

What did we learn so far?

- DFA: Deterministic finite automator
 - Limited memory
 - Deterministic behaviour.
- NFA: Nondeterministic finite automator
 - Limited memory
 - Nondeterministic behaviour.

However, $NFA = DFA$

Any language that is recognized by
a DFA/NFA is called a regular language.

Regular Expressions.

In simple terms, regular expressions describes languages.

Basically, it's a notation for describing strings in a set.

We use regular operations to build more complicated expressions.

- regular operations

- Union (\cup)
- Concatenation (\circ)
- Kleen Star ($*$)

$$\text{Ex: } (0 \cup 1)^* \equiv (\{0\} \cup \{1\})^* = \{0, 1\}^*$$

If Σ is an alphabet, the regular expression Σ describes the language consisting of all length 1 strings over the alphabet Σ

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

using Σ as a regular expression

$$\Sigma \equiv (0 \cup 1) = \{0, 1\}$$

Σ^* describes the language consisting of all strings over the alphabet Σ

$$\Sigma = \{0, 1, 2\}$$

$$\Sigma^* = (01101)^* = \{\epsilon, 0, 1, 2, 00, 000, \\ 01, 010, \dots\}$$

In regular expressions star operations are done first, followed by concatenation, and finally, union, unless parenthesis change the usual order.

1. Paranthesis
2. Star
3. Concatenation
4. Union.

Def, Regular Expressions

R is a regular expression,

if R is,

1. $a \xrightarrow{\{a\}}$, for some $a \in \Sigma$

2. $\epsilon \xleftarrow{\{\epsilon\}}$ } these are
3. $\phi \xleftarrow{\{\}} \text{not equal}$

4. $R_1 \cup R_2$, where R_1 & R_2 are
regular expressions

5. $R_1 \circ R_2$, where R_1 & R_2 are regular
expressions.

6. R_1^* , where R_1 is a regular
expression.

$$\text{Ex: } R \cup \emptyset = R$$

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

$$R = 1$$

$$L(R) = \{1\}$$

$$L(R \cup \emptyset) = \{1\} \cup \{\} = \{1\}$$

$$R \cup \epsilon \neq R \text{ (not always)}$$

$$R = 1$$

$$L(R) = \{1\}$$

$$L(R \cup \epsilon) = \{1\} \cup \{\epsilon\} = \{1, \epsilon\}$$

Remember: regular expressions
describe a language

$$0^* 1 0^* = \{w \mid w \text{ contains a single 1}\}$$

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

Simple exercise; let $\Sigma = \{0, 1\}$
what are the languages described by
the following regular expressions?

1. $(01)^* = \{w \mid w \text{ is a string that has } n \text{ copies of } 01 \text{ any number of times}\}$

$$2. 1^* \circ \epsilon = 1^*$$

$$3. 0^* \circ \emptyset = 0^* \circ \{\} = \emptyset$$

$$4. (01)^* \cup \epsilon = (01)^* \quad | \quad (01) \cup \epsilon = \{\epsilon, (01)\}$$

$$5. (0 \cup \epsilon) 1^* = 0 1^* \cup 1^*$$

$$6. (0 \cup \epsilon) \circ (1 \cup \epsilon) = \{0, \epsilon\} \circ \{1, \epsilon\} = \{0, 01, 1, \epsilon\}$$

$$7. \emptyset^* = \{\}^* = \epsilon$$

Regular Expressions



Regular Languages .

Any regular language (a language that is recognized by a DFA/NFA) can be expressed by a regular expression.

Theorem 1.54

A language is regular iff some regular expression describes it.

Let A be a language,

A is regular \iff some regular expression describes A .

"If part" \Leftarrow (some regular expression describes $A \Rightarrow A$ is regular)

Lemma 1.55,

If a language A is described by a regular expression, then A is regular.

Proof: Let R be a regular expression then we will convert it to an NFA.

$$L(R) = A$$

$$1. R = a, a \in \Sigma, L(R) = \{a\}$$



$$2. R = \epsilon, L(R) = \{\epsilon\}$$



$$3. R = \emptyset, L(R) = \{\}$$



$$4. R = R_1 \cup R_2 \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \begin{array}{l} \text{use the already created} \\ \text{NFA's of } R_1, R_2 \text{ then} \\ \text{create larger NFA's} \end{array}$$

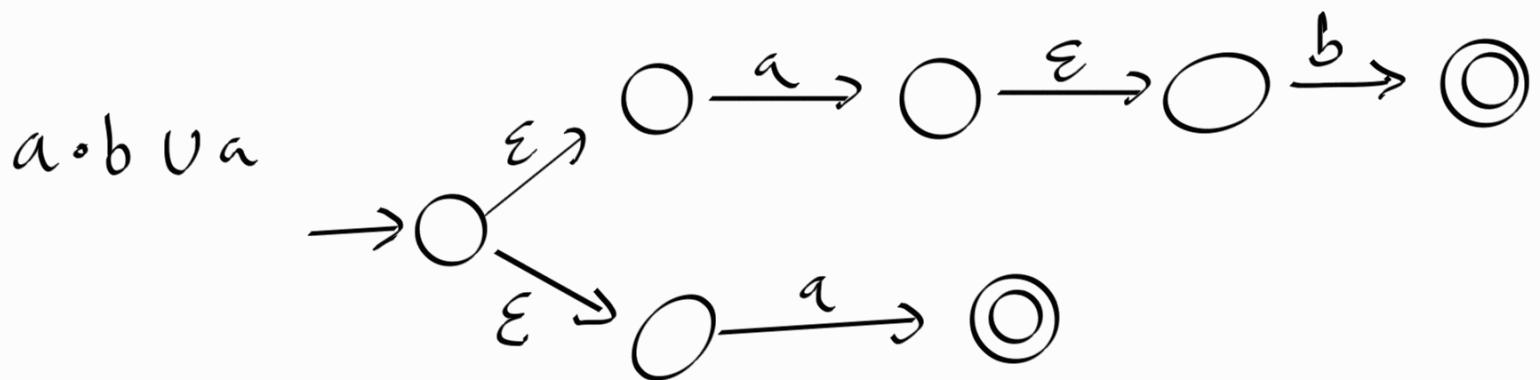
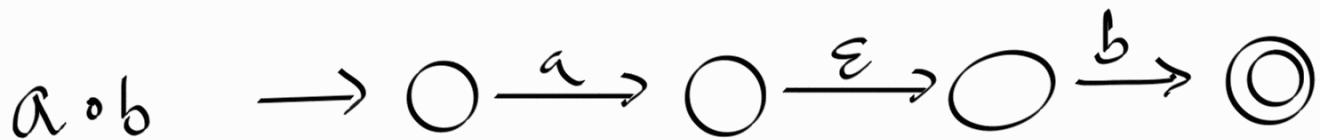
$$5. R = R_1 \circ R_2 \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \begin{array}{l} \\ \\ \end{array}$$

