Automata theory and its applications
Notes

toshinari tong

March 10, 2023

1 Finite Automata

1.1 Finite Automata

omega: $\omega = \{0, 1, 2, ...\}$
alphabet: $\Sigma = \{a_1, ..., a_n\}$
input: $u = \sigma_1\sigma_2...\sigma_m$ where $\sigma_i \in \Sigma$
empty input: λ
finite words: $\Sigma^* = \{\sigma_1\sigma_2...\sigma_m | \sigma_1, \sigma_2, ..., \sigma_m \in \Sigma, m \in \omega\}$
Σ-language: any subset of Σ^*
set of states: $S = \{s_0, ..., s_k\}$
sequentiality: the next state of the system is one of the states determined by the pair (s, σ)

nondeterministic finite automaton: $\mathcal{A} = (S, T, I, F)$ where
S is a finite nonempty set called the set of states
$T \subseteq S \times \Sigma \times S$ is a nonempty set called the transition table
$I \subseteq S$ is called the set of initial states
$F \subseteq S$ is called the set of final states
Finite automata are the simplest mathematical abstractions of **discrete sequential systems**.
$T(s, \sigma) = \{s' \in S | (s, \sigma, s') \in T\}$
deterministic automaton: $\mathcal{A}$ is deterministic if $|I| = 1$ and $\forall s \in S, \forall \sigma \in \Sigma, |T(s, \sigma)| = 1$.

computation / run: a sequence of states of length $m + 1$ from given input of length m:
stage 1: $\mathcal{A}$ choose an initial state $s_1 \in I$
stage n: suppose sequence $s_1 s_2 ... s_{n-1}$ is produced. the automaton chooses an $s_n \in S$ s.t. $(s_{n-1}, \sigma_{n-1}, s_n) \in T$
If $\mathcal{A}$ is deterministic, then it has a computation on any input and this computation is unique.
successful computation / $\mathcal{A}$ accepts an input: $s_{m+1} \in F$

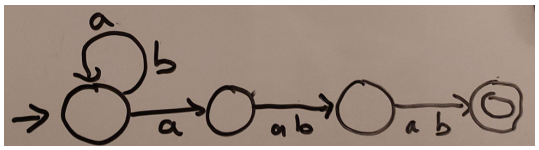
transition diagram: this automaton accepts words $(ab)^n aa$

accessibility: $G_0(s) = \{s\}$, $G_1(s) = \{s_1 | \exists \sigma \in \Sigma, (s, \sigma, s_1) \in T\} \cup G_0(s)$,$G_{n+1}(s) = G_n(s) \cup \bigcup_{s_1 \in G_n(s)} G_1(s_1)$
accessibility operator: $G(s) = \bigcup_n G_n(s)$

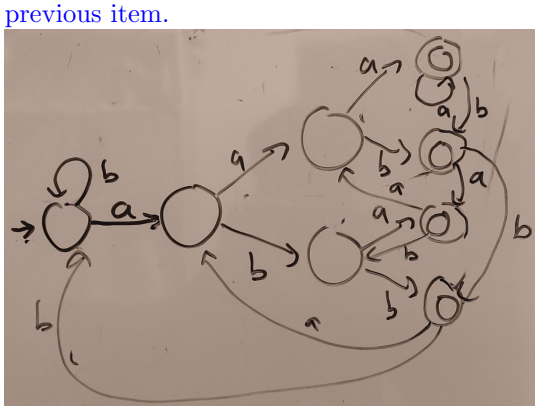
finite automaton (FA) recognizable language: $L \subseteq \Sigma^*$ s.t. $\exists \mathcal{A}, L = L(\mathcal{A}) = \{w \in \Sigma^* | \mathcal{A} \text{ accepts } w\}$

EXERCISE 2.2.7

1. Give an example of a nfa with 4 states over the alphabet $\{a, b\}$ s.t. $\mathcal{A}$ recognizes language L containing exactly those w that have a at the 3rd position from the right of w.



