Building the Minimal Automaton of A^*X in Linear Time, when X is of Bounded Cardinality

Omar AitMous [1] Frédérique Bassino [1] Cyril Nicaud [2]

[1]LIPN UMR 7030
Université Paris 13

[2]LIGM UMR CNRS 8049
Université Paris Est

23 June 2010

Introduction

Let $\mathcal{T}$ be a text.

- Pattern = a non-empty language which does not contain the empty word ε.
- Pattern matching = finding occurrences of words of the pattern in the text $\mathcal{T}$.

We might be interested in :

- the appearance of the pattern.
- the set of positions where the pattern occurs.
- the number of occurrences.
- ...

Introduction

A usual approach, given a pattern X over an alphabet A,

- build a deterministic automaton recognizing A^*X.
- use it to sequentially treat the text.

We can reasonably consider three categories of patterns:

- a single word.
- a finite set of words.
- a regular language.

Introduction

Depending on the nature of the pattern, the usual algorithms build a deterministic automaton that is not necessary minimal.

pattern	algorithm	automaton
single word	KMP ’77	minimal
finite set of words	Aho-Corasick ’75	deterministic
regular language	Mohri ’97	deterministic

Introduction

We consider the case of a set of m non-empty words whose sum of lengths is n, where m is fixed and n tends towards infinity.

$\Longrightarrow$ how to efficiently build the minimal automaton recognizing the language A^*X ?

Aho-Corasick Algorithm

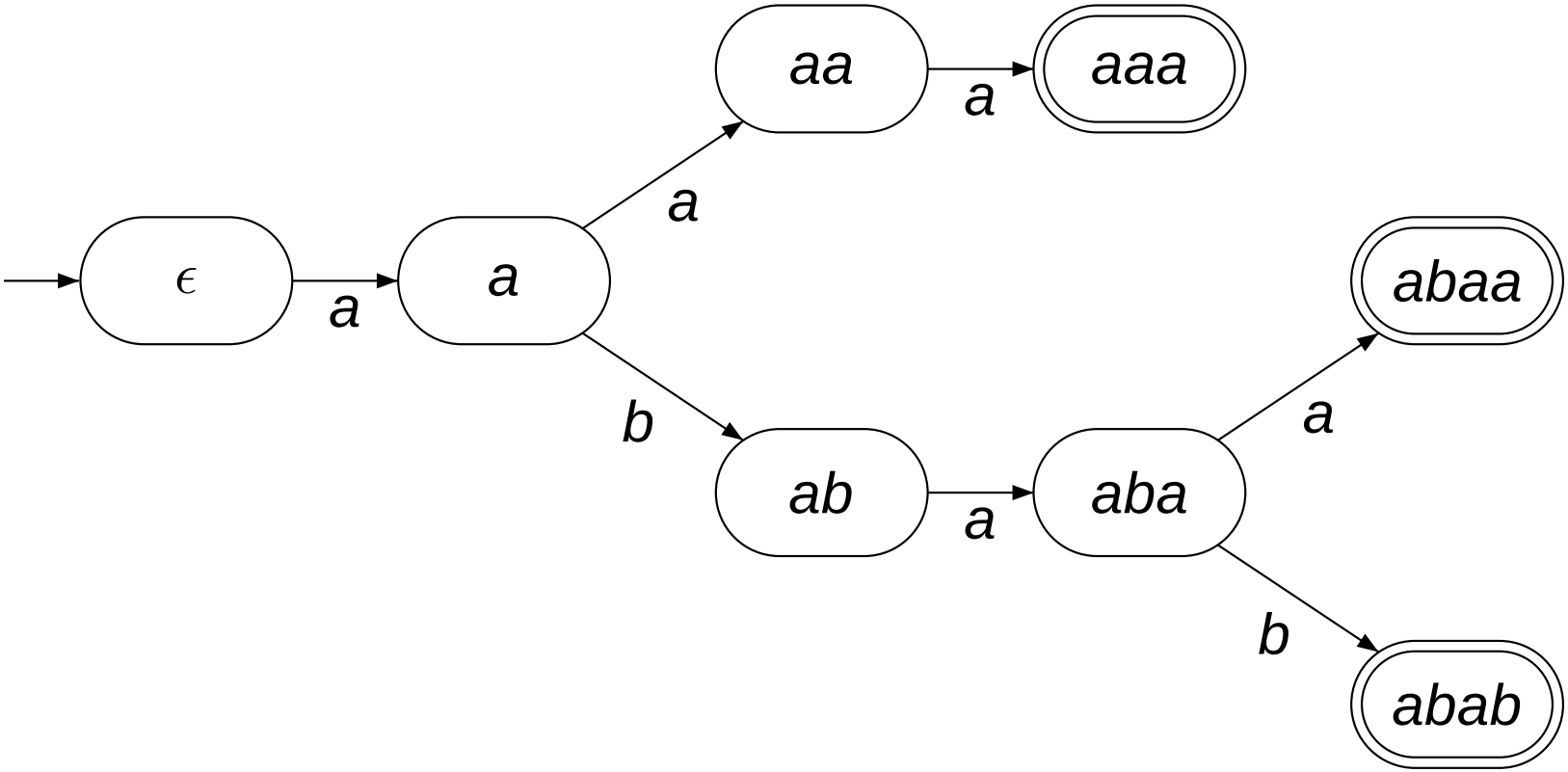
First step : A co-deterministic automaton recognizing A^*X

