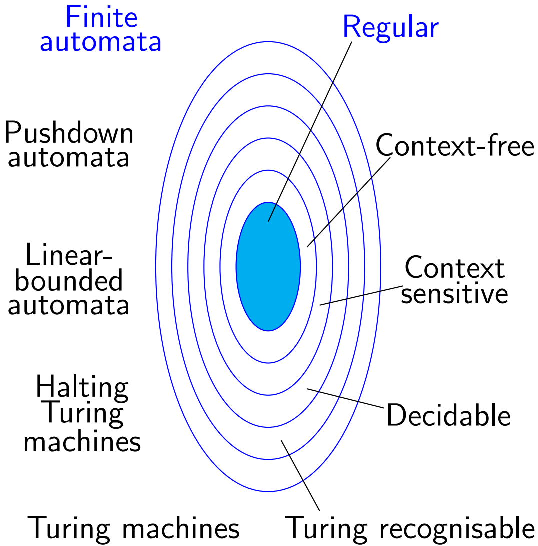
COMP30026 Models of Computation

Lecture 9: Finite Automata

Mak Nazecic-Andrlon and William Umboh

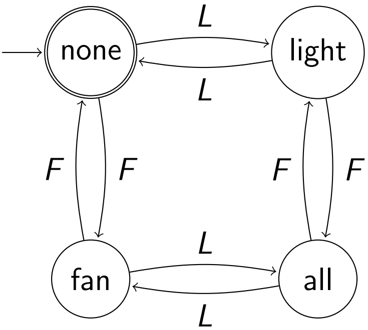
Semester 2, 2024

Machines vs Languages



An Example Automaton

A state diagram for simple controller for lighting and ventilation:

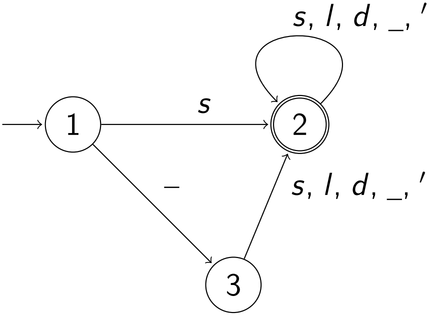


Start in the “none” state. We can then toggle the light (L) or the fan (F). $LFLFFF$ is accepted: it finishes in the “none” state.

$LLFFL$ is rejected: it finishes in the “light” state.

Example 2

Here is an automaton for recognising Haskell variable identifier:



s is an abbreviation for $a, \ldots, z$ (the small or lower-case letters)
l is an abbreviation for $A, \ldots, Z$ (the large or upper-case letters)
d is an abbreviation for $0, \ldots, 9$ (the digits)

Formal Definition

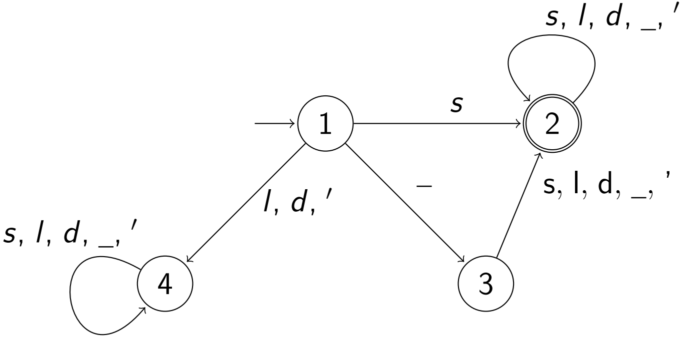
A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ are the accept states.

Here δ is a total function, that is, δ must be defined for all possible inputs.

Back to Example 2

To make it clear that the transition function is total, we should add a new state 4 and arcs to state 4 from state 1:



Strings and Languages

An alphabet Σ can be any non-empty finite set.

The elements of Σ are the symbols of the alphabet. Usually we choose symbols such as $\mathtt{a}$, $\mathtt{b}$, $\mathtt{c}$, 1, 2, 3, ...

A string over Σ is a finite sequence of symbols from Σ.

We write the concatenation of string x with a string y as xy.

The empty string is denoted by ϵ.

