(k, l)-Unambiguity and Quasi-Deterministic Structures

P. Caron, M. Flouret, L. Mignot

Université de Rouen, LITIS, Équipe Combinatoire et Algorithmes

March 12th 2014

LATA 2014

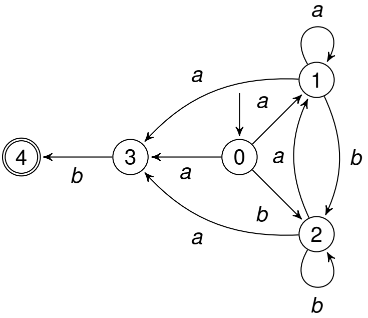
Summary

Theoretical Background

Automaton

An automaton is a 5-tuple $A = (\Sigma, Q, I, F, \delta)$ where:

- Σ is a finite alphabet,
- Q is a finite set of states,
- $I \subset Q$ is the set of initial states,
- $F \subset Q$ is the set of final states,
- $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition function.

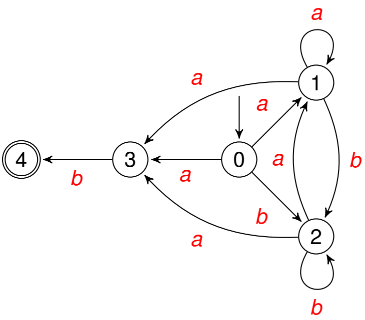


Theoretical Background

Automaton

An automaton is a 5-tuple $A = (\Sigma, Q, I, F, \delta)$ where:

- Σ is a finite alphabet,
- Q is a finite set of states,
- $I \subset Q$ is the set of initial states,
- $F \subset Q$ is the set of final states,
- $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition function.

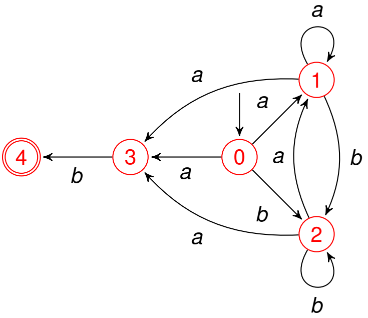


Theoretical Background

Automaton

An automaton is a 5-tuple $A = (\Sigma, Q, I, F, \delta)$ where:

- Σ is a finite alphabet,
- Q is a finite set of states,
- $I \subset Q$ is the set of initial states,
- $F \subset Q$ is the set of final states,
- $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition function.

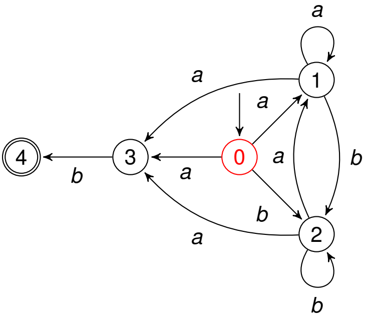


Theoretical Background

Automaton

An automaton is a 5-tuple $A = (\Sigma, Q, I, F, \delta)$ where:

- Σ is a finite alphabet,
- Q is a finite set of states,
- $I \subset Q$ is the set of initial states,
- $F \subset Q$ is the set of final states,
- $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition function.

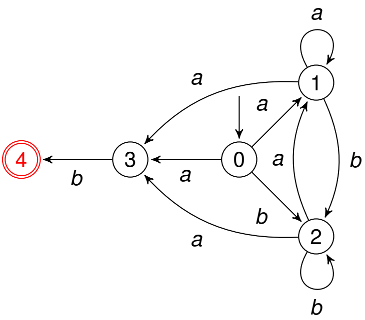


Theoretical Background

Automaton

An automaton is a 5-tuple $A = (\Sigma, Q, I, F, \delta)$ where:

- Σ is a finite alphabet,
- Q is a finite set of states,
- $I \subset Q$ is the set of initial states,
- $F \subset Q$ is the set of final states,
- $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition function.

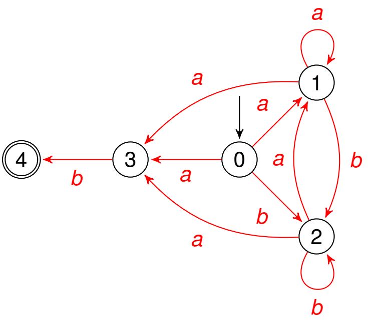


Theoretical Background

Automaton

An automaton is a 5-tuple $A = (\Sigma, Q, I, F, \delta)$ where:

- Σ is a finite alphabet,
- Q is a finite set of states,
- $I \subset Q$ is the set of initial states,
- $F \subset Q$ is the set of final states,
- $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition function.