Second step : The minimal automaton

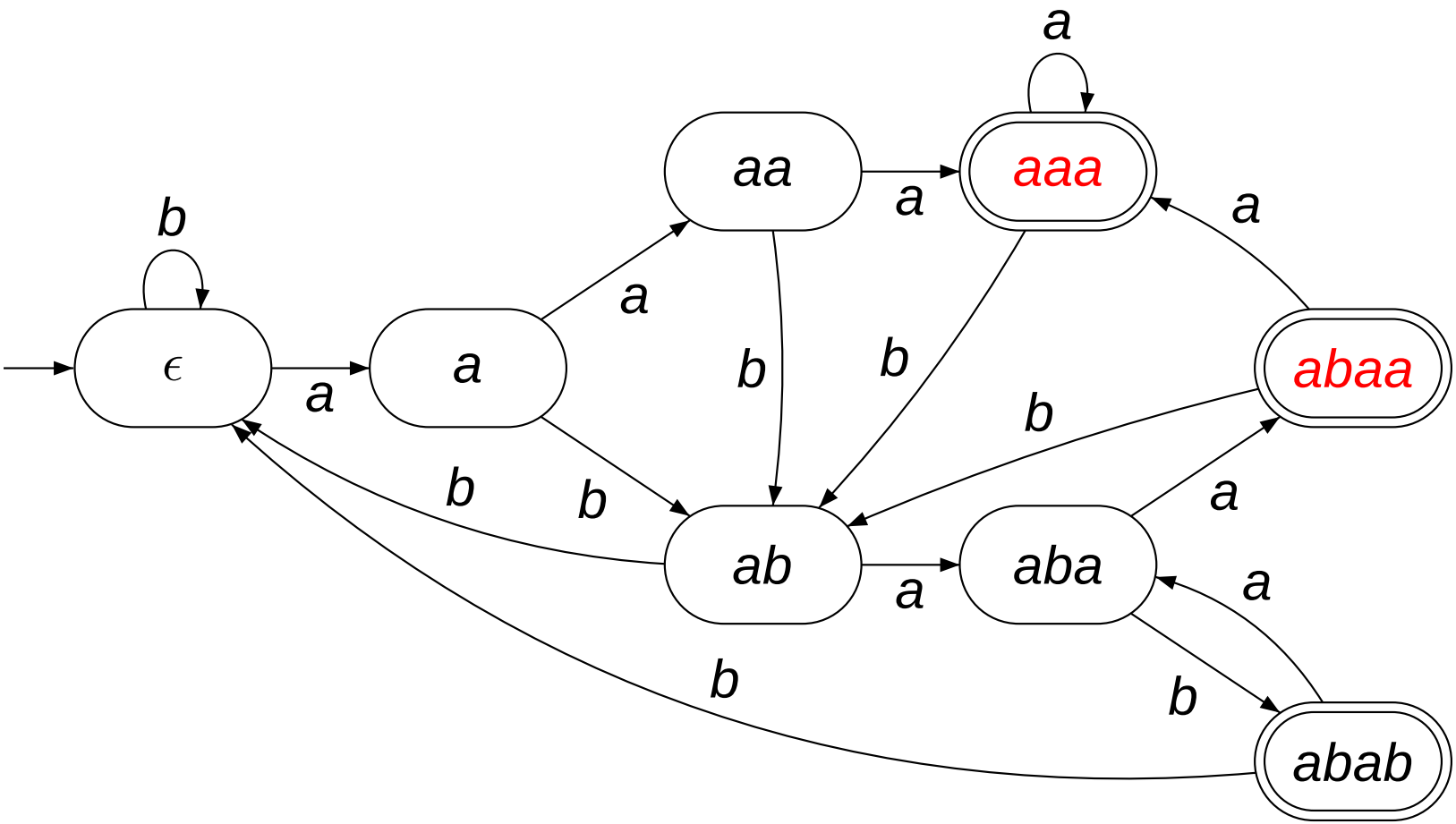
Complexity

Conclusion

Aho-Corasick Algorithm [Aho Corasick'75]



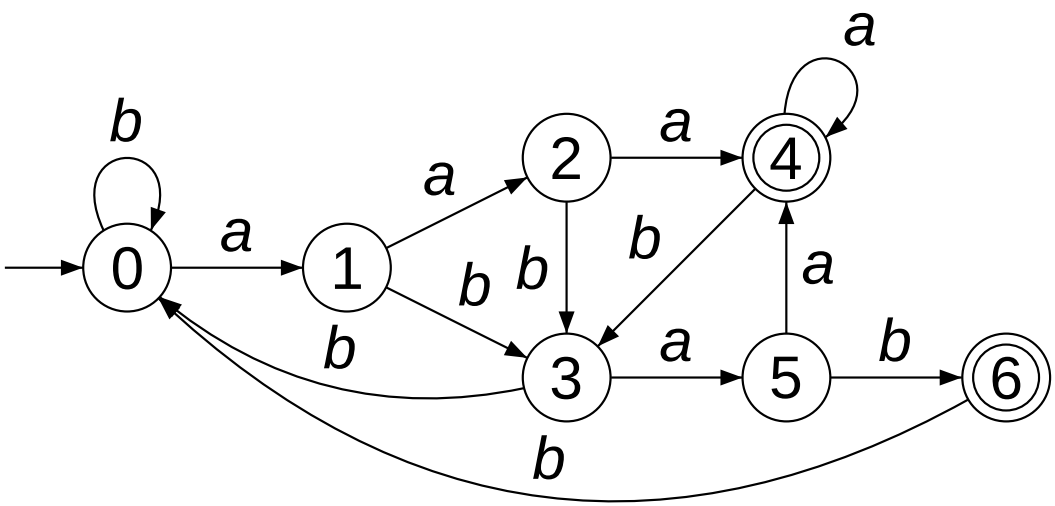
Automaton $\mathcal{A}(X)$, where $A = \{a, b\}$ and $X = \{aaa, abaa, abab\}$.



Deterministic automaton $\mathcal{D}(X)$, where $A = \{a, b\}$ and $X = \{aaa, abaa, abab\}$.

Is the automaton minimal?

Aho-Corasick automaton is not minimal in the general case.



Minimal automaton recognizing A^*X, where $X = \{aaa, abaa, abab\}$.

Experimentally the **probability** for Aho-Corasick automaton to be minimal is **very small** when n is large.

Example: $|A| \geq 10, m \geq 5$ and $n \geq 1000$

the probability is lower than **0.05**

To build the minimal automaton :

$$X \xrightarrow[O(n)]{Aho-Corasick} \mathcal{D}(X) \xrightarrow[O(n \log n)]{Hopcroft} \mathcal{M}in(X)$$

Our contribution

Algorithm to directly build the minimal automaton A^*X

- time complexity : $\mathcal{O}(n)$
- space complexity
  - sparse lists $\rightarrow \mathcal{O}(n^2)$
  - hash functions $\rightarrow \mathcal{O}(n)$

Brzozowski's algorithm [Brzozowski'62]

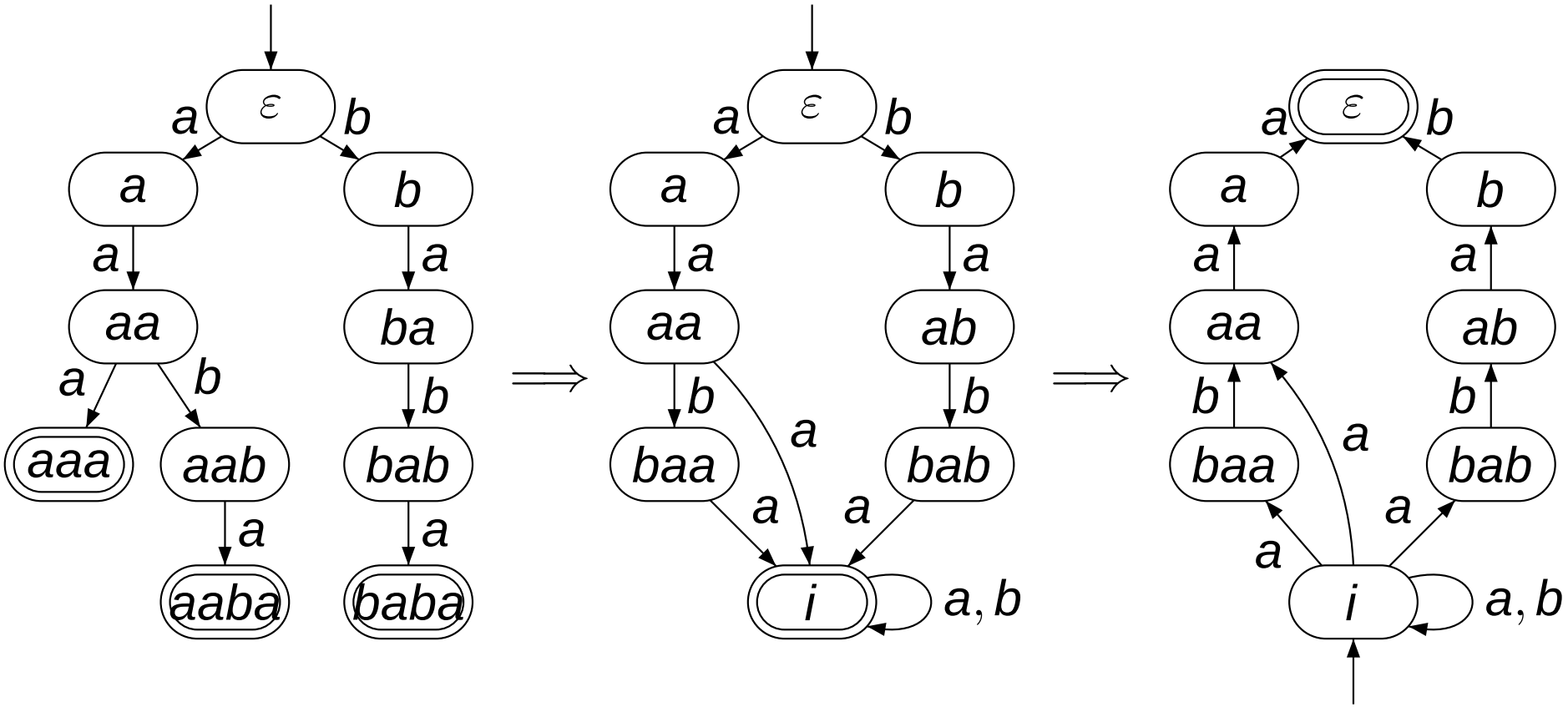
- Given a non-deterministic automaton $\mathcal{A}_X$ recognizing A^*X
- **Brzozowski's algorithm**
  - Co-deterministic automaton $\mathcal{C}_X$ recognizing A^*X
    - reverse $\mathcal{A}_X$.
    - determinize it.
    - reverse the result.
  - Minimal automaton recognizing A^*X
    - determinize $\mathcal{C}_X$.
- Our adaptation of Brzozowski's algorithm:
  - direct construction of a co-deterministic automaton recognizing A^*X
  - ad-hoc algorithm to determinize $\mathcal{C}_X$

Proposition

Let X be a set of m non-empty words whose sum of lengths is n. There exists a deterministic automaton recognizing the language $\widetilde{X}A^*$ whose number of states is at most $n - m + 2$, where $\widetilde{X}$ is the set of mirror words of words in X.