2. Give an example of a deterministic automaton that recognizes the languatge L defined in the previous item.



1.2 Closure Properties

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be automata. Then there exists an automaton $\mathcal{A}$ s.t. $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.
First suppose $S_1 \cap S_2 = \varnothing$. If they have common states, we can just rename elements of S_2 and create a new copy of $\mathcal{A}_2$. Define $\mathcal{A} = (S, T, I, F)$ as follows:

1. $S = S_1 \cup S_2$.
2. $T = T_1 \cup T_2$.
3. $I = I_1 \cup I_2$.
4. $F = F_1 \cup F_2$.

Denote $\mathcal{A} = \mathcal{A}_1 \oplus \mathcal{A}_2$.

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be automata. Then there exists an automaton $\mathcal{A}$ s.t. $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

1. $S = S_1 \times S_2$.
2. $T = \{((s_1, s_2), \sigma, (t_1, t_2)) \mid (s_1, \sigma, t_1) \in T_1, (s_2, \sigma, t_2) \in T_2\}$.
3. $I = I_1 \times I_2$.
4. $F = F_1 \times F_2$.

Denote $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$.

EXERCISE 2.3.2

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be finite automata. Construct an automaton which accepts the union $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ and whose set of states is $S_1 \times S_2$ and the transition table is the one for $\mathcal{A}_1 \times \mathcal{A}_2$.

$F = \{(f_1, x_2) \mid f_1 \in F_1, x_2 \in S_2\} \cup \{(x_1, f_2) \mid f_2 \in F_2, x_1 \in S_1\}$

EXERCISE 2.3.3

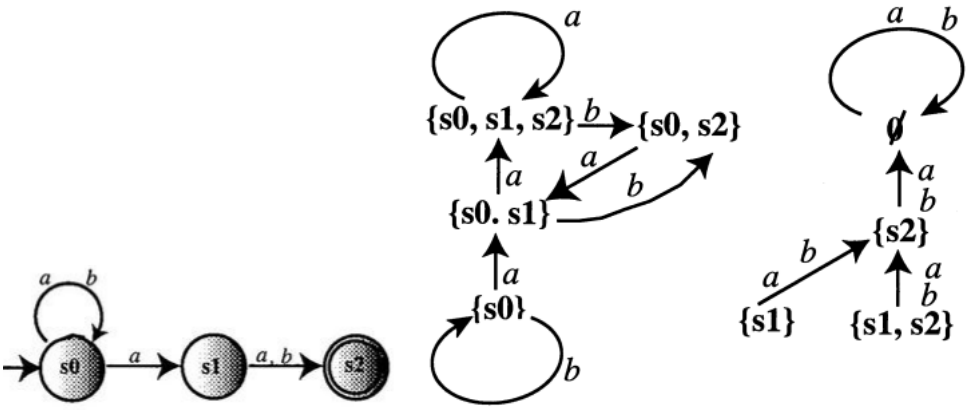
Consider the 2 automata. Construct the automaton that accepts the intersection.

If $\mathcal{A}$ is a deterministic automaton, then $\mathcal{A}^c = (S, T, I, S \backslash F)$ accepts the complement of $L(\mathcal{A})$, that is $L(\mathcal{A}^c) = \Sigma^* \backslash L(\mathcal{A})$.

However if the original automaton is not deterministic, then this doesn't work: Even if there is a computation $s_1, s_2, ..., s_{m+1}$ s.t. $s_{m+1} \notin F$, this does not guarantee that u is not accepted by $\mathcal{A}$. For example, maybe choosing some other initial state or some other path may result in a successful run. In other words, u is not accepted iff all computations of $\mathcal{A}$ on u are unsuccessful.

