# WORDS, TREES AND AUTOMATA MINIMIZAZION


Marinella Sciortino

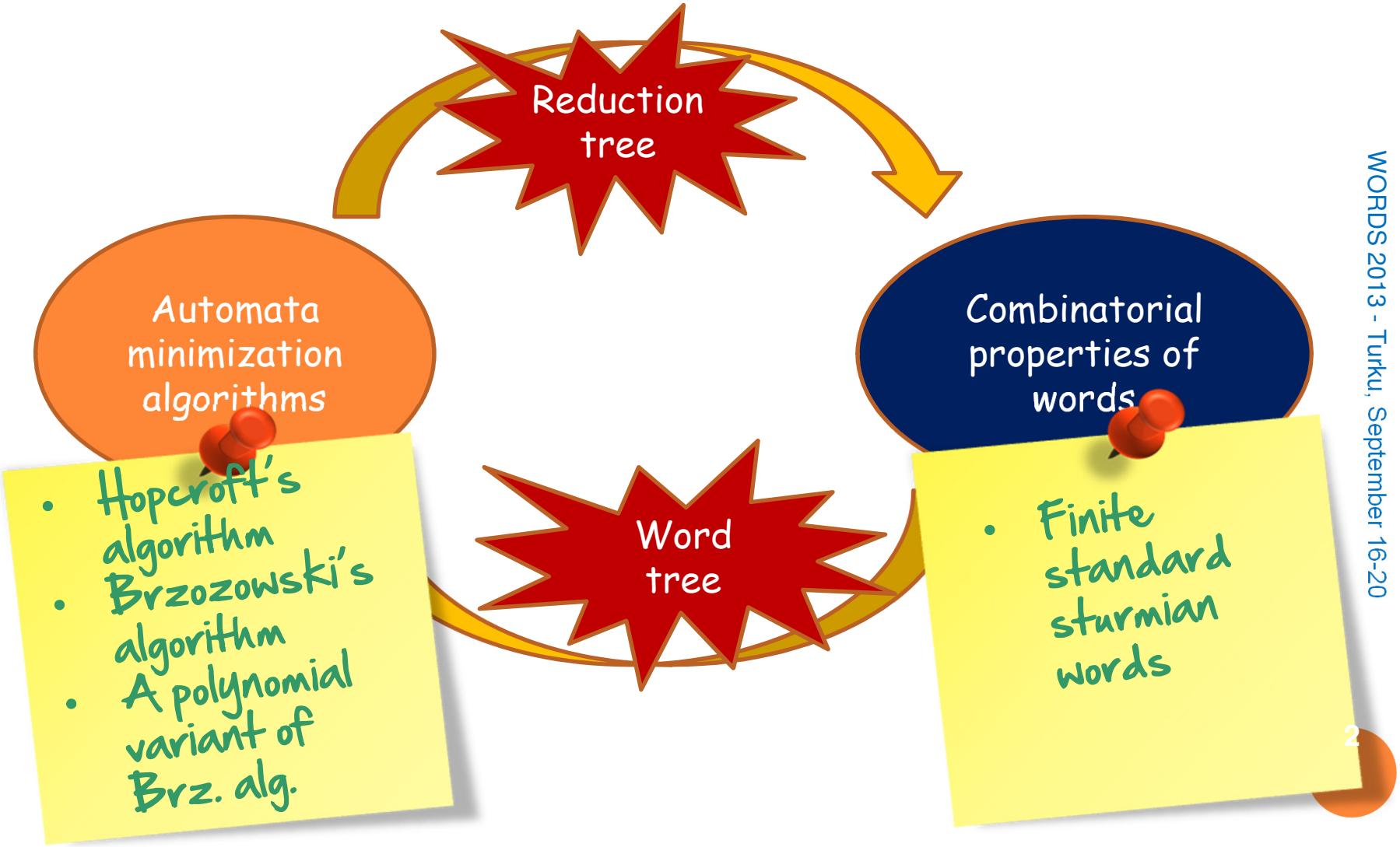
Joint work with Giusi Castiglione

University of Palermo



WORDS 2013 - Turku, September 16-20

# Graphical summary



# Regular languages and FSA

- Regular languages can be represented equivalently as rational expressions or as deterministic finite automata.
- A finite state automaton is denoted by

$$A=(Q,\Sigma,I,\delta,F),$$

where Q is the set of states, Σ is the finite alphabet, I the set of initial states, F the set of final states, δ the transition function from $Q\times\Sigma$ to $2^Q$.

- The automaton is deterministic (DFA), denoted by $A=(Q,\Sigma,i,\delta,F)$, if it has a unique initial state and if δ is a function from $Q\times\Sigma$ to Q.
- There is a unique DFA (up to a renaming of the states) with a minimal number of states, called minimal automaton, recognizing a regular language L.
- The description of a language with its minimal automaton is important when space considerations matter in implementations of applications, such as in Pattern Matching, Lexical Analysis, Coding Systems, and so on.

# Nerode equivalence

- The minimal automaton equivalent to a given automaton A (i.e. recognizing the same language) is defined by the right-invariant Nerode equivalence.
- Given a state $p \in Q$, we consider the language

$$L_p(A)=\{v \in \Sigma^* \mid \delta(p,v) \in F\}$$

- The Nerode equivalence is defined as follows:

$$p \sim q \text{ if } L_p(A)=L_q(A)$$

- It is a congruence, i.e. for any $a \in \Sigma$,

$$p \sim q \text{ implies } \delta(p,a) \sim \delta(q,a),$$

and it is the coarsest congruence such that F is union of classes of the congruence.