This automaton is then reversed to obtain a co-deterministic automaton recognizing A^*X.

Example



Co-deterministic automaton $\mathcal{C}_X$ recognizing A^*X, where $X = \{aaa, abaa, abab\}$.

Summary

- Construction of $\mathcal{C}_X$: linear time and space complexities
- Determinization of $\mathcal{C}_X$ produces the minimal automaton (Brzozowzki)
- The minimal automaton has at most $n + 1$ states (Aho-Corasick)

Subset construction

Any finite automaton $\mathcal{A} = (A, Q, I, F, \delta)$ can be transformed by the *subset construction* into a deterministic automaton $(A, \mathcal{P}(Q), \{I\}, F_{\mathcal{M}}, \delta_{\mathcal{M}})$ recognizing the same language and in which:

- $F_{\mathcal{M}} = \{P \in \mathcal{P}(Q) \mid P \cap F \neq \emptyset\}$
- $\delta_{\mathcal{M}}(P, a) = \bigcup_{p \in P} \delta(p, a)$

We denote by $\mathcal{M}$ the accessible and complete part of this automaton.

To build the automaton $\mathcal{M}$,

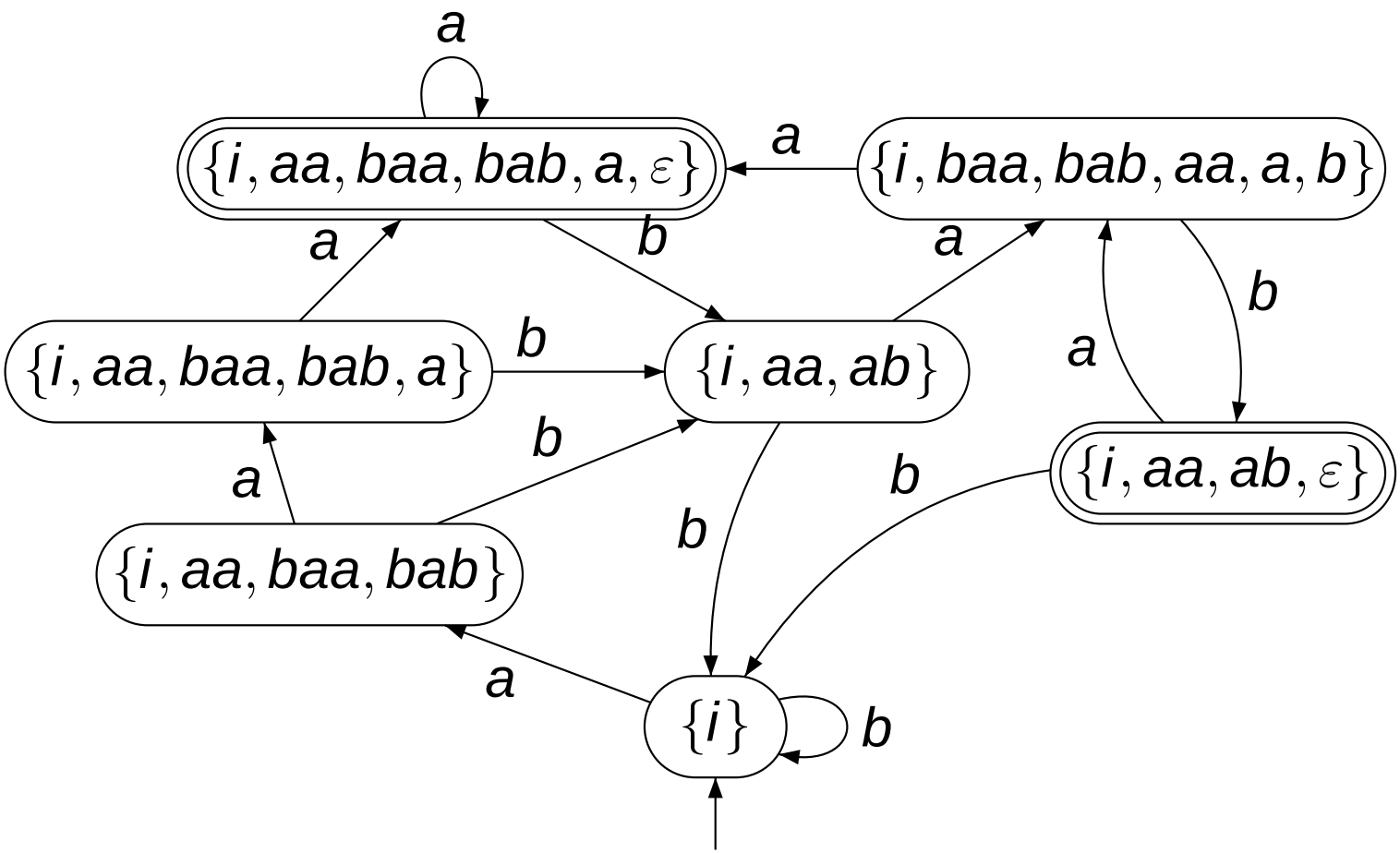
- start from the initial state I
- build the accessible states

For each state $P \in \mathcal{P}(Q)$,

- compute $\delta_{\mathcal{M}}(P, a)$ (iteration over the elements in P)
- check whether the state $\delta_{\mathcal{M}}(P, a)$ has already been created

$\Longrightarrow$ The time complexity is at least linear in $\sum_P |P|$

Example (by *subset construction*)



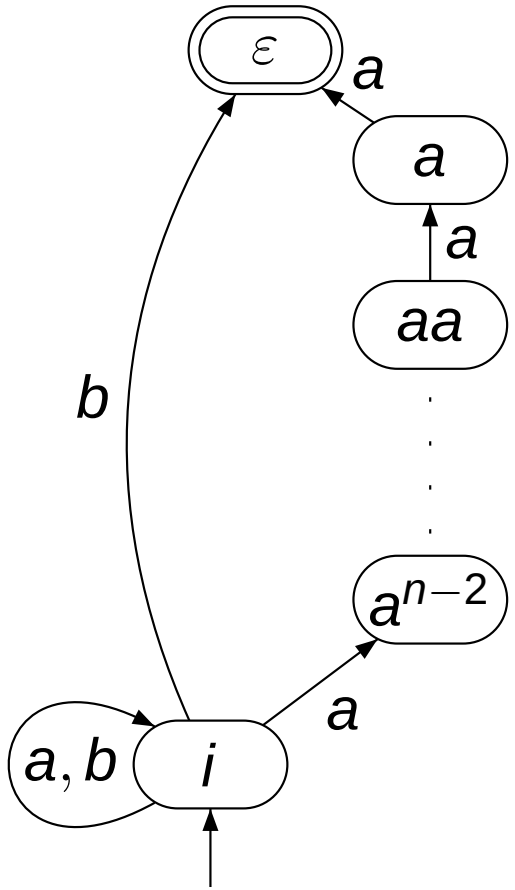
Minimal automaton recognizing A^*X, $X = \{aaa, abaa, abab\}$.

Cost of the construction

Let $X = \{a^{n-1}, b\}$ be a pattern.

To determinize this automaton, we compute:

- $\delta_{\mathcal{M}}(\{i\}, a) = \{i, a^{n-2}\}$
- $\delta_{\mathcal{M}}(\{i, a^{n-2}\}, a) = \{i, a^{n-2}, a^{n-3}\}$
- ...



