Language Processors F29LP2, Lecture 4

Jamie Gabbay

February 2, 2014

Finite automata

The plural of automaton is automata.

For this course "automata" refers to the mathematical concept we are about to define—not to the a clockwork device (as in the film "Hugo").

Deterministic Finite Automata (DFA)

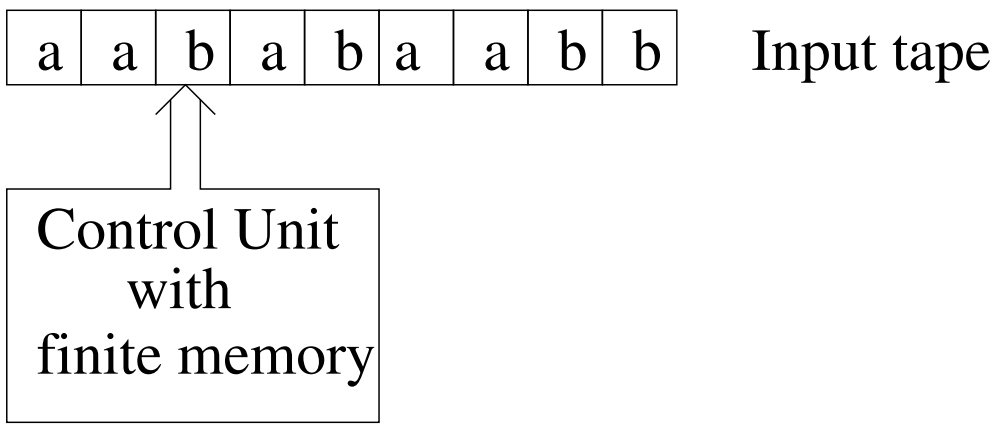
DFA describe languages.

DFA are accepting devices: you input a word and the DFA accepts or rejects it.

The set of words accepted by a DFA determines a language.

The DFA 'eats' the letters of the word from left to right, one by one. It has finite memory. At each input letter, the DFA's state changes.

The word is accepted if the DFA ends in an 'accepting state' after 'eating' the word.



DFA definition

A DFA $A = (Q, \Sigma, \delta, q_0, F)$ is specified by:

1. Finite state set Q.
2. Input alphabet Σ. The machine operates on words over Σ.
3. Transition function

$$\delta : Q \times \Sigma \longrightarrow Q$$

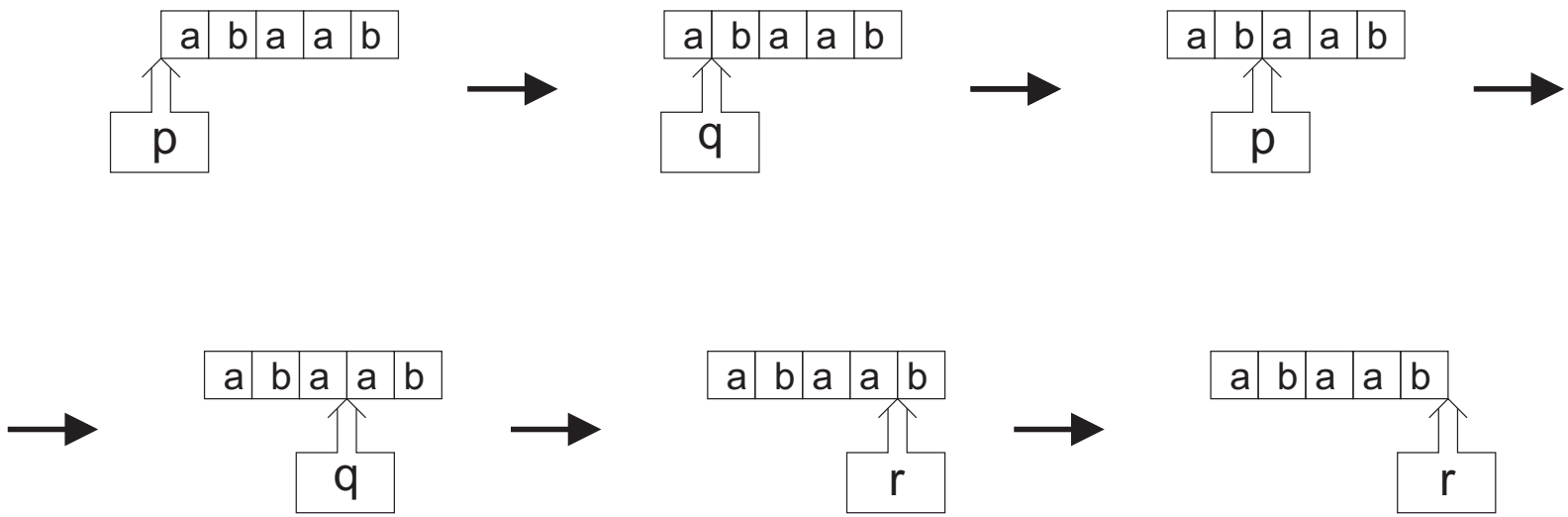
   If the machine is in state q and the present input letter is a then the machine changes its internal state to $\delta(q, a)$ and moves to the next input letter.
4. Initial state $q_0 \in Q$.
5. Set $F \subseteq Q$ of final states specifies which states are accepting. If the machine, after reading the whole input, ends at a state in F, then the word is accepted, otherwise rejected.

