Théorie des Langages Rationnels

Introducing Automata

Adrien Pommellet, LRE

February 13, 2023

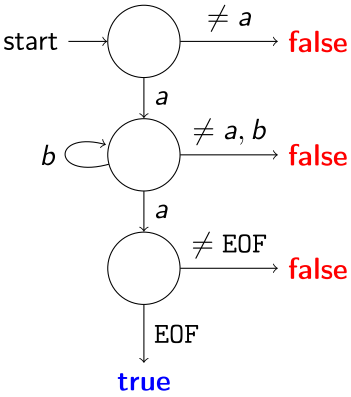
Practical Application

Exercise 1. Find an algorithm that decides the language of the regular expression ab^*a on $\Sigma = \{a, b, c\}$. This algorithm must be efficient: there is **no backtracking**, the word must be read from the left to the right in a single pass.

Answer I

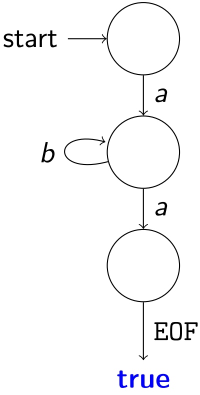
```
def algo(w):
  i = 0
  if w[i] != a:
    return false
  i += 1
  while w[i] == b
    i += 1
  if w[i] != a:
    return false
  i += 1
  if i == len(w):
    return true
  else:
    return false
```

A **graphical** representation



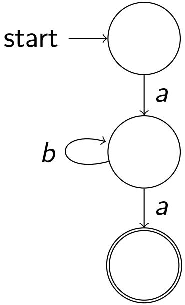
Answer II

We remove the edges leading to a **false** result and make them **implicit**: any letter that can't be matched to an existing edge triggers a **false** result.



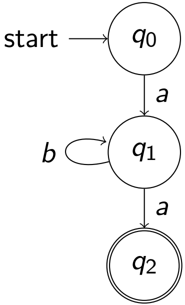
Answer III

We remove the `EOF` edge leading to the **true** result: instead, we mark the node as **accepting**.



Answer IV

Finally, we **name** the nodes so it is easier to identify them. The resulting structure is called a **finite automaton**.



Finite automaton

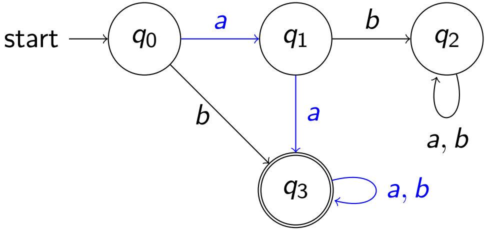
A finite automaton $\mathcal{A}$ is made of the five following components:

- A finite set Q of **states**. A state is a node of the graph.
- A finite **alphabet** Σ.
- A set of **edges** $\delta \subseteq Q \times \Sigma \times Q$: an edge is a triplet, and δ is a set of triplets.
- A set of **initial states** I.
- A set of **accepting states** F.

We then write $\mathcal{A} = (Q, \Sigma, \delta, I, F)$.

Finite Automata

Paths



A **path** labelled by a word $w = w_1 \dots w_n$ is a sequence of consecutive edges labelled by the letters $w_1, \dots, w_n$ **starting from an initial state**.

Here, $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_3 \xrightarrow{b} q_3$ is a path labelled by the word aab. We write $q_0 \xrightarrow{aab}{}^*_{\mathcal{A}} q_3$, meaning that using zero or more edges, we can reach q_3 from q_0 with a path labelled by aab.

Accepting a word

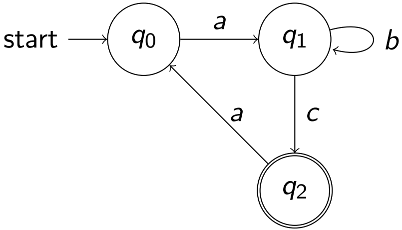
An automaton $\mathcal{A}$ accepts a word $w \in \Sigma$ if there exists a path from an initial state to a final state labelled by w. Such a path is said to be accepting.

Language of an automaton

The language $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \mathcal{A} \text{ accepts } w\}$ of an automaton $\mathcal{A}$ is the set of all words in Σ^* accepted by $\mathcal{A}$. $\mathcal{L}(\mathcal{A})$ is said to be **recognized** by $\mathcal{A}$.