The states of the minimal automaton are labeled : $\{i\}$, $\{i, a^{n-2}\}$, $\{i, a^{n-2}, a^{n-3}\}$, ..., $\{i, a^{n-2}, a^{n-3}, \ldots, a\}$, $\{i, a^{n-2}, a^{n-3}, \ldots, a, \varepsilon\}$ and $\{i, \varepsilon\}$.

Hence, $\sum_P |P| = \Omega(n^2)$,

$\Longrightarrow$ the complexity is quadratic in the worst case.

Alternative : limit the size of the labels.

Idea of the construction

- given an automaton $\mathcal{C}_X$
- separate the words in X

$$\{i\}, \{i\}, \{i\}$$

- if the letter a is read, we reach the states

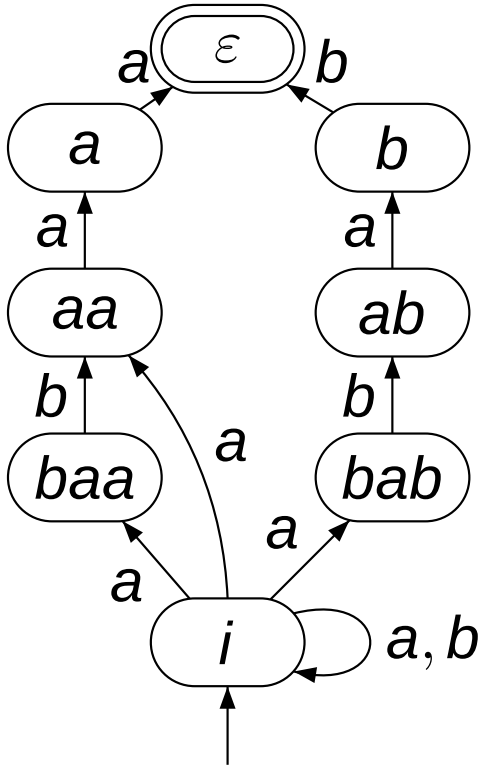
$$\{i, aa\}, \{i, baa\}, \{i, bab\}$$

- if we read again a,

$$\{i, aa, a\}, \{i, baa\}, \{i, bab\}$$

For each word, we store the farthest state from i,

$$a, baa, bab$$



Automaton $\mathcal{C}_X$, where $X = \{aaa, abaa, abab\}$.

To compute a transition by a

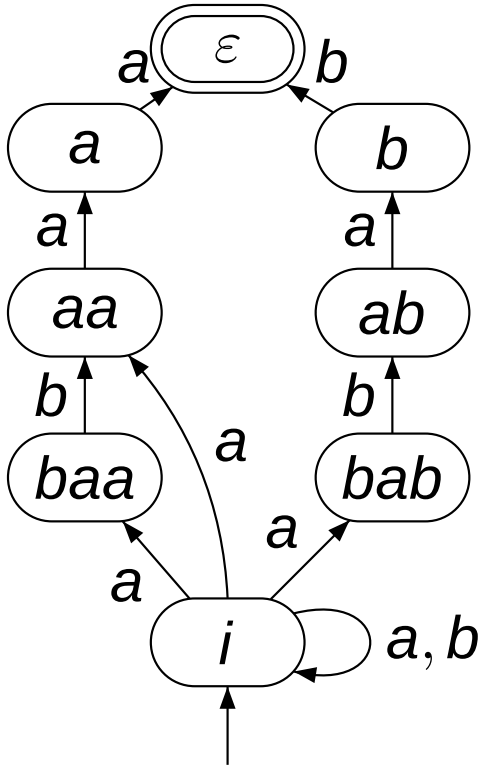
- we start from: a, baa, bab
- for each word, if the transition does not exist
  - we fail to the previous state

Thus by reading the letter a, we reach:

$$\varepsilon, baa, bab$$

et en lisant b

$$i, aa, ab$$



Automaton $\mathcal{C}_X$, where $X = \{aaa, abaa, abab\}$.

Failure function f

We define a *failure* function $f(u, p, a)$, for $u \in X$, $p \in Suff(u) \setminus \{u, \varepsilon\}$ (there exists w such that $u = w \cdot p$) and $a \in A$, by:

$$f(u,p,a) = \begin{cases} p & \text{if } \delta(p,a) \text{ is defined and } \delta(p,a) \neq \varepsilon \\ q & \text{if } |Border(w \cdot a)| \geq 2 \text{ and } Border(w \cdot a) = q \cdot a \\ i & \text{otherwise.} \end{cases} \quad (1)$$

Extented border array

Let u be a word of length ℓ.

We define an *extended* border array from $\{0, 1, \ldots, \ell - 1\} \times A$ to $\{0, 1, \ldots, \ell - 1\}$ by:

- $border\_ext[0][u[0]] = 0$
- $border\_ext[i][a] = |Border(u[0 \cdot \cdot i - 1] \cdot a)|$ for all $i \in \{1, \ldots, \ell - 1\}$ and $a \in A$

Proposition

The extented border array is computed in linear time and space (for $|A|$ fixed).

Example

We compute the extended border array for every word in $X = \{aaa, abaa, abab\}$.
Bold values are those of a standard border array.

Letter	Prefix w of u_2 such that $w \neq u_2$			
	ε	a	ab	aba
a	**0**	1	**1**	**1**
b	/	**0**	0	2

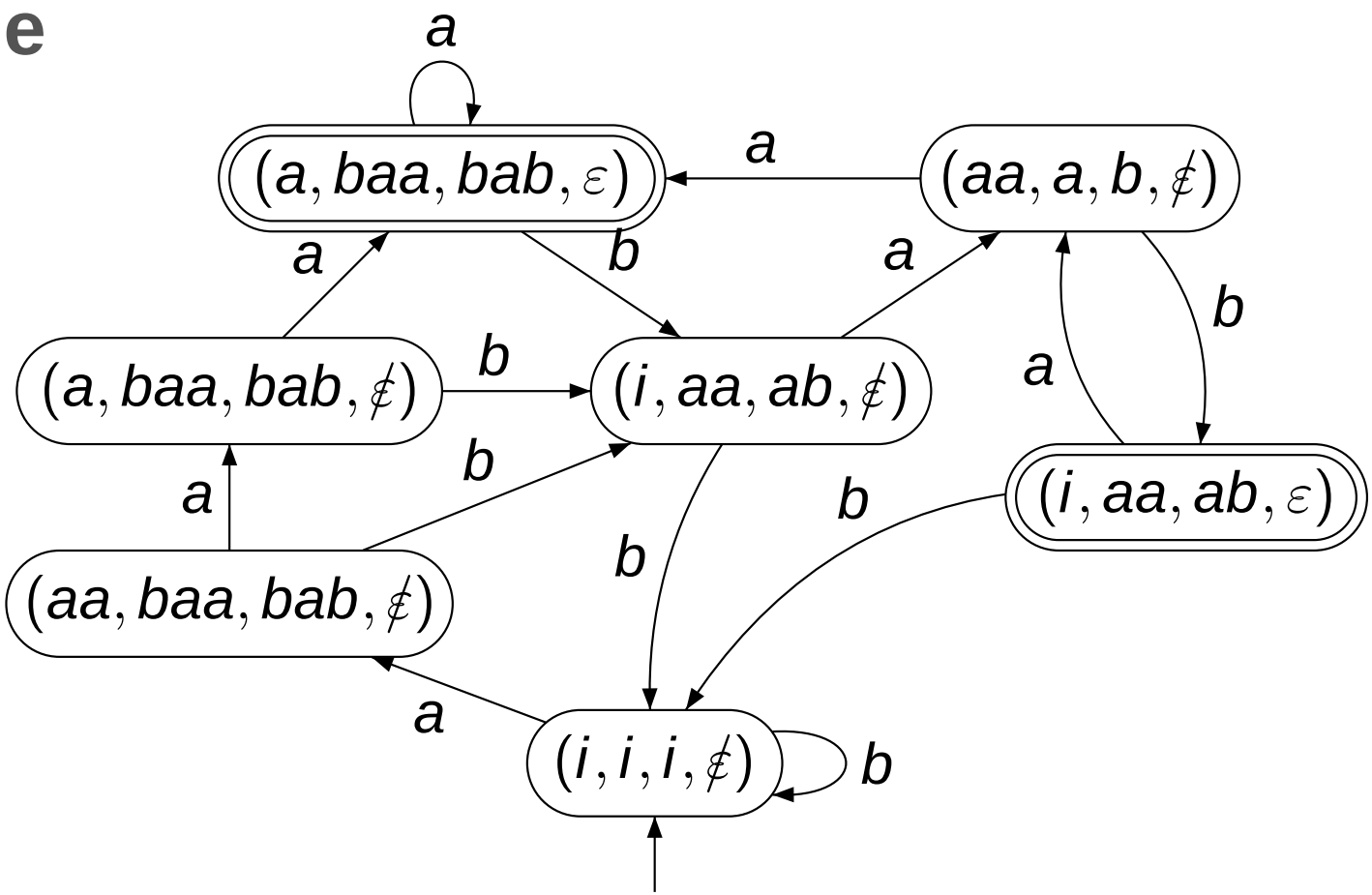
Extended border array of the word $u_2 = abaa \in X$.

