then is $\overline{L}$ also regular?



Set of all regular Languages

Proof: Given a DFA M, such that $L(M) = L$, construct the **toggled DFA** M' from M, by

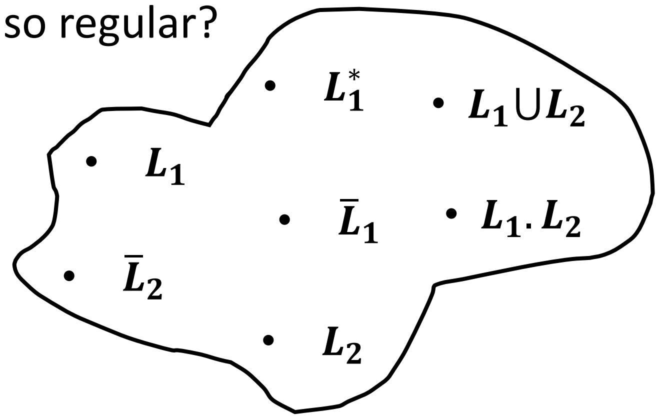
(i) changing all the non-final states of M to be the final states of M' and
(ii) changing all the final states M to be the non-final states of M'.

$$\boldsymbol{L(M') = \overline{L}}$$

Q: If L is the language accepted by an NFA, does "toggling" its states result in an NFA that accepts $\overline{L}$?

Closure of Regular Languages

Q: Is the set of all regular languages **closed under complement**? If L is regular, then is $\overline{L}$ also regular?



Set of all regular Languages

Proof: Given a DFA M, such that $L(M) = L$, construct the **toggled DFA** M' from M, by

(i) changing all the non-final states of M to be the final states of M' and
(ii) changing all the final states M to be the non-final states of M'.

$$\boldsymbol{L(M') = \overline{L}}$$

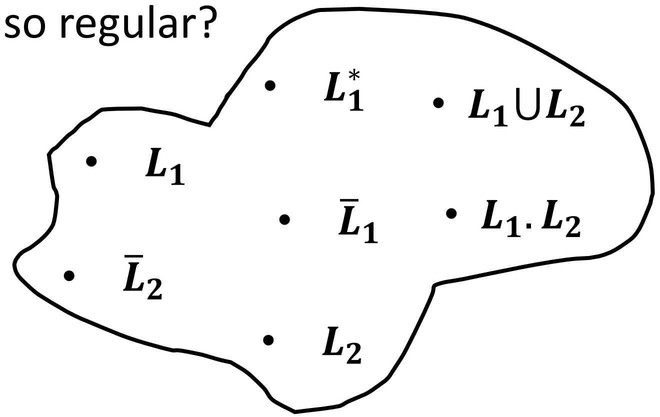
Q: If L is the language accepted by an NFA, does "toggling" its states result in an NFA that accepts $\overline{L}$?

Proof: Consider that for an input string $x \in L$, such that N accepts it. Suppose there is an rejecting run and an accepting run for input x. (See Table)

	NFA N	**Toggled NFA N'**
Run 1	Rejecting	
Run 2	Accepting	

Closure of Regular Languages

Q: Is the set of all regular languages **closed under complement**? If L is regular, then is $\overline{L}$ also regular?



Set of all regular Languages

Proof: Given a DFA M, such that $L(M) = L$, construct the **toggled DFA** M' from M, by

(i) changing all the non-final states of M to be the final states of M' and
(ii) changing all the final states M to be the non-final states of M'.

$$\boldsymbol{L(M') = \overline{L}}$$

Q: If L is the language accepted by an NFA, does "toggling" its states result in an NFA that accepts $\overline{L}$?

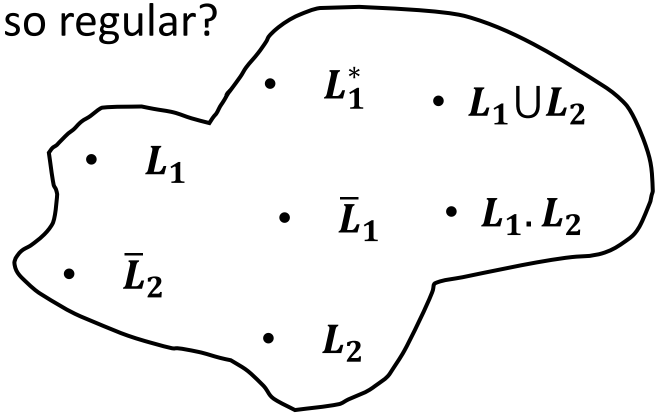
Proof: Consider that for an input string $x \in L$, such that N accepts it. Suppose there is an rejecting run and an accepting run for input x. (See Table)

For toggled NFA N' too, there are two runs for x. However, the rejecting run N is an accepting run for N'. Thus x is accepted by both N and N'.