The **power automaton** $\mathcal{A}^d = (S^d, T^d, I^d, F^d)$ is defined as follows:

1. $S^d = \mathcal{P}(S)$
2. $T^d = \{(S_1, \sigma, S_2) \mid S_2 = \{s' \mid \exists s \in S_1 \text{ s.t. } (s, \sigma, s') \in T\}\}$
3. $I^d = \{I\}$
4. $F^d = \{G \subseteq S \mid ; G \cap F \neq \varnothing\}$ if $F \neq \varnothing$.



Let $\mathcal{A}$ be a nfa. Then the power automaton $\mathcal{A}^d$ is deterministic and $L(\mathcal{A}) = L(\mathcal{A}^d)$.
If $u = \sigma_1\sigma_2...\sigma_m$ is accepted by $\mathcal{A}$, then by definition of F^d it is accepted by $\mathcal{A}^d$.
If u is accepted by $\mathcal{A}^d$, let $S_1, S_2, ..., S_{m+1}$ be a computation of $\mathcal{A}^d$ on u; then there exists an $s_{m+1} \in S_{m+1} \cap F$. By definition of T^d, there exists a sequence $s_1, ..., s_{m+1}$ which is a succesful computation of $\mathcal{A}$ on u.

EXERCISE 2.3.5

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be deterministic finite automata. Consider the automata $\mathcal{B} = \mathcal{A}_1 \oplus \mathcal{A}_1$ and $\mathcal{C} = \mathcal{A}_1 \times \mathcal{A}_1$. Construct an automaton recognizing the complements of $L(\mathcal{B})$ and $L(\mathcal{C})$ without using the subset construction.

$\mathcal{B}^c = \mathcal{A}_1^c \times \mathcal{A}_2^c, \mathcal{C} = \mathcal{A}_1^c \oplus \mathcal{A}_2^c$

For every finite automaton there exists an automaton such that it accepts the complement of the language.

EXERCISE 2.3.6

Exercise 2.3.6 *Take a nondeterministic finite automaton $\mathcal{A} = (S, I, T, F)$. Construct a new set S' by stages as follows.*

Stage 0. *Set $S_0 = \{I\}$.*

Stage $n + 1$. *Suppose that S_n has been built and is $\{X_1, \ldots, X_k\}$. Set*

$$S_{n+1} = S_n \bigcup \{T'(X_i, \sigma) \mid \sigma \in \Sigma,\ 1 \leq i \leq k\}.$$

If $S_{n+1} = S_n$, then let $S' = S_n$. Otherwise go to the next stage.
Prove the following two facts.

1. *There exists an n such that $S_{n+1} = S_n$.*
2. *Set $F' = \{X \mid X \in S', X \bigcap F \neq \emptyset\}$. Define $\mathcal{A}' = (S', \{I\}, T', F')$. Show that $\mathcal{A}'$ is a deterministic automaton that accepts $L(\mathcal{A})$.*

S is finite and $|S_n|$ is always increasing so there must be a point where $S_{n+1} = S_n$.

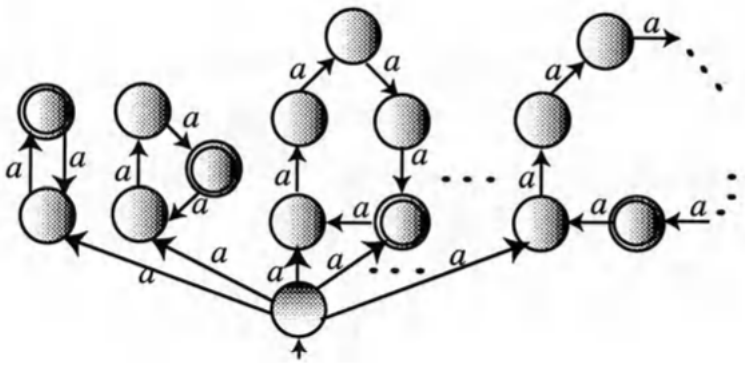
For every natural number $n \leq 1$ there exists an automaton with exactly $n + 1$ states such that no deterministic automaton with less than 2^n states recognizes the complement of $L(\mathcal{A})$.
Consider $\Sigma = \{a, b\}$ and $L = \{uav \mid u, v \in \Sigma^*, |v| = n - 1\}$. Let $\mathcal{A} = (S, T, I, F)$ be the following:

