02/07/2025

What did we learn so far?

DFA - Deterministic finite Automators

- Limited memory
- Deterministic behaviour

NFA - Non deterministic finite Automata

- Limited memory
- Non deterministic behaviour.

- However NFA = DFA
- Any language that is recognized by a DFA/NFA is called a Regular Language

Regular Expressions

- we can use regular operations to build up expressions that describes languages.

  - regular operations
    - Union ($\cup$)
    - concatenation ($\circ$)
    - Kleene star * $(\{0\} \cup \{1\}) \circ \{0\}^*$

Ex! $(0 \cup 1) \circ 0^*$

↳ This expression describes a language

It basically describes strings in the language where starting with either 0 or 1 followed by zero or more 0s

$$(0 \cup 1)0^* \equiv (\{0\} \cup \{1\}) \circ \{0\}^*$$

star

union

concatenation

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

alphabet.

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

Σ ← using Σ as a regular expression

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

Σ^* describes the language consisting of all strings over the alaphabet.

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

$$\Sigma^* = (0 \cup 1 \cup 2)^* = \{\varepsilon, 0, 1, 2, 01, 000 \ldots\}$$

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

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

$5 + 3 \times 2 + 6$

$5 + 6 + 6$

$11 + 6$

17

Def

Regular Expressions

R is a regular expression, if R is:

1. a, for some $a \in \Sigma$ ← $\{a\}$
2. ε ← $\{\varepsilon\}$
3. $\emptyset$ ← $\{\}$

} these are 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 expressions.

Ex: $R \cup \phi = R$ ✓

e.g $R = \{1\}$

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

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

$R \cup \{\varepsilon\} \neq R$

e.g $R = \{1\}$

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

$L(R \cup \{\varepsilon\}) =$

$\{1, \varepsilon\}$

Remember regular expression describes a language

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

simple exercise: Let $\Sigma = \{0, 1\}$
what are the languages described by following regular expressions?

1. $(01)^* = (\{01\})^*$
2. $1^* \circ \varepsilon = 1^*$
3. $0^* \circ \phi = \{0\}^* \circ \phi = \{0\}^* \circ \{\} = \phi$
4. $(01)^* \cup \varepsilon = \{\varepsilon, (01)^*\}$
5. $(0 \cup \varepsilon) 1^* = 0^* 1^* \cup 1^*$

$01^* \neq (01)^*$

Regular Expressions = Regular Languages.

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

Theorem 1.54

A language is regular iff some regular expression describes it.

"If part"

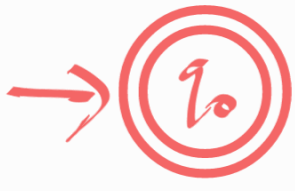
Lemma 1.55 If a Language is described by a regular expression, then it is regular.

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

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

NFA: 

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



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

4. $R = R_1 \cup R_2$
5. $R = R_1 \circ R_2$
6. $R = R_1^*$

} use the already created NFA's of R_1, R_2 then create larger NFA's. follow Thm 1.45, 1.47, 1.49