	NFA N	**Toggled NFA N'**
Run 1	Rejecting	Accepting
Run 2	Accepting	Rejecting

Closure of Regular Languages

Q: Is the set of all regular languages **closed under complement**? If L is regular, then is $\overline{L}$ also regular?



Set of all regular Languages

Proof: Given a DFA M, such that $L(M) = L$, construct the **toggled DFA** M' from M, by

(i) changing all the non-final states of M to be the final states of M' and
(ii) changing all the final states M to be the non-final states of M'.

$$\boldsymbol{L(M') = \overline{L}}$$

Q: If L is the language accepted by an NFA, does "toggling" its states result in an NFA that accepts $\overline{L}$?

Proof: Consider that for an input string $x \in L$**,** such that N accepts it. Suppose there is an rejecting run and an accepting run for input x. (See Table)

For toggled NFA N' too, there are two runs for x. However, the rejecting run N is an accepting run for N'. Thus x is accepted by both N and N'.

Contradiction! So No, the **toggled NFA does not accept $\overline{L}$**.

	NFA N	**Toggled NFA N'**
Run 1	Rejecting	Accepting
Run 2	Accepting	Rejecting

Closure of Regular Languages

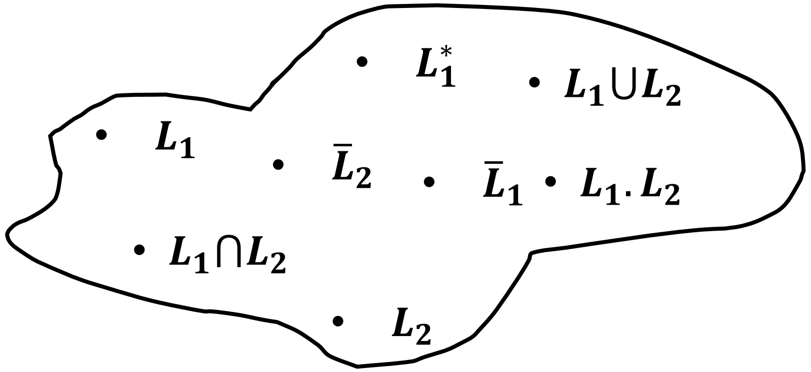
Q: Is the set of all regular languages **closed under intersection**? If L_1 and L_2 are regular, then is $L = L_1 \cap L_2$ also regular?

Proof: We shall use the fact that regular languages are **closed** under union and complement.

Note that using De Morgan's laws:

$$\boldsymbol{L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}}$$

Given a DFA for L_1 and a DFA for L_2, we know how to construct an NFA for $\overline{L_1}, \overline{L_2}$ as well as for $L_1 \cup L_2$. Using these constructions and the aforementioned relationship, we can construct an NFA for $L = L_1 \cap L_2$



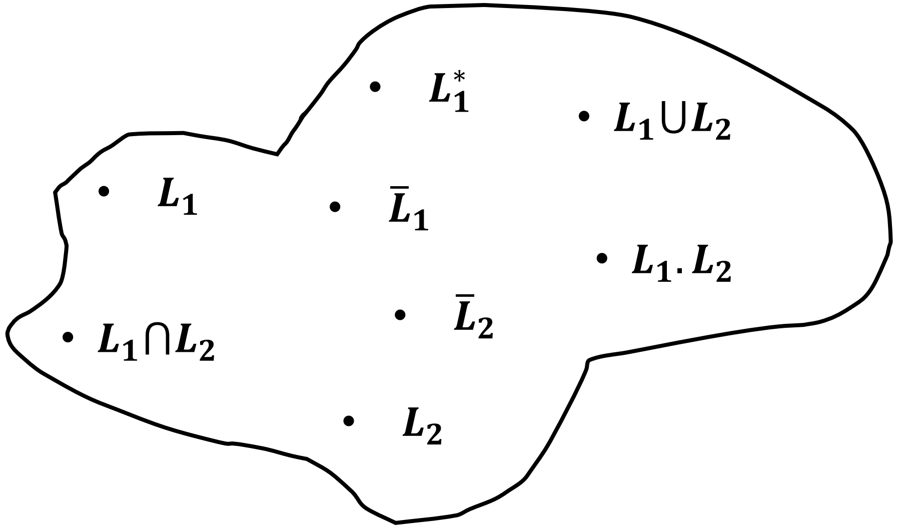
Set of all regular Languages

Closure of Regular Languages

Summary:

Regular Languages are closed under:

- **Union**
- **Intersection**
- **Star**
- **Complement**
- **Concatenation**



Set of all regular Languages

Regular Languages

If Σ is an alphabet, then

- $\Sigma^0 = \{\epsilon\}$
- $\Sigma^2 = \{a_1 a_2 | a_1 \in \Sigma,\ a_2 \in \Sigma\}$
- $\Sigma^k = \{a_1 a_2 \cdots a_k | a_i \in \Sigma\ |0 \leq i \leq k\}$
- $\Sigma^* = \{\bigcup_{i \geq 0} \Sigma^i\} = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2\ \cdots\} = \{a_1 a_2 \cdots a_k | k \in \{0,1,\cdots\}\ \&\ a_j \in \Sigma, \forall j \in \{1,2,\cdots,k\}\}$

A Language $L \subset \Sigma^*$ and $L^* = \{\bigcup_{i \geq 0} \mathrm{L}^i\}$

Regular Languages

If Σ is an alphabet, then

- $\Sigma^0 = \{\epsilon\}$
- $\Sigma^2 = \{a_1 a_2 | a_1 \in \Sigma,\ a_2 \in \Sigma\}$
- $\Sigma^k = \{a_1 a_2 \cdots a_k | a_i \in \Sigma \,|0 \leq i \leq k\}$
- $\Sigma^* = \{\bigcup_{i\geq 0} \Sigma^i\} = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2\ \cdots\} = \{a_1 a_2 \cdots a_k | k \in \{0,1,\cdots\} \,\&\, a_j \in \Sigma, \forall j \in \{1,2,\cdots,k\}\}$

A Language $L \subset \Sigma^*$ and $L^* = \{\bigcup_{i\geq 0} \mathrm{L}^i\}$

Regular Language (alternate definition): Let Σ be an alphabet. Then the following are the regular languages over Σ:

- The empty language Φ is regular
- For each $a \in \Sigma, \{a\}$ is regular.
- Let L_1, L_2 be regular languages. Then $L_1 \cup L_2$, $L_1 . L_2$, L_1^* are regular languages.

Regular Expressions

A regular expression describes regular languages algebraically. The algebraic formulation also provides a powerful set of tools which will be leveraged to prove

- languages are regular
- derive properties of regular languages

Regular Expressions

A regular expression describes regular languages algebraically. The algebraic formulation also provides a powerful set of tools which will be leveraged to prove

- languages are regular
- derive properties of regular languages

Syntax for regular expressions (Recursive definition): R is said to be a regular expression if it has one of the following forms:

- Φ is a regular expression, $L(\Phi) = \Phi$
- ϵ is a regular expression, $L(\epsilon) = \{\epsilon\}$
- Any $a \in \Sigma$ is a regular expression, $L(a) = \{a\}$

Regular Expressions

A regular expression describes regular languages algebraically. The algebraic formulation also provides a powerful set of tools which will be leveraged to prove

- languages are regular
- derive properties of regular languages

Syntax for regular expressions (Recursive definition): R is said to be a regular expression if it has one of the following forms:

- Φ is a regular expression, $L(\Phi) = \Phi$
- ϵ is a regular expression, $L(\epsilon) = \{\epsilon\}$
- Any $a \in \Sigma$ is a regular expression, $L(a) = \{a\}$
- $R_1 + R_2$ is a regular expression if R_1 and R_2 are regular expressions, $L(R_1 + R_2) = L(R_1) \cup L(R_2)$
- R^* is a regular expression if R is a regular expression, $L(R^*) = \left(L(R)\right)^*$

Regular Expressions

A regular expression describes regular languages algebraically. The algebraic formulation also provides a powerful set of tools which will be leveraged to prove