- The equivalence classes are the states of the minimal automaton.

# How to compute it?

# Minimization algorithms

- There are lots of methods which can be used to minimize a finite automaton, i.e. to find the minimal automaton equivalent to a given DFA.
- Some of them operate by successive refinements of a partition of the states:
  - Moore, 1956. Time complexity $O(kn^2)$, $k=|\Sigma|$
  - Hopcroft, 1971. Time complexity $O(knlogn)$, $k=|\Sigma|$
- The Bzrozowski's method (1962) operates by reversal and determinization repeated twice.

  Time complexity: exponential worst case, but good performance in practice.
- A taxonomy of finite automata minimization algorithms is given by B. Watson, 1994.
- "Minimization of automata" by Berstel, Boasson, Carton, Fagnot, 2010.
- Polynomial variants of Brzozowski's method have been recently introduced.
- Many papers on experimental comparison between minimization algorithms.

# Hopcroft's algorithm: the fastest

- No faster algorithm is known for general deterministic automata
- In order to obtain the Nerode equivalence, a sequence of refinements of equivalence relations is realized.
- The sequence starts with the partition in two classes separating final and not final states.
- It is based on the so-called "smaller half" strategy

# EXECUTION OF HOPCROFT'S ALGORITHM

Let $\Pi$ a partition of Q:

- A pair $(C,a)$ where $C \in \Pi$ and $a \in \Sigma$ is a **splitter** if there exists a class P in $\Pi$ such that

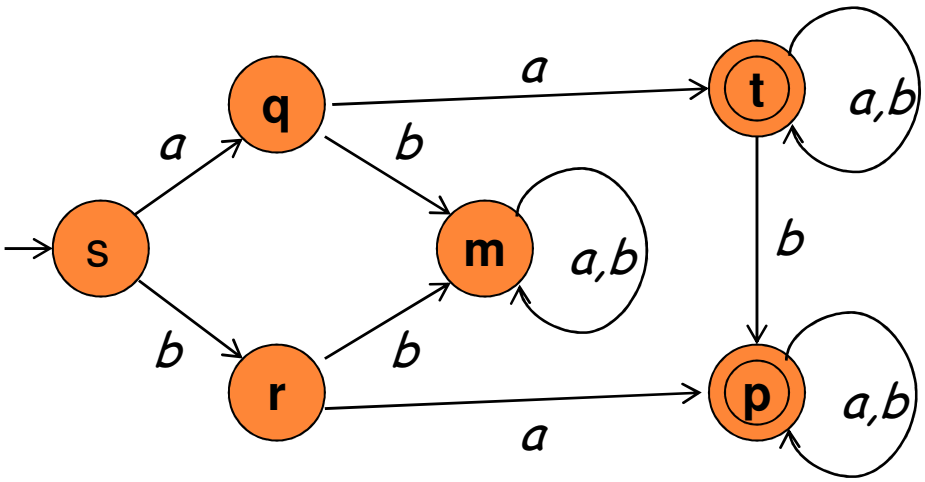
$$\emptyset \not\subset \delta^{-1}{}_a(C) \cap P \not\subset P$$

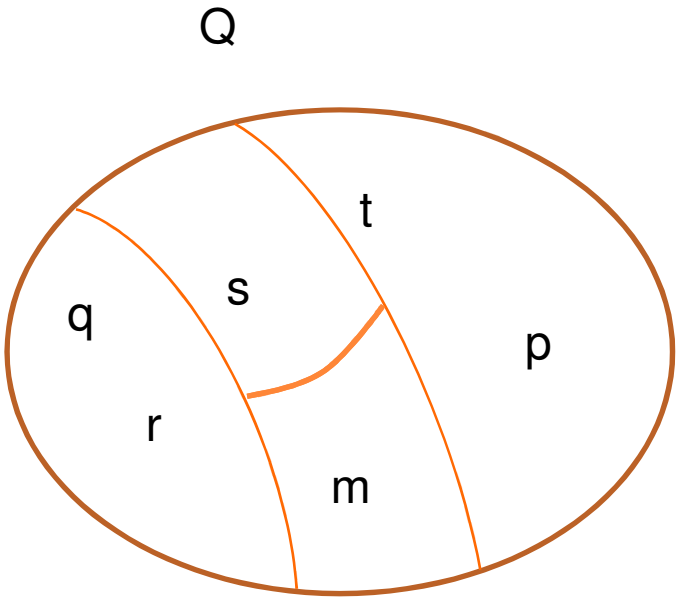
- The class P is split in $\delta^{-1}{}_a(C) \cap P$ and $P / \delta^{-1}{}_a(C)$
- The smallest set coupled with some symbol of the alphabet goes into the waiting set.
- At each step a pair from the waiting set is extracted and the algorithm stops when the waiting set is empty.