Deterministic Automaton

Definition

An automaton $M = (\Sigma, Q, I, F, \delta)$ is deterministic if and only if

- the automaton has a unique initial state : $|I| = 1$,
- Each state is the origin of at most one transition labeled by each letter of the alphabet : $\forall q \in Q, a \in \Sigma, |\delta(q, a)| \leq 1$.

Theoretical background

Regular Expression

A regular expression E over an alphabet Σ is defined by :

$$\begin{array}{lll} E = \emptyset, & E = \varepsilon, & E = a, \\ E = F + G, & E = F \cdot G, & E = F^*, \end{array}$$

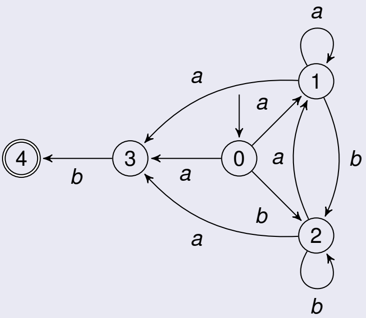
with a a symbol of Σ and F and G two regular expressions.

$$E = (a + b)^* \cdot a \cdot b$$

Theoretical background

From an Expression to its position Automaton

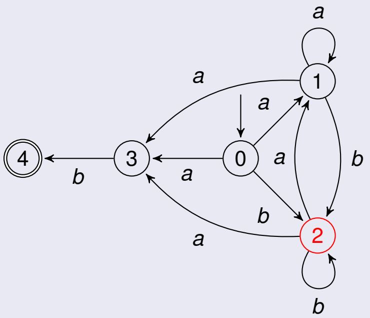
$$E = (a+b)^* \cdot a \cdot b$$
$$E^{\#} = (a_1 + b_2)^* \cdot a_3 \cdot b_4$$



Theoretical background

From an Expression to its position Automaton

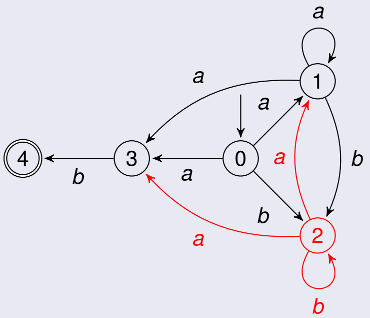
$$E = (a+b)^* \cdot a \cdot b$$
$$E^\# = (a_1 + b_2)^* \cdot a_3 \cdot b_4$$



Theoretical background

From an Expression to its position Automaton

$$E = (a+b)^* \cdot a \cdot b$$
$$E^{\#} = (a_1 + b_2)^* \cdot a_3 \cdot b_4$$

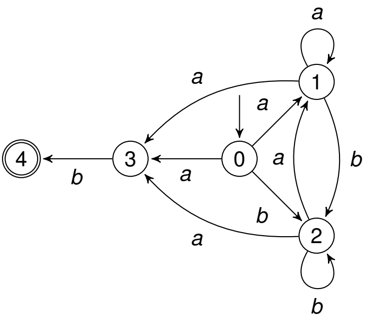


Theoretical Background

Automaton Recognition

A word w is recognized by an automaton if it is the label of a path going from an initial to a final state.

A language L is recognized by an automaton if it is equal to the set of all words recognized by this automaton.



This automaton recognizes the words ending with *ab*.

Theoretical Background

Membership test in an NFA A

Time: $O(|w| \times n^2)$
Space: $O(n)$
(n: number of states of A)

Membership test in an DFA D

Time: $O(|w|)$
Space: $O(1)$

Theoretical Background

Membership test in an NFA A

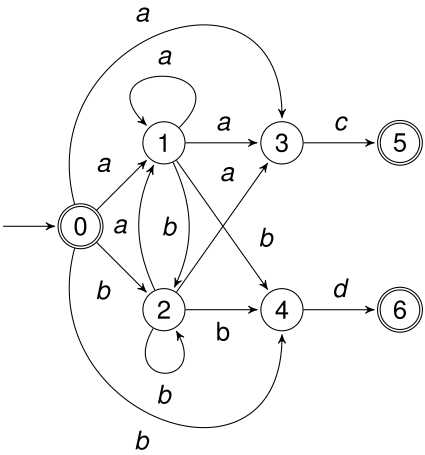
Time: $O(|w| \times n^2)$
Space: $O(n)$
(n: number of states of A)

