What did we learn so far?

- DFA: Deterministic finite automaton
  - Limited memory
  - Deterministic behaviour.
- NFA: Nondeterministic finite automatons
  - 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 ($*$)

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

$(\{0,1\}) \circ \{0\}^*$

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

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

$$\Sigma^* = (0 \cup 1 \cup 2)^* = \{\varepsilon, 0, 1, 2, 00, 000, 01, 010, \ldots\ldots\}$$

In regular expressions star operation 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, for some $a \in \Sigma$ ← $\{a\}$
2. ε ← $\{\varepsilon\}$
3. ϕ ← $\{\}$

} 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 expression.

Ex: $R \cup \phi = R$ $\quad \Sigma = \{0,1\}$

$R = 1$

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

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

$R \cup \varepsilon \neq R$ (not always)

$R = 1$

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

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

Remember: regular expressions describe a language

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

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

Simple exercise; let $\Sigma = \{0, 1\}$

what are the languages described by the following regular expressions?

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

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

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

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

5. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$

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

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

Regular Expressions $\equiv$ 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. $R = \varepsilon$, $L(R) = \{\varepsilon\}$

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

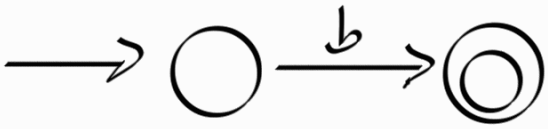
4. $R = R_1 \cup R_2$
5. $R = R_1 \circ R_2$
6. $R = R_1^*$

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

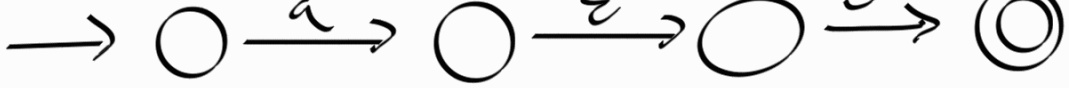
Ex: $(a \circ b \cup a)^*$ $\quad \Sigma = \{a, b\}$