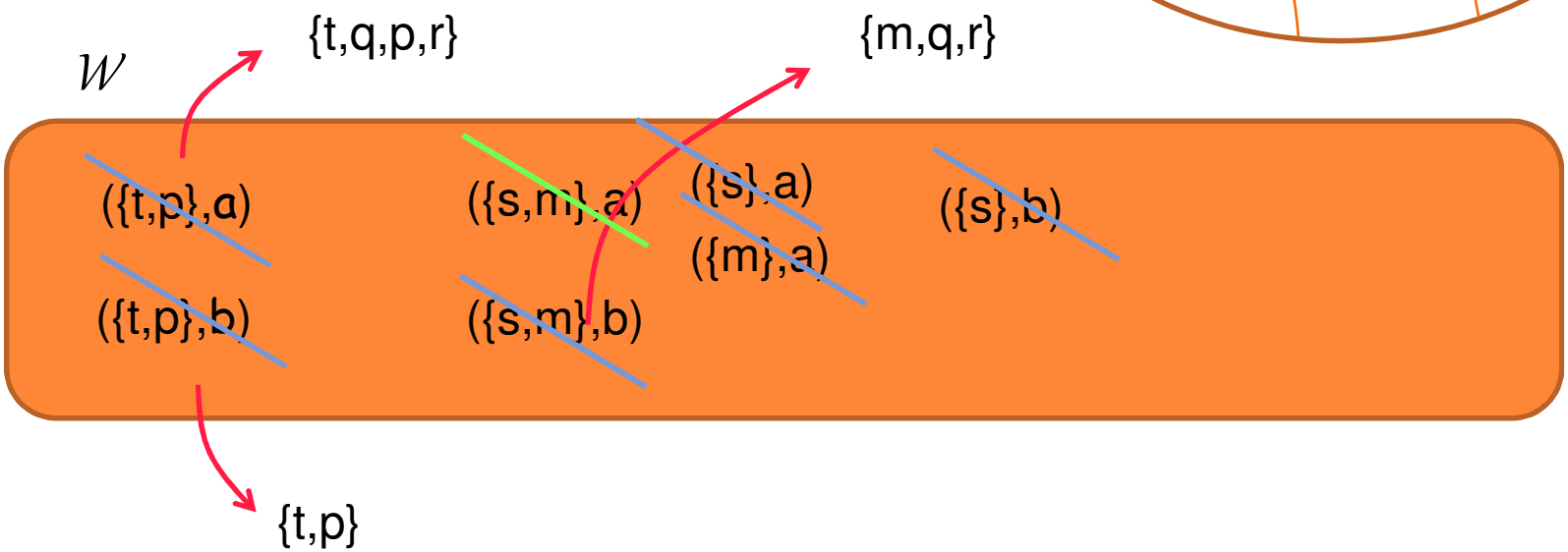
**Let A =(Q,Σ,i,δ,F) be a DFA.**

```
1: Π ← {F, F^c}
2: for all a ∈ Σ do
3:         Add((min(F, F^c ), a),W)
4: while W ≠ ∅ do
5:         (C, a) ← TakeSome(W)
6:         for each P ∈ Π which is
   split   by (C, a) do
7:         P', P'' ← (C, a)|P
8:         Replace P by P' and P'' in Π
9:         for all b ∈ Σ do
10:            if (P, b) ∈ W then
11:            Replace (P, b) by (P', b)
           and (P'' , b) in W
12:            else
13:            Add((min(P', P''), b),W)
```