1. $S = \{s_0, ...s_n\}$
2. $T = \{(s_0, \sigma, s_0), (s_0, a, s_1), (s_i, \sigma, s_{i+1})\}$
3. $I = \{s_0\}$
4. $F = \{s_n\}$

$\mathcal{A}$ recognizes L. Now suppose there exists a deterministic automaton $\mathcal{B}$ with less than 2^n states that accept $\Sigma^* \backslash L$. Then there must be uav_1 and ubv_2 of length n that transforms the initial state to the same state. Now $uav_1u \in L$ and $ubv_2u \notin L$. However the 2 words transforms the initial state to the same state. So no $\mathcal{B}$ exists.

EXERCISE 2.3.7

Exercise 2.3.7 *For every natural number n consider the automaton $\mathcal{A}_n$ pictured in Figure 2.17. This automaton has n disjoint cycles of length 2, 3, 5, etc., that is, the cycle i from the left is of length p_i, where p_i is the ith prime number. Prove that no deterministic automaton with less than $p_1 \cdot 2 \cdot \cdots \cdot p_n$ states accepts the language recognized by $\mathcal{A}_n$.*



Let $\mathcal{B}$ be a deterministic automaton with less than $p_1p_2...p_n$ states that recognizes $L(\mathcal{A}_n)$. Then there must be a cycle of length $P < p_1p_2...p_n$ in $\mathcal{B}$ if it follows the instruction $aaa...$. Therefore if instruction $aaa...a$ of length x is not accepted, then $x+P$ is also not accepted in $\mathcal{B}$. $x \not\equiv 0 \pmod{p_i}$ for all i, and $x+P \not\equiv 0 \pmod{p_i}$ for all i. If P is not divisible by p_i, then there must exist a y where $x + yP \equiv 0 \pmod{p_i}$ because the set $\{x \pmod{p_i}, x + P \pmod{p_i}, x + 2P \pmod{p_i}, ...\}$ visits all elements. Therefore P is divisible by all p_i, but $P < p_1p_2...p_n$. So no $\mathcal{B}$ exist.

Suppose L_1 and L_2 are languages.

$$pref(L_1, L_2) = \{u \mid u \in L_1 \text{ and for some string } w, uw \in L_2\}$$

Let L_1 and L_2 be FA recognizable languages. Then $pref(L_1, L_2)$ is FA recognizable.
$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$, except that $F = \{(f, s) \mid f \in F_1 \text{ and } G(s) \cap F_2 \neq \varnothing\}$.

$$suff(L_1, L_2) = \{u \mid u \in L_1 \text{ and for some string } w, wu \in L_2\}$$

EXERCISE 2.3.8

Let L_1 and L_2 be FA recognizable languages. Then $suff(L_1, L_2)$ is FA recognizable.
$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$, except that $I = \{(i, s) \mid i \in I_1 \text{ and } s \in G(I_2)\}$.

EXERCISE 2.3.9

Exercise 2.3.9 *Let Σ be an alphabet. A* **nondeterministic finite automaton with silent moves** *is a quadruple $\mathcal{A} = (S, I, T, F)$, where S, I, and F are exactly the same as in the definition of finite automaton (see Definition 2.2.2) and T is a union of T_1 and T_2 such that*

1. $T_1 \subset S \times \Sigma \times S$,
2. $T_2 \subset S \times S$.

Thus, if $\mathcal{A}$ is in state s and $(s, s') \in T$, then $\mathcal{A}$ is allowed to silently move from s to s' without yet reading the next input. Do the following:

1. *Define the notions of computation, acceptance, and recognizable languages for nondeterministic automata with silent moves.*
2. *Prove then that for any $L \subset \Sigma^\star$, the language L is FA recognizable if and only if L is recognized by an NFA with silent moves. (Hint: Use the accessibility operator).*

If L is FA recognizable, then it is obviously recognized by an NFA with silent moves because FA is a subset of NFA with silent moves.

