Théorie des Langages Rationnels

Simplification of ε-NFA

Adrien Pommellet, LRE

March 15, 2023

The problem with ε-NFA

There must be a simpler answer.

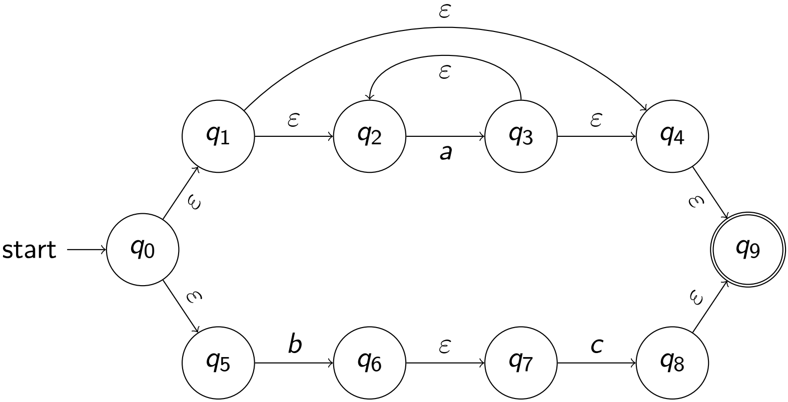
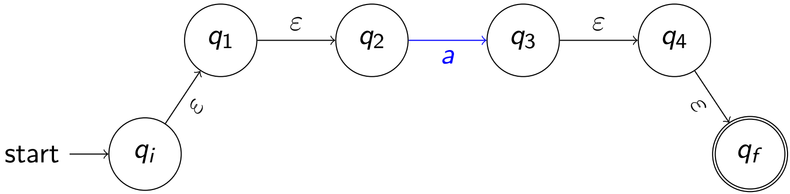


Figure 1: An automaton $\mathcal{A}$ recognizing $\mathcal{L}(a^* + bc)$.

The problem with ε-NFA

An inefficient model

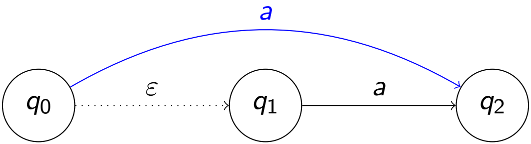
ε-NFA are flawed: a short word may require an **arbitrary long** path. Here, it takes 5 edges to accept the word a of length 1.



But in a finite automata without spontaneous transitions, a path labelled by a word w is **exactly as long** as w itself.

The problem with ε-NFA

Working around spontaneous transitions



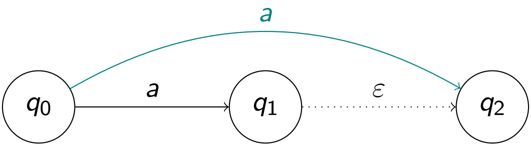
Consider a path $q_0 \xrightarrow{\varepsilon}{}^{*}_{\mathcal{A}} q_1$ and an edge $q_1 \xrightarrow{a}_{\mathcal{A}} q_2$ in an automaton.

An edge $q_0 \xrightarrow{a}_{\mathcal{A}} q_2$ could achieve the same result without using any spontaneous transition.

This pattern is known as the **backward removal of ε-transitions**.

The problem with ε-NFA

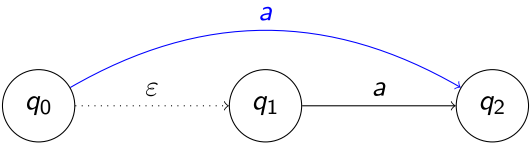
Forward removal



In a similar manner, we could perform a **forward elimination of spontaneous transitions**.

It is known as forward (resp. backward) because the ε-path is in front of (resp. behind) the actual edge labelled by a.

The Backward Removal Algorithm
A rough outline



Our algorithm should feature the following steps:

1. Find all the ε-paths $q_0 \xrightarrow{\varepsilon}^*_{\mathcal{A}} q_1$.
2. Find all the patterns $q_0 \xrightarrow{\varepsilon}^*_{\mathcal{A}} q_1 \xrightarrow{a} q_2$ and for each pattern add a relevant edge $q_0 \xrightarrow{a} q_2$.
3. Remove all the ε-transitions.