- languages are regular
- derive properties of regular languages

Syntax for regular expressions (Recursive definition): R is said to be a regular expression if it has one of the following forms:

- Φ is a regular expression, $L(\Phi) = \Phi$
- ϵ is a regular expression, $L(\epsilon) = \{\epsilon\}$
- Any $a \in \Sigma$ is a regular expression, $L(a) = \{a\}$
- $R_1 + R_2$ is a regular expression if R_1 and R_2 are regular expressions, $L(R_1 + R_2) = L(R_1) \cup L(R_2)$
- R^* is a regular expression if R is a regular expression, $L(R^*) = \left(L(R)\right)^*$
- R_1R_2 is a regular expression if if R_1 and R_2 are regular expressions, $L(R_1R_2) = L(R_1).L(R_2)$
- (R) is a regular expression if R is a regular expression, $L\left((R)\right) = R$

Regular Expressions

Syntax for regular expressions:

Regular Expression	Regular Language	Comment
Φ	$\{\}$	The empty set
ϵ	$\{\epsilon\}$	The set containing ϵ only
a	$\{a\}$	Any $a \in \Sigma$
$R_1 + R_2$	$L(R_1) \cup L(R_2)$	For regular expressions R_1 and R_2
$R_1 R_2$	$L(R_1).L(R_2)$	For regular expressions R_1 and R_2
R^*	$\left(L(R)\right)^*$	For regular expressions R
(R)	$L(R)$	For regular expressions R

Order of precedence: (), *, . , +

A language L is regular if and only if for some regular expression R, $L(R) = L$.

RE's are equivalent in power to NFAs/DFAs

Regular Expressions

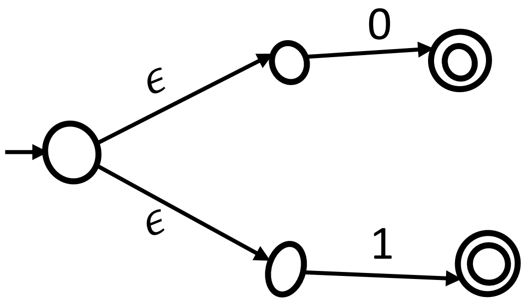
Syntax for regular expressions:

Regular Expression R	$\boldsymbol{L(R)}$
01	$\{01\}$
$01 + 1$	$\{01,1\}$
$(0 + 1)^*$	$\{\epsilon, 0, 1, 00,01, \cdots\}$
$(01 + \epsilon)1$	$\{011,1\}$
$(0 + 1)^*01$	$\{01,001,101,0001, \cdots\}$
$(0 + 10)^*(\epsilon + 1)$	$\{\epsilon, 0, 10, 00, 001,010, 0101, \cdots\}$

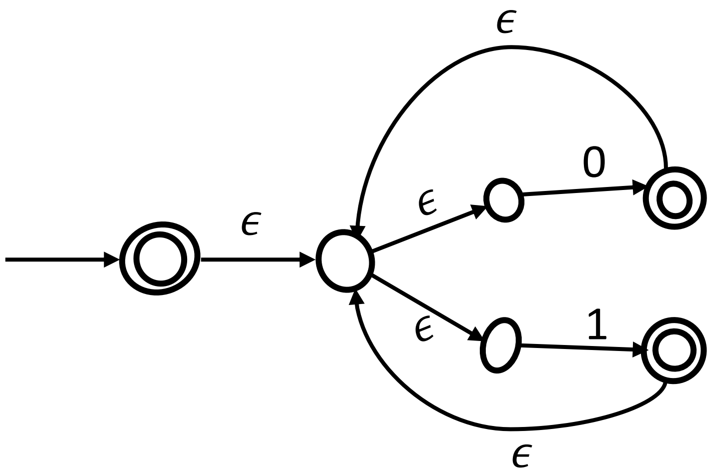
Regular Expressions

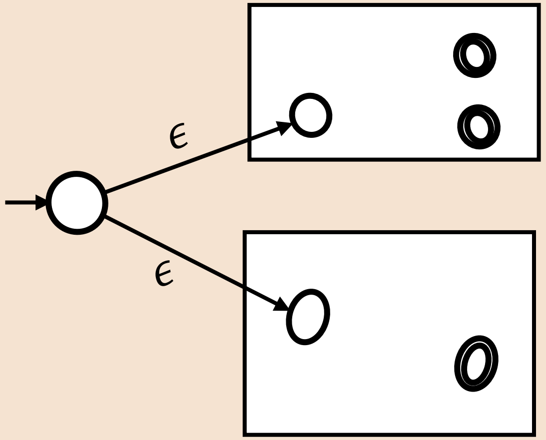
NFA for RE: $(\mathbf{0}+\mathbf{1})^*\mathbf{01}$