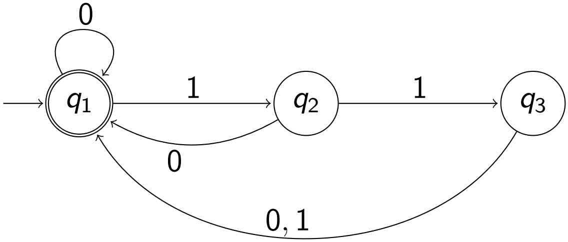
A language (over alphabet Σ) is a (finite or infinite) set of strings over Σ.

Σ^* denotes the set of all strings over Σ.

Examples of Languages over Alphabet $\Sigma = \{0, 1\}$

- $\emptyset$
- $\{\epsilon\}$
- $\{\epsilon, 0, 1\}$
- $\{00, 01, 10, 11\}$
- $\{\epsilon, 0, 00, 000, \dots\}$
- $\{\epsilon, 0, 1, 00, 11, 000, 111, \dots\}$
- $\{\epsilon, 01, 0011, 000111, \dots\}$
- $\{w \in \Sigma^* \mid w \text{ contains odd number of } 0\}$
- $\{w \in \Sigma^* \mid \text{the length of } w \text{ is a multiple of } 3\}$
- $\{w \in \Sigma^* \mid w \text{ is not empty string}\}$
- $\{w \in \Sigma^* \mid w \text{ does not contain } 001\}$
- Σ^*

Example 3



The automaton M_1 (above) can be described precisely as

$$M_1 = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_1\}) \qquad \text{with}$$

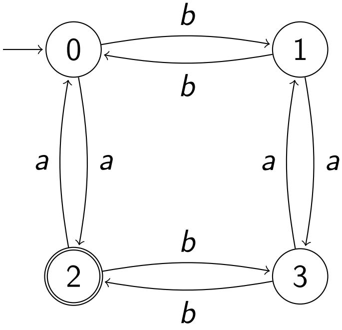
δ	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_1	q_1

$$L(M_1) = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w \text{ is } \epsilon, \textit{ or } \text{ends with `0'}, \textit{ or } \text{the number of} \\ \text{`1' symbols ending } w \text{ is a multiple of 3} \end{array} \right\}$$

is the language recognised by M_1.

Example 4

Which language is recognised by this machine?



Acceptance, Formally

What does it mean for an automaton to accept a string?

Let $M = (Q, \Sigma, \delta, q_0, F)$ and let $w = v_1 v_2 \cdots v_n$ for some $v_i \in \Sigma$.

M accepts w iff there is a sequence of states $r_0, r_1, \ldots, r_n$, with each $r_i \in Q$, such that

1. $r_0 = q_0$
2. $\delta(r_i, v_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$
3. $r_n \in F$

Let A be the set of all strings accepted by a machine M. We say A is the language of M and write $L(M) = A$.

We also say M recognises A.

Regular Languages

Definition

A language is regular iff there is a finite automaton that recognises it.

We shall soon see that there are languages which are not regular.

Regular Operations

Let A and B be languages (i.e. sets of strings).

The regular operations are:

- **Union:** $A \cup B$
- **Concatenation:** $A \circ B = \{xy \mid x \in A, y \in B\}$
- **Kleene star:** $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0, \text{ each } x_i \in A\}$

Note that the empty string, ϵ, is always in A^*.

Regular Operations: Example

Let $A = \{\mathrm{aa}, \mathrm{abba}\}$ and $B = \{\mathrm{a}, \mathrm{ba}, \mathrm{bba}, \mathrm{bbba}\}$. Then,

$$
\begin{aligned}
A \cup B &= \{\mathrm{a}, \mathrm{aa}, \mathrm{abba}, \mathrm{ba}, \mathrm{bba}, \mathrm{bbba}\}, \\
A \circ B &= \{\mathrm{aaa}, \mathrm{abbaa}, \mathrm{aaba}, \mathrm{abbaba}, \mathrm{aabba}, \mathrm{abbabba}, \ldots\}, \\
A^* &= \left\{ \begin{array}{l} \epsilon, \mathrm{aa}, \mathrm{abba}, \mathrm{aaaa}, \mathrm{aaabba}, \mathrm{abbaaa}, \mathrm{abbaabba}, \\ \mathrm{aaaaaa}, \mathrm{aaaaabba}, \mathrm{aaabbaaa}, \mathrm{aaabbaabba}, \ldots \end{array} \right\}.
\end{aligned}
$$

Closure

Theorem

If A and B are regular languages, then so are $A \cup B$, $A \circ B$, and A^*.