a

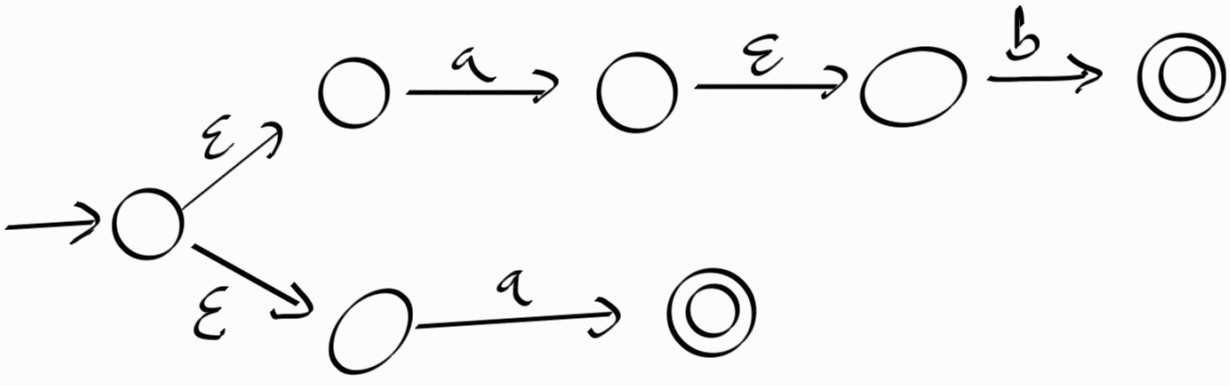
b



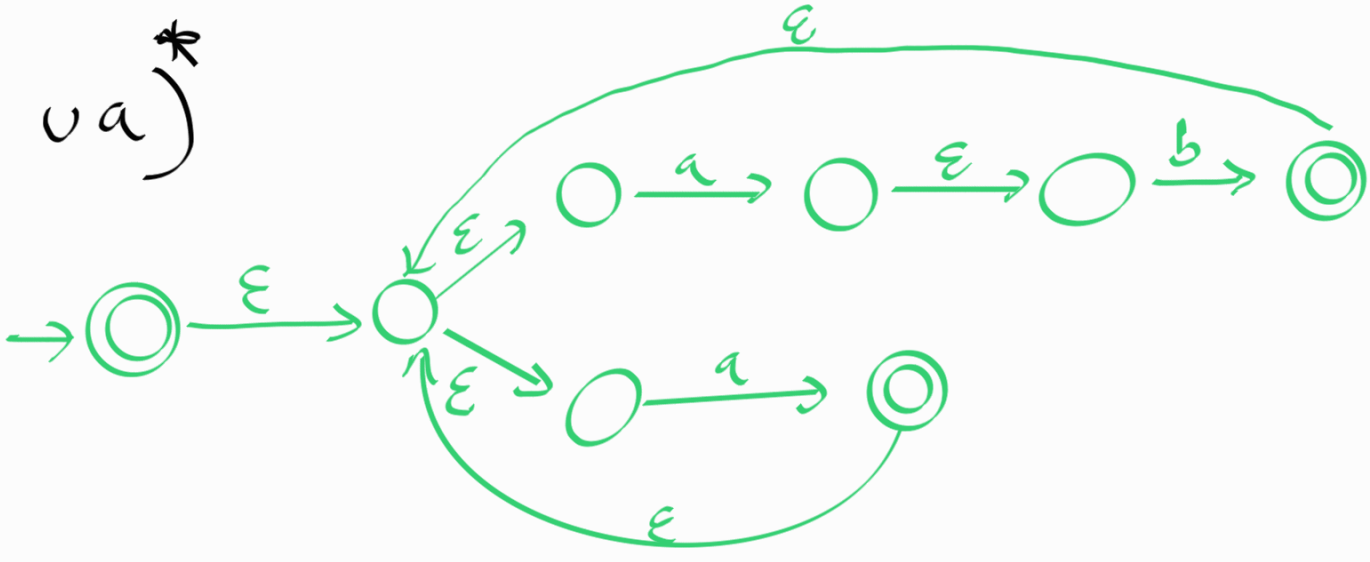
$a \circ b$



$a \circ b \cup a$



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



"only if" part ⇒

Lemma 1.60

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

Idea!

new type of finite automaton

Step 1: DFA ⟶ GNFA

Step 2: GNFA ⟶ regular expression

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.

GNFA follows a special set of conditions.

