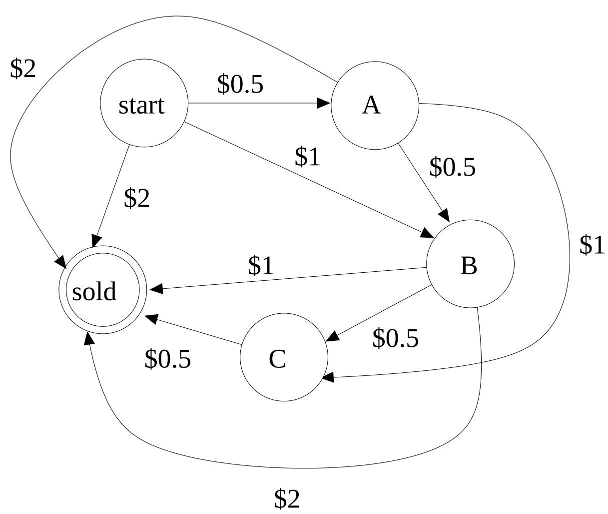
Lecture 4: Finite Automata

A **finite automaton** is a machine (controller) with only a finite number of states.
It is the simplest and most restricted model of computers.

Such a controller is used in many electromechanical devices, e.g., automatic door, lift, vending machine, washing machine, microwave oven, parts of digital watches and calculators, traffic light systems.

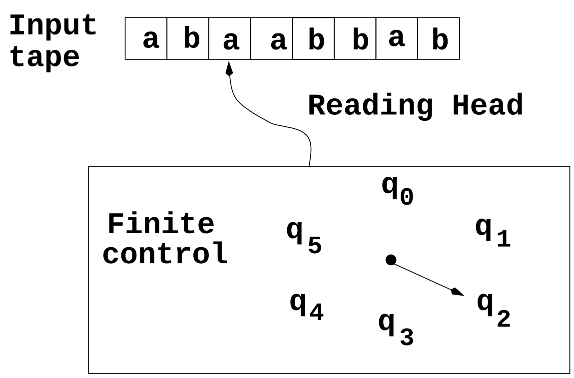
Example: vending machine



Deterministic finite automata (DFA):

A language recognizer consisting of

1. An *input tape* — divided into squares, for storing the string input.
2. A *finite control* — a controller that can be in a finite number of states.
3. A *reading head* — move one position to the right each step.



How does a DFA work?

1. Initialization:
   - The reading head is at the leftmost square.
   - The finite control is in the *initial state*.
2. After reading a symbol,
   - The reading head moves one square to the right.
   - The finite control enters a new state, which is deterministically dependent on the current state and current input symbol.
3. After reading the entire input string,
   - If the finite control is in a *final state*, the input string is considered *accepted*.
   - Otherwise, the input string is not *accepted*.

Deterministic finite automata

A DFA is a quintuple $M = (K, \Sigma, \delta, s, F)$ where

- K — a finite set of states (of the machine),
- Σ — an alphabet (symbols that the machine recognizes),
- $s \in K$ — the initial state,
- $F \subseteq K$ — a set of final states (a.k.a. accepting states)
- δ — the transition function:
  - $\delta : K \times \Sigma \to K$,

current state	current symbol	next state
q	σ	$\delta(q, \sigma)$

  - meaning it is deterministic – each pair of state and symbol is mapped to exactly one state.

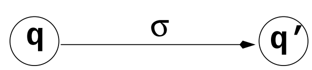
Example 1: $M = (K, \Sigma, \delta, s, F)$ where

$K = \{q_0, q_1\}$, $s = q_0$, $F = \{q_0\}$, $\Sigma = \{a, b\}$

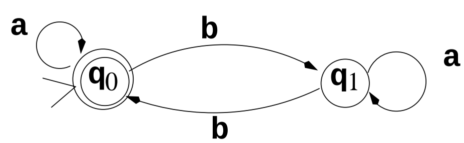
δ	a	b
q_0	q_0	q_1
q_1	q_1	q_0

State diagram: a pictorial representation of DFAs.

- Nodes represent states,
  - initial state is indicated by >,
  - final states are indicated by double cycles.
- Arcs represent transitions:
  - If $\delta(q, \sigma) = q'$, then there is an arrow from q to q', labeled with σ.

