Théorie des Langages Rationnels

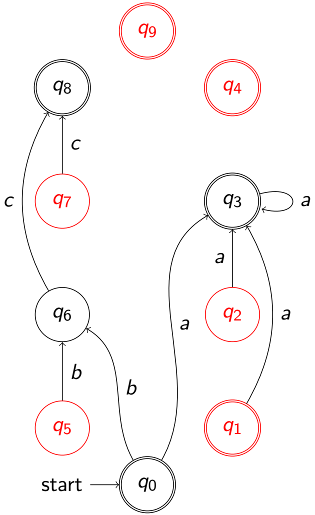
Pruning and Determinization

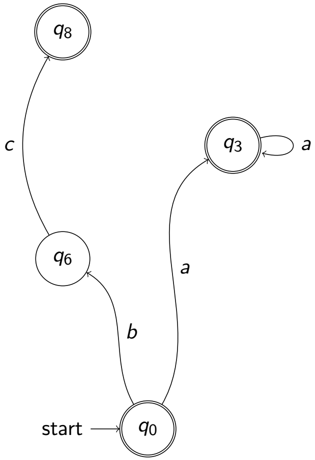
Adrien Pommellet, LRE

March 29, 2023

Pruning the Automaton

Two automata accepting $bc + a^*$

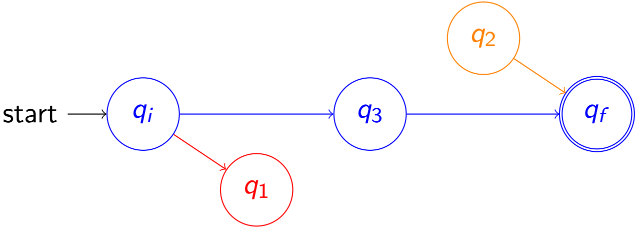




The NFA generated by Thompson's algorithm followed by the backward removal of ε-transitions are needlessly complicated. How can we make them **smaller**?

Pruning the Automaton

A property of accepting paths



A word w is only accepted if there is an **accepting path**, that is, a path from an initial state to a final state labelled by w.

Thus, a state that cannot lead to a final state or cannot be reached from an initial state will never belong to an accepting path.

Let us formalize this intuition:

Usefulness of states

Let q be a state of an automaton $\mathcal{A}$.

- q is said to be **accessible** if it can be reached from an initial state.
- q is said to be **co-accessible** if from q, a final state can be reached.
- q is said to be **useful** if it is both accessible and co-accessible. It is otherwise said to be **useless**.

In the previous example, states q_1 and q_2 are useless, but state q_3 is useful.

Pruning the Automaton

A simpler automaton

Pruning the Automaton

A simpler automaton

Theorem

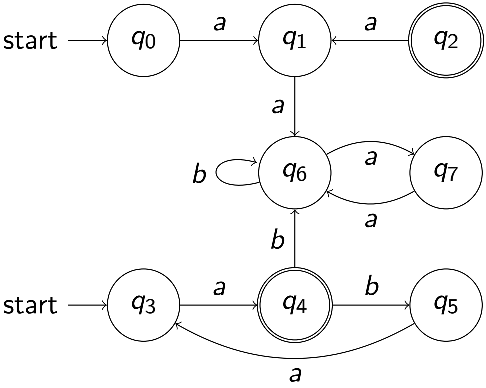
Let $\mathcal{A}$ be an automaton, and let $\mathcal{A}'$ be the automaton obtained by removing the useless states from $\mathcal{A}$ (also called the ***pruned*** *automaton). Then $\mathcal{A}$ and $\mathcal{A}'$ are equivalent.*

This theorem yields the following **pruning** algorithm:

1. Find the accessible states by performing a **depth-first or breadth-first search** from the initial states.
2. Find the co-accessible states by performing a search from the final states, **reversing the edges**.
3. Keep only the states that are both accessible and co-accessible.

Practical Application

Exercise 1. Prune the automaton $\mathcal{A}$ on the alphabet $\Sigma = \{a, b\}$ below.