If L is recognized by an NFA with silent moves $\mathcal{A} = (S, T, I, F)$, then it is recognied by FA $\mathcal{B} = (S, T_B, I_B, F)$ where

$$T_B = \bigcup_{(s,\sigma,s') \in T_1} \{(s, \sigma, g) \mid g \in G_{T_2}(s')\} \text{ and } I_B = \bigcup_{i \in I} G_{T_2}(i)$$

Suppose our alphabet is of the form

$$\Sigma_1 \times \Sigma_2 = \{(a, b) \mid a \in \Sigma_1, b \in \Sigma_2\}$$

Any word of this alphabet is a sequence of pairs, which can be identified with 2 words each from Σ_1^* and Σ_2^*.

$$pr_1(L) = \{u \in \Sigma_1^* \mid (u, v) \in L \text{ for some } v \in \Sigma_2^*\}$$
$$pr_2(L) = \{u \in \Sigma_2^* \mid (v, u) \in L \text{ for some } v \in \Sigma_1^*\}$$

If language L over $\Sigma = \Sigma_1 \times \Sigma_2$ is FA recognizable, then so are the projections $pr_1(L)$ and $pr_2(L)$.
$\mathcal{A}_1 = \mathcal{A}$, except $S_1 = \{(s, \sigma_1, s') \mid (s, (\sigma_1, \sigma_2), s') \in T\}$

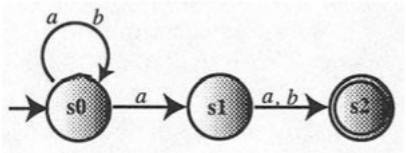
1.3 The Myhill-Nerode Theorem

2 inputs u and v are **in relation** $\sim$ if for all states s and s' from S, u transforms s to $s' \iff v$ transforms s to s'.

Let $\mathcal{A}$ be an automaton. The relation $\sim$ is an equivalence relation on Σ^*. Moreover the index of the equivalence relation $\sim$ is finite.
It is reflexive, symmetric and transitive.
Take any input u and define $f_u : S \to \mathcal{P}(S)$ to be $f_u(s) = \{s' \mid \text{input } u \text{ transforms } s \text{ to } s'\}$.
For all u and v they are $\sim$-equivalent iff $f_u = f_v$. Since the number of functions from S to $\mathcal{P}(S)$ is finite, the index of $\sim$ is finite.

EXERCISE 2.4.2

Exercise 2.4.2 *Consider the automaton pictured in Figure 2.18. Define the equivalence classes of the relation $\sim$ for this automaton. What is the index of the equivalence relation $\sim$?*



$\{\varnothing\}, \{b, bb, bbb, ...\}, \{a, ba, bba, ...\}, \{\text{remaining that ends with } b\}, \{\text{remaining that ends with } a\}$

right invariance: if u is $\sim$-equivalent to v, then uw is $\sim$-equivalent to vw.

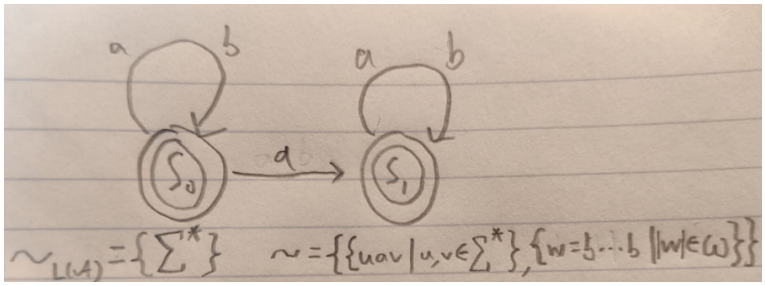
Therefore, if $u \sim w$, then $uw \in L(\mathcal{A})$ iff $vw \in L(\mathcal{A})$.