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

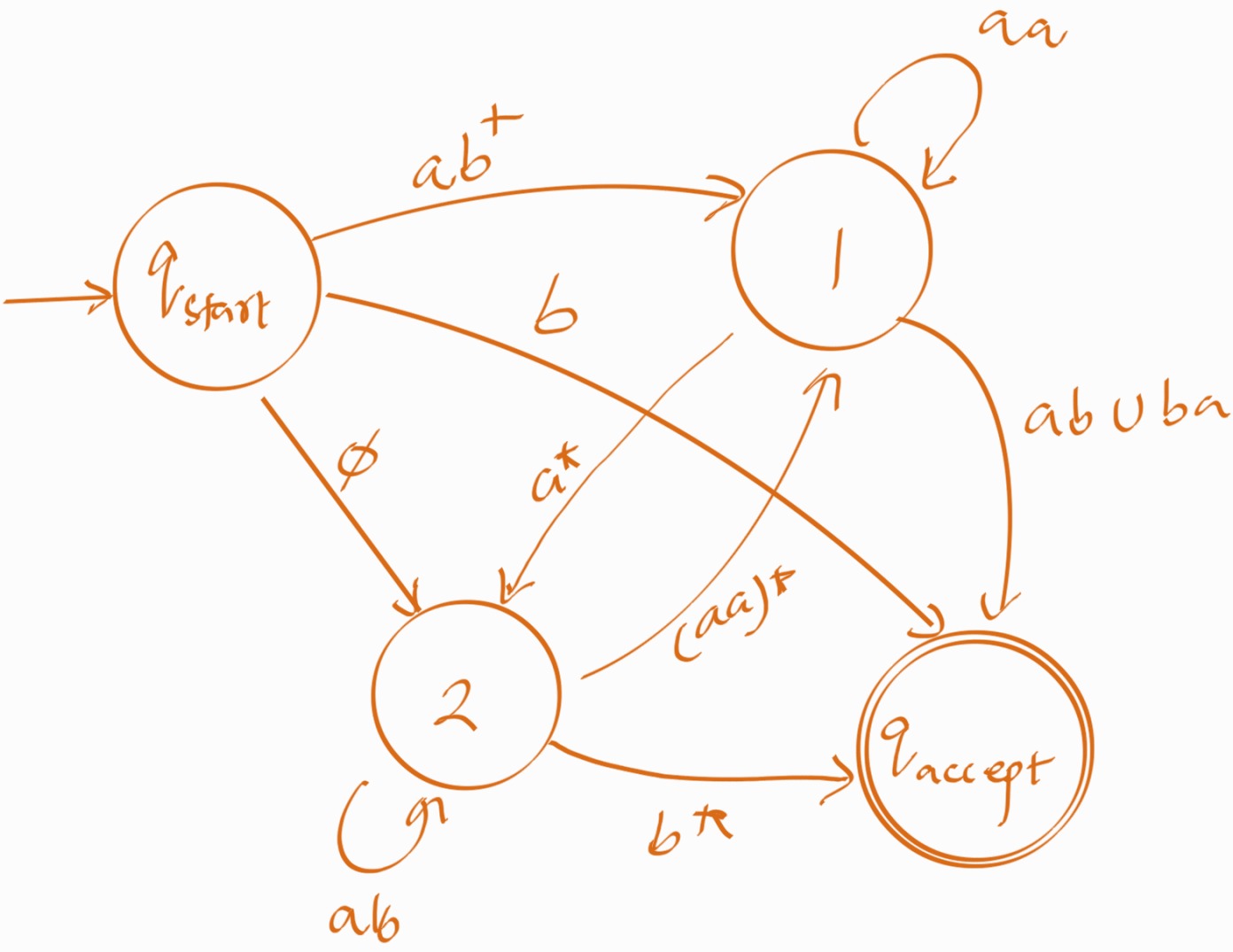
* Start state has transition arrows going to every other state, but no arrows coming in from any state.

* 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.

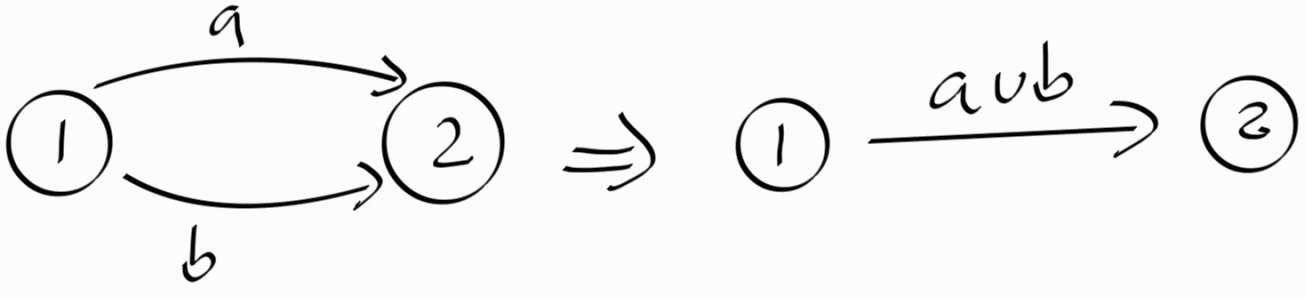
Let's look at how we can transform a DFA to a GNFA.

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.