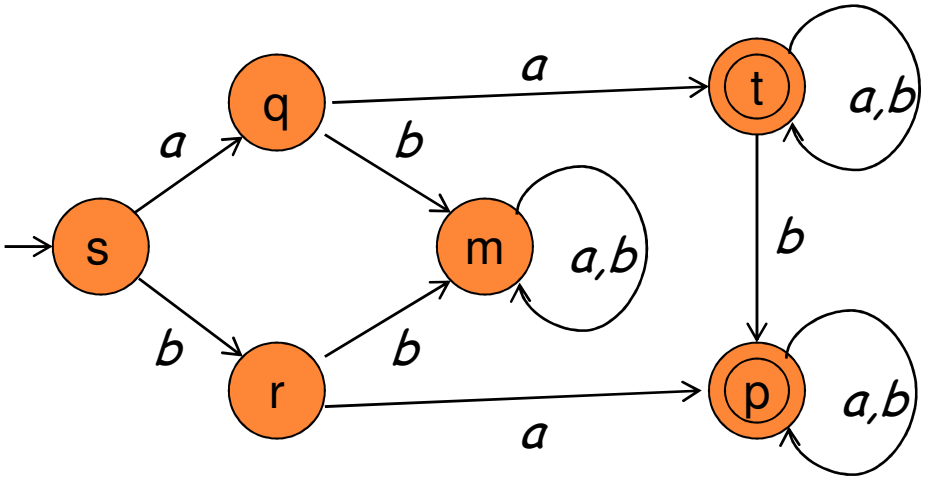
# Hopcroft's algorithm: an example

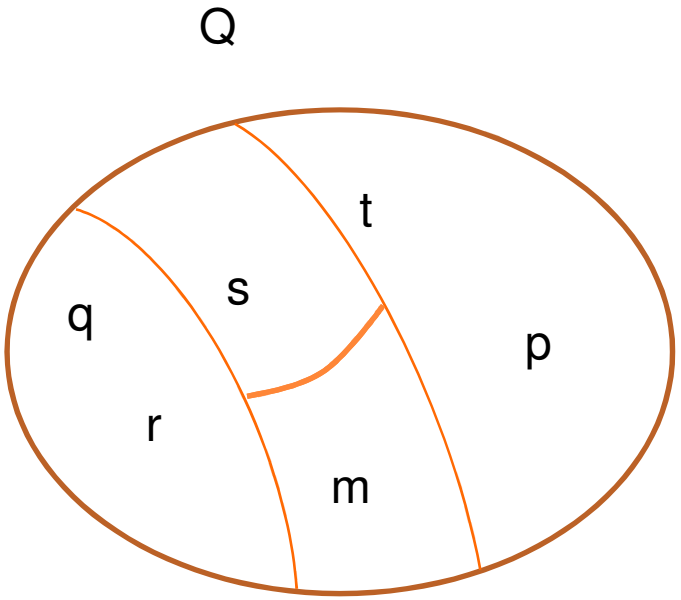


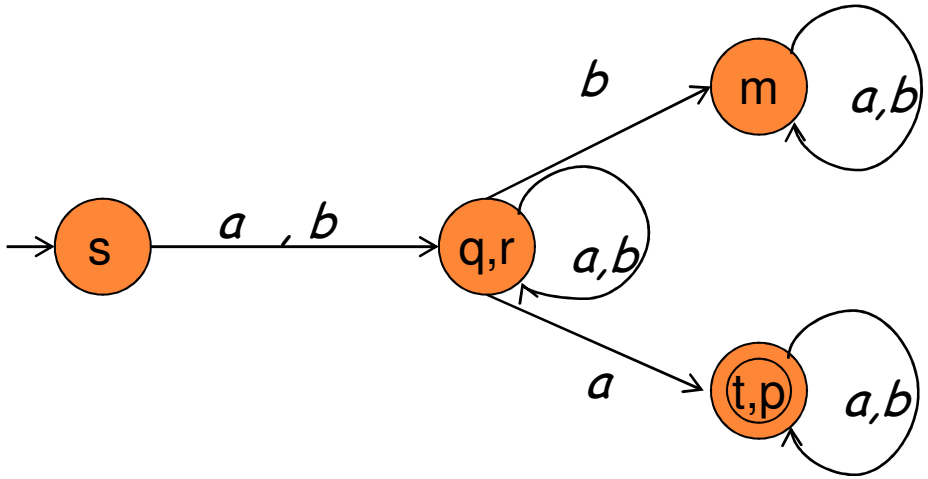



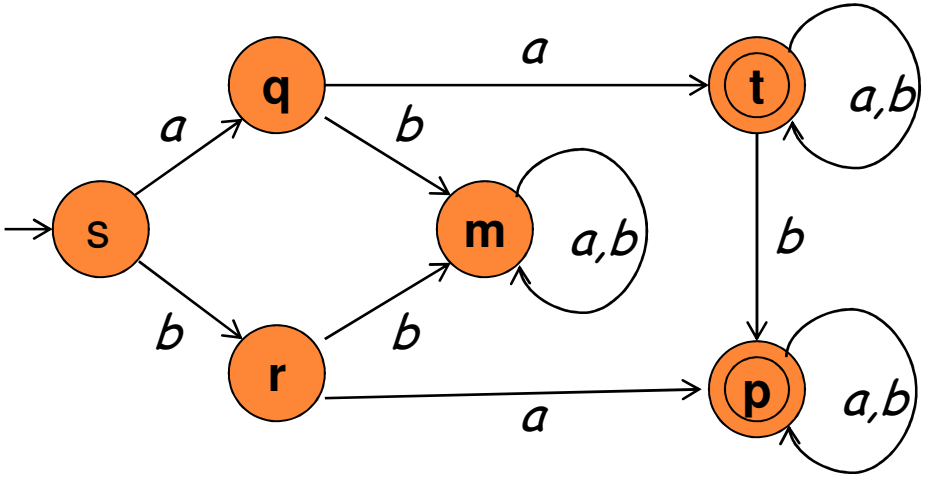
# Hopcroft's algorithm: an example

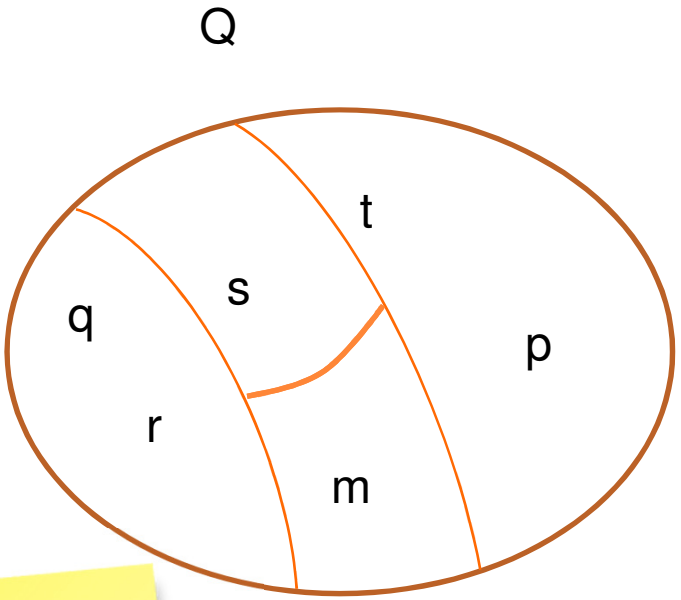


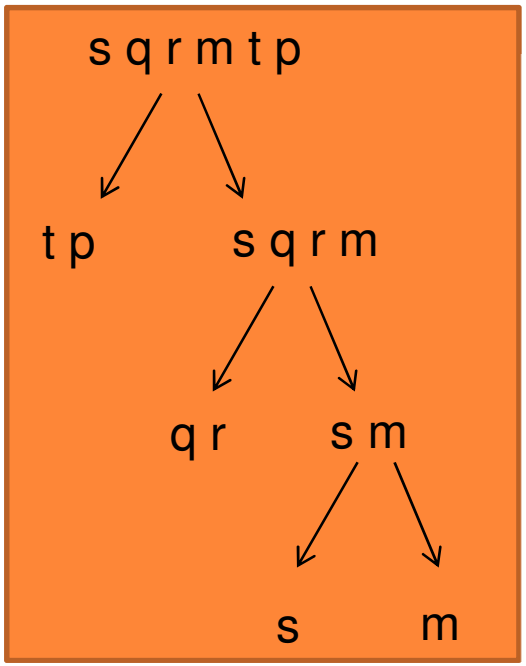



# Hopcroft's algorithm: the derivation tree




The Derivation tree describes the refinement process

# General features of the algorithm

- The refinement process leading to the final partition could be not unique.
- For a given automaton, there could be different executions with different time complexity.

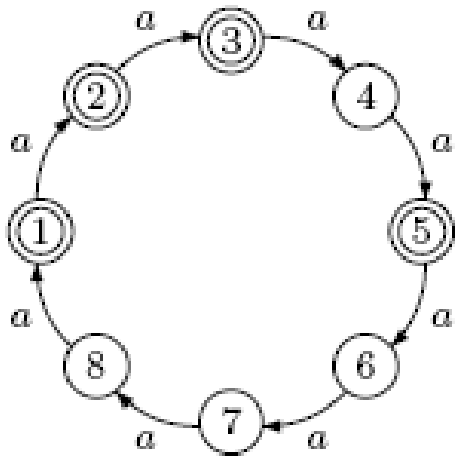
Question: Is the time complexity tight?

[Berstel, Carton 2004]
There exist cyclic unary automata where you need $\Omega(n \log n)$ time if you are "unlucky".
They are related to De Bruijn words.

# CYCLIC UNARY AUTOMATA

$w$= 11101000

$A_w$




w is a word
$a_1a_2\ldots a_n$ over A={0,1}.

$A_w$=(Q,Σ,δ,F) the cyclic automaton associated to w:
Q={1,2,…,n}, Σ={a}

δ(i,a)=(i+1), i<n
δ(n,a)=1

F={ i ∈ Q | $a_i$=1}

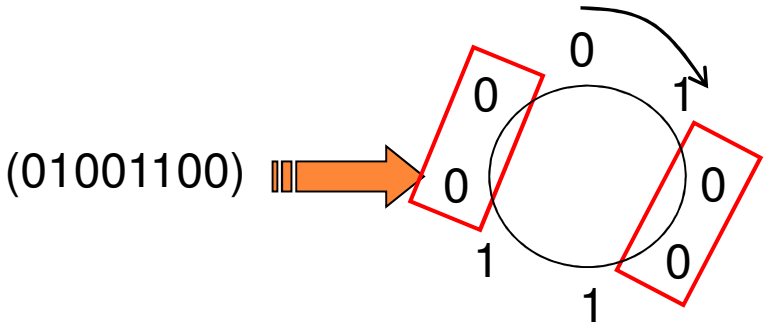
A natural question is:
do there exist infinite families of automata for which each execution of Hopcroft's algorithm runs in O(n log n)?

# Circular words

$A=\{0,1\}$ binary alphabet.

- $u,v \in A^*$ **conjugate**: $\exists$ $z,w$ in $A^*$ s.t. $v=zw$ and $u=wz$
- conjugacy is an equivalence relation: let $w \in A^*$, by **(w)** we denote the class of all the conjugates of the word w and we call it **circular word**;
- a **factor** of (w) is a factor of ww of length not greater than |w|;
- u is a **special factor** of (w) if both u0 and u1 are factors of (w).

**Example:**




00 is a special factor

# Circular standard words

Recall the notion of **standard word**:

$d_1, d_2, \ldots, d_n, \ldots$ a sequence of natural integers $d_1 \geq 0$, $d_i > 0$

$\{s_n\}_{n \geq 0}$ sequence of words over $A=\{0,1\}$ where

$$s_1 = 0,\ s_{n+1} = s_n^{d_n} s_{n-1} \text{ for } n \geq 1$$

Each finite word $s_n$ is called standard word.

$(d_1, d_2, \ldots, d_n, \ldots)$ **directive sequence**

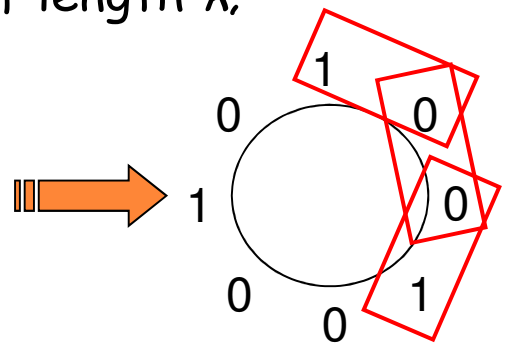
$(1,1,\ldots,1,\ldots)$ corresponds to Fibonacci words.

*(w)* is a **circular standard word** if *w* is a standard word.