Membership test in an DFA D

Time: $O(|w|)$
Space: $O(1)$

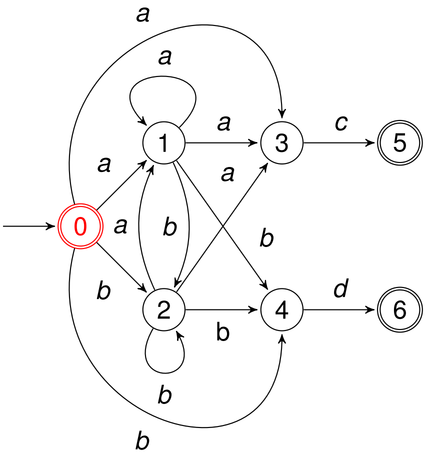
number of states:	n	$\longrightarrow$	$O(2^n)$
number of transitions:	$O(n^2 \times \lvert\Sigma\rvert)$	$\longrightarrow$	$O(2^n \times \lvert\Sigma\rvert)$

Buffer and k-Lookahead Determinism



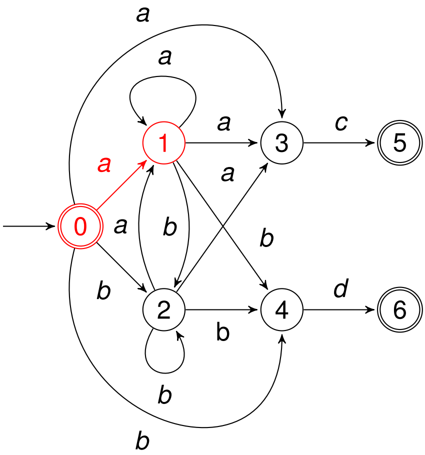
	a	b
		4
	1	
0		2
	3	$\emptyset$

Buffer and *k*-Lookahead Determinism



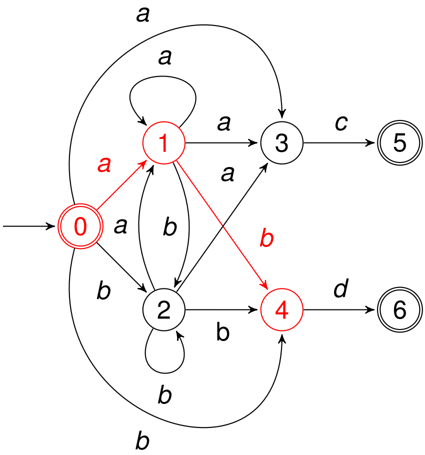
	a	*b*
0	1 3	4 2 ∅

Buffer and k-Lookahead Determinism



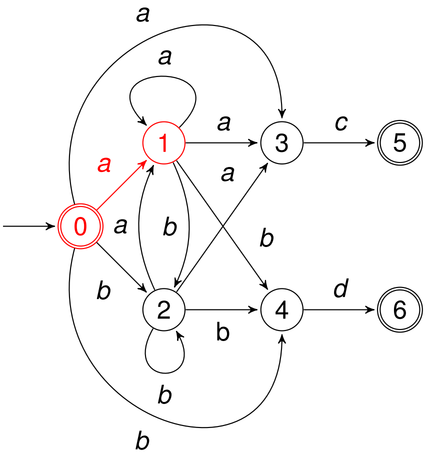
	a	b
0	1	4
		2
	3	∅

Buffer and k-Lookahead Determinism



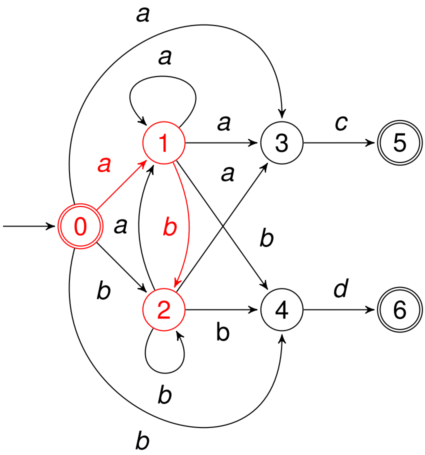
	a	b
		4
	1	
0		2
	3	∅

Buffer and *k*-Lookahead Determinism



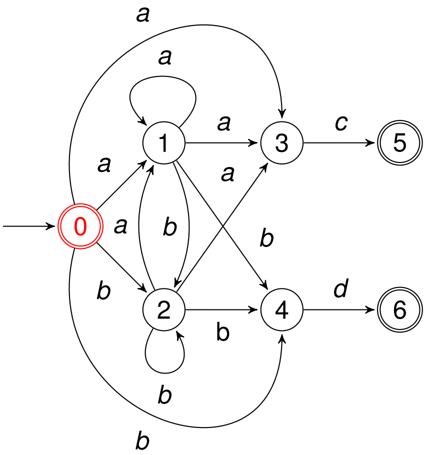
	a	b
		4
	1	
0		2
	3	∅

Buffer and k-Lookahead Determinism