That is, the regular languages are closed under regular operations.

How to prove? Nondeterminism.

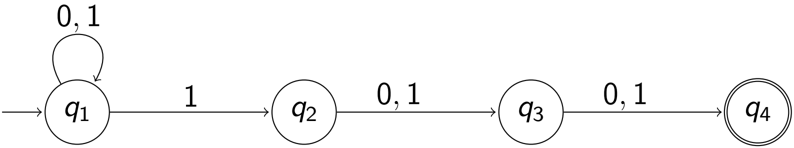
Nondeterminism

The type of machine we have seen so far is called a deterministic finite automaton, or DFA.

We now turn to non-deterministic finite automata, or NFAs.

Here is an NFA that recognises the language

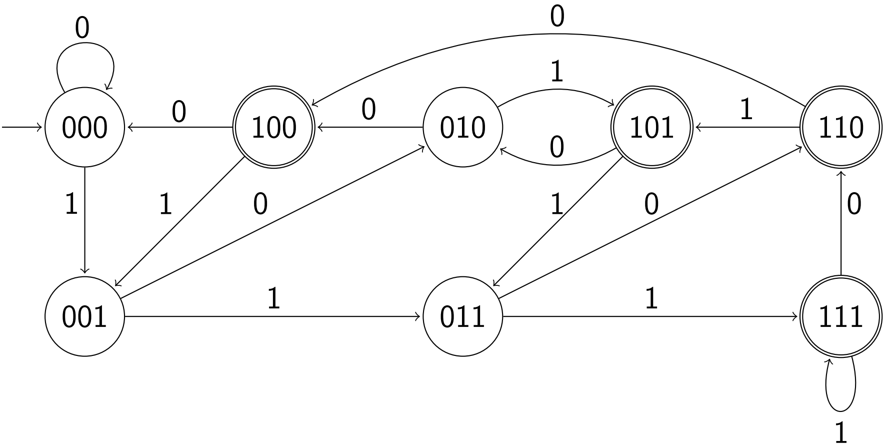
$$\left\{ w \;\middle|\; \begin{array}{l} w \in \{0,1\}^* \text{ has length 3 or more,} \\ \text{and the third last symbol in } w \text{ is 1} \end{array} \right\}$$



Note: No transitions from q_4, and two possible transitions when we meet a 1 in state q_1.

Nondeterminism

The NFA is more intelligible than a DFA for the same language:



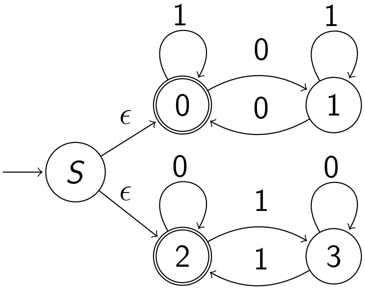
This is the simplest DFA that will do the job!

Epsilon Transitions

NFAs may also be allowed to move from one state to another without consuming input.

Such a transition is an ϵ transition.

Among other things, this gives us an easy way to construct a machine to recognise the union of two languages:



Formal Definition of NFA

For any alphabet Σ let Σ_ϵ denote $\Sigma \cup \{\epsilon\}$.

An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ are the accept states.

NFA Acceptance, Formally

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and let $w = v_1 v_2 \cdots v_n$ where each v_i is a member of Σ_ϵ.

N accepts w iff there exists a sequence of states $r_0, r_1, \ldots, r_n$, with each $r_i \in Q$, such that

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, v_{i+1})$ for $i = 0, \ldots, n-1$
3. $r_n \in F$

Next Lecture: Being Regular

More on regular languages in the next lecture.

In particular we shall see that NFAs are no more powerful than DFAs.