$$6. R = R_1^*$$

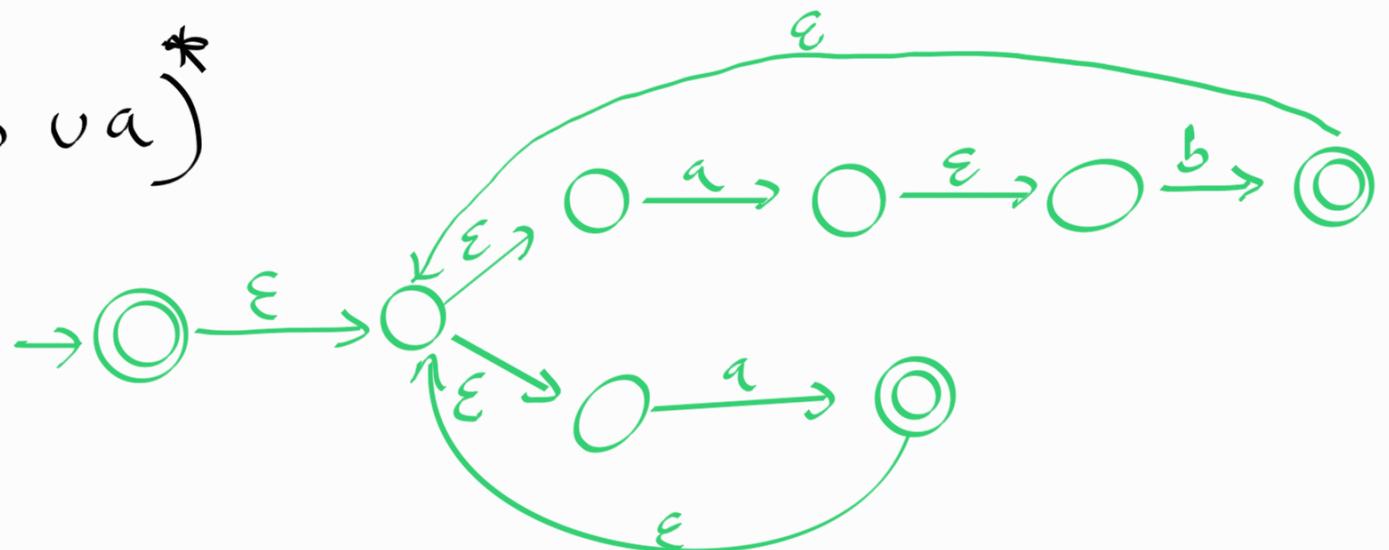
by following Thm 1.45, 1.47, 1.49

Ex: $(a \cdot b \cup a)^*$

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



$(a \cdot b \cup a)^*$



"only if" part \Rightarrow

Lemma 1.60

If a language is regular,
then it is described by a
regular expression.

Idea!

Step 1: DFA \longrightarrow GNFA

Step 2: GNFA \longrightarrow regular expression

new type of
finite automaton

Generalize NFA is simply a
Nondeterministic finite automaton wherein
the transition arrows may have any
regular expression as labels, instead
of only one symbol or ϵ .



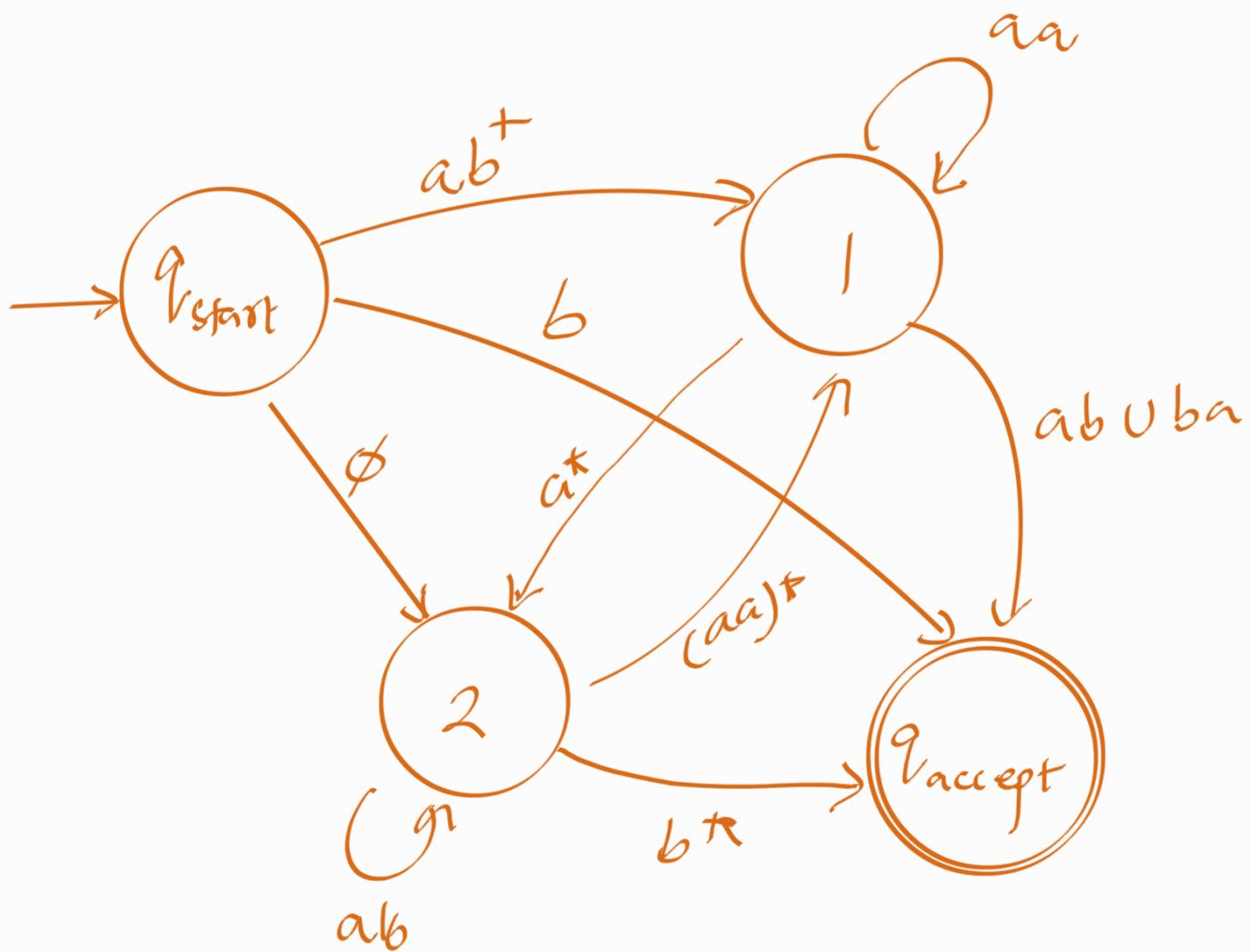
We want the DFA to convert into a
GNFA of above pattern.

GNFA follows a special set of conditions.

following conditions needs to be met if we want to call a machine GNFA.

- * Start state has transition arrows going to every other state , but no arrows coming in from any state.
- * There is only a single accept state , and it has arrows coming in from every other state but no arrows going to any other state.
- * Accept state cannot be the start state .
- * Except for start state and accept state , one arrow goes from every state to every other state and also from each state to itself.