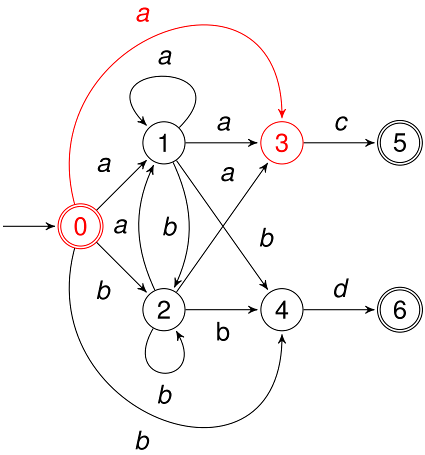
	a	b
		4
0	1	2
	3	∅

Buffer and k-Lookahead Determinism



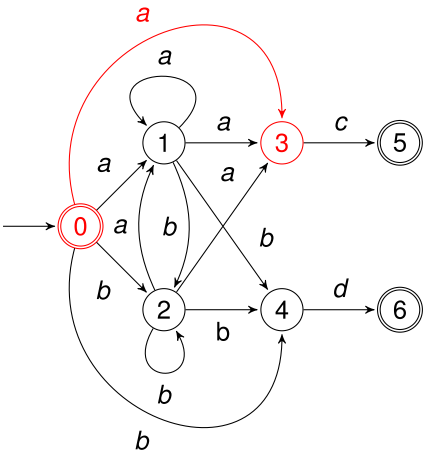
	a	b
0	1	4
		2
	3	$\emptyset$

Buffer and k-Lookahead Determinism



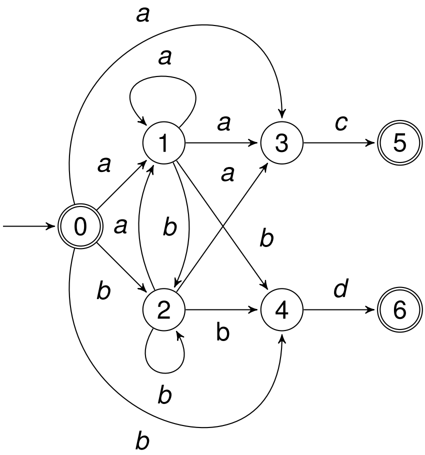
	a	b
0	1	4, 2
	3	∅

Buffer and *k*-Lookahead Determinism



	a	b
0	1	4
		2
	3	$\emptyset$

Buffer and k-Lookahead Determinism



	a	b
0	1, 3	4, 2, $\emptyset$

Buffer and k-Lookahead Determinism

Basic Idea

A sliding window of length k on the input word is sufficient to determine the next state in the automaton.

Brzozowski and Santean [2008]

k-predictable automata
$\Rightarrow$ Characterizing these automata and finding the bound.

Han and Wood [2008]

Deterministic k-lookahead position automata
$\Rightarrow$ Defining the notion of deterministic k-lookahead expressions and languages.