# Circular standard words

**Proposition (Borel, Reutenauer).** Let $v$ be a word of length $n \geq 2$. $(v)$ is a circular standard word iff for $k=0,\ldots,n-1$, $(v)$ has exactly $k+1$ factors of length $k$.

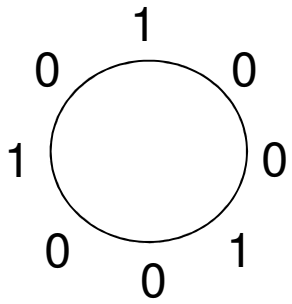
(10010010) is standard




For k=2, there are 3 distinct factors of length 2

**Proposition.** Let $v$ be a word of length $n$. $(v)$ is standard **iff** $\forall$ $k=0,\ldots,n-2$, $(v)$ has a unique special factor of length $k$.

(10010010) is standard




Special factors:

| | |
|---|---|
| k=0 | ε |
| k=1 | 0 |
| k=2 | 10 |
| k=3 | 010 |
| k=4 | 0010 |
| k=5 | 10010 |
| k=6 | 010010 |

# Some answers on unary automata

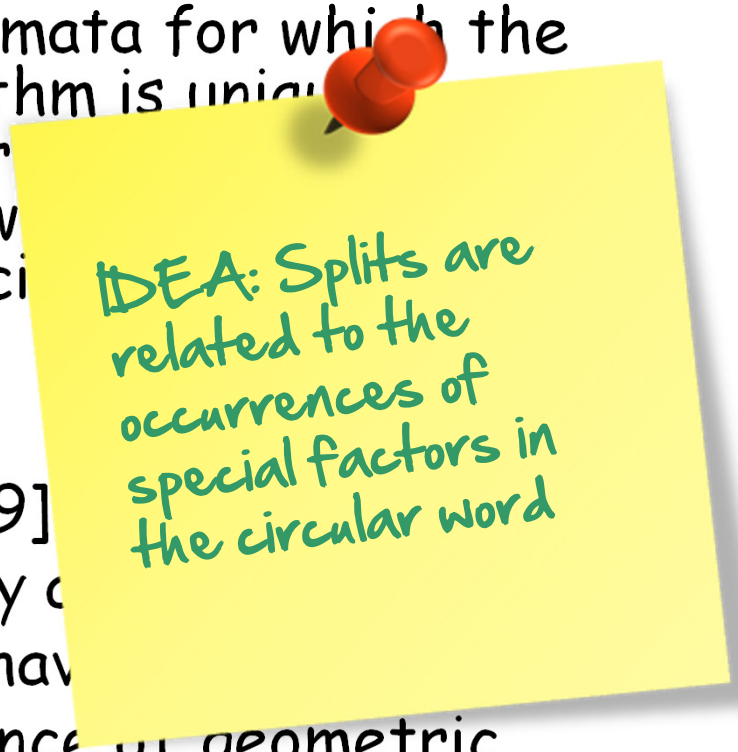
- [Castiglione, Restivo, S. 2008]
  - There exist cyclic unary automata for which the execution of Hopcroft's algorithm is unique. They are associated to circular standard words.
  - There exist unary automata where you need always $\Omega(n \log n)$ time. They are associated to circular Fibonacci words.

- [Berstel, Boasson, Carton, 2009]
  The same holds for cyclic unary automata associated to all circular standard words having directive sequences for which the sequence of geometric means is bounded.

# Some answers on unary automata

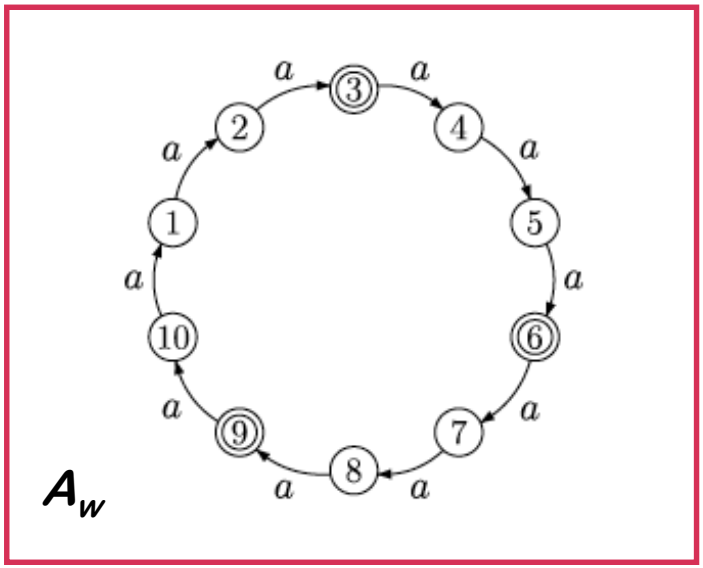
- [Castiglione, Restivo, S. 2008]
  - There exist cyclic unary automata for which the execution of Hopcroft's algorithm is unique. They are associated to circular
  - There exist unary automata w
  $\Omega(n \log n)$ time. They are associ
  Fibonacci words.