Example GNFA

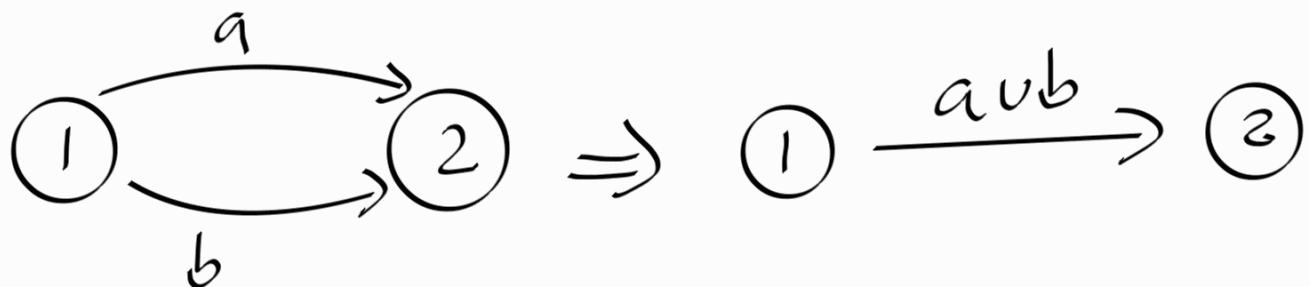


Note that \emptyset transition means that there is no direct transition between the two states.

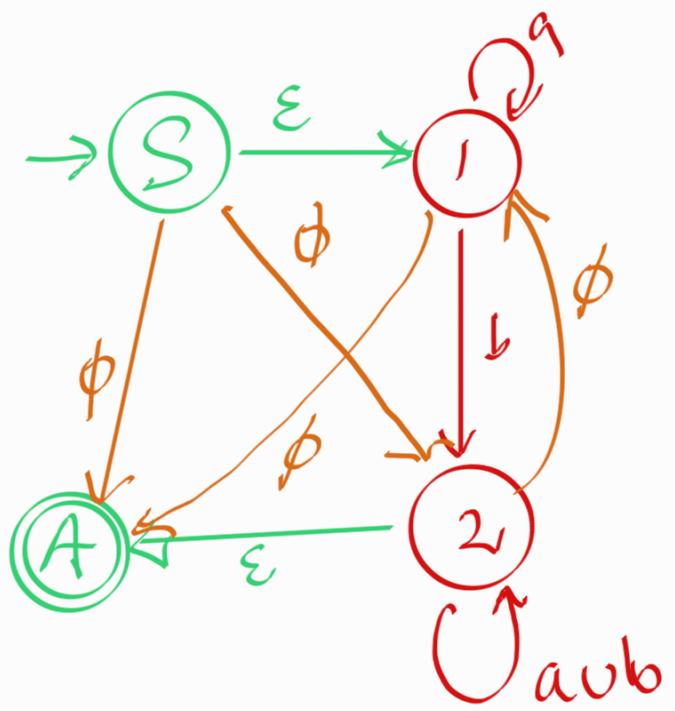
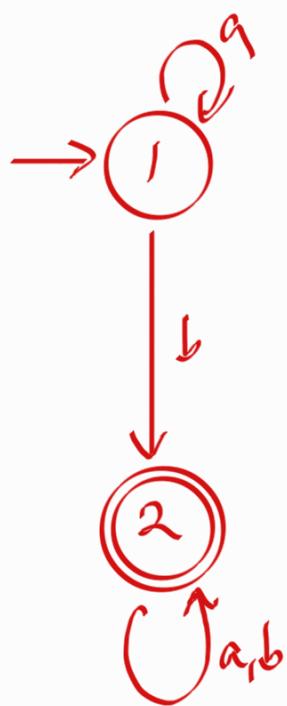
Let's look at how we can transform a DFA to a GNFA.

1. Add a new start state and add ϵ transitions to the old start state.
2. Add a new accept state and put ϵ -transitions to new accept state from all old accept states.
3. If any arrows have multiple labels (or if there are multiple arrows going between two states in the same direction), we replace each with a single arrow whose label is the union of the previous labels.

ex:



Let's look at a simple DFA
and convert it to a GNFA.



Now, we need to convert the GNFA into a regular expression.

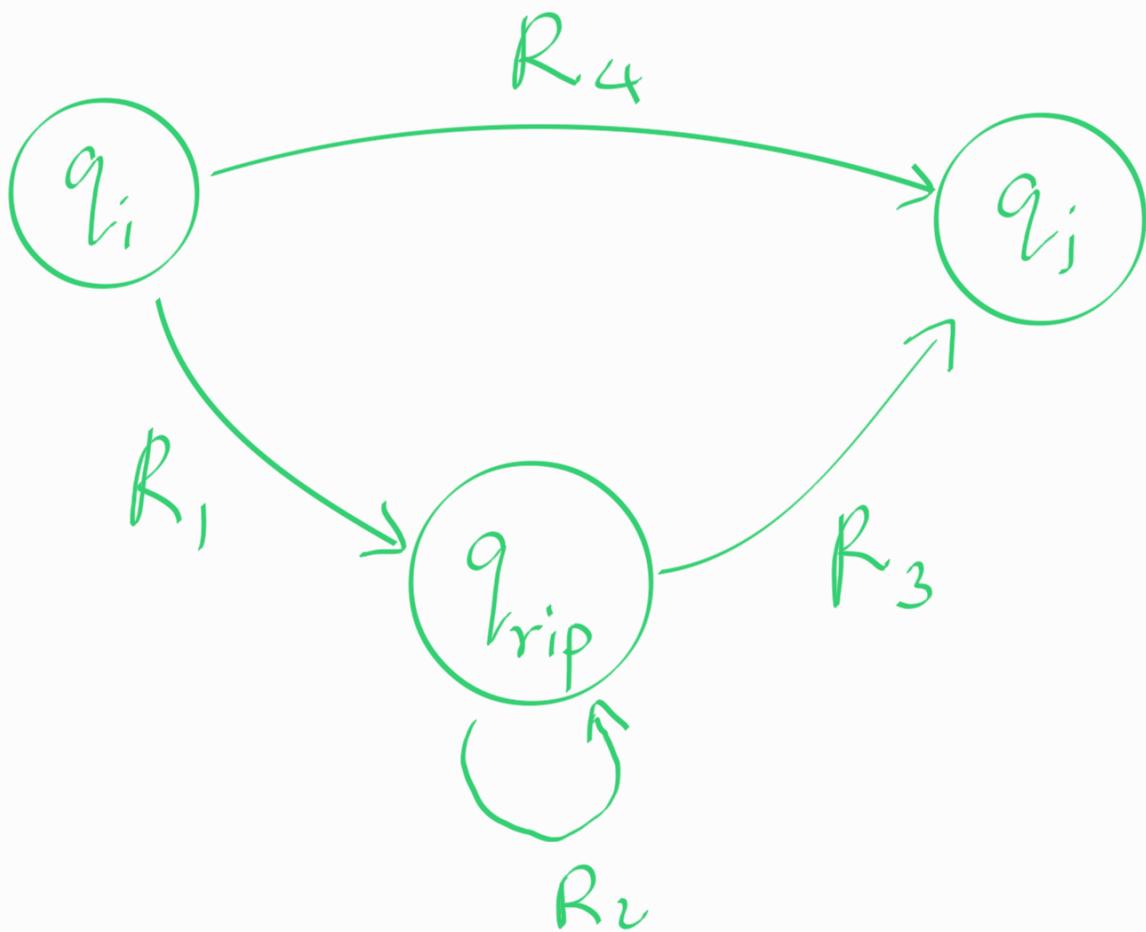
We systematically reduce the number of states of the given GNFA one by one until it has 2 states.