(i) NFA for $(0+1)$

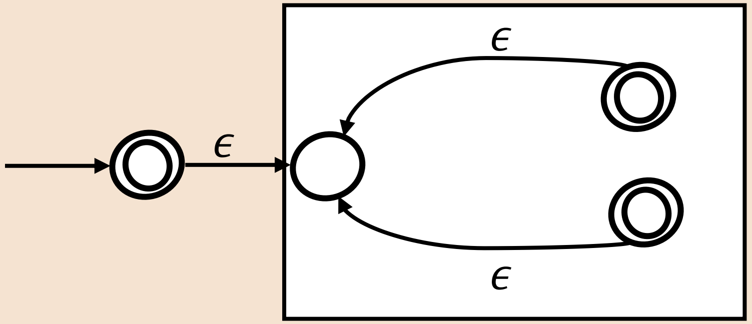


(ii) NFA for $(0+1)^*$





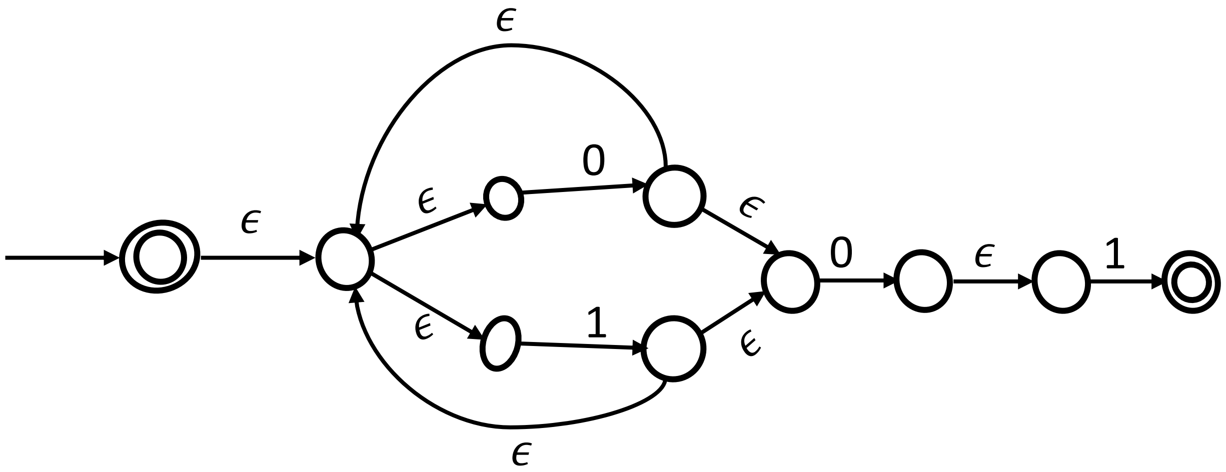
$L_1 \cup L_2$

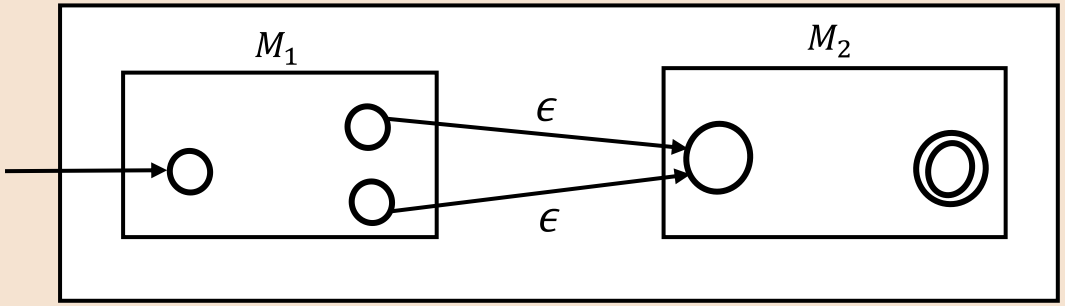


L_1^*

Regular Expressions

NFA for $(\mathbf{0}+\mathbf{1})^{*}\mathbf{01}$





$\boldsymbol{L_1.L_2}$

Regular Expressions

Let $\Sigma = \{a, b\}$.