Finite Automata

Refusing words



This automaton accepts the word abc. However:

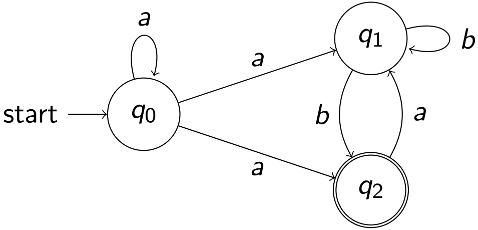
- The word aca is refused as the only path it labels does not **end** in an accepting path. **Passing through** an accepting state is not enough.
- The word acb is refused because there is **no path** starting from q_0 labelled by acb.

Thus, there are **two ways** to reject words.

Practical Application

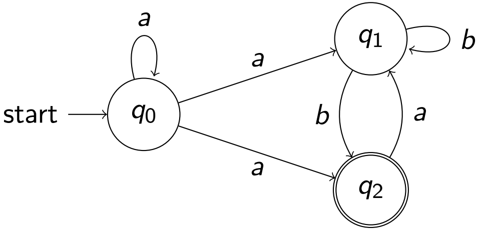
Exercise 2. Do the following words belong to the language of the automaton $\mathcal{A}$ below?

$$baba, abab, aaab, aaaa, \varepsilon, \text{ any word in } \mathcal{L}(a^* ab^* b)$$



Answer

The automaton $\mathcal{A}$ accepts *aaab*, *abab*, *aaaa*, and any word in $\mathcal{L}(a^*ab^*b)$ but refuses *baba* and ε.



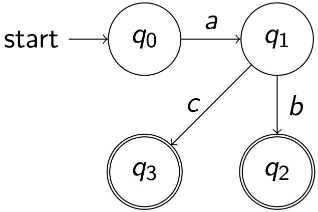
Properties of Finite Automata

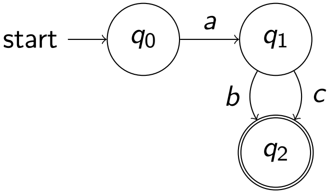
Equivalent automata

Equivalence

Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are said to be equivalent if $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

As an example, the two following automata are different yet equivalent:





Properties of Finite Automata

Deterministic automata

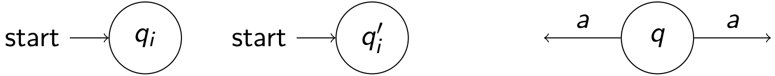
Determinism

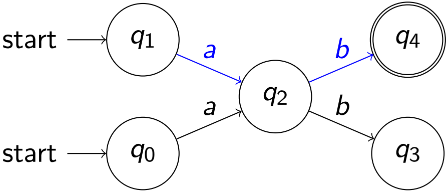
An automaton $\mathcal{A}$ is said to be deterministic and called a **DFA** if:

- It has **exactly one** initial state.
- For each letter a in Σ and each state q of $\mathcal{A}$, there is from q **at most one outgoing edge** labelled by a.

It is otherwise non-deterministic (and called a **NFA**).

The following patterns are **not** allowed:





A NFA may for a given word w feature multiple paths starting from an initial path and labelled by w. Here, there are **four** paths labelled by ab, only two of these being accepting.

A **single accepting path** is enough for the NFA to accept the word ab, regardless of the number of rejecting paths.

The previous definition leads to a simpler property:

Theorem

If an automaton $\mathcal{A}$ is deterministic, then for each word w in Σ^* there is ***at most one path*** *labelled by w.*

Intuitively, determinism applied to algorithms means that for each input, there is **a single, well-defined matching execution**. There is no **arbitrary choice** to be made.

Practical Application

Exercise 3. Find two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ on the alphabet $\Sigma = \{a, b\}$ such that:

- $\mathcal{A}_1$ recognizes the set of all words starting with ab.
- $\mathcal{A}_2$ recognizes the set of all words ending with ab.

Answer

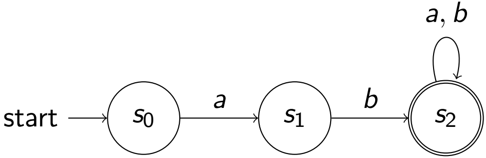


Figure 1: Automaton $\mathcal{A}_1$.

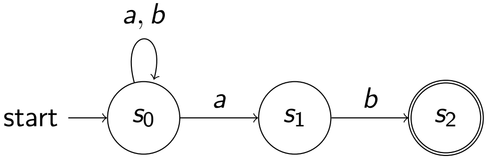


Figure 2: Automaton $\mathcal{A}_2$.

What About Determinism?

The automaton $\mathcal{A}_2$ is a NFA. But designing a DFA in that context is a more complex issue that we may answer later...

See you next class!