Determinization algorithm

Let $\mathcal{C}_X = (A, Q, \{i\}, \{\varepsilon\}, \delta)$ be a co-deterministic automaton recognizing A^*X, where $X = \{u_1, \ldots, u_m\}$. We denote by $\mathcal{B}_X$ the accessible part of the automaton $(A, I_{\mathcal{B}}, Q_{\mathcal{B}}, F_{\mathcal{B}}, \delta_{\mathcal{B}})$, where:

- $Q_{\mathcal{B}} = (Q \setminus \{\varepsilon\})^m \times \{\varepsilon, \notin\}$,
- $I_{\mathcal{B}} = \{(i, \ldots, i, \notin)\}$
- for all $P \in F_{\mathcal{B}}$, $P = (v_1, v_2, \ldots, v_m, \varepsilon)$, where $v_r \in Q \setminus \{\varepsilon\}$ for all $r \in \{1, \ldots, m\}$.

Example



Minimal automaton $\mathcal{B}_X$ recognizing A^*X, where $X = \{aaa, abaa, abab\}$.

Theorem

Theorem

$\mathcal{B}_X$ is the minimal automaton recognizing A^*X.

Summary

Let $X = \{u_1, \ldots, u_m\}$ be a set of m words whose sum of lengths is n.
Recall that m is fixed.

- co-deterministic automaton $\mathcal{C}_X$ recognizing A^*X
  - $\mathcal{O}(n)$ time and space complexities
- the determinization of $\mathcal{C}_X$ produces the minimal automaton $\mathcal{B}_X$
  - $\mathcal{O}(n)$ states labeled by sequences of length $m + 1$

During the determinization process, *sparse lists* are used to store the states.

Sparse list

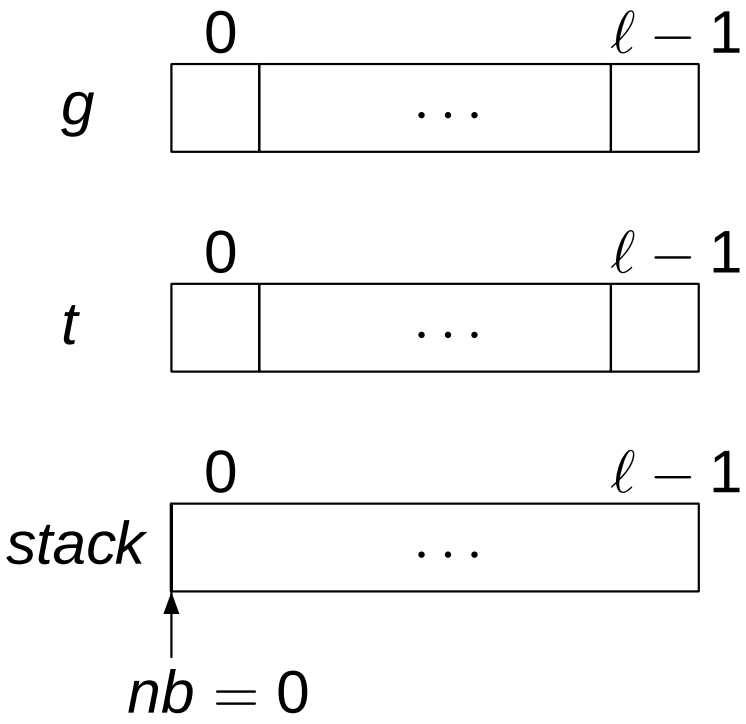
Let $g : \{0, \ldots, \ell - 1\} \to F$ be a partial function and denote by $Dom(g)$ the domain of g.

A *sparse list* is a data structure that implements g and performs the following operations in constant time:

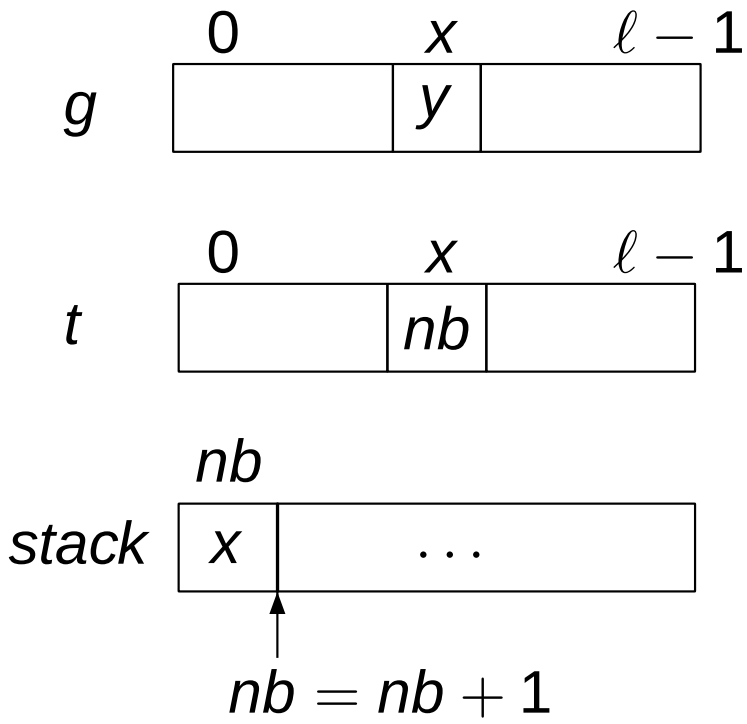
- initializing g with $Dom(g) = \emptyset$ (malloc)
- setting a value $g(x)$ for a given $x \in \{0, \ldots, \ell - 1\}$
- testing whether $g(x)$ is defined or not
- finding the value for $g(x)$ if it is defined
- removing x from $Dom(g)$

The space complexity of a sparse list is $\mathcal{O}(\ell)$.

Sparse list

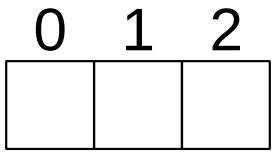


Initialization.



Insertion of $g(x) = y$.

State insertion

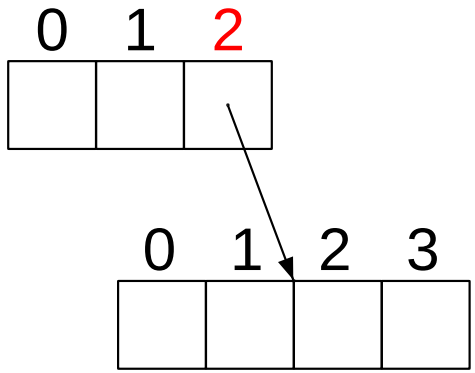


$X = \{aaa, abaa, abab\}$.

$P = (aa, baa, bab, \notin)$.