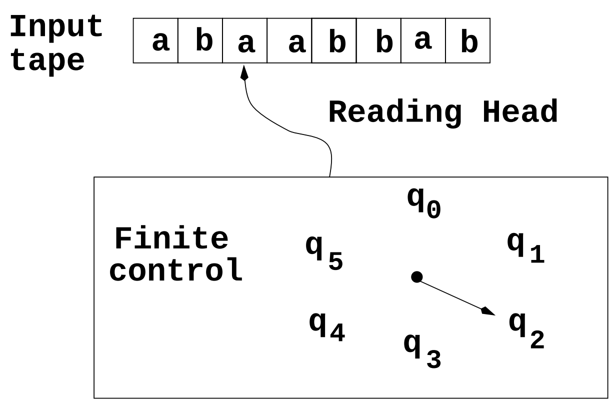


Example: for the FA M in the previous page:



A configuration indicates the current status of the machine, i.e., which state it is currently at, and remaining content of the input tape.

- A configuration is denoted as $(q, w) \in K \times \Sigma^*$, where q is the current state and Σ^* is the unread part of the input string.
- Example: (q, baa) means that the FA is currently in state q and the tape content is baa (the next symbol to be read is b).



The current configuration is $(q_2, aabbab)$.

Algebraic description of computations by DFAs

One step of computation in a DFA:

- Suppose the current configuration is (q, w) where $w = \sigma w'$ and $\delta(q, \sigma) = q'$. Then the configuration after one step of computation is (q', w').
- That is, (q, w) **yields** (q', w') in **one step**;

$$(q, w) \vdash_M (q', w').$$

- $\vdash_M$: $K \times \Sigma^+ \to K \times \Sigma^*$.

Example:

- Consider the FA in Example 1, suppose the input string is *abba*. Then the initial configuration is

$$(q_0, abba)$$

- The configuration after **one step of computation** is

$$(q_0, bba)$$

- What is the configuration after the second step?

In short: $(q_0, abba) \vdash_M (q_0, bba) \vdash_M (q_1, ba)$

Multiple steps of computations:

- If

  $$(q, w) \vdash_M (q_1, w_1) \vdash_M \ldots \vdash_M (q_{n-1}, w_{n-1}) \vdash_M (q', w')$$

  we say (q, w) **yields** (q', w') in $n \geq 0$ steps, denoted

  $$(q, w) \vdash_M^* (q', w').$$

- Since any configuration yields itself in zero step, we can write

  $$(q, w) \vdash_M^* (q, w).$$

- That is, $\vdash_M^*$ is the reflexive transitive closure of $\vdash_M$.

Definitions

- A string w is **accepted** by M iff

$$(s, w) \vdash_M^* (q, e) \text{ and } q \in F$$

  - That is, start at the initial state; when the entire string is read, the controller must be in one of the final states.

- The **language accepted** by an FA M is

$$L(M) = \{w : w \text{ accepted by } M\}.$$

Example: Consider the FA M in Example 1, i.e., $M = (K, \Sigma, \delta, s, F)$ where

$K = \{q_0, q_1\}$, $s = q_0$, $F = \{q_0\}$

$\Sigma = \{a, b\}$

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_0

Symbol "b" flips the state, while "a" doesn't.

- $$\begin{aligned} (q_0, aabba) &\vdash_M (q_0, abba) \\ &\vdash_M (q_0, bba) \\ &\vdash_M (q_1, ba) \\ &\vdash_M (q_0, a) \\ &\vdash_M (q_0, e) \end{aligned}$$
- So, $(q_0, aabba) \vdash_M^* (q_0, e)$
- Therefore, M accepts $aabba$.

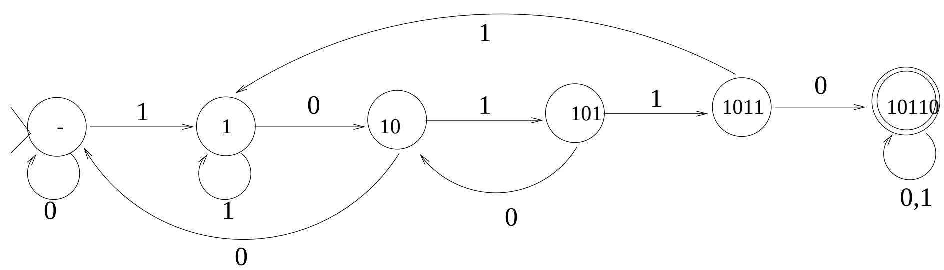
$L(M) = \{w \in \{a, b\}^* : w$ contains even number of b's$\}$.