- Starting state
- Accept state
- Arrows between these 2 states with the final regular expression.

* The key idea about reducing the number of states is called ripping.

Basically, we rip a state and repair the damage, so that the same language is recognized by the GNFA.

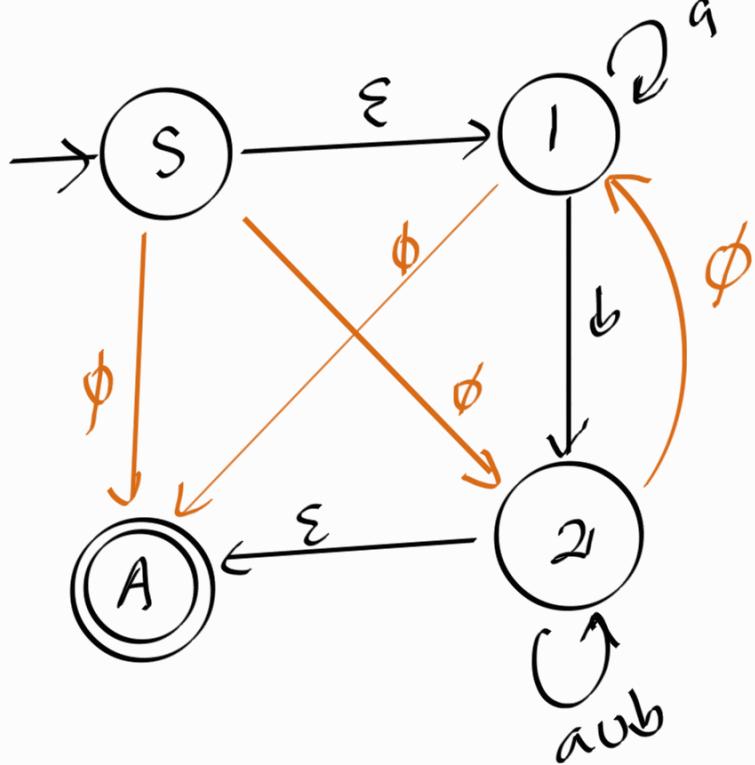
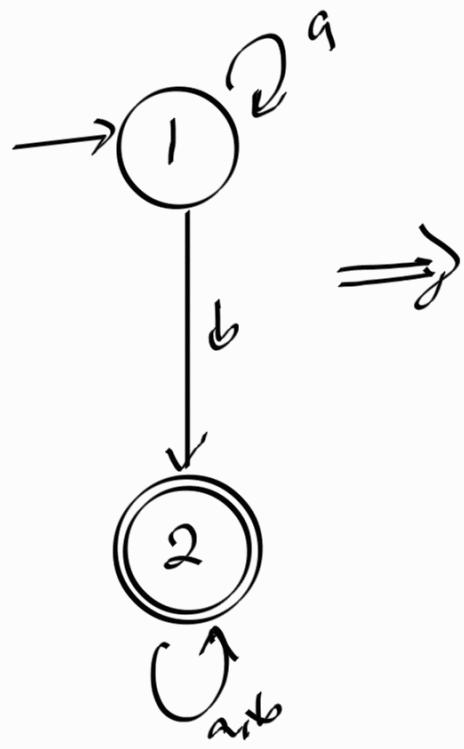
we can rip any state except start and accept state.



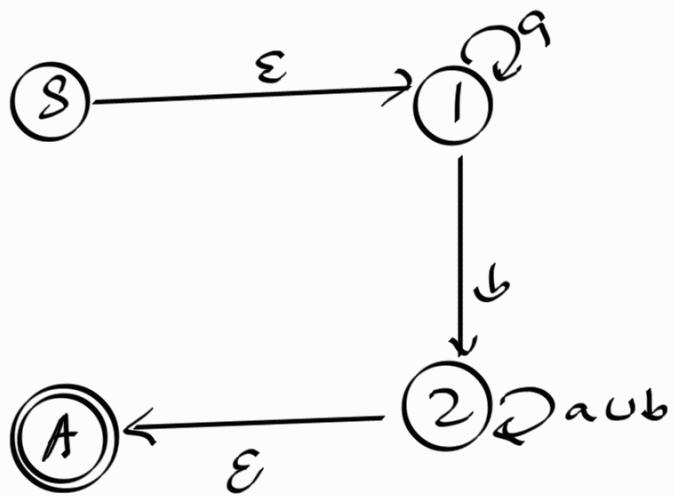
$$q_i \xrightarrow{(R_1 \cdot R_2^* \cdot R_3) \cup R_4} q_j$$

We have to consider all the paths that can go through the rip state that we consider and repair them.

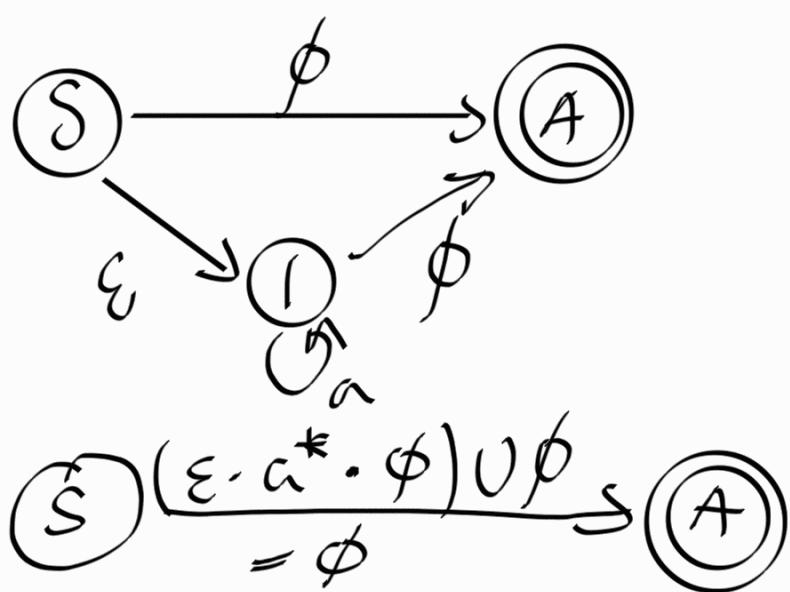
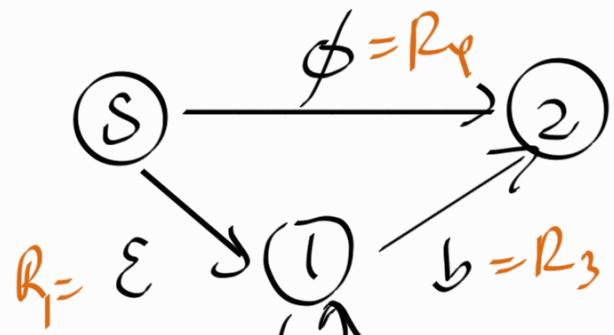
Example: DFA \rightarrow GNFA \rightarrow Regular Expression