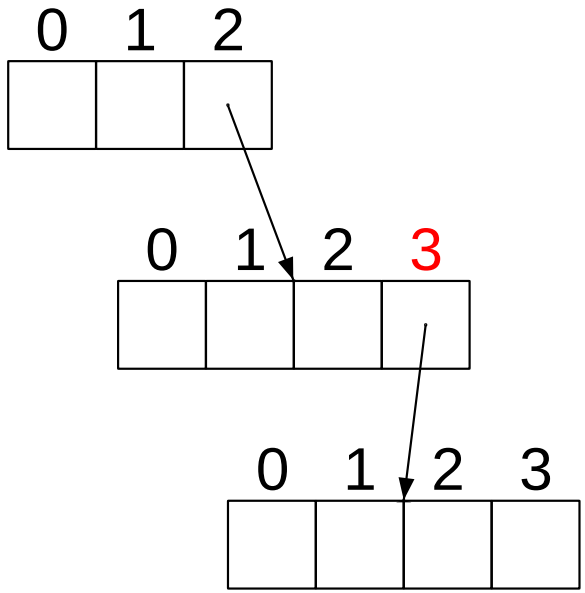
$X = \{aaa, abaa, abab\}$.

$P = (aa, baa, bab, \notin)$.



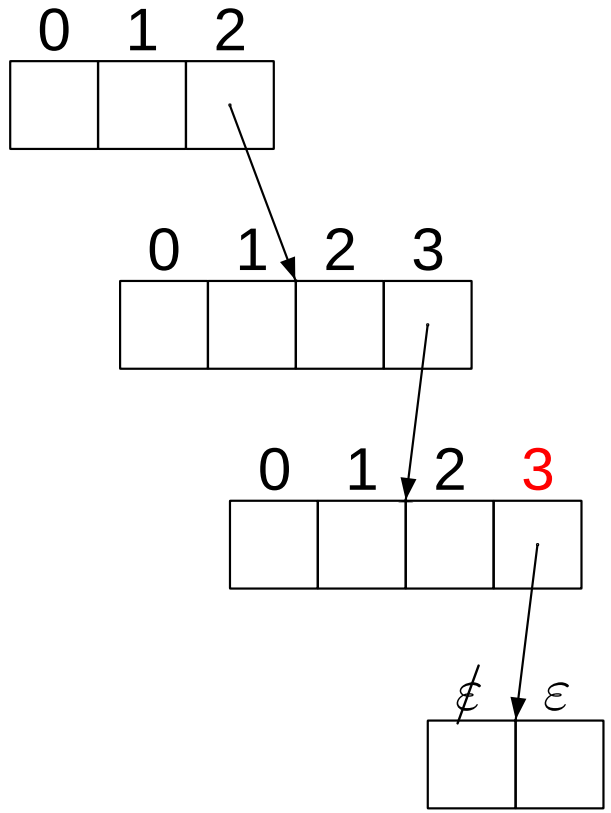
$X = \{aaa, abaa, abab\}$.

$P = (aa, baa, bab, \notin)$.



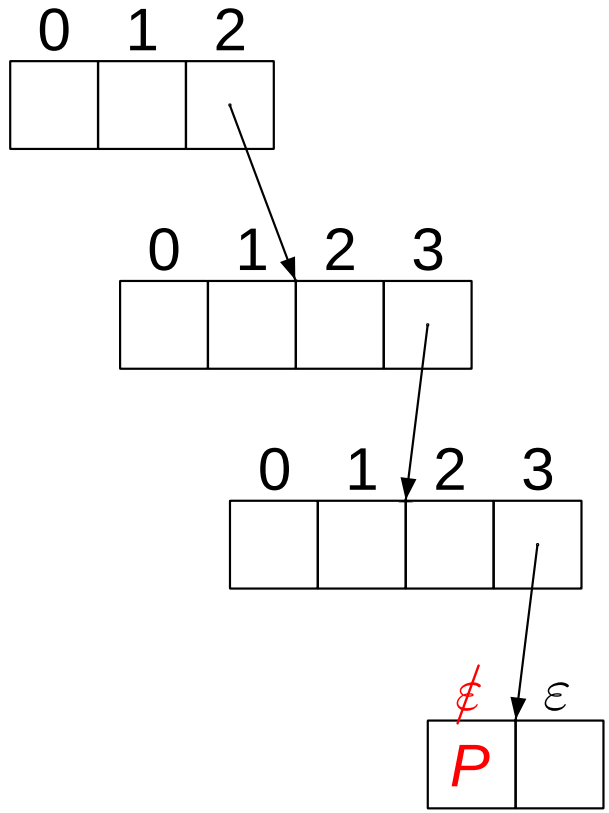
$X = \{aaa, abaa, abab\}$.

$P = (aa, baa, bab, \notin)$.



$X = \{aaa, abaa, abab\}$.

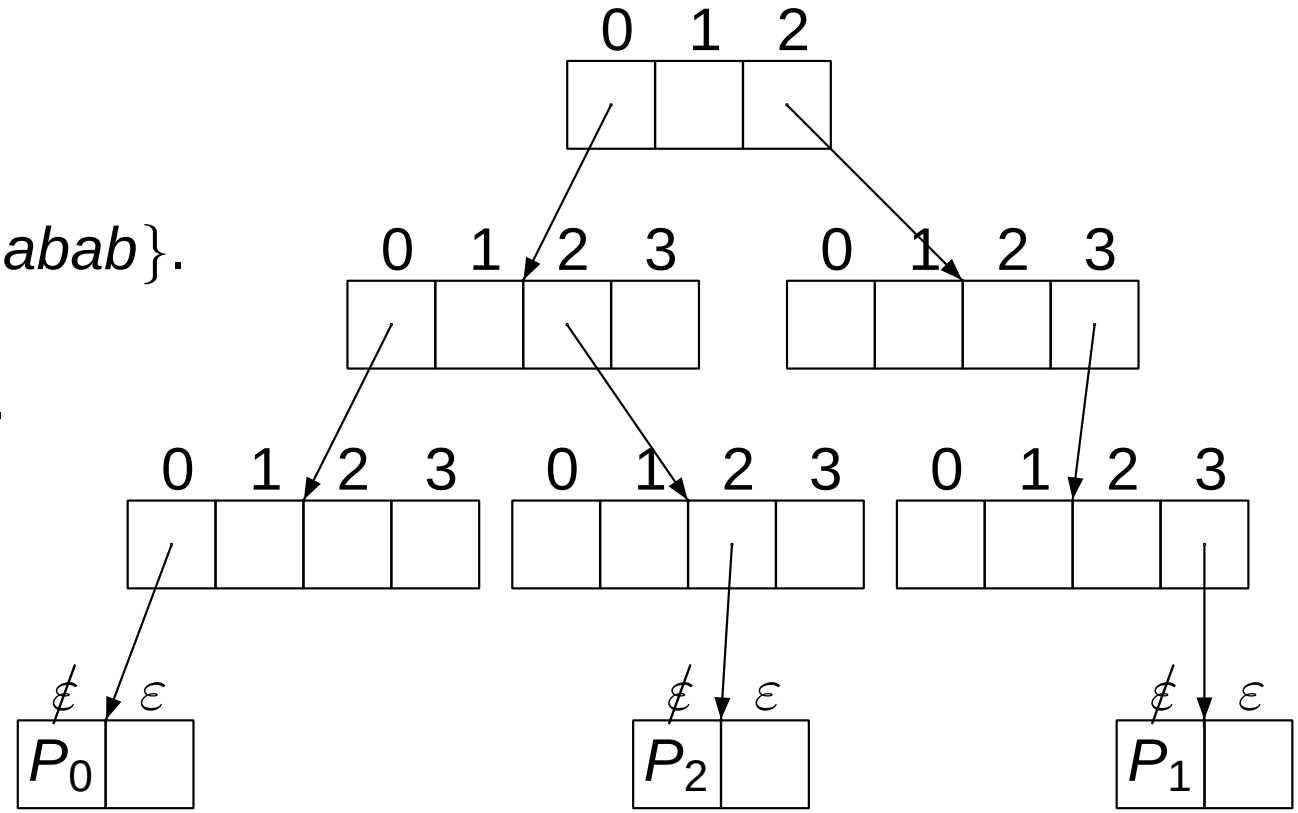
$P = (aa, baa, bab, \notin)$.