Answer I

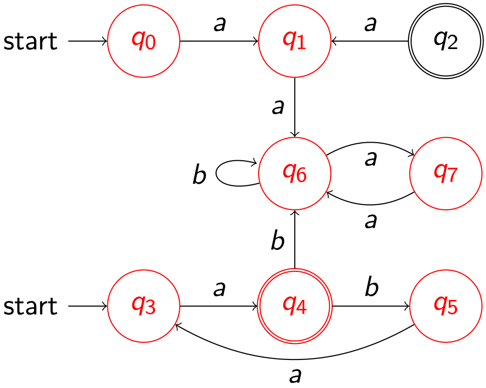


Figure 1: Accessible states of the automaton $\mathcal{A}$.

Answer II

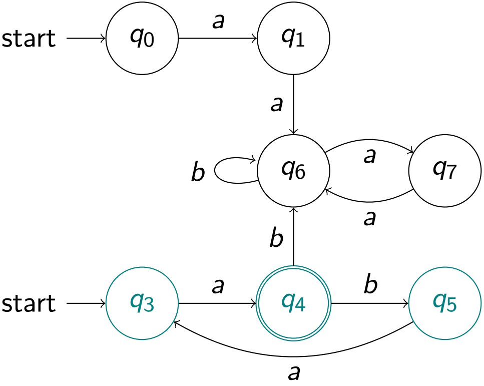


Figure 2: Co-accessible states of the partially pruned automaton $\mathcal{A}$.

Answer III

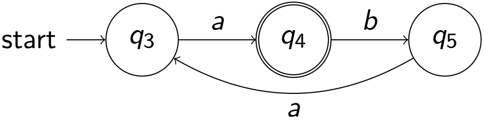
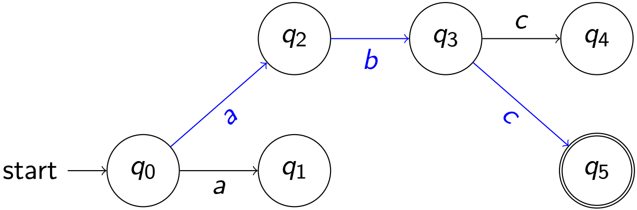


Figure 3: The pruned automaton $\mathcal{A}'$.

Determinizing a NFA

A reminder



A NFA may feature **multiple paths labelled by the same word**, but it only takes one accepting path to accept said word.

The NFA above accepts the word *abc*, but it may take **up to three attempts** before finding an accepting path.

Determinizing a NFA
A better model

NFA are not an efficient model: to check whether a word is accepted, one may have to browse an **arbitrary** number of paths.

On the other hand, a DFA features at most one path per word: checking whether a word is accepted or not takes takes a **linear** number of operations **in the size of the input**.

We will introduce an algorithm that proves the following theorem:

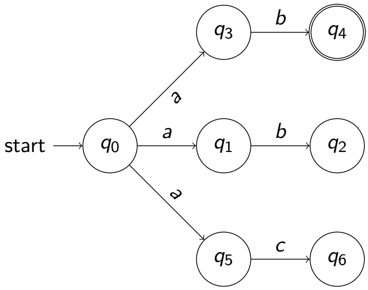
Theorem

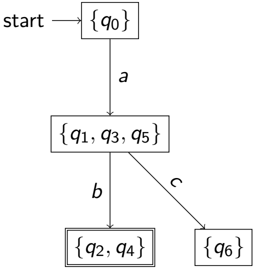
Given a NFA $\mathcal{A}$ on an alphabet Σ, there exists an equivalent DFA $\mathcal{A}'$ on Σ.

Determinizing a NFA

An intuition

We will use sets to list **all the possible states the automaton could be in**: $\{q_2, q_4\}$ stands for "being in state q_2 or in state q_4".





Determinizing a NFA

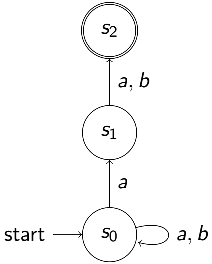
The theoretical powerset construction

1. The states of $\mathcal{A}'$ are labelled by sets of states of the original NFA $\mathcal{A}$.
2. The initial state of $\mathcal{A}'$ is labelled by I, the set of initial states of $\mathcal{A}$.
   A run may start in any initial state.
3. If a state of $\mathcal{A}'$ is labelled by a set S, then its successor by the letter $a \in \Sigma$ is the state labelled by $S_a = \{q \mid \exists p \in S, p \xrightarrow{a}_{\mathcal{A}} q\}$.
   *Note that S_a is the set of all states that can be reached with the letter a by **at least one** state of S, but not necessarily **every** state.*
4. A state of $\mathcal{A}'$ is accepting if and only if **at least one** accepting state of $\mathcal{A}$ belongs to its label.
   It only takes one accepting path to accept a word.