Myhill-Nerode equivalent relations: Let L be a language. u and v are L-equivalent ($u \sim_L v$) if for all w, $uw \in L \iff vw \in L$.

Myhill-Nerode Theorem: A language L is FA recognizable iff $\sim_L$ is of finite index.
Suppose $L = L(\mathcal{A})$. By above, $\sim$ of $\mathcal{A}$ is of finite index. And if $u \sim v$ then $u \sim_L v$. Therefore every $\sim_L$ equivalence class is a union of $\sim$-equivalence classes. Hence $\sim_L$ is of finite index. To see that L is a union of $\sim_L$ equivalence classes note that if $u \in L$ and $v \sim_L u$, then $v \in L$ because $u\lambda, v\lambda \in L$.
Now suppose $\sim_L$ is an equivalence relation of finite index. Let $\mathcal{A} = (S, T, I, F)$ be the following:
$S = \{[u] \mid u \in \Sigma^*\}$
$T = \{([u], \sigma, [v]) \mid u\sigma \sim_L v\}$
$I = \{[\lambda]\}$
$F = \{[u] \mid u \in L\}$
$\mathcal{A}$ is a deterministic finite automaton. $u \in L \iff [u] \in F \iff u \in L(\mathcal{A})$

EXERCISE 2.4.5

Give an example of an automaton $\mathcal{A}$ for which $\sim_{L(\mathcal{A})} \neq \sim$.



A deterministic finite automaton $\mathcal{A}$ is **minimal** if the number of states of every deterministic automaton recognizing $L(\mathcal{A})$ is greater than or equal to the number of states of $\mathcal{A}$.

EXERCISE 2.4.7

Let L be a FA recognizable language. Show that $\mathcal{A}$ constructed in the proof of the Myhill-Nerode theorem is minimal.

Let $\mathcal{B}$ be a deterministic automaton with number of states less than number of equivalence classes of $\sim_L$. Then, there must exist u, v s.t. $[u] \neq [v]$ but the final transformed state is the same. This means that $\exists w\ uw \in L, vw \notin L$ but this is not possible since it is deterministic and transforms to the same final state (either both are accepted or both are not accepted).

We say that automata $\mathcal{A}, \mathcal{B}$ over Σ are **isomorphic** if there exists a bijective $h : S_A \to S_B$ s.t.:
$(s_1, \sigma, s_2) \in T_A \iff (h(s_1), \sigma, h(s_2)) \in T_B$
$s \in I_A \iff h(s) \in I_B$
$s \in F_A \iff h(s) \in F_B$

EXERCISE 2.4.8

Two minimal automata recognizing the same language are isomorphic.

$\mathcal{A}$ and $\mathcal{B}$ are isomorphic $\iff$ ($\forall u, v$ (u, v go to same state in $\mathcal{A} \iff u, v$ go to same state in $\mathcal{B}$) and $F_A = F_B$)
$\mathcal{A}$ and $\mathcal{B}$ are not isomorphic $\iff$ ($\exists u, v$ (u, v go to same state in $\mathcal{A} \iff u, v$ go to diff state in $\mathcal{B}$) or $F_A \neq F_B$)
Suppose there exists non-isomorphic minimal automata $\mathcal{A}$ and $\mathcal{B}$. Assume first condition is true. Since u, v go to the same state in $\mathcal{A}$, they are $\sim_L$-equivalent. This means 2 $\sim_L$-equivalent strings

go to different states in $\mathcal{B}$. Construct new automata $\mathcal{B}_2 = (S, T, I, F)$:
Denote the state u goes to as s_u and the state v goes to as s_v.
$S = S_B \backslash \{s_v\}$
$T = T_B$ but every arrow that points to s_v now points to s_u
$I = I_B$
$F = F_B$ without s_v if its in $(s_u \in F_B \iff s_v \in F_B)$
We have constructed an automaton that recognizes the same language with number of states less than that of the minimal automaton. Therefore first condition cannot hold. It is clear that if the first condition does not hold and $F_A \neq F_B$ then they dont recognize the same language. Therefore it is impossible that $\mathcal{A}$ and $\mathcal{B}$ are not isomorphic.