Existence of a state

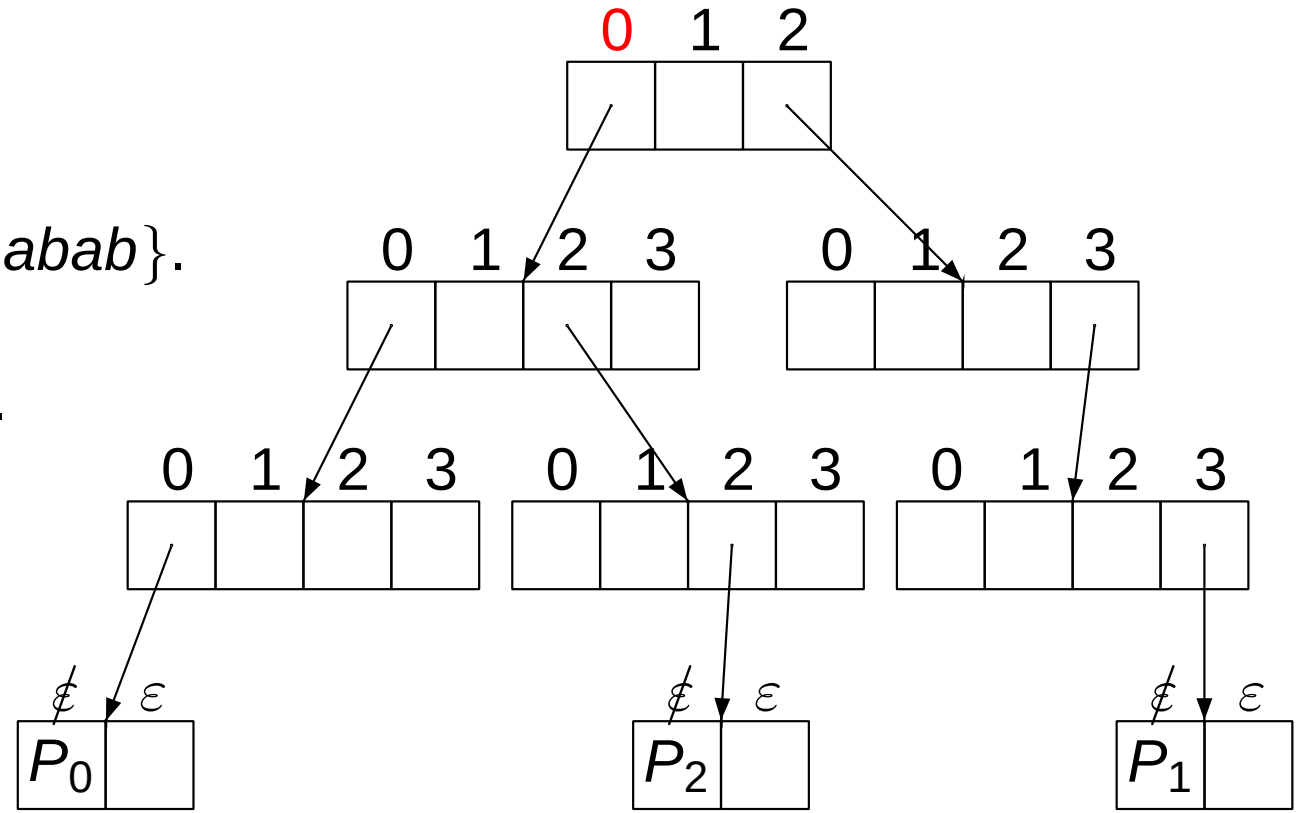
$X = \{aaa, abaa, abab\}$.

$P = (i, aa, ab, \notin)$.



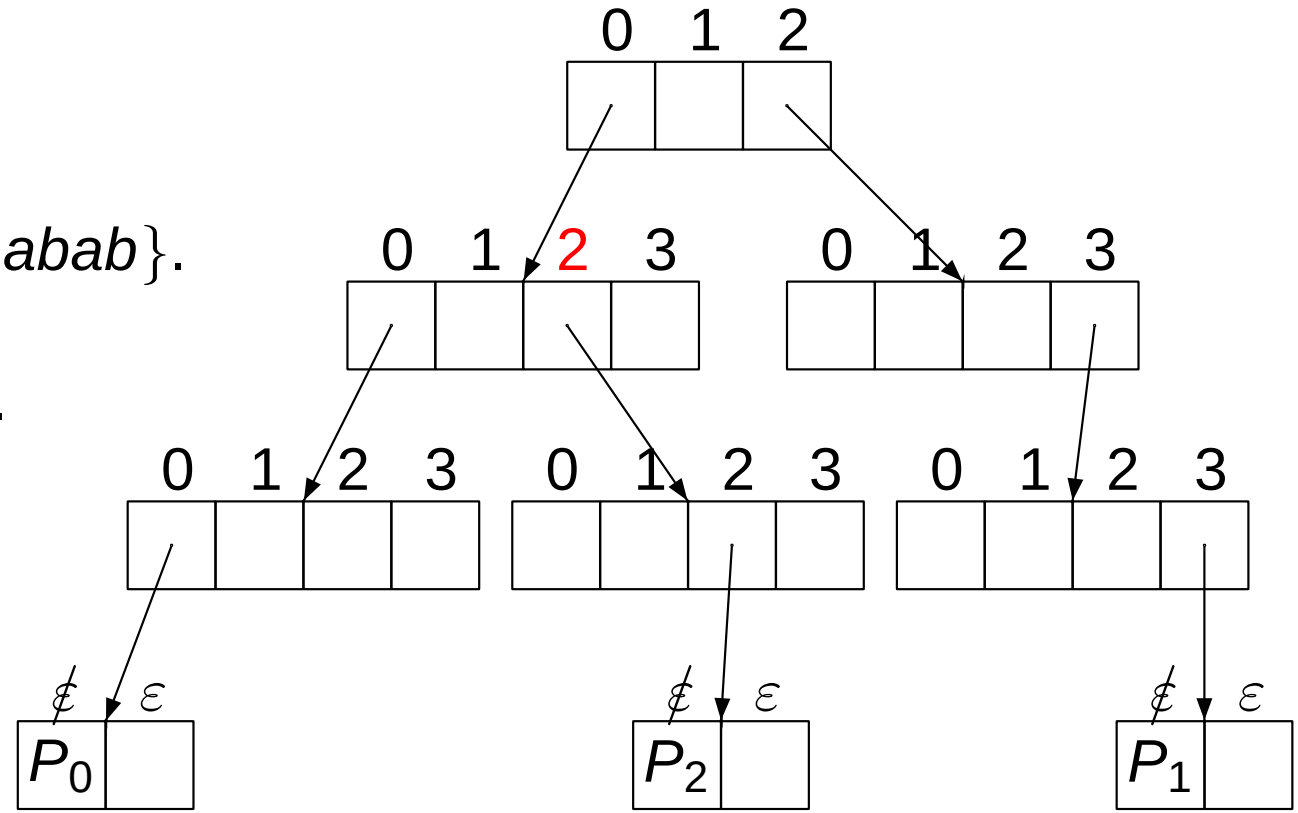
$X = \{aaa, abaa, abab\}$.

$P = (i, aa, ab, \notin)$.