Theoretical Background

Membership test in an NFA A

Time: $O(|w| \times n^2)$
Space: $O(n)$
(n: number of states of A)

Membership test in a buffered NFA

Time: $O(k \times |w|)$
Space: $O(1)$

Theoretical Background

Membership test in an NFA A

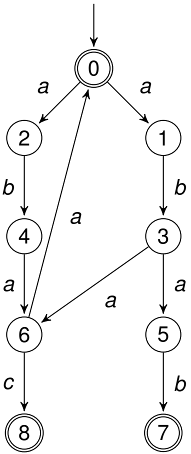
Time: $O(|w| \times n^2)$
Space: $O(n)$
(n: number of states of A)

Membership test in a buffered NFA

Time: $O(k \times |w|)$
Space: $O(1)$

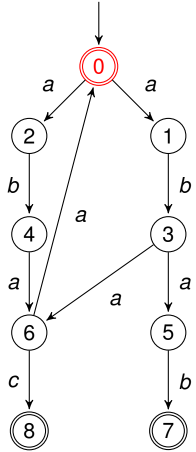
$$\begin{array}{rccc} \text{number of states:} & n & \longrightarrow & n \\ \text{number of transitions:} & O(n^2 \times |\Sigma|) & \longrightarrow & O(n \times |\Sigma|^k) \end{array}$$

(k, l)-Unambiguous Automaton



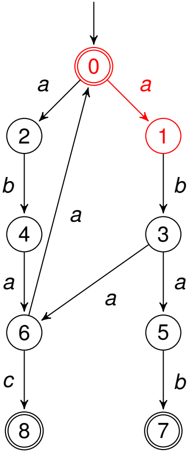
	a	*b*	*a*	*a*
0	1	3	5	∅
	2	4	6	0

(k, l)-Unambiguous Automaton



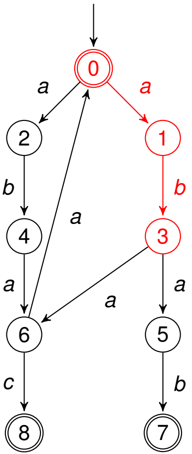
	a	b	a	a
0	1	3	5	∅
	2	4	6	0

(*k*, *l*)-Unambiguous Automaton



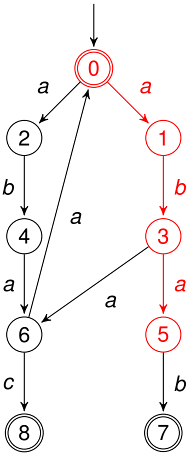
	a	*b*	*a*	*a*
0	1	3	5	∅
	2	4	6	0

(k, l)-Unambiguous Automaton



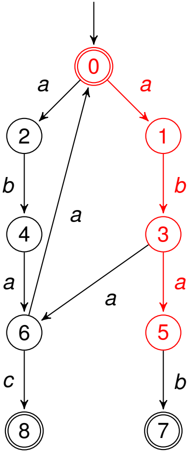
	a	b	a	a
0	1	3	5	$\emptyset$
	2	4	6	0

(k, l)-Unambiguous Automaton



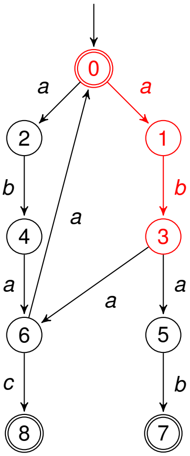
	a	b	a	a
0	1	3	5	∅
	2	4	6	0

(*k*, *l*)-Unambiguous Automaton



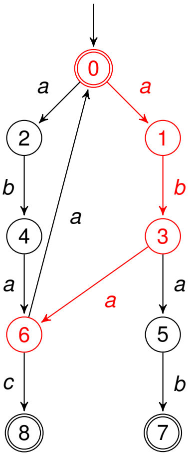
	a	*b*	*a*	*a*
0	1	3	5	∅
	2	4	6	0

(k, l)-Unambiguous Automaton



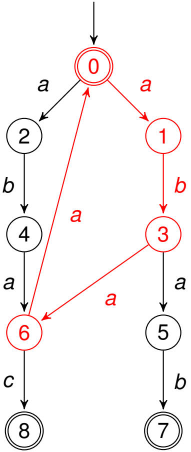
	a	b	a	a
0	1	3	5	$\emptyset$
			6	0
	2	4		

(k, l)-Unambiguous Automaton



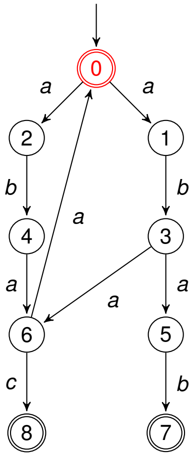
	a	b	a	a
			5	$\emptyset$
0	1	3		
			6	0
	2	4		