Language	**Regular Expression**
$\{\omega \mid \omega$ ends in "ab"$\}$	$(a+b)^*ab$
$\{\omega \mid \omega$ has a single a $\}$	b^*ab^*
$\{\omega \mid \omega$ has at most one $a\}$	$b^* + b^*ab^*$
$\{\omega \mid \, \lvert\omega\rvert$ is even$\}$	$((a+b)(a+b))^* = (aa+bb+ab+ba)^*$
$\{\omega \mid \omega$ has "ab" as a substring$\}$	$(a+b)^*ab(a+b)^*$
$\{\omega \mid \lvert\omega\rvert$ is a multiple of 3$\}$	$\big((a+b)(a+b)(a+b)\big)^*$

Regular Expressions

Let $\Sigma = \{a, b\}$.

Language	Regular Expression
$\{\omega \mid \omega$ ends in "ab"$\}$	$(a+b)^*ab$
$\{\omega \mid \omega$ has a single a $\}$	b^*ab^*
$\{\omega \mid \omega$ has at most one $a\}$	$b^* + b^*ab^*$
$\{\omega \mid \|\omega\|$ is even$\}$	$((a+b)(a+b))^* = (aa+bb+ab+ba)^*$
$\{\omega \mid \omega$ has "ab" as a substring$\}$	$(a+b)^*ab(a+b)^*$
$\{\omega \mid \|\omega\|$ is a multiple of 3$\}$	$\big((a+b)(a+b)(a+b)\big)^*$

Some algebraic properties of Regular Expressions:

- $R_1 + (R_2 + R_3) = (R_1 + R_2) + R_3$
- $R_1(R_2R_3) = (R_1R_2)R_3$
- $R_1(R_2 + R_3) = R_1R_2 + R_1R_3$
- $(R_1 + R_2)R_3 = \mathrm{R_1R_2 + R_2R_3}$
- $R_1 + R_2 = R_2 + R_1$
- $R_1^*R_1^* = R_1^*$
- $(R_1^*)^* = R_1^*$
- $R\epsilon = \epsilon R = R$
- $R\Phi = \Phi R = \Phi$
- $R + \Phi = R$
- $\epsilon + RR^* = \epsilon + R^*R = R^*$
- $(R_1 + R_2)^* = (R_1^*R_2^*)^* = (R_1^* + R_2^*)^*$

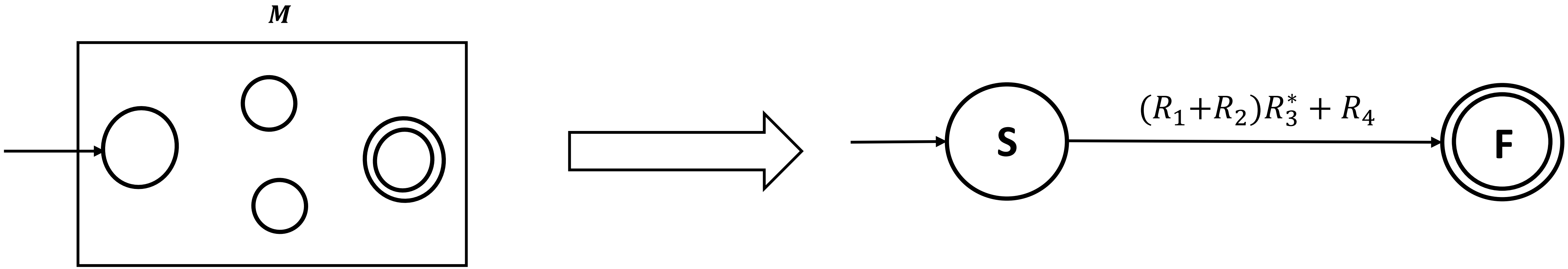
DFA to Regular Expressions

If a language is regular then it accepts a regular expression. We could draw equivalent NFAs for Regular Expressions.

How can we obtain Regular expressions given a DFA?

Given a DFA M, we **recursively** construct a two-state Generalized NFA (GNFA) with

- A start state and a final state
- A single arrow goes from the start state to the final state
- The label of this arrow is the regular expression corresponding to the language accepted by the DFA M.



Thank You!