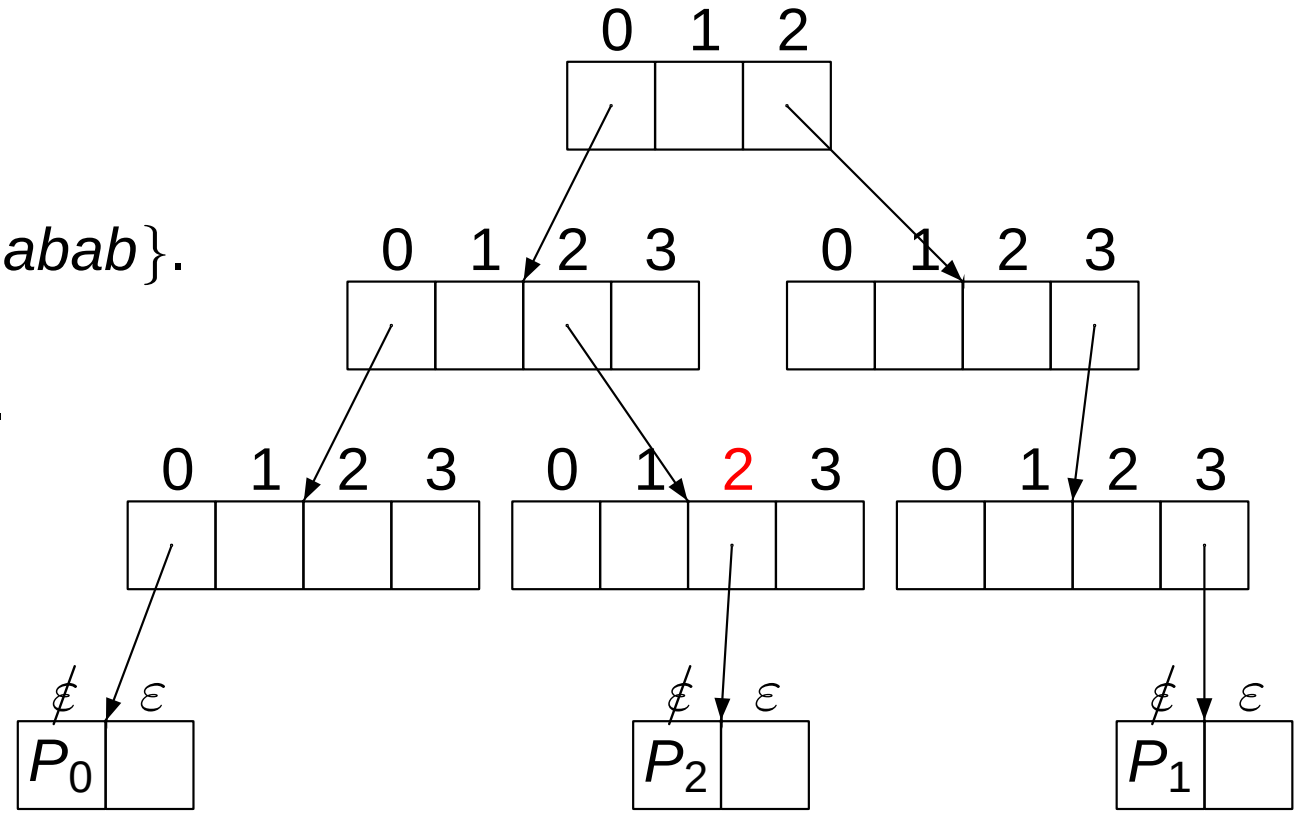
$X = \{aaa, abaa, abab\}$.

$P = (i, aa, ab, \notin)$.



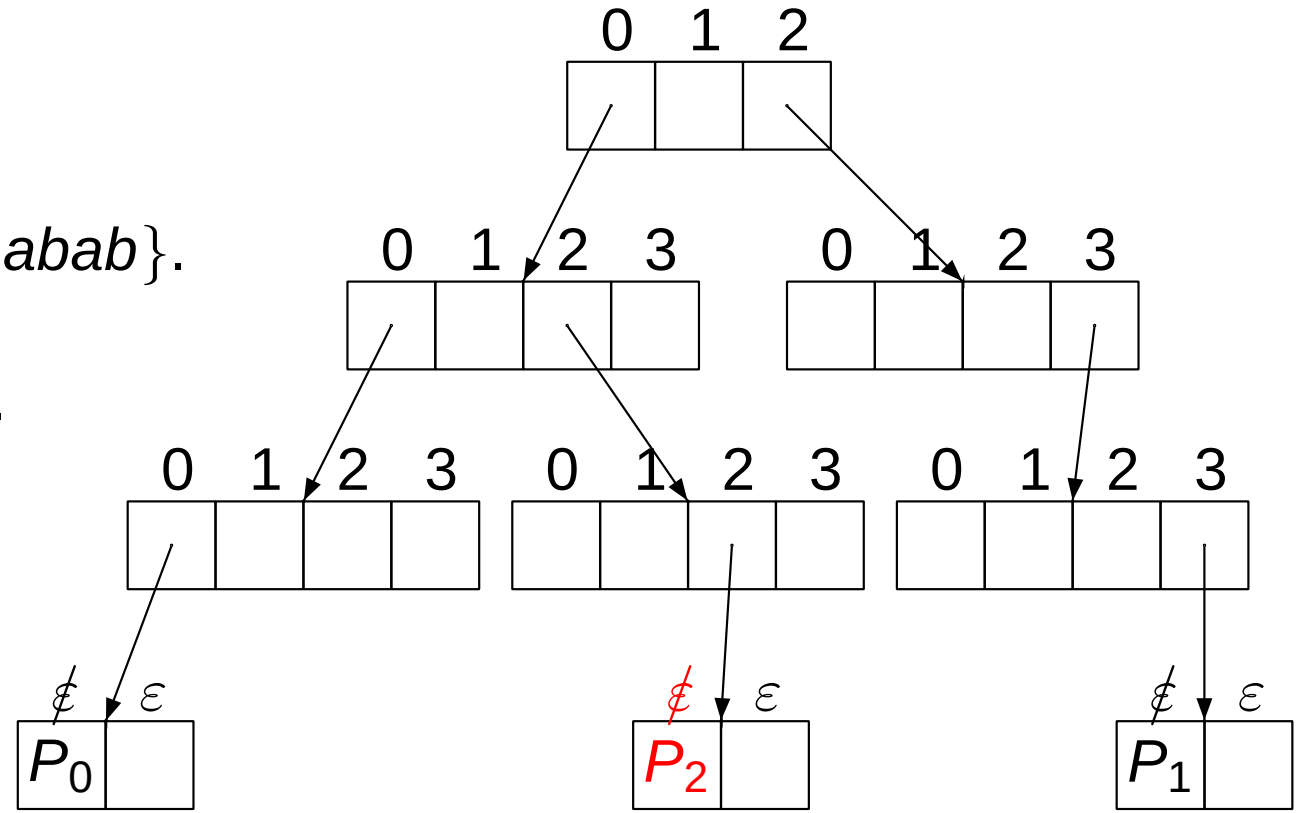
$X = \{aaa, abaa, abab\}$.

$P = (i, aa, ab, \notin)$.



$X = \{aaa, abaa, abab\}$.

$P = (i, aa, ab, \notin)$.



Complexity

Theorem

Using sparse lists, the construction of the minimal automaton recognizing A^*X runs in time $\mathcal{O}(n)$ and requires $\mathcal{O}(n^2)$ space, where n is the length of the pattern X.

Proof.

The determinization produces the minimal automaton, of size at most $n + 1$ (from Aho-Corasick's result). Each state requires $m + 1$ sparse lists of size $|u_1|, |u_2|, \ldots, |u_m|, 2$.

$$\Longrightarrow \text{ the total space complexity is quadratic in } n.$$

Searching and inserting a state in the tree of sparse lists takes a constant time. □

Remark: Hashing

In practice a hash table can be used to store these states.

Under the hypothesis of a simple uniform hashing,

$\Longrightarrow$ the average time and space complexities of the determinization are linear

Conclusion

$X = \{u_1, \ldots, u_m\}$ a pattern of length n, where m is fixed.

- direct construction : co-deterministic automaton A^*X.
- ad-hoc determinization algorithm (using sparse lists).

$$\Longrightarrow \mathcal{O}(n) \text{ time complexity, } \mathcal{O}(n^2) \text{ space complexity.}$$

Future work

- natural continuation: m not fixed anymore.
- regular language.

Thank you for your attention.