(*k*, *l*)-Unambiguous Automaton



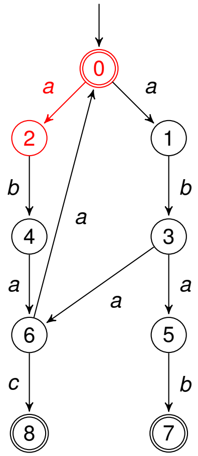
	a	*b*	*a*	*a*
			5	∅
0	1	3		
			6	0
	2	4		

(k, l)-Unambiguous Automaton



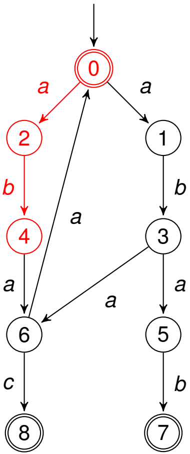
	a	b	a	a
0	1	3	5	∅
	2	4	6	0

(k, l)-Unambiguous Automaton



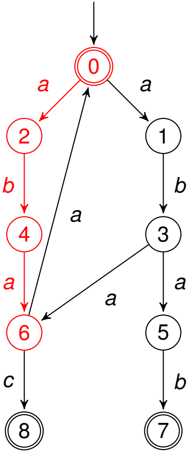
	a	b	a	a
0	1	3	5	$\emptyset$
	2	4	6	0

(*k*, *l*)-Unambiguous Automaton



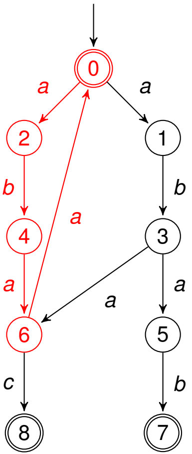
	a	*b*	*a*	*a*
0	1	3	5	∅
			6	0
	2	4		

(k, l)-Unambiguous Automaton



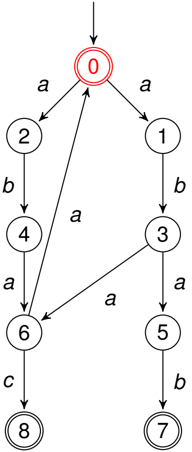
	a	b	a	a
0	1	3	5	$\emptyset$
	2	4	6	0

(k, l)-Unambiguous Automaton



	a	b	a	a
0	1	3	5	$\emptyset$
	2	4	6	0

(k, l)-Unambiguous Automaton



	a	b	a	a
0	1	3	5	∅
	2	4	6	0

(k, l)-Unambiguous Automaton

Basic Idea

A sliding window of length k on the input word is sufficient to determine a unique state within the next l input symbols.

Theorem [Caron, Flouret and Mignot]

One can decide if there exists a couple (k, l) such that a given automaton is (k, l)-unambiguous. Furthermore such a couple is computable.

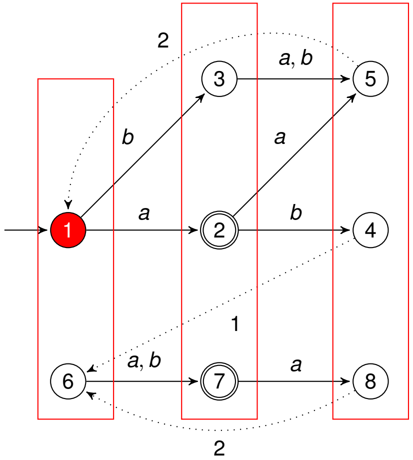
A Convenient Deterministic Structure for (k, l)-Unambiguous Automaton

Quasi-deterministic Structure

In a (k, l)-unambiguous automaton, given a state q and a word w of length k, a unique state is reached in $i < l$ steps.
A particular structure called Quasi-Deterministic Structure can be defined from triples (q, w, i).