1.4 The Kleene Theorem

The **trivial languages** over Σ are $\varnothing, \{\lambda\}, \{a_1\}, ..., \{a_n\}$.

The concatenation of L_1 and L_2 is $L_1 \cdot L_2 = \{uv \mid u \in L_1, v \in L_2\}$. The concatenation operation makes the set of all languages a monoid, and is associative:

$$(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$$

Another operation is union: $L_1 + L_2 = L_1 \cup L_2$

The third operation is the **Kleene star**: $L^* = \{\lambda\} + L + (L \cdot L) + (L \cdot L \cdot L) + ... = L^0 + L^1 + L^2 + L^3 + ...$

That is why the notation Σ^* is the way it is.

Stage 0. R_0 is the class of trivial languages.
Stage $t+1$. Suppose $R_t = \{L_1, L_2, ..., L_k\}$ has been defined. Let R_{t+1} be

$$R_t \cup \{L_i^* \mid i = 1, ..., k\} \cup \{L_i \cdot L_j \mid i, j = 1, ..., k\} \cup \{L_1 + L_j \mid i, j = 1, ..., k\}$$

Define R:

$$R = \bigcup_{t \in \omega} R_t$$

A language $L \subseteq \Sigma^*$ is **regular** if L is in R.

EXERCISE 2.5.2

Let $\Sigma = \{a, b\}$. Show that the following languages are regular.
$\{ab^n \mid n \in \omega\}$ $\{a\} \cdot \{b\}^*$
$\{(ab)^{2n} \mid n \in \omega\}$ $(\{a\} \cdot \{b\} \cdot \{a\} \cdot \{b\})^*$
any finite language Use only $\cdot$ and $+$
any cofinite language Use only $\cdot$ and $+$, then $\cdot(\{a\} + \{b\})^*$
$\{uabv \mid u, w \in \Sigma^*\}$ $(\{a\} + \{b\})^* \cdot \{a\} \cdot \{b\} \cdot (\{a\} + \{b\})^*$
$\{aub \mid u \in \Sigma^*\}$ $\{a\} \cdot (\{a\} + \{b\})^* \cdot \{b\}$

Name the regular languages $\varnothing, \{\lambda\}, \{a_1\}, ..., \{a_n\}$ with **symbols** $\mathbf{e}, \mathbf{0}, \mathbf{a_1}, ..., \mathbf{a_n}$. **Regular expressions** represent regular languages and are special type of words over the new alphabet: $\mathbf{e}, \mathbf{0}, \mathbf{a_1}, ..., \mathbf{a_n}, +, *, \cdot, (,)$

The **atomic regular expressions** are $\mathbf{e}, \mathbf{0}, \mathbf{a_1}, ..., \mathbf{a_n}$. $L(\mathbf{e}) = \varnothing, L(\mathbf{0}) = \{\lambda\}, L(\mathbf{a_i}) = \{a_i\}$
Suppose regular expressions $\mathbf{r}_1, \mathbf{r}_2$ and $L(\mathbf{r}_1), L(\mathbf{r}_2)$ are defined. Then $(\mathbf{r}_1 \cdot \mathbf{r}_2), (\mathbf{r}_1 + \mathbf{r}_2), (\mathbf{r})^*$ are **regular expressions**. The languages defined by these expressions are obvious. operator precedance: $* < \cdot < +$; we denote the set of all regular expressions **RE**.

A language L is regular iff $L = L(\mathbf{r})$ for some regular expression $\mathbf{r}$.
For forward, show every language in every R_t can be represented by a regular expression by induction; for backward, show L is regular using the definition of regular expressions.

Kleene Theorem: A language L is regular iff L is FA recognizable.