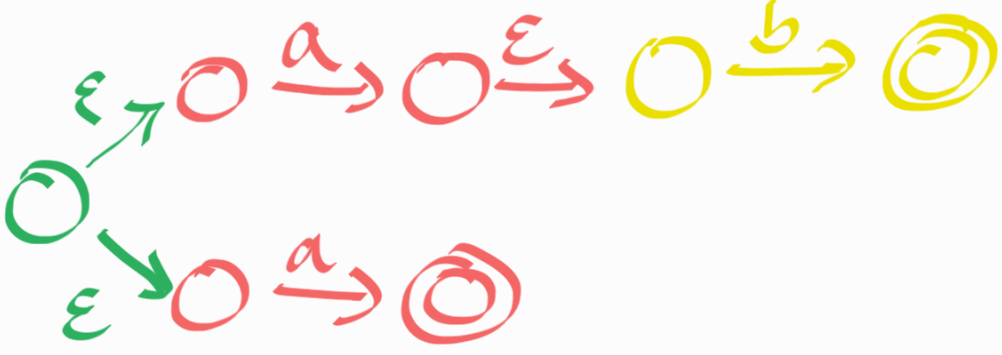
Ex: $(a \circ b \cup a)^*$

a

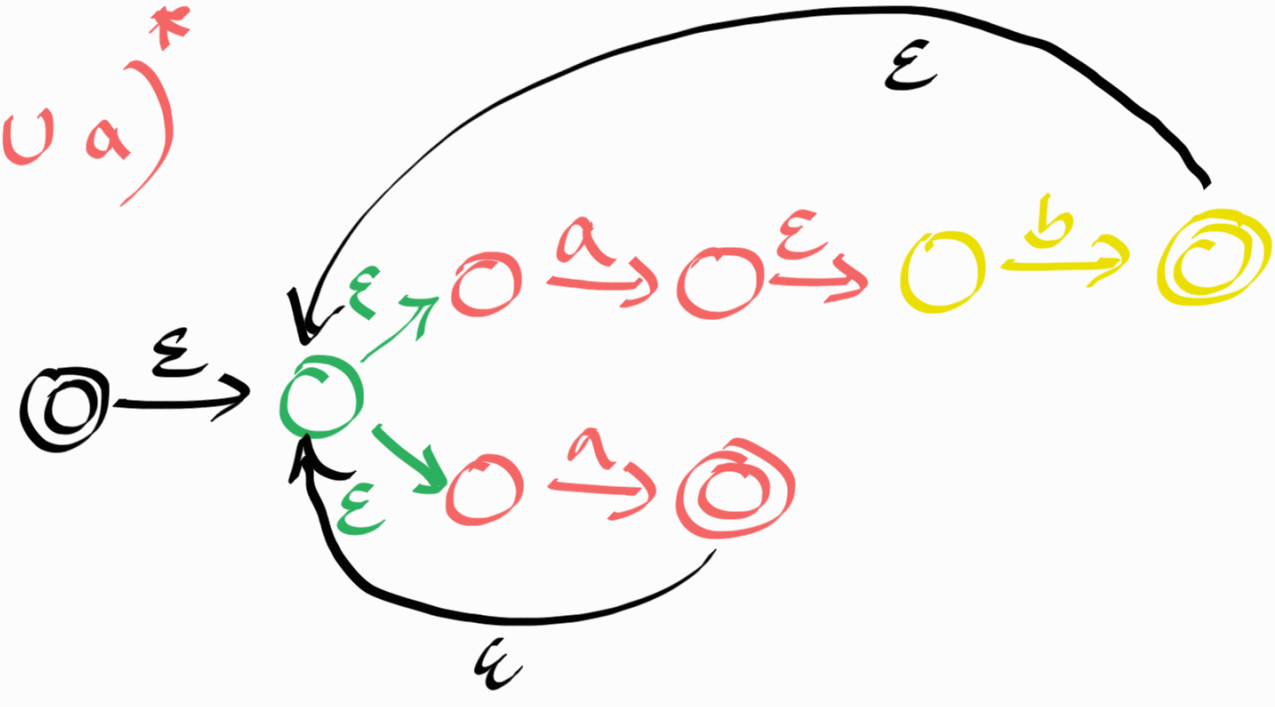
b

$a \circ b$

$a \circ b \cup a$



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



"Only if" part

Lemma 1.60

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

Idea!

step 1: DFA ⟶ GNFA

step 2: GNFA ⟶ regular expression

Generalized NFA are simply Nondeterministic finite automatons wherein the transition arrows may have any regular expressions as labels, instead of only one symbol or ε.

we want the DFA to convert to a GNFA of following 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 and accept state, one arrow goes from every state to every other state and also from each state to itself.

Example GNFA

Note about transitions.

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

1. Add a new start state and add a ε transition 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.

Converting a DFA to a GNFA is easy.

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.

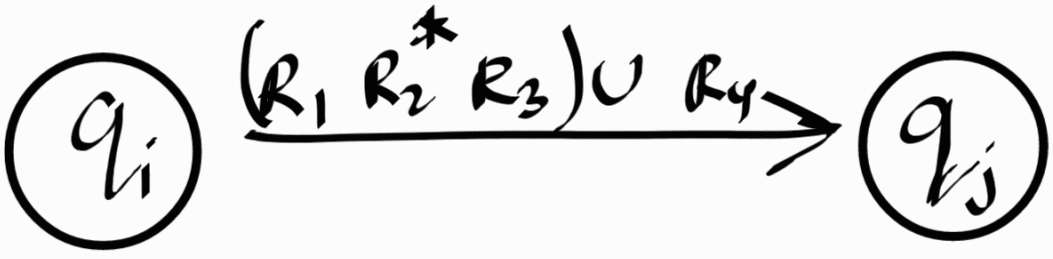
- Starting state.
- Accept state.
- arrow between these 2 states with the final regular expression.

