Automata Minimization and operations with regular languages.

DSIC - UPV

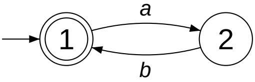
Closure Operations

- A set C is closed under an operation $\cdot$ iff for any elements $x, y \in C$, $x \cdot y \in C$.
- Examples
  - Let $C = \{L \subseteq \Sigma^* \mid L \text{ es finite } \}$. The union and the intersection are closed in C, whereas the complement is not.

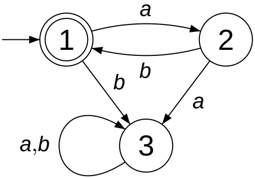
Automata

Automaton A_1 (not complete)

$$L(A_1) = \{(ab)^n \mid n \geq 0\} = \{ab\}^*$$



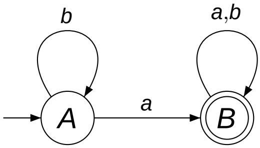
Automaton A_2 (complete). Note that $L(A_2) = L(A_1)$



Automata

Automaton A_3

$$L(A_3) = \{x \in \{a,b\}^* \mid |x|_a > 0\} = \{a,b\}^*\{a\}\{a,b\}^* = \\ = \{b\}^*\{a\}\{a,b\}^*$$



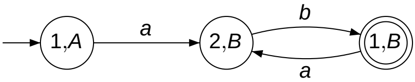
Boolean operations
Intersection

Regular languages are closed under intersection:
Let $L_1, L_2 \in \mathcal{L}_3$, then there exist two automata A_1, A_2 such that $L_1 = L(A_1), L_2 = L(A_2)$, where
$A_i = (Q_i, \Sigma, \delta_i, q_i, F_i), i = 1, 2$
We build $A' = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = Q_1 \times Q_2$
- $q0 = [q1, q2]$
- $F = F1 \times F2$
- $\delta([p_1, p_2], a) = [\delta_1(p_1, a), \delta_2(p_2, a)], p_1 \in Q1, p_2 \in Q2, a \in \Sigma$

$$\Rightarrow L(A') = L(A_1) \cap L(A_2)$$

Automaton for $L(A_1) \cap L(A_3)$.



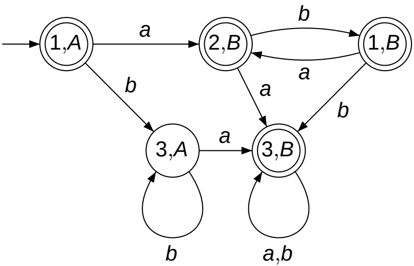
Regular languages are closed under **Union**:
Let $L_1, L_2 \in \mathcal{L}_3$, then there exist two *complete* automata A_1, A_2 such that $L_1 = L(A_1), L_2 = L(A_2)$, where $A_i = (Q_i, \Sigma, \delta_i, q_i, F_i), i = 1, 2$
We build $A' = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = Q_1 \times Q_2$
- $q0 = [q1, q2]$
- $F = F1 \times Q \times \cup Q \times F2$
- $\delta([p_1, p_2], a) = [\delta_1(p_1, a), \delta_2(p_2, a)], p_1 \in Q1, p_2 \in Q2, a \in \Sigma$

$$\Rightarrow L(A') = L(A_1) \cup L(A_2)$$

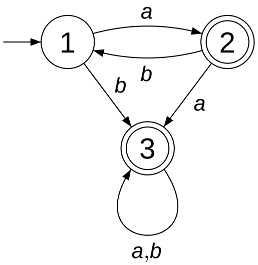
Boolean operations
Union

Automaton for $L(A_2) \cup L(A_3)$.



- Regular languages are closed under **Complement**. Let $L \in \mathcal{L}_3$, then there exists a complete automaton A such that $L = L(A)$ where $A = (Q, \Sigma, \delta, q_0, F)$. Automaton $A' = (Q, \Sigma, \delta, q_0, Q\text{-}F)$ accepts L^c
- Regular languages are closed under **Difference**. Let $L_1, L_2 \in \mathcal{L}_3$. Note that $L_1\text{-}L_2 = L_1 \cap L_2^c$.

Automaton for $L(A_2)^c$.

