Regular operations can be used to study whether languages are regular or not.

That is, these operations can help us to check if for a given language, whether there are finite automata to recognize it or not.

We mainly consider three operations.

Assume $A, B$ are given languages.

- union

$$A \cup B$$
concatenation

\[ A \circ B = \{ xy \mid x \in A, y \in B \} \]

star:

\[ A^* = \{ x_1 \cdots x_k \mid k \geq 0, x_i \in A \} \]
If $k = 0$, what do we mean $x_1 \cdots x_k$?

We define

$\epsilon : \text{empty string}$

in this situation.

Thus

$\epsilon \in A^*$
Example

\[ \Sigma = \{a, \ldots, z\} \]
\[ A = \{\text{good, bad}\} \]
\[ B = \{\text{boy, girl}\} \]
\[ A \circ B = \{\text{goodboy, \ldots}\} \]
\[ A^* : \{\epsilon, \text{good, bad, goodgood, \ldots}\} \]

We say an operation \( R \) is **closed** if the following property holds

if \( x \in A, y \in A \), then \( xRy \in A \)
Example: $\mathbb{N} = \{1, 2, \ldots\}$ is closed under multiplication

Th 1.25: regular languages are closed under the union operation

\[ A_1, A_2 \text{ are regular languages} \]
\[ \Rightarrow A_1 \cup A_2 \text{ is regular} \]

Proof
Assume we are given two automata

\[ M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \]
\[ M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \]

- **Question**: you want to think about why we can consider the same \( \Sigma \)
- **Idea**: we construct a parallel machine to run two machines simultaneously
Definition of our new machine

\[ M = (Q, \Sigma, \delta, q_0, F) \]
\[ Q = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\} \]
\[ \delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \]
\[ q_0 = (q_1, q_2) \]
\[ F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\} \]

Example: combining

\{w \mid w \text{ has an odd } \# \text{ 1's}\} \cup \{w \mid w \text{ has an odd } \# \text{ 0's}\}
Is this proof rigorously enough?
A formal proof should be done by induction. But we don’t provide it here

Th 1.26: closed under concatenation

If $A, B$ are regular, then $A \circ B$ is regular

But the proof is not easy

It’s unclear where to break the input

To easily do the proof, we introduce a new technique called **nondeterminism**