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

We basically rip out a state and repair the damage, so that the same language is recognized.

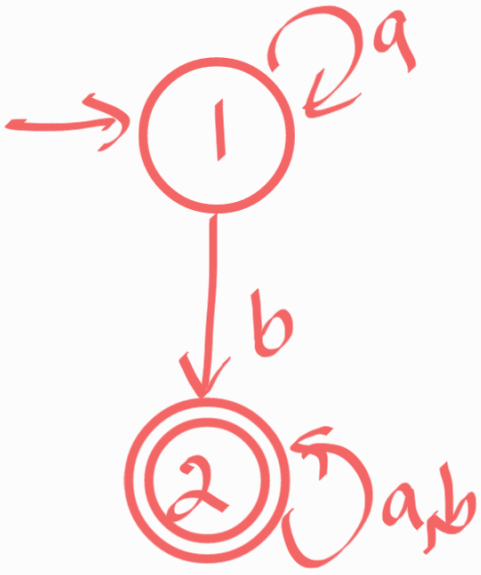
We can rip any state except Start and accept state.

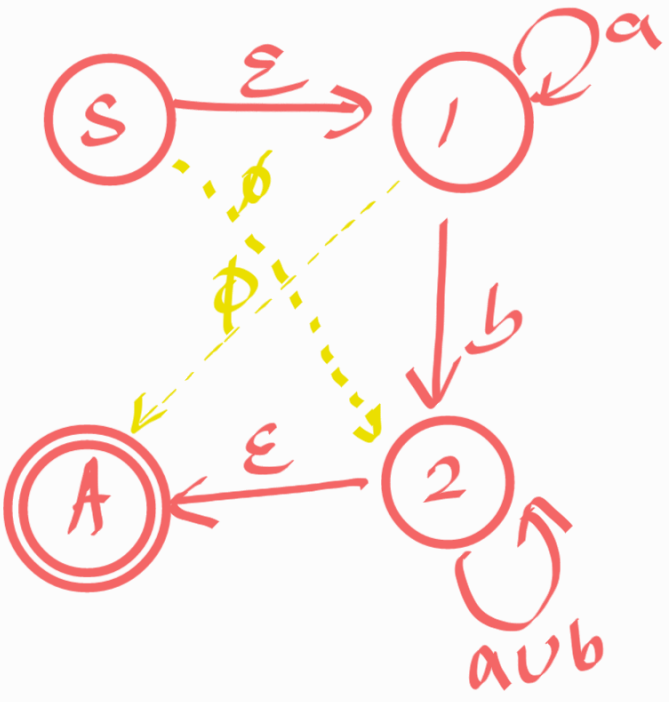
before

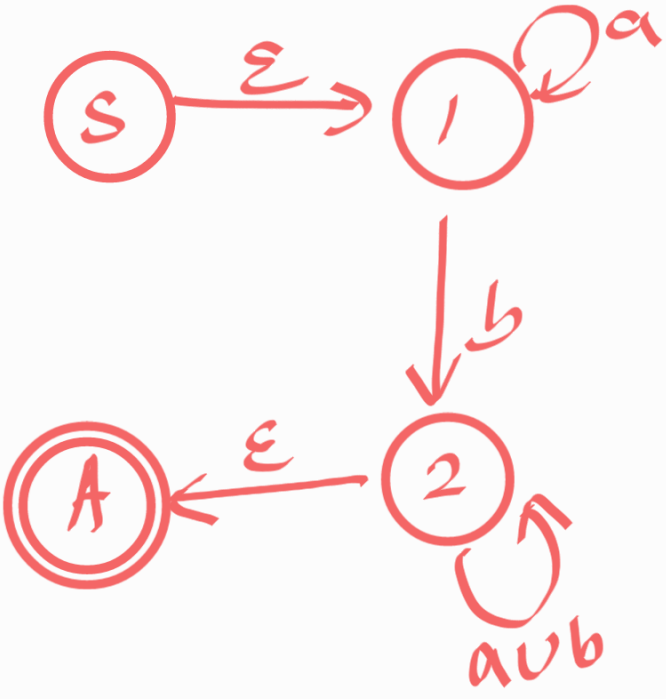


We make this change for each arrow going from any state q_i to any state q_j including the case where $q_i = q_j$