Language of a DFA

The language recognised (or accepted) by a DFA A is the set of words that A accepts: write this $L(A)$. Take

$$A = (\{p, q, r\}, \{a, b\}, \delta, p, \{r\}) \quad \text{where} \quad \delta = \begin{array}{c|cc} & a & b \\ \hline p & q & p \\ q & r & p \\ r & r & r \end{array}.$$

The operation of A on the word $w = abaab$ is as follows:



Transition diagrams

Fine, but less readable than it could be.

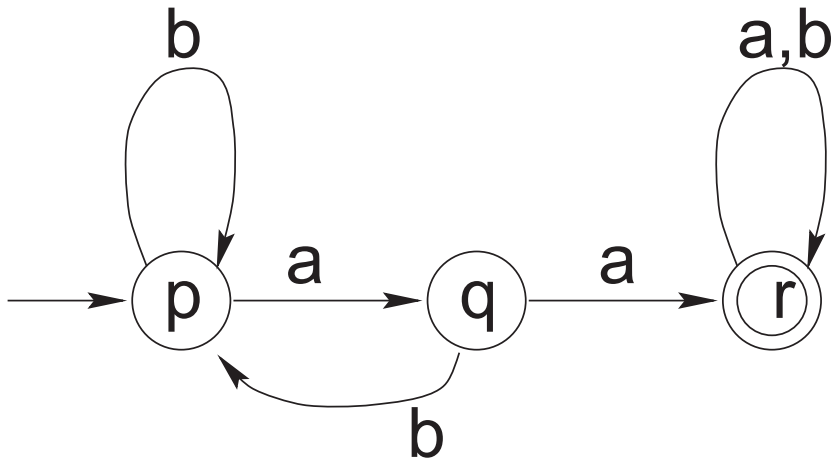
A transition diagram is a labeled directed graph. Vertices = states & edges = transitions.

Transition $\delta(q, a) = p$ is an arc labeled a from vertex q to vertex p:

Final states are double circles. The initial state is indicated by a short incoming arrow . . .

Transition diagrams

. . . and here's the transition diagram for the DFA A of Slide 5:



Transition diagrams

To determine whether w is accepted by A, follow the path in A labeled with the input letters of w, starting from the initial state.

If this leads you to a final state, the word is accepted.

$w = abaab$ leads to state r, which is a final state, so the word is accepted.

$w' = abba$ is rejected because it leads to q, which is not a final state.

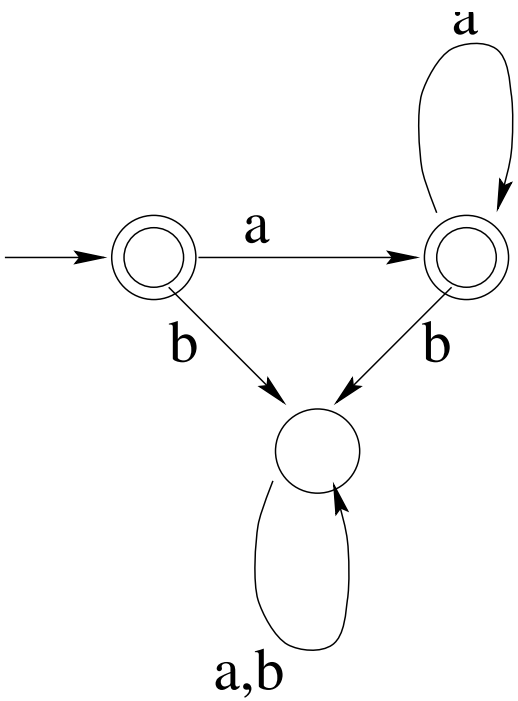
Summary so far

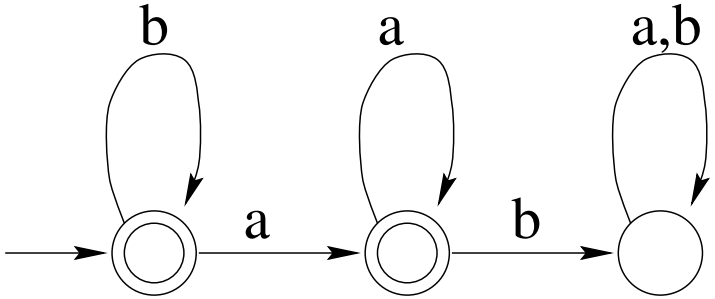
A DFA A finitly describes its language $L(A)$.