We define the set of states reachable from a given state using nothing but ε-transitions:

Forward ε-closure of a state

The forward ε-closure $\varepsilon^{\mathcal{A}}_{\text{forward}}(q)$ of a state q of an automaton $\mathcal{A}$ is the set $\{p \in Q \mid q \xrightarrow{\varepsilon}^*_{\mathcal{A}} p\}$.

Note that $q \xrightarrow{\varepsilon}^*_{\mathcal{A}} q$, always: from q we can reach q by not reading anything and using 0 edges. Thus $q \in \varepsilon^{\mathcal{A}}_{\text{forward}}(q)$.

We must compute this set for every state q of the automaton.

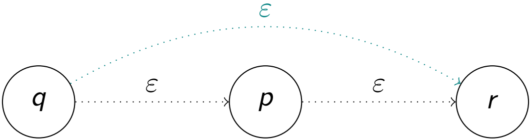
The Backward Removal Algorithm

An useful property

Theorem

If $p \in \varepsilon^{\mathcal{A}}_{forward}(q)$ and $r \in \varepsilon^{\mathcal{A}}_{forward}(p)$ then $r \in \varepsilon^{\mathcal{A}}_{forward}(q)$.

Indeed, remember that $\varepsilon \cdot \varepsilon = \varepsilon$. Therefore, if there is an ε-path from q to p and an ε-path from p to r, then there is an ε-path from q to r.



The Backward Removal Algorithm

Making our knowledge grow

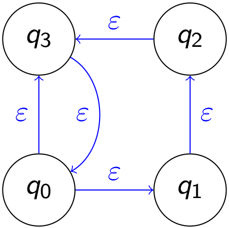
In order to compute a set E, it is sometimes possible to use a pattern known as an **iterative fixpoint algorithm**:

1. We know a non-empty set of **base cases** $B \subseteq E$.
2. We also known a set of **rules** R such that, given $e_1, \ldots, e_k \in E$ and a rule $r \in R$, we can build an element $r(e_1, \ldots, e_k) \in E$.
3. We compute an **increasing sequence of sets** $(S_i)_{i \geq 0}$ such that $S_0 = B$ and $\forall i \geq 0$, $S_i \subseteq S_{i+1} \subseteq E$, where S_{i+1} has been grown from S_i by applying rules from R to existing elements in S_i.
4. If there exists a rank n such that $S_n = S_{n+1}$, we say that **a fixed point S_n has been reached**.
5. Assuming the algorithm is well-designed, $S_n = E$.

The Backward Removal Algorithm

Computing the forward ε-closure

States	Step 0	Step 1	Step 2	Step 3
0	0 1 3	0 1 3 2	0 1 3 2	0 1 3 2
1	1 2	1 2 3	1 2 3 0	1 2 3 0
2	2 3	2 3 0	2 3 0 1	2 3 0 1
3	3 0	3 0 1	3 0 1 2	3 0 1 2



The Backward Removal Algorithm

A summary of the closure algorithm

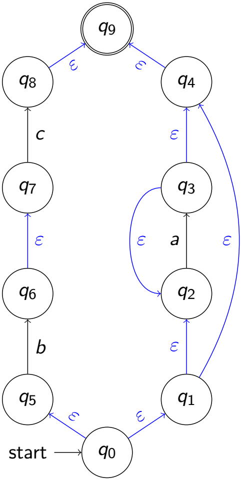
1. We design a **table** with $|Q|$ lines, one for each state of $\mathcal{A}$.
   Cell (q, i) will contain our knowledge of $\varepsilon^{\mathcal{A}}_{forward}(q)$ after i iterations.
2. For each state q, we write in cell $(q, 0)$ the state q itself as well as any state p such that there exists an edge $q \xrightarrow{\varepsilon}_{\mathcal{A}} p$.
   *We **initialize** the table with knowledge we can directly infer from the edges and the states of the automaton itself.*
3. Assuming column i is known, we compute column $i+1$ by adding to cell $(q, i+1)$ the content of cell (p, i) for each p in cell (q, i).
   *We extend our knowledge by **combining** ε-paths identified previously.*
4. We iterate until columns i and $i+1$ are the same.
   *We reach a fixpoint when **no new extra information** can be added to the table.*

Practical Application

Exercise 1. Compute $\varepsilon^{\mathcal{A}}_{\mathsf{forward}}(q)$ for each state q of the automaton $\mathcal{A}$ of Figure 1.