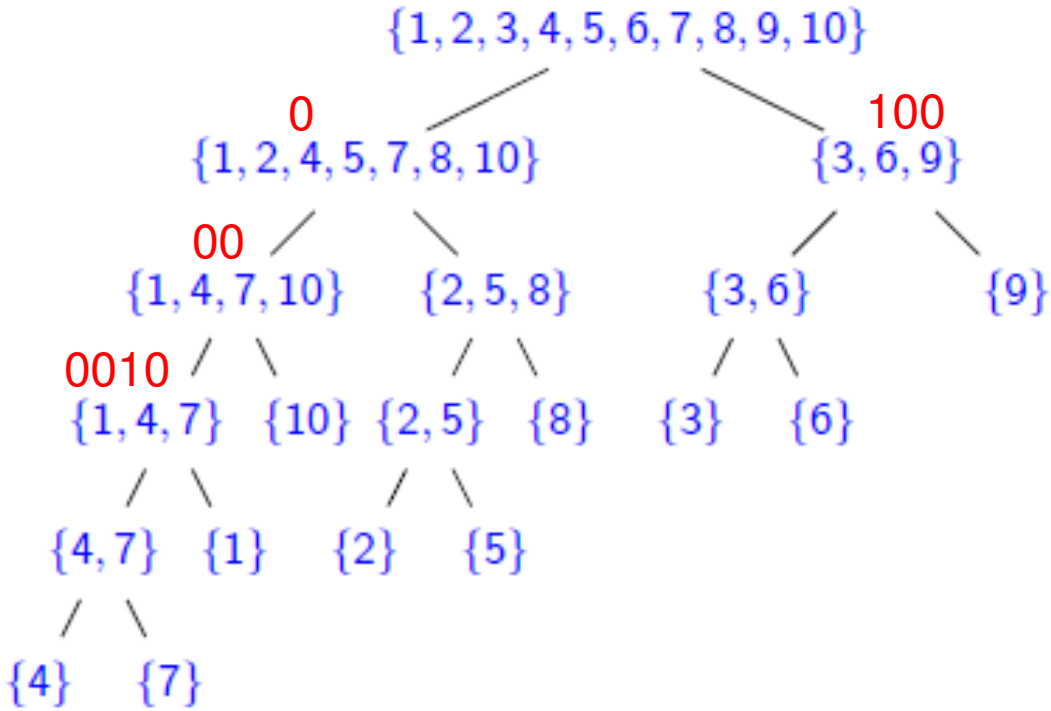
- [Berstel, Boasson, Carton, 2009]
  The same holds for cyclic unary
  to all circular standard words hav
  sequences for which the sequence of geometric means is bounded.



