

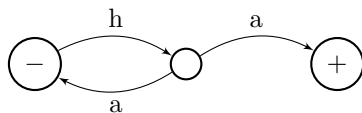
# 1 Finite State Automata

## 1.1 Alphabets & Strings

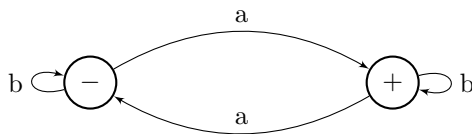
- Let  $A$  be a set; then  $A^n$  is the set of all finite sequences  $a_1 \dots a_n$  with  $a_i \in A$ ,  $1 \leq i \leq n$ 
  - Elements of  $A$  are *letters* or *symbols*
  - Elements of  $A^n$  are *words* or *strings* over  $A$  of length  $n$
- $\varepsilon$  is the special *empty string*, the only string of length 0
- $A^+ = \bigcup_{m \geq 1} A^m$  – the set of non-empty strings over  $A$  of any length
- $A^* = A^+ \cup \varepsilon = \bigcup_{m \geq 0} A^m$  – the set of (possibly empty) strings over  $A$  of any length
- If  $\alpha = a_1 \dots a_m$ ,  $\beta = b_1 \dots b_n \in A^*$ , then define  $\alpha\beta$  to be  $a_1 \dots a_m b_1 \dots b_n \in A^{m+n}$ . This gives binary ‘product’ or *concatenation* on  $A^*$
- For  $\alpha \in A^+$ , define  $\alpha^n$ ,  $n \in \mathbb{N}$  by  $\alpha^0 = \varepsilon$ , and  $\alpha^{n+1} = \alpha^n \alpha$
- A *language* with alphabet  $A$  is a subset of  $A^*$

## 1.2 Definition of an FSA

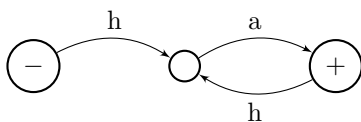
- A Finite State Automaton (FSA) is a tuple  $M = (Q, F, A, \tau, q_0)$ 
  - $Q$  is a finite set of states
  - $F \subseteq Q$  is the set of final states
  - $A$  is the alphabet
  - $\tau \subseteq Q \times A \times Q$  is the set of transitions
  - $q_0 \in Q$  is the initial state
- The transition diagram of an FSA is a directed graph with:
  - Vertex set  $Q$
  - An edge for each transition;  $(q, a, q') \in \tau$  corresponds to an edge from  $q$  to  $q'$  with label  $a$
  - Initial state  $q_0$  labelled with  $-$
  - Final states labelled with  $+$
  - Example: a *non-deterministic* ‘haha machine’, with  $A = \{h, a\}$



- A *computation* of  $M$  is a sequence  $q_0, a_1, q_1, a_2, \dots, a_n, q_n$  with  $n \geq 0$  where  $(q_i, a_{i+1}, q_{i+1}) \in \tau$  for  $0 \leq i \leq n-1$ 
  - The *label* on the computation is  $a_1 \dots a_n$
  - The computation is *successful* if  $q_n \in F$
  - A string  $a_1 \dots a_n$  is *accepted* by  $M$  if there is a successful computation with label  $a_1 \dots a_n$ , and it is *rejected* otherwise
- The language recognised by  $M$  is  $\mathcal{L}(M) = \{w \in A^* \mid w \text{ is accepted by } M\}$
- There is a one-to-one correspondence between computations of  $M$  and paths in the graph from  $q_0$
- Example:  $A = \{a, b\}$  of an FSA accepting only words with an odd number of ‘a’s



- An FSA is deterministic (a DFA) if for all  $q \in Q, a \in A$  there is exactly one  $q' \in Q$  such that  $(q, a, q') \in \tau$
- Example: DFA for the ‘haha machine’



- Note this machine lacks a transition for  $a$  when in the initial state – though technically required for a DFA, it is easily fixed by adding an ‘error state’ to catch what would otherwise be missing transitions