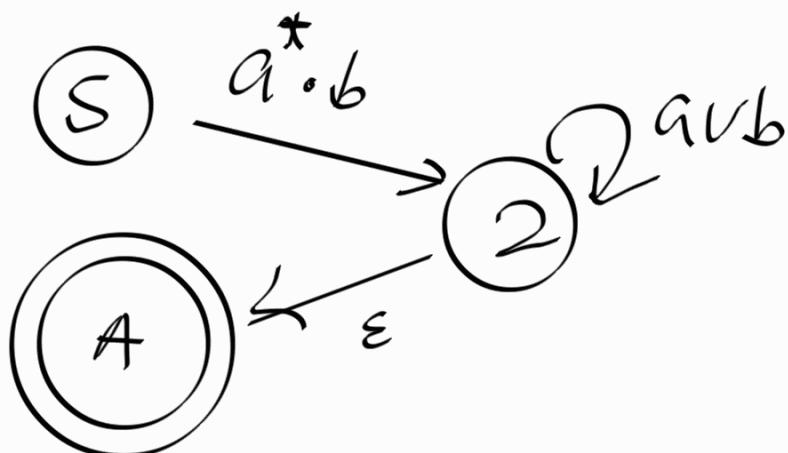
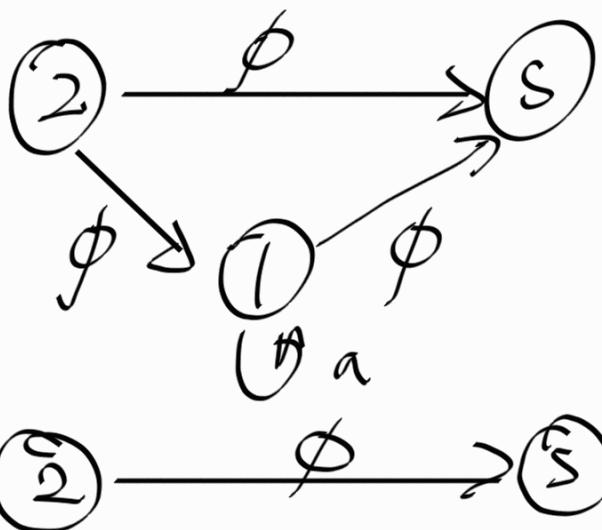
Sometimes we do not include the unnecessary \emptyset transitions as it clutters the figure.



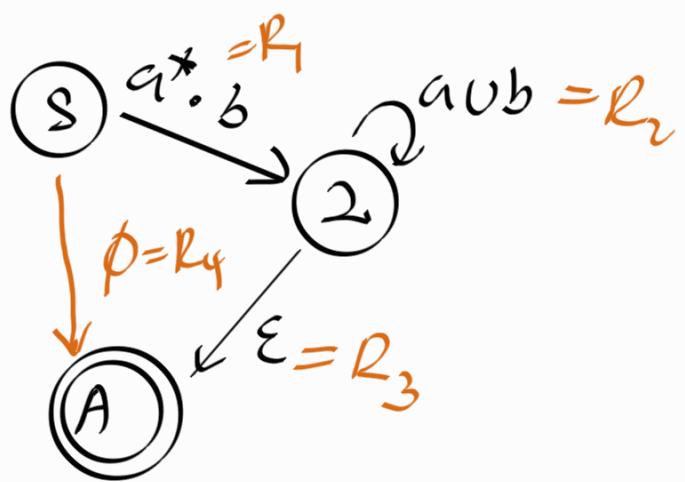
ripping ①



$$\begin{aligned}
 &S \xrightarrow{\phi} A \\
 &S \xrightarrow{\epsilon} 1 \\
 &1 \xrightarrow{a} 1 \\
 &1 \xrightarrow{\phi} 2 \\
 &2 \xrightarrow{(e \cdot a^* \cdot b) \cup \phi} A \\
 &= a^* \cdot b
 \end{aligned}$$

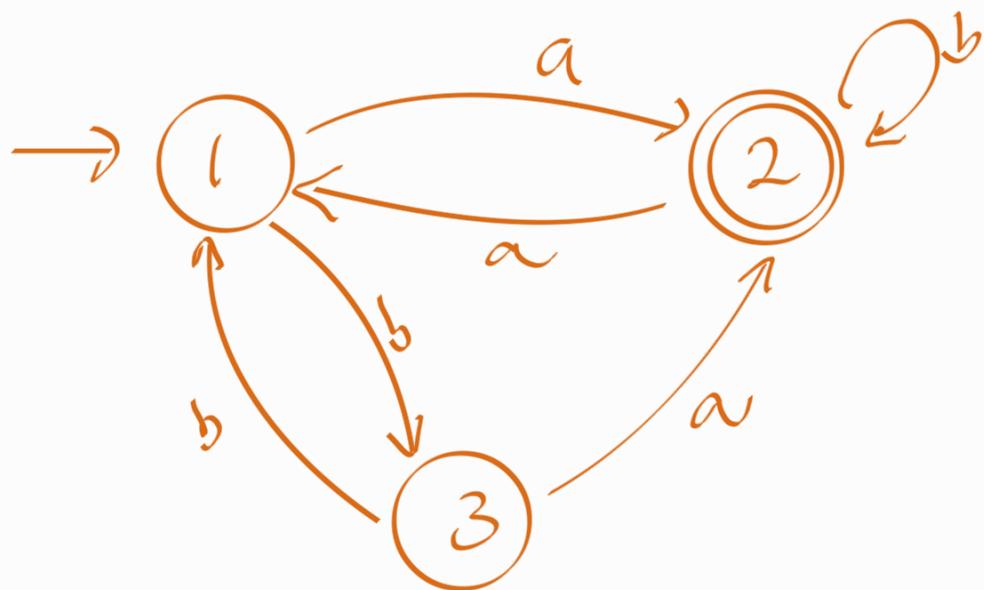


$$\begin{aligned}
 &2 \xrightarrow{\epsilon} A \\
 &2 \xrightarrow{\phi} 1 \\
 &1 \xrightarrow{a} 1 \\
 &1 \xrightarrow{\phi} 2 \\
 &2 \xrightarrow{(\phi \cdot a^* \cdot \phi) \cup \epsilon} A
 \end{aligned}$$