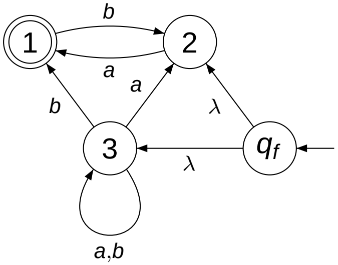


Reverse

Regular languages are closed under the operation **Reverse**.
Let $L \in \mathcal{L}_3$, then there exists an automaton $A = (Q, \Sigma, \delta, q_0, q_f)$ If $|F| > 1$, A can be modified to have one final state (How?).
We build $A' = (Q, \Sigma, \delta', q_f, q_0)$ where:
$q \in \delta(p, a) \leftrightarrow p \in \delta'(q, a)$.

$$\Rightarrow L(A') = L(A^r)$$

Automaton for $(L(A_2)^c)^r$.



Concatenation

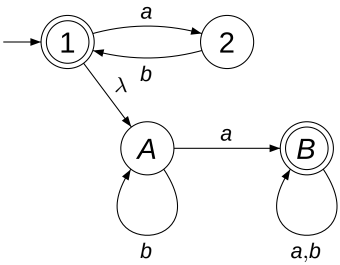
Regular languages are closed under Concatenation.
Let $L_1, L_2 \in \mathcal{L}_3$, then there exist two automata A_1, A_2 such that $L_1 = L(A_1), L_2 = L(A_2)$, where
$A_i = (Q_i, \Sigma, \delta_i, q_i, F_i), (i = 1, 2)$ and such that $Q1 \cap Q2 = \emptyset$
We build $A' = (Q, \Sigma, \delta', q_1, F_2)$ donde:

- $Q = Q_1 \cup Q_2$
- $\delta' = \delta_1 \cup \delta_2 \cup \delta''$ where $q_2 \in \delta''(p, \lambda)$, for any $p \in F_1$

$$\Rightarrow L(A') = L(A_1) \cdot L(A_2)$$

Concatenation

Automaton for $L(A_1) \cdot L(A_3)$.

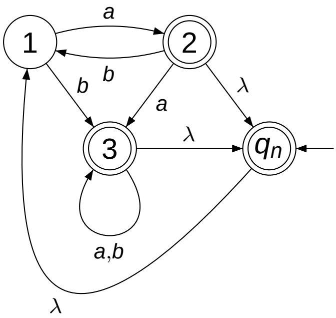


Star Closure

Regular languages are closed under Star Closure.
Let $L \in \mathcal{L}_3$, then there exists an automaton A such that $L = L(A)$ where $A = (Q, \Sigma, \delta_0, q_0, F)$ We build $A' = (Q', \Sigma, \delta', q_n, F)$ where:

- $Q' = Q \cup \{q_n\}$, $q_n \notin Q$
- $F = F \cup \{q_n\}$
- $\delta'(p, a) = \delta(p, a)$, for every $p \in Q$ and every $a \in \Sigma$
- $q_n \in \delta'(p, \lambda)$, for every $p \in F$
- $\delta'(q_n, \lambda) = \{q_0\}$

Automaton for $(L(A_2)^c)^*$.



Homomorphisms

- Regular languages are closed under **Homomorphisms**.
- Regular languages are closed under **Inverse Homomorphisms**.
  Let $h : \Sigma \rightarrow \Delta^*$ and $L \in \mathcal{L}_3$,there exists an automaton A such that $L = L(A)$, where $A = (Q, \Sigma, \delta, q_0, F)$.
  We build $A' = (Q, \Sigma, \delta', q_0, F)$ with:

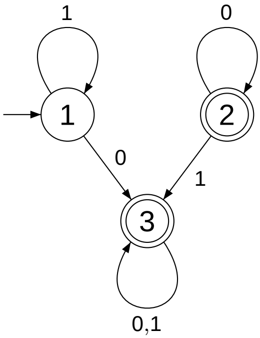
$$\delta'(p, a) = \begin{cases} \delta(p, h(a)) & \text{if}\, \delta(p, h(a)) \neq \emptyset \\ \emptyset & otherwise \end{cases}$$

Homomorphism

- $\Sigma = \{a, b\}$, $\Delta = \{0, 1, 2\}$
- $h(a) = 0$, $h(b) = 12$

Inverse Homomorphism

$\Sigma = \{0, 1\}$, $\Delta = \{a, b\}$. $g(0) = ab$, $h(1) = ba$. Automaton for $g^{-1}(L(A_2)^c)$.