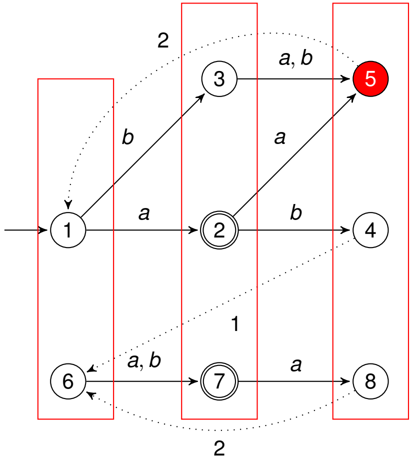
Quasi-Deterministic Structure: An Example

b	*b*	*b*	*a*	*a*	*b*	*a*	*b*

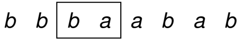


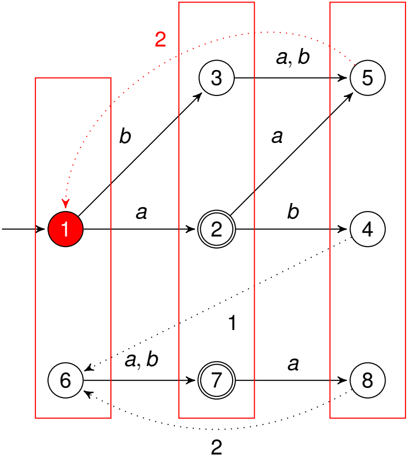
Quasi-Deterministic Structure: An Example

$\boxed{b\ \ b}\ b\ \ a\ \ a\ \ b\ \ a\ \ b$



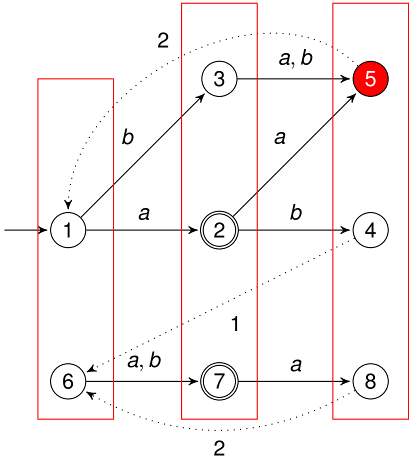
Quasi-Deterministic Structure: An Example





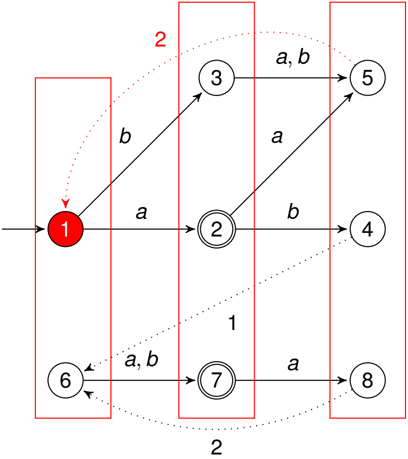
Quasi-Deterministic Structure: An Example

$b\quad b\quad \boxed{b\quad a}\quad a\quad b\quad a\quad b$



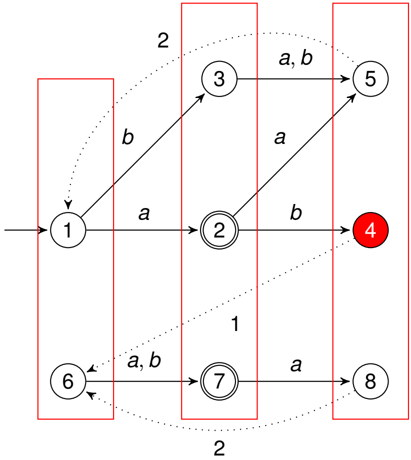
Quasi-Deterministic Structure: An Example

b b b a | a b | a b



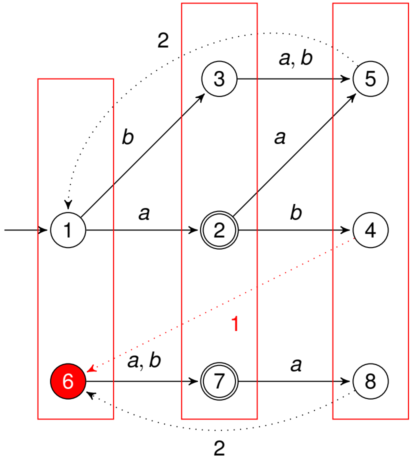
Quasi-Deterministic Structure: An Example