### 1.3 Deterministic FSAs

- For a DFA  $M$ , define the transition function  $\delta : Q \times A \rightarrow Q$  by  $q' = \delta(a, q)$ , where  $q'$  is the unique element such that  $(q, a, q') \in \tau$
- If  $\mathcal{L}$  is a language with alphabet  $A$ , then the following are equivalent:
  1.  $\mathcal{L}$  is recognised by an FSA
  2.  $\mathcal{L}$  is recognised by a DFA
- Given a non-deterministic FSA  $M = (Q, F, A, \tau, q_0)$ , an equivalent DFA  $M' = (Q', F', A, \tau', q'_0)$  may be generated by the *powerset method*:
  - $Q' = \mathcal{P}(Q) \setminus \emptyset$  (i.e. the set of all subsets of  $Q$  that aren't empty)
  - $F' = \{X \in Q' \mid q \in X \text{ for some } q \in F\}$
  - For  $X \in Q', a \in A$ , define  $\delta(X, a) := \{q \in Q \mid (x, a, q) \in \tau \text{ for some } x \in X\}$
  - $\tau' = \{(X, a, \delta(X, a)) \mid X \in Q', a \in A\}$
  - $q'_0 = \{q_0\}$
- Proof: show that  $\mathcal{L}(M) = \mathcal{L}(M')$ 
  - $\mathcal{L}(M) \subseteq \text{Lang}(M')$ :
    - \* Given  $w \in \mathcal{L}(M)$ ,  $q_0 a_1 \dots a_n q_n$  is a successful computation of  $M$
    - \* Then define  $q'_i = \delta(q'_{i-1}, a_i)$  for  $1 \leq i \leq n$
    - \*  $q'_0, a_1, q'_1 \dots a_n, q'_n$  will be a successful computation of  $M'$
    - \* Therefore  $w \in \mathcal{L}(M')$
  - $\mathcal{L}(M') \subseteq \text{Lang}(M)$ :
    - \* Let  $w = a_1 \dots a_n \in L(M')$ , and  $q'_0, a_1, q'_1 \dots a_n, q'_n$  be a successful computation of  $M'$
    - \* Each  $q'_i$  cannot be the empty set
    - \* By definition of  $\tau'$ ,  $\exists q_1 \in q'_1$  s.t.  $(q_0, a_1, q_1) \in \tau$
    - \* Then we can find  $q_i \in q'_i$  s.t.  $(q_{i-1}, a_i, q_i) \in \tau$  for  $1 \leq i \leq n$
    - \* For  $q_n$  we further require  $q_n \in F$
    - \* Therefore,  $q_0, a_1, q_1, a_2, \dots a_n, q_n$  is a successful computation
    - \* Therefore  $w \in \mathcal{L}(M)$

## 1.4 The Pumping Lemma

- The Pumping Lemma says that for any  $\mathcal{L}$  recognised by an FSA  $M$ , there is a certain word length beyond which all words can be split into sections as  $xyz$ , where  $xy^nz$  is also in the language
- Formally there is an integer  $p > 0$  s.t. any word  $w \in L$  with  $|w| \geq p$  is of the form  $w = xyz$ , where  $|y| > 0$ ,  $|xy| \leq p$  and  $xy^iz \in \mathcal{L}$  for  $i \geq 0$
- Proof:
  - Let  $p$  be the number of states in  $M$ , and suppose  $w = a_1 \dots a_n \in \mathcal{L}$ , where  $n \geq p$
  - A successful computation  $q_0, a_1, \dots, q_n$  has to pass through a certain state at least twice (by the pigeonhole principle)
  - Therefore,  $\exists r < s$  s.t.  $q_r = q_s$ ; choose minimal such  $s$
  - Now put  $x = a_1 \dots a_r$ ,  $y = a_{r+1} \dots a_s$  (note  $|y| > 0$ ), and  $z = a_{s+1} \dots a_n$
  - By minimality of  $s$ ,  $q_0, \dots, q_{s-1}$  are distinct, and  $|xy| = s \leq p$
  - Then, note that  $q_r, a_{r+1}, \dots, q_s$  is a loop, which may be validly repeated  $i \geq 0$  times
  - Therefore,  $xy^iz \in \mathcal{L}$
- Corollary: here exist languages which are not computable by an FSA
- Example: there is no FSA which can recognise  $\mathcal{L} = \{a^n b^n \mid n \in \mathbb{N}\}$
- Proof:
  - Assume for a contradiction there exists an FSA  $M$  which can recognise  $\mathcal{L}$
  - Let  $p$  be the number from the pumping lemma, and choose  $n \geq p$  and consider  $w = a^n b^n$
  - By the pumping lemma,  $\exists x, y, z$  s.t.  $a^n b^n = xyz$ , with  $|y| \geq 1$  and  $|xy| \leq p \leq n$
  - Then  $y$  is written entirely in terms of the letter  $a$ , and  $|y| \geq 1$
  - By the pumping lemma,  $xy^iz \in \mathcal{L}$  for all  $i$
  - So choose  $i = 0$ , then some  $w = a^k b^n \in \mathcal{L}$  s.t.  $k < n$ , which is a contradiction

## 2 Turing Machines

### 3 Partial Recursive Functions

## 4 First Order Logic