Automata Minimization

- A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is reachable if for every $q \in Q$ there exists a word $x \in \Sigma$ such that $\delta(q_0, x) = q$
- Let $A = (Q, \Sigma, \delta, q_0, F)$ be a complete and reachable DFA. The indistinguishability relation $\sim$ en Q is defined $\forall q, q' \in Q$:
  $(q \sim q' \leftrightarrow \forall x \in \Sigma^*(\delta(q, x) \in F \leftrightarrow \delta(q', x) \in F))$
- Let $A = (Q, \Sigma, \delta, q_0, F)$ be a complete and reachable DFA and let $\sim$ be the indistinguishability relation. We define the quotient automaton $A/\sim = (Q, \Sigma, \delta, q_0, F)$ as:
  - $Q = [q]_\sim \mid q \in Q$
  - $q_0 = [q_0]_\sim$
  - $F = \{[q] \mid q \in F\}$
  - $\delta([q]_\sim, a) = [\delta(q, a)]_\sim$

Automata Minimization

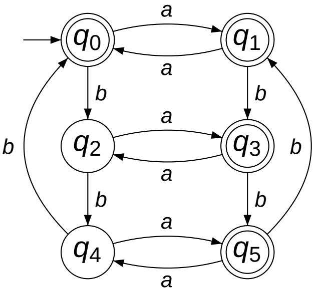
- Sea $A = (Q, \Sigma, \delta, q_0, F)$ be a complete and reachable DFA and let $\sim$ be the indistinguishability relation. The automaton $A/\sim$ is the minimum DFA accepting $L(A)$
- Let $A = (Q, \Sigma, \delta, q_0, F)$ be a complete and reachable DFA and let $k \geq 0$ be an integer. The k-indistinguishability relation $\sim_k$ is defined:
  $\forall q, q' \in Q : (q \sim_k q \leftrightarrow$
  $\forall x \in \Sigma^*, |x| \leq k, (\delta(q, x) \in F \leftrightarrow \delta(q, x) \in F))$
- Properties of $\sim_k$:
  - $\forall k \geq 0, p \sim_{k+1} q \rightarrow p \sim_k q$
  - $\forall k \geq 0, p \sim q \rightarrow p \sim_k q$
  - $\forall k \geq 0, p \sim_{k+1} q \leftrightarrow (p \sim_k q \wedge \forall a \in \Sigma, \delta(p, a) \sim_k \delta(q, a))$

Automata Minimization

Minimization Algorithm:

- 1. $\pi_0 = \{Q - F, F\}$
- 2. Obtain π_{k+1} from π_k $B(p, \pi_{k+1}) == B(q, \pi_{k+1})$ if and only if
  - $B(p, \pi_k) == B(q, \pi_k)$
  - y For every $a \in \Sigma, B(\delta(p, a), \pi_k) == B(\delta(q, a))$
- 3. If π_{k+1} is different from π_k go to 2

Automata Minimization



Automata Minimization

π_0 :

		a	b
B_0	q_0	B_0	B_1
	q_1	B_0	B_0
	q_3	B_1	B_0
	q_5	B_1	B_0
B_1	q_2	B_0	B_1
	q_4	B_0	B_0

Automata Minimization

π_1 :

		a	b
B_0	q_0	B_1	B_3
$B1$	q_1	B_0	B_2
$B2$	q_3	B_3	B_2
	q_5	B_4	B_1
B_3	q_2	B_2	B_4
B_4	q_4	B_2	B_0

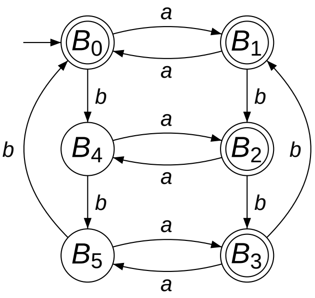
Automata Minimization

π_2 :

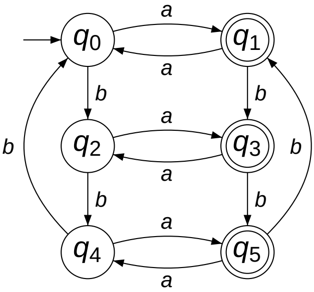
		a	b
B_0	q_0	B_1	B_4
$B1$	q_1	B_0	B_2
$B2$	q_3	B_4	B_3
$B3$	q_5	B_5	B_1
B_4	q_2	B_2	B_5
B_5	q_4	B_3	B_0

Automata Minimization

$$\pi_3 = \pi_2$$



Automata Minimization



Automata Minimization

π_0 :

		a	b
B_0	q_1	B_1	B_0
	q_3	B_1	B_0
	q_5	B_1	B_0
B_1	q_0	B_0	B_1
	q_2	B_0	B_1
	q_4	B_0	B_1

Automata Minimization

$$\pi_1 = \pi_0$$