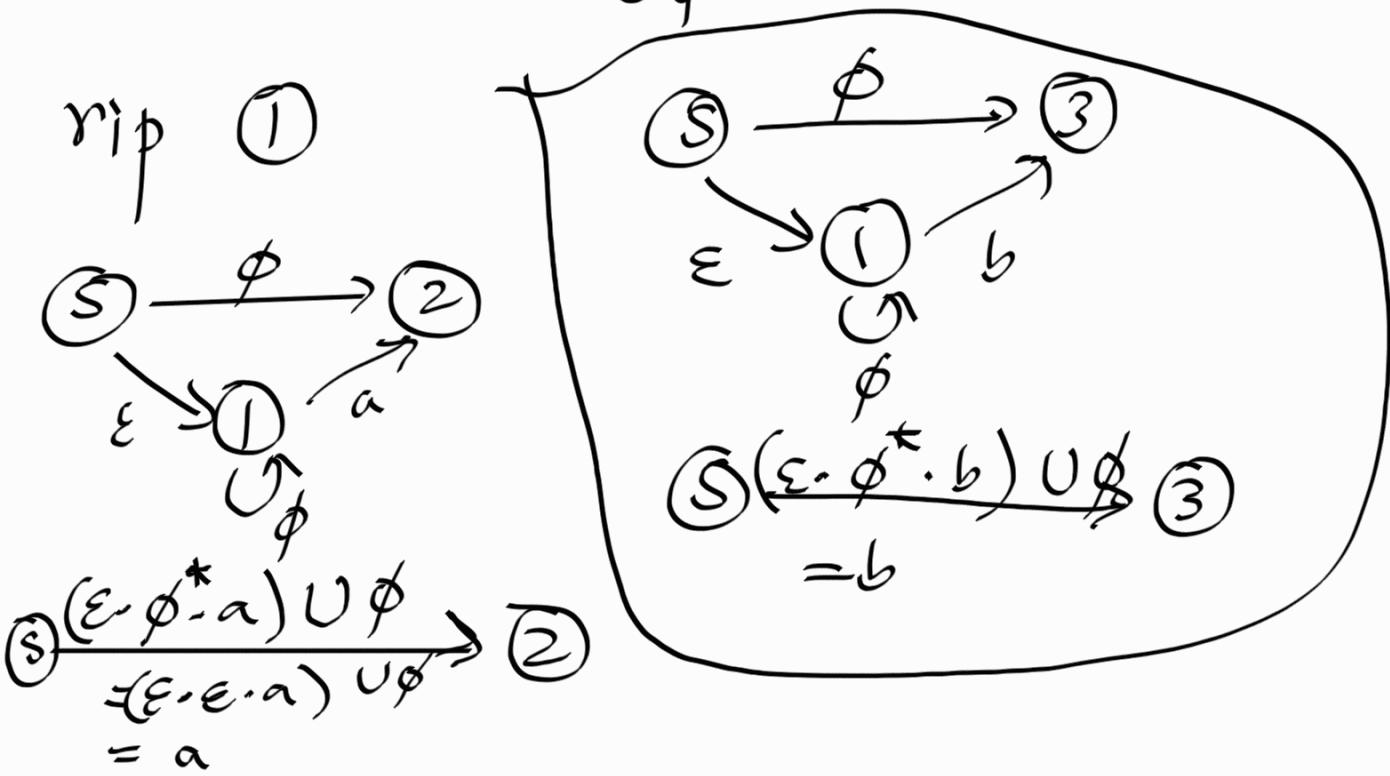
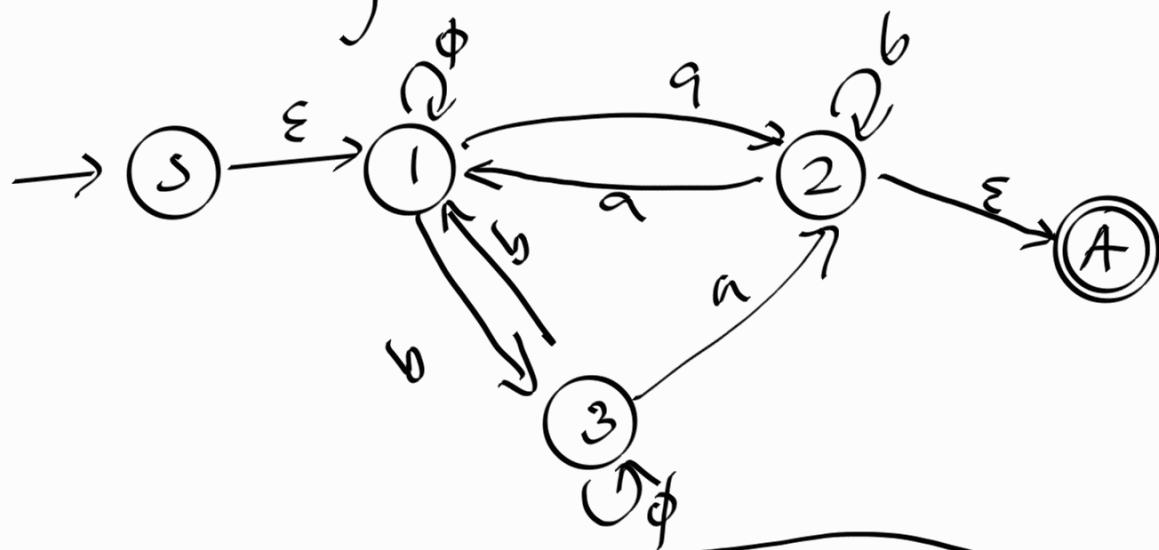


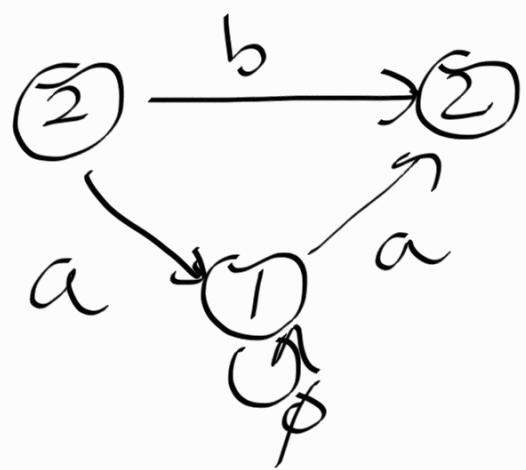
$$\begin{aligned}
 & \textcircled{S} \xrightarrow{(a^* \cdot b \cdot (a \cup b)^*) \cup \phi} \textcircled{A} \\
 &= (a^* \cdot b \cdot (a \cup b)^*)
 \end{aligned}$$

Another Example.



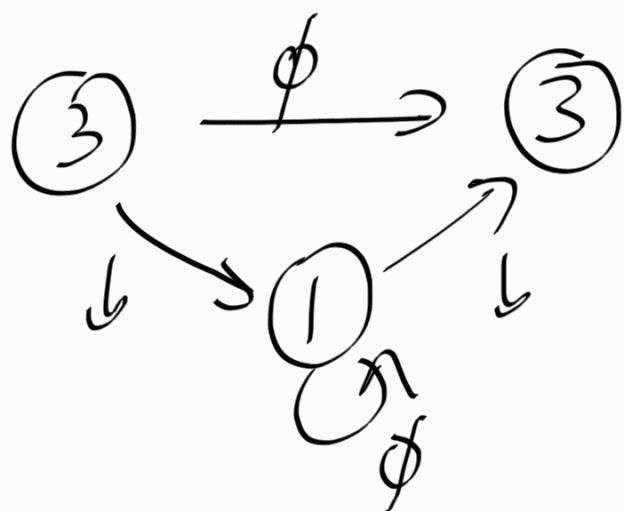
Let's try to convert to a GNFA





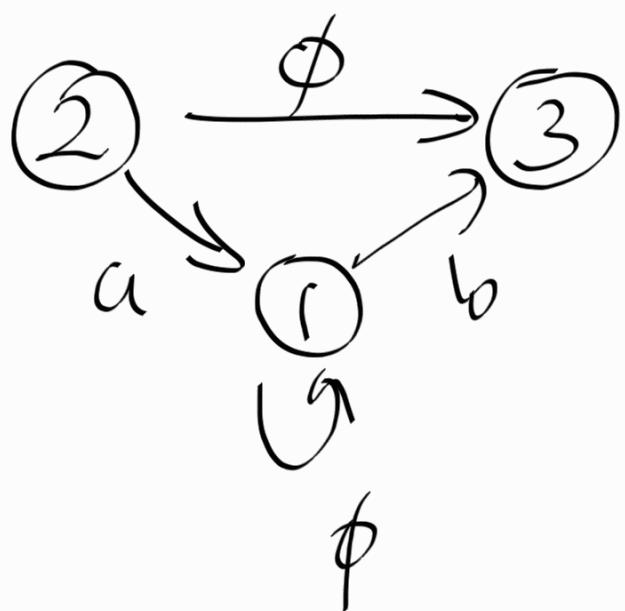
$$② \xrightarrow{(a \cdot \phi^* \cdot a) \cup b} ②$$