Determinizing a NFA

A practical implementation

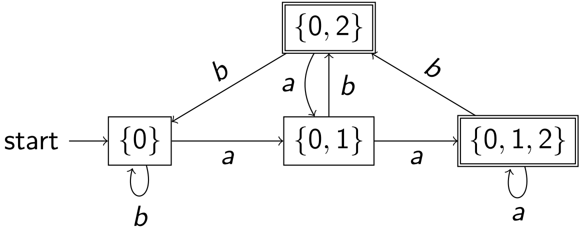
States	a	b
$I = \{0\}$	$\{0, 1\}$	$\{0\}$
$\{0, 1\}$	$\{0, 1, 2\}$	$\{0, 2\}$
$\{0, 1, 2\}$	$\{0, 1, 2\}$	$\{0, 2\}$
$\{0, 2\}$	$\{0, 1\}$	$\{0\}$



Determinizing a NFA

The result

Note that a same state of $\mathcal{A}$ **may appear in multiple labels**: these sets merely list the possible configurations at a given point of the execution.

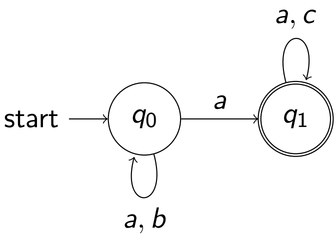


Determinizing a NFA

A practical summary

1. We will design a **table** representing $\mathcal{A}'$; its columns are labelled by the alphabet Σ, and its lines, by the sets labelling each state of $\mathcal{A}'$.
2. The first line is labelled by the set I of **initial states** of $\mathcal{A}$.
3. Write in cell (E, x) the set E_x of **successors** of E by letter x.
4. If a set E_x that has **not been explored yet** appears in a cell, add a new line labelled by E_x and compute its successors.
5. **Iterate** until there is no new set left to explore.
6. Any set whose label contains an **accepting** state of $\mathcal{A}$ is made accepting.

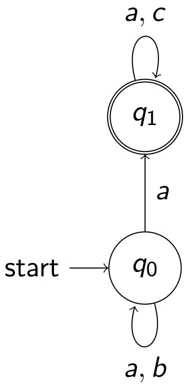
Exercise 2. Compute a DFA on the alphabet $\Sigma = \{a, b, c\}$ equivalent to the NFA below.



Determinizing a NFA

Answer I

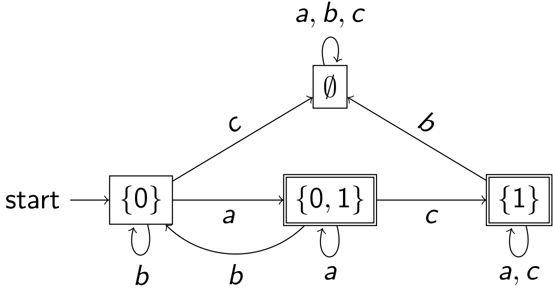
States	a	b	c
$I = \{0\}$	$\{0, 1\}$	$\{0\}$	$\emptyset$
$\{0, 1\}$	$\{0, 1\}$	$\{0\}$	$\{1\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\{1\}$	$\{1\}$	$\emptyset$	$\{1\}$



Determinizing a NFA

Answer II

A node labelled by $\emptyset$ may appear. It is a **sink state** that never accepts.



Determinizing a NFA

Properties of the resulting DFA

Theorem

Given a NFA $\mathcal{A}$ on an alphabet Σ with n states:

- *There exists a **complete**, equivalent DFA $\mathcal{A}'$ on Σ.*
- *This equivalent automaton may have **up to** 2^n **states**.*

Intuitively, if a set Q is of size n, the set of its subsets (also known as its **powerset**, written 2^Q) contains exactly 2^n elements.

Thus, if Q is the set of states of $\mathcal{A}$, since the states of $\mathcal{A}'$ are identified by labels in 2^Q, $\mathcal{A}'$ has at most 2^n states.

Note that the previous example has $4 = 2^2$ states and is complete.