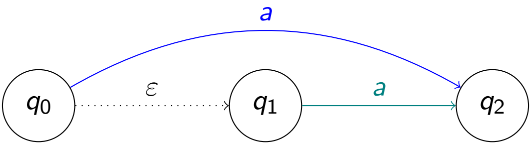
Answer

States	Step 0	Step 1	Step 2	Step 3
0	0 1 5	0 1 5 2 4	0 1 5 2 4 9	0 1 5 2 4 9
1	1 2 4	1 2 4 9	1 2 4 9	1 2 4 9
2	2	2	2	2
3	3 2 4	3 2 4 9	3 2 4 9	3 2 4 9
4	4 9	4 9	4 9	4 9
5	5	5	5	5
6	6 7	6 7	6 7	6 7
7	7	7	7	7
8	8 9	8 9	8 9	8 9
9	9	9	9	9



The Backward Removal Algorithm

The next step

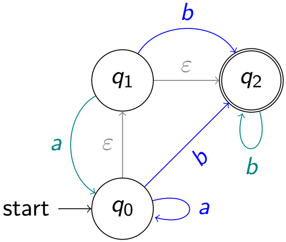


Note that if $q_1 \in \varepsilon^{\mathcal{A}}_{\text{forward}}(q_0)$ and there exists an edge $q_1 \xrightarrow{a}_{\mathcal{A}} q_2$, then we can add an edge $q_0 \xrightarrow{a}_{\mathcal{A}} q_2$.

The Backward Removal Algorithm

Computing the NFA

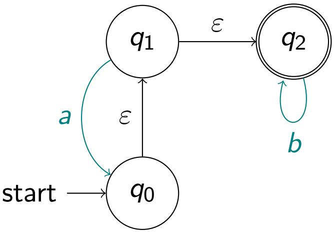
States	$\varepsilon^{\mathcal{A}}_{\text{forward}}$
0	0 1 2
1	1 2
2	2

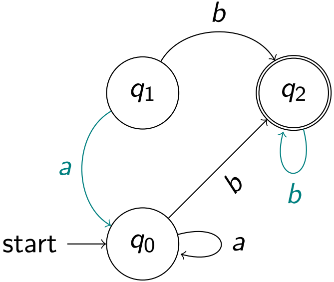


The Backward Removal Algorithm

It's not over yet

The original automaton accepts a, but the new one **does not**. We need to fix our algorithm.

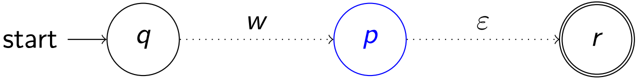




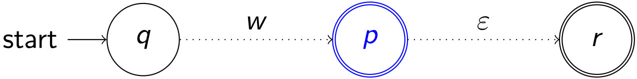
The Backward Removal Algorithm

New accepting states

If there is an accepting state r in the forward ε-closure of a state p, then any path $q \xrightarrow{w}^*_{\mathcal{A}} p$ can be extended to an accepting path $q \xrightarrow{w}^*_{\mathcal{A}} r$ with the same label.

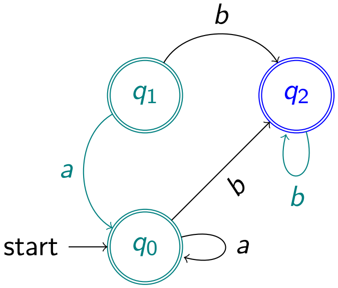


Thus, we may treat p as **being accepting as well**.



We will therefore apply the following rule: if $r \in \varepsilon^{\mathcal{A}}_{\text{forward}}(p)$ is an accepting state of the original automaton, then we should make p accepting as well.