$a a \cup b$



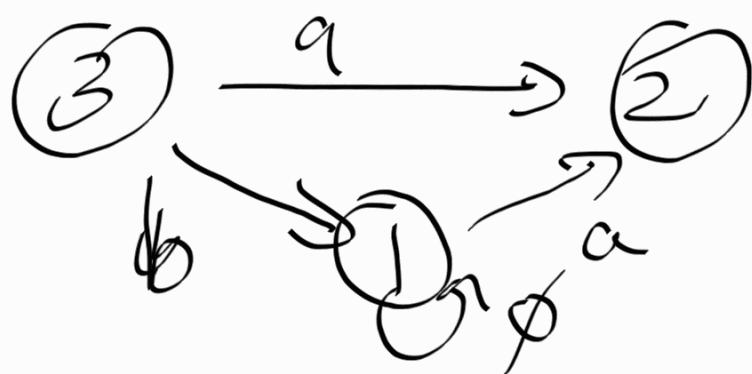
$$③ \xrightarrow{(b \cdot \phi^* \cdot b) \cup \phi} ③$$

$= b b$



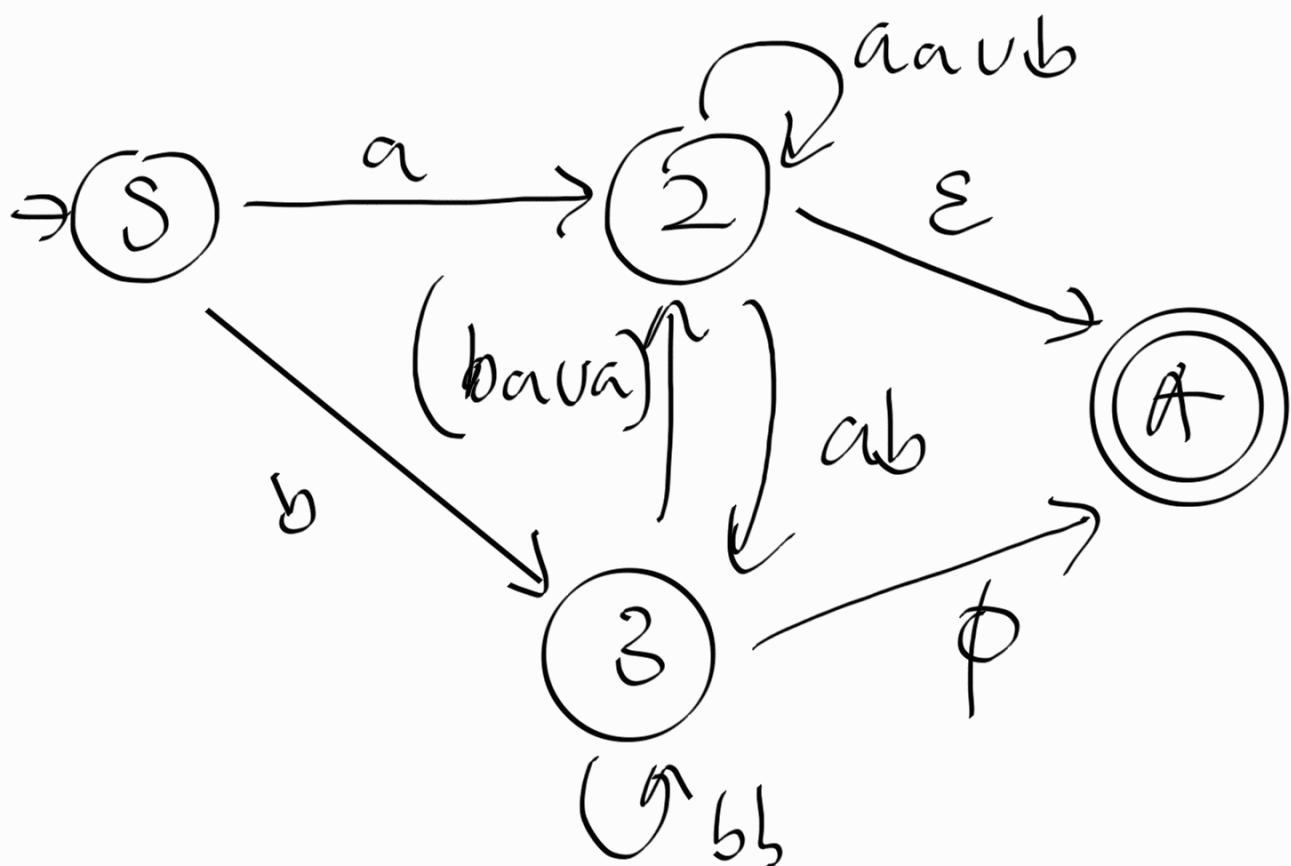
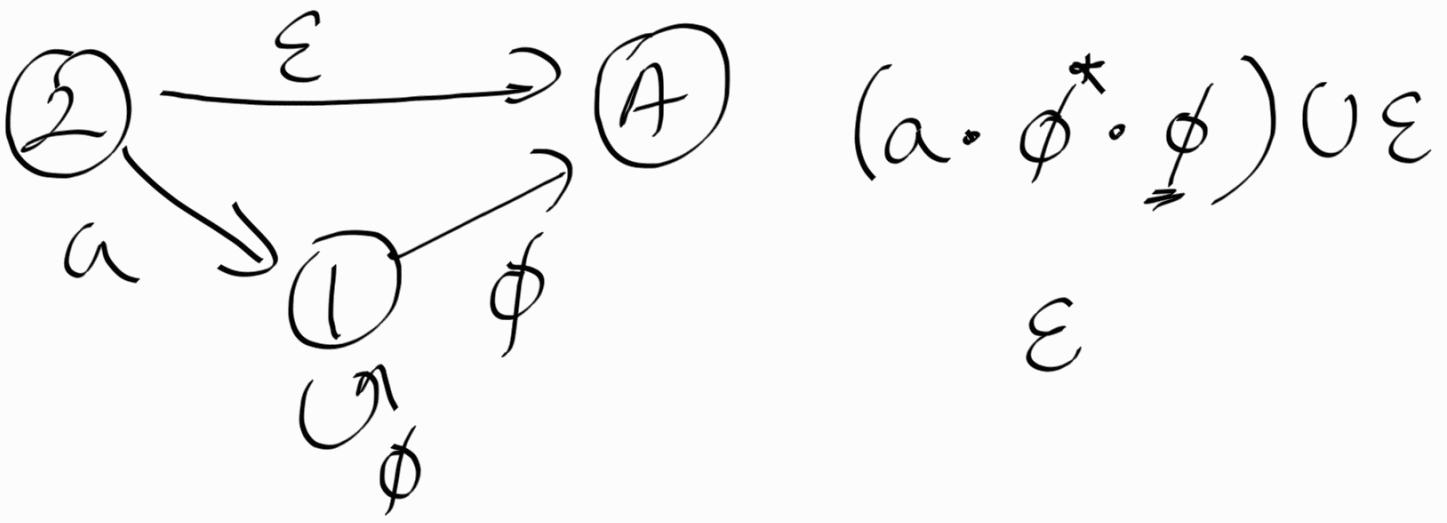
$$② \xrightarrow{(a \cdot \phi^* \cdot b) \cup \phi} ③$$