$b \quad b \quad b \quad a \quad \boxed{a \quad b} \quad a \quad b$



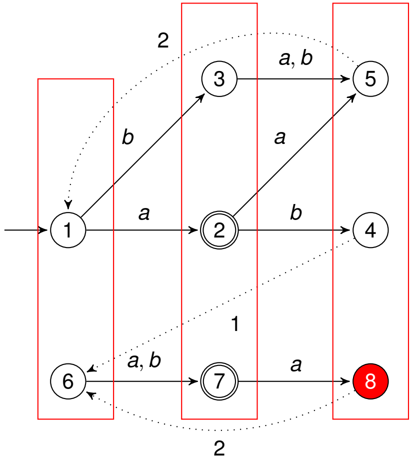
Quasi-Deterministic Structure: An Example

b b b a a | *b a* | *b*



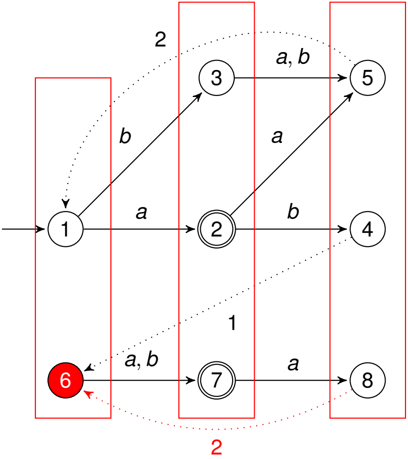
Quasi-Deterministic Structure: An Example

b b b a a [*b a*] *b*



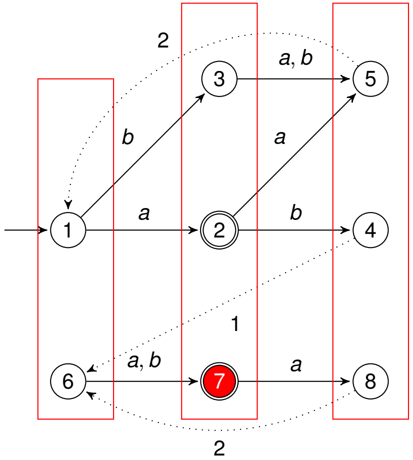
Quasi-Deterministic Structure: An Example

b b b a a b a | *b* |



Quasi-Deterministic Structure: An Example

b b b a a b a | *b* |



Quasi-Deterministic Structure

Definition

A Quasi-Deterministic Structure is a 6-tuple $(\Sigma, \mathcal{Q}, i, F, \delta, \gamma)$ with

- Σ a finite alphabet,
- $\mathcal{Q} = (Q_1, Q_2, \ldots Q_k)$ a list of disjoint finite sets of states,
- $i \in Q_1$ the initial state,
- $F \subset \bigcup_{j=1}^{k} Q_j$ the final states,
- $\delta : Q_j \times \Sigma \longrightarrow Q_{j+1}$ with $1 \leq j < k$ the transition function.
- $\gamma : Q_k \times [\![1, k]\!] \longrightarrow Q_1$ the shift function.

Quasi-Deterministic Structure

Membership test in an NFA A

Time: $O(|w| \times n^2)$
Space: $O(n)$
(n: number of states of A)

Membership test in a QDS

Time: $O(\frac{k}{s} \times |w|)$
Space: $O(1)$
(s: minimal shift of γ)

Quasi-Deterministic Structure

Membership test in an NFA A

Time: $O(|w| \times n^2)$
Space: $O(n)$
(n: number of states of A)

Membership test in a QDS

Time: $O(\frac{k}{s} \times |w|)$
Space: $O(1)$
(s: minimal shift of γ)

number of states:	n	$\longrightarrow$	$O(n \times \|\Sigma\|^k)$
number of transitions:	$O(n^2 \times \|\Sigma\|)$	$\longrightarrow$	$O(n \times \|\Sigma\|^{k+1})$

Conclusion

- (k, l)-unambiguity leads to Quasi-deterministic structure
- QDS are smaller than DFAs and faster than NFAs
- (k, l)-unambiguity is a generalization of k-lookahead determinism:
  $(k, 1)$-unambiguity $=$ k-lookahead determinism

Perspectives

- Algorithm over quasi-deterministic structure.
  - Minimization (using a Myhill/Nerode-like equivalence).
  - Computation of a QDS from a regular expression.
- Open problems over (k, l)-unambiguous languages.
  - are they strictly included in regular languages ?
  - is there a hierarchy ?