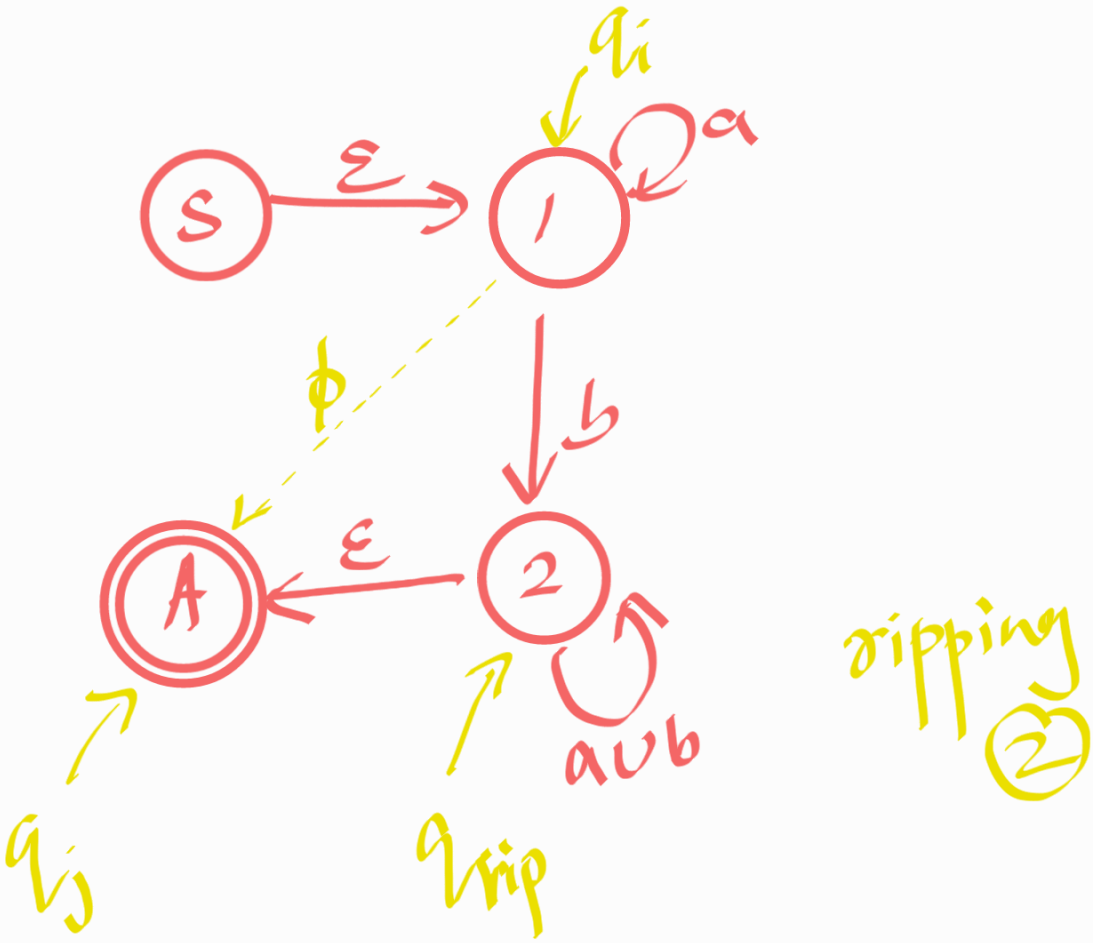
Example: DFA → GNFA → Regular Expression.