States	$\varepsilon^{\mathcal{A}}_{\text{forward}}$
0	0 1 2
1	1 2
2	2



The Backward Removal Algorithm

A summary of the final step

1. We remove all the existing ε-transitions, and we only keep the edges whose label is in Σ.
2. For each state q and each $p \in \varepsilon^{\mathcal{A}}_{\text{forward}}(q)$, if there's an existing edge $p \xrightarrow{x}_{\mathcal{A}} r$ in the original automaton, then we add an edge $q \xrightarrow{x}_{\mathcal{A}} r$.
3. if $r \in \varepsilon^{\mathcal{A}}_{\text{forward}}(p)$ is an **accepting state of the original automaton**, then we should make p **accepting as well**.

The Backward Removal Algorithm

The whole algorithm

1. Compute the closure $\varepsilon^{\mathcal{A}}_{\text{forward}}$ of the ε-NFA $\mathcal{A}$.
2. Remove all the ε-transitions, then add new edges to the automaton based on $\varepsilon^{\mathcal{A}}_{\text{forward}}$ and the remaining original edges.
3. Depending on $\varepsilon^{\mathcal{A}}_{\text{forward}}$, you may have to change some states to be accepting.

As a consequence:

Theorem

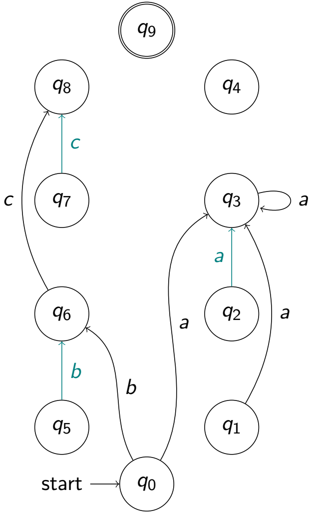
Given a ε-NFA $\mathcal{A}$ on the alphabet Σ, there exists an equivalent NFA $\mathcal{A}'$ on Σ with ***the same number of states****.*

Practical Application

Exercise 2. Use $\varepsilon^{\mathcal{A}}_{\text{forward}}$ to convert the automaton $\mathcal{A}$ of Figure 1 into an equivalent NFA.

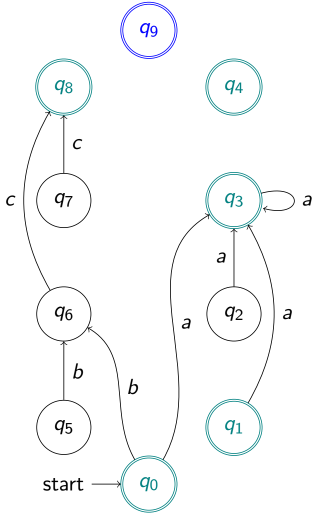
Answer I

States	$\varepsilon^{\mathcal{A}}_{\text{forward}}$
0	0 1 5 2 4 9
1	1 2 4 9
2	2
3	3 2 4 9
4	4 9
5	5
6	6 7
7	7
8	8 9
9	9



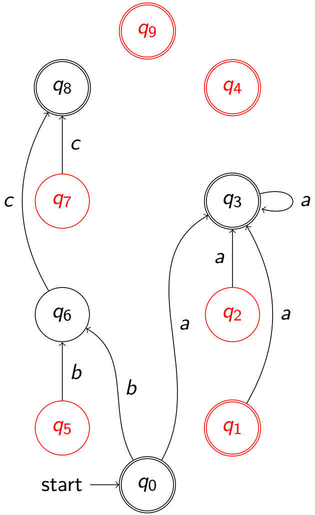
Answer II

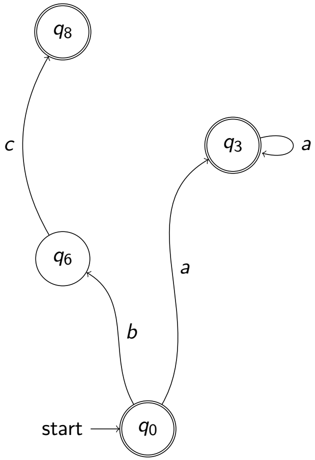
States	$\varepsilon^{\mathcal{A}}_{\text{forward}}$
0	0 1 5 2 4 9
1	1 2 4 9
2	2
3	3 2 4 9
4	4 9
5	5
6	6 7
7	7
8	8 9
9	9



The Backward Removal Algorithm

Simplifying the solution





See you next class!