DFA and pattern matching:

FA are useful in recognizing patterns in data.

Example: Design a machine that matches an occurrence of 10110 in a string.

Note: the prefixes of 10110 are e, $1, 10, 101, 1011, 10110$. Each corresponds to one state of the FA.



Lemma: Let L be a language accepted by a DFA, then $\overline{L}(= \Sigma^* - L)$ is accepted by a DFA.

Proof:

Suppose $L = L(M)$ where $M = (K, \Sigma, \delta, s, F)$ is a DFA.
Define $M' = (K, \Sigma, \delta, s, K - F)$.
Let $w \in \Sigma^*$. Then
$w \in L(M')$
$\Leftrightarrow (s, w) \vdash^*_{M'} (q, e)$ for some $q \in K - F$
$\Leftrightarrow (s, w) \vdash^*_{M'} (q, e)$ for some $q \notin F$
$\Leftrightarrow (s, w) \vdash^*_{M} (q, e)$ for some $q \notin F$
since M' and M have the same δ.
$\Leftrightarrow w \notin L$
So $L(M') = \overline{L}$, i.e., $\overline{L}$ is accepted by M'.

Given two DFAs M_1 and M_2 that accept the languages L_1 and L_2 respectively. Can you design an algorithm to construct a DFA that accepts the language $L_1 \cap L_2$?

Lemma: Let L_1 and L_2 be languages accepted by DFAs, then $L_1 \cap L_2$ and $L_1 \cup L_2$ are accepted by DFAs.

Proof:
Suppose $L_1 = L(M_1)$ and $L_2 = L(M_2)$ where
$M_1 = (K_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (K_2, \Sigma, \delta_2, s_2, F_2)$ are DFAs.
Define $M = (K_1 \times K_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$,
where $\delta((p, q), \sigma) = (\delta_1(p, \sigma), \delta_2(q, \sigma))$.
Let $w = \sigma_1....\sigma_n \in L(M)$.
$\Leftrightarrow ((s_1, s_2), \sigma_1....\sigma_n) \vdash_M ((p_2, q_2), \sigma_2....\sigma_n)$ where
$\delta_1(s_1, \sigma_1) = p_2$ and $\delta_2(s_2, \sigma_1) = q_2$
$\vdash_M ((p_3, q_3), \sigma_3....\sigma_n)$ where
$\delta_1(p_2, \sigma_2) = p_3$ and $\delta_2(q_2, \sigma_2) = q_3$
$\vdash_M$ ....
$\vdash_M ((p_{n+1}, q_{n+1}), e)$ where $(p_{n+1}, q_{n+1}) \in F_1 \times F_2$
$\Leftrightarrow (s_1, \sigma_1....\sigma_n) \vdash_{M_1} (p_2, \sigma_2....\sigma_n) \vdash_{M_1} ... \vdash_{M_1} (p_{n+1}, e)$
where $p_{n+1} \in F_1$, and

$(s_2, \sigma_1....\sigma_n) \vdash_{M_2} (q_2, \sigma_2....\sigma_n) \vdash_{M_2} ... \vdash_{M_2} (q_{n+1}, e)$
where $q_{n+1} \in F_2$.

$\Leftrightarrow \sigma_1....\sigma_n \in L(M_1)$ and $\sigma_1....\sigma_n \in L(M_2)$

$\Leftrightarrow \sigma_1....\sigma_n \in L_1$ and $\sigma_1....\sigma_n \in L_2$.

So $L(M) = L_1 \cap L_2$.

We can prove that $L_1 \cup L_2$ is accepted by a DFA in a similar way.

Alternatively:

L_1, L_2 accepted by DFAs.

$\Rightarrow \overline{L_1}, \overline{L_2}$ accepted by DFAs.

$\Rightarrow L_3 = \overline{L_1} \cap \overline{L_2}$ accepted by a DFA.

$\Rightarrow \overline{L_3}$ accepted by a DFA.

$\Rightarrow L_1 \cup L_2$ accepted by a DFA

(since $\overline{L_3} = \overline{\overline{L_1} \cap \overline{L_2}} = L_1 \cup L_2$.)

Can we prove that L^* and $L_1 L_2$ are accepted by DFAs?