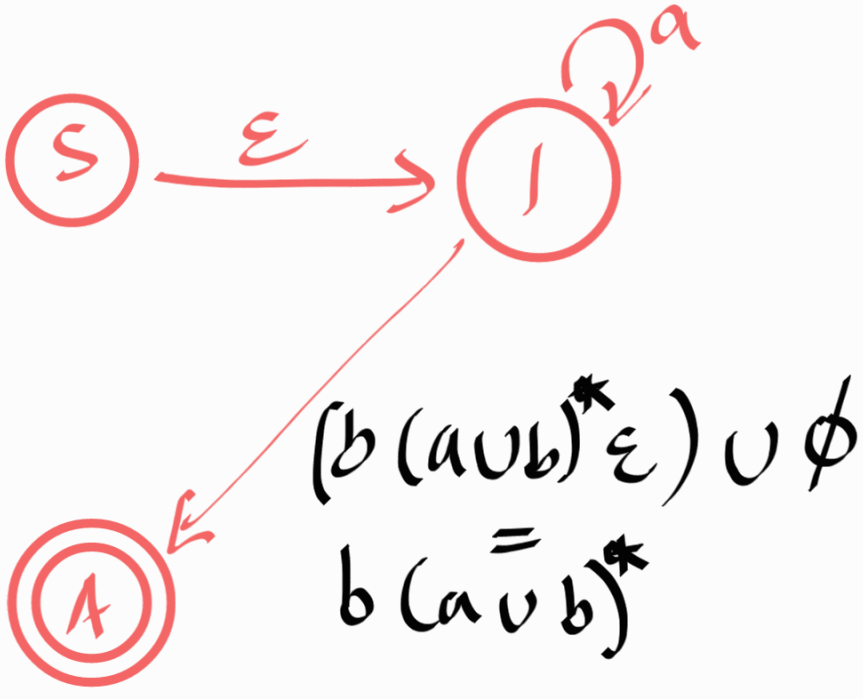






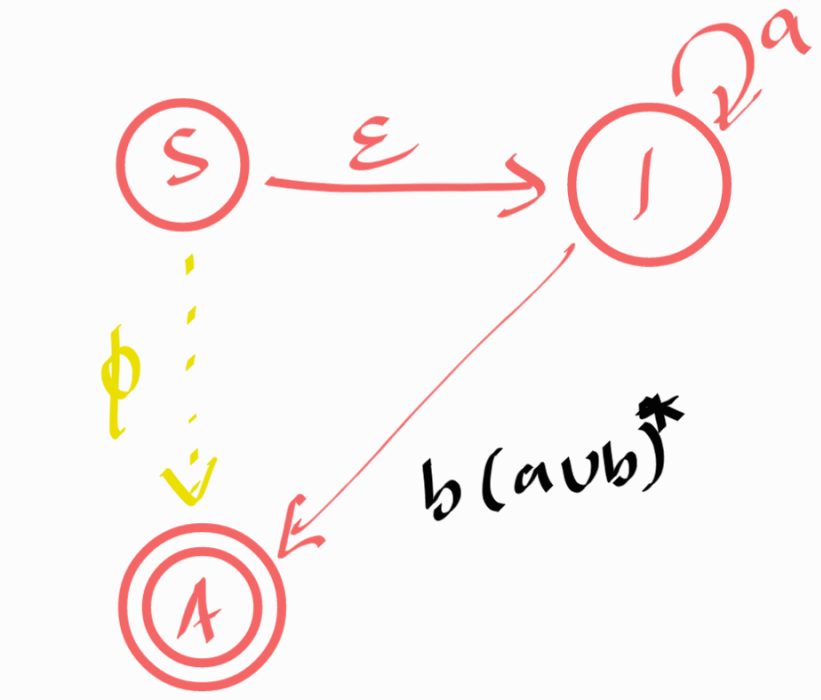
sometimes we do not include unnecessary ϕ transitions, since it clutters the graph.



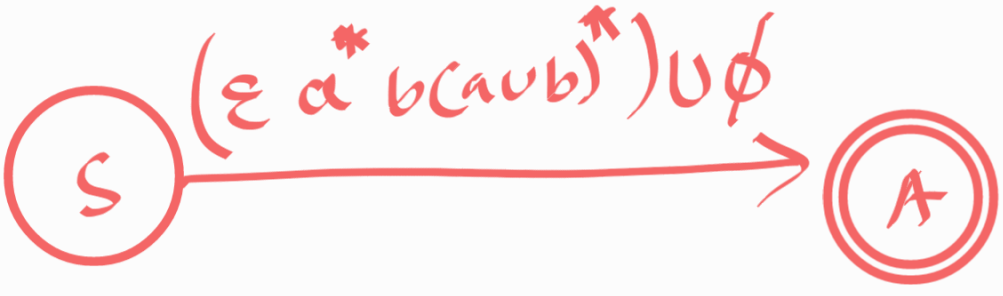


$$(b(a \cup b)^* \varepsilon) \cup \phi$$

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



ripping (1)



$a^* b(aUb)^*$

This is the final regular expression.