$= a b$

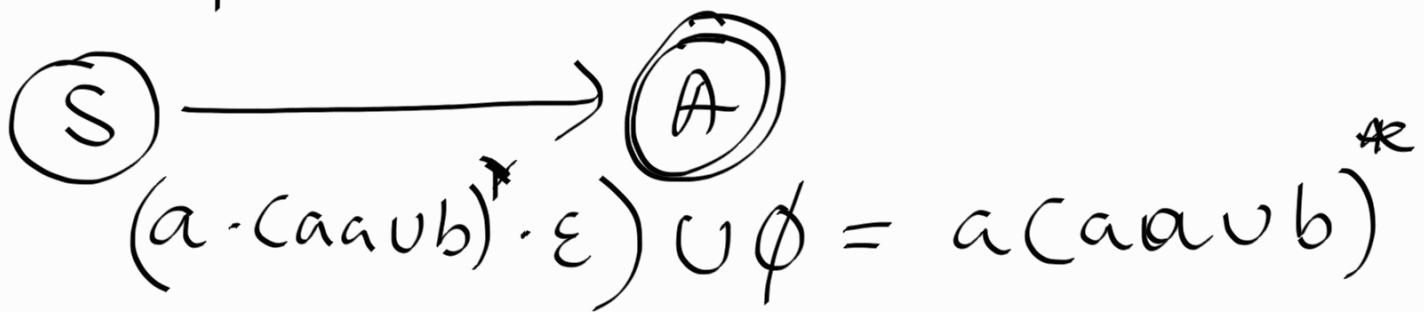


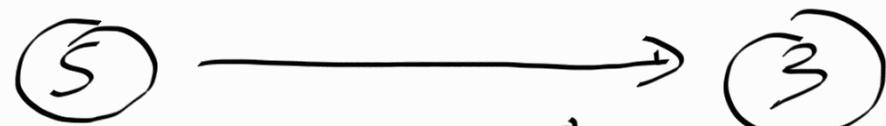
$$③ \xrightarrow{(l \cdot \phi^* \cdot a) \cup a} ②$$

$= (ba \cup a)$



nip 2





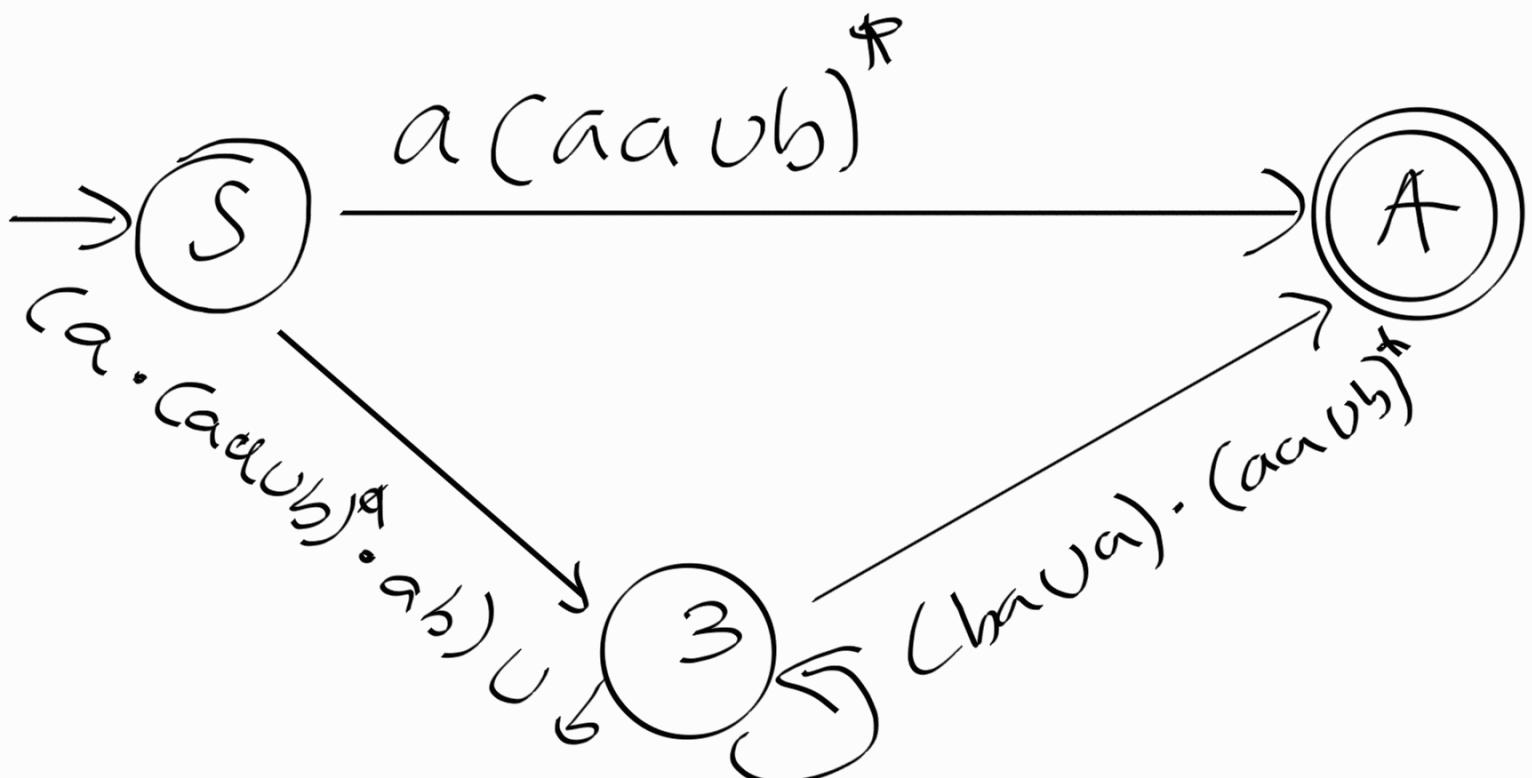
$$(a \circ (aa \cup b)^* \cdot ab) \cup b$$



$$((ba \cup a) \circ (aa \cup b)^* \cdot ab) \cup b \cup b$$



$$((ba \cup a) \circ (aa \cup b)^* \cdot \epsilon) \cup \emptyset$$



$$((ba \cup a) \cdot (aa \cup b)^*, ab) \cup bb$$

After this rip ③

Whole lot of proof
just to show

NFA = DFA = Regular
Expressions.