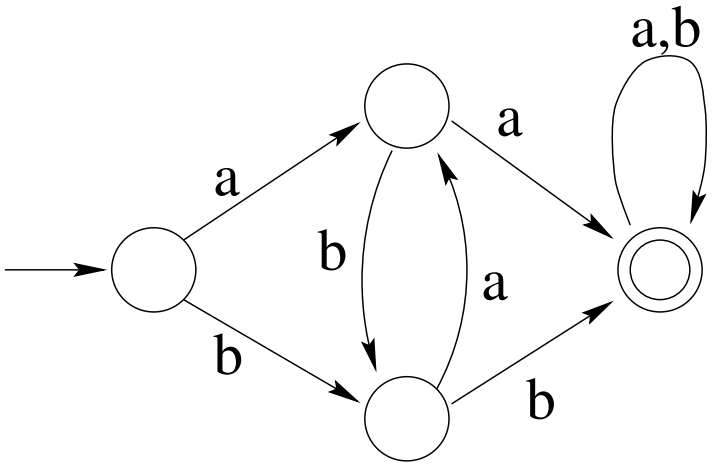
The DFA of Slides 5 and 7 represents the language "all words over the alphabet $\{a, b\}$ that contain aa as a subword".

Examples

Let's determine the languages recognised by the following DFAs:







Examples

Let's draw DFAs to recognise languages over the alphabet $\{a, b\}$.

Note: the automaton must recognise the language exactly. This means:

- Every word of the language should be accepted.
- Every word not in the language should be rejected.

1. Words that end in ab.
2. Words with an odd number of a's.
3. Words that contain aba as a subword.
4. Words that start with a and end in a.
5. The finite language $\{\varepsilon, a, b\}$.
6. All words over $\{a, b\}$, i.e. $\{a, b\}^*$.

DFAs and regular languages

Not every languages can be defined by a DFA. E.g. no DFA accepts

$$\{a^p \mid p \in \mathbb{N} \text{ is prime}\}.$$

We'll prove later that even simple languages such as

$$\{a^n b^n \mid n \geq 0\}$$

cannot be recognised by any DFA.

Languages that can be recognised by DFA are called regular (cf. regexps!).

Extended transition function

Recall the transition function $\delta : Q \times \Sigma \longrightarrow Q$ from slide 4 ($\delta(q, a)$ is the state we reach from q if we input a).

Extend this to $\hat{\delta}$, giving the state after an arbitrary string of letters is read:

$$\hat{\delta} : Q \times \Sigma^* \longrightarrow Q.$$

For state q and word w, $\hat{\delta}(q, w)$ is the state we reach from q if we input w.

Extended transition function

Let's define that recursively:

1. $\hat{\delta}(q, \varepsilon) = q$ (no input = no change).
2. $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$.
   (If w takes us from q to $q' = \hat{\delta}(q, w)$ then wa takes us to $\delta(q', a)$.)

$\hat{\delta}$ extends δ, in the sense that $\hat{\delta}(q, a) = \delta(q, a)$.

So we can legitimately simplify notation and write δ for both δ and $\hat{\delta}$ from now on; $\delta(q, w)$ means $\hat{\delta}(q, w)$.

Example

Recall the three-state DFA from slides 5 and 7. In this DFA,

$$\begin{aligned} \delta(r, ab) &= r, \\ \delta(q, bb) &= p, \\ \delta(p, abaab) &= r. \end{aligned}$$

Formal definition of $L(A)$

Suppose $A = (Q, \Sigma, \delta, q_0, F)$ is a DFA.

Then $L(A)$, the language it accepts, can be precisely formulated as follows:

$$L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}.$$

This is the set of words w over the alphabet Σ such that, if the machine reads input w in the initial state q_0, then it reaches a final state.

Formal definition of $L(A)$

DFA according to our definition are often called a complete DFA because each state must have an outgoing transition with each letter in the alphabet.

Alternatively, allow partial transition functions, not necessarily defined for all states and input letters.

If there is no transition from q for an input letter a, the machine is deemed to halt (it 'crashes'). The word is rejected.

Exactly the same languages can be recognised using complete or partial transition functions (exercise: prove it).

We stick to δ being a total function; it's mathematically easier to do so.