# Derivation Tree

w= 0010010010

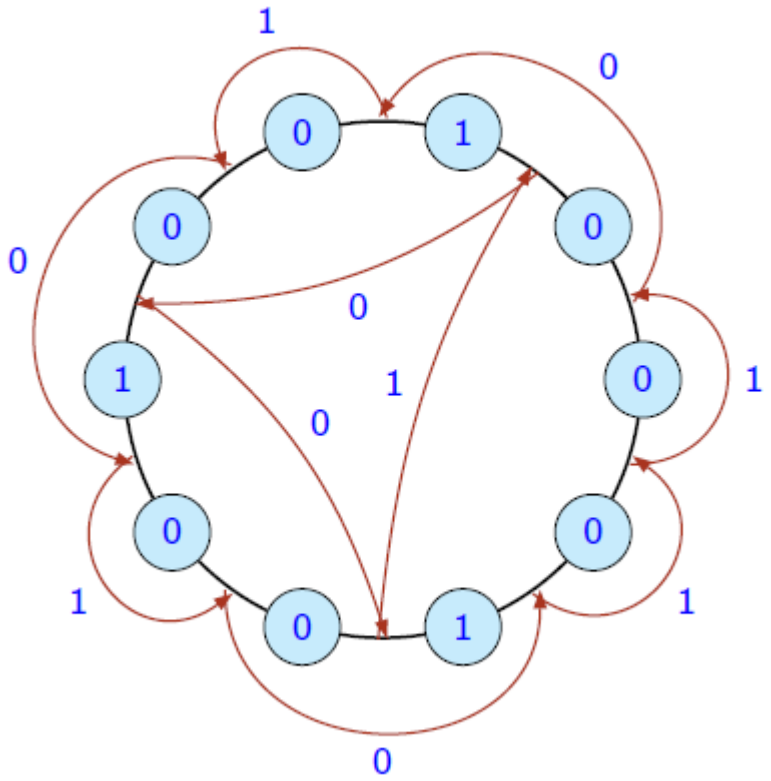



And so on…

# Factorization of circular standard words

w= 0010010010



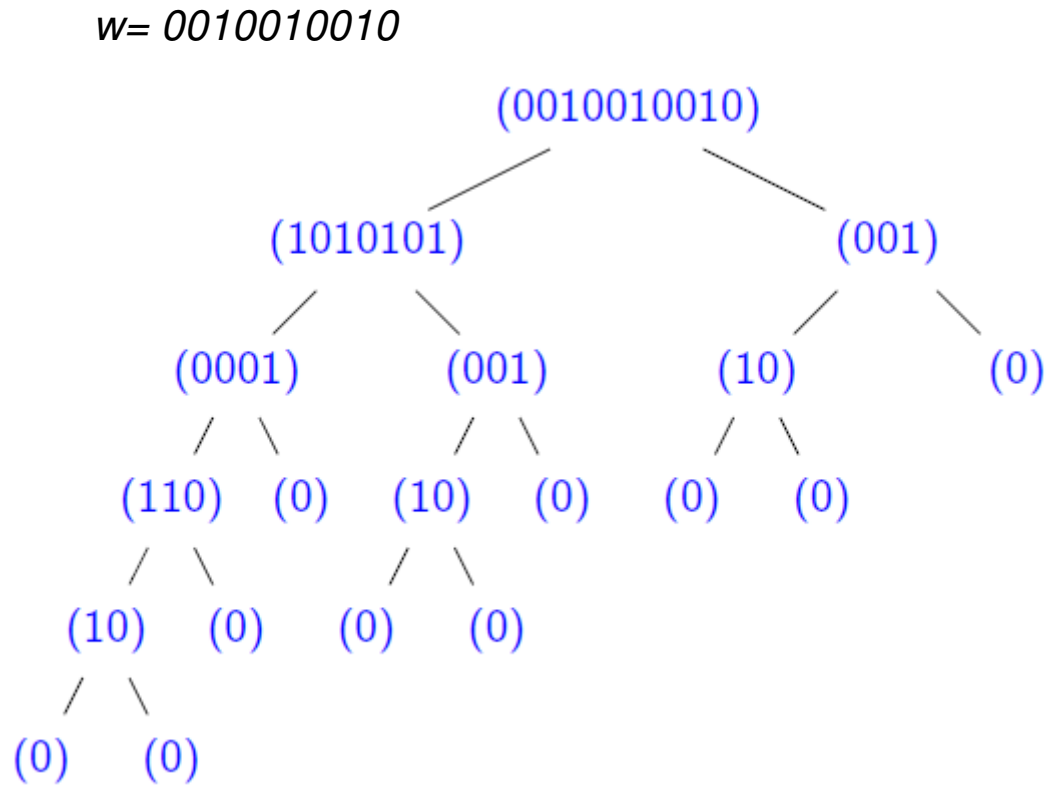

- Each circular standard words can be uniquely "circularly" factored in {0,01} and $\{10^p, 10^{p+1}\}$.
- By using such encoding we obtain again circular standard words.
- Example:

w=0010010010=

- 0|01|0|01|0|01|0|
  ->1010101 denoted by L(w)
- 00|100|100|10
  ->100 denoted by R(w)

# Reduction tree of a circular standard word

- if (w)=(0) or (w)=(1), $\tau(w)$ is a single node with label (0);
- if $|w| > 1$, $\tau(w)$ is a tree with root labeled by (w) having respectively as left and right subtrees $\tau(L(w))$ and $\tau(R(w))$.

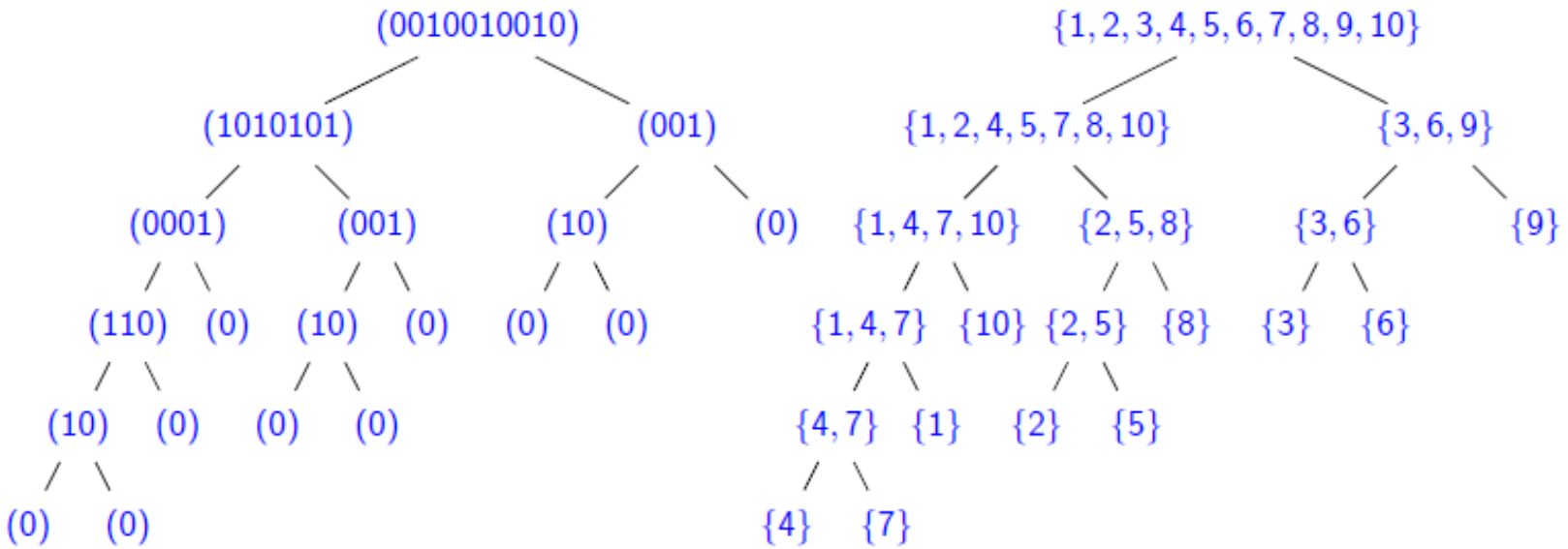



- THEOREM (Castiglione, Restivo, S. 2009): Let (w) and (v) be circular standard words, then $\tau(w)=\tau(v)$ iff (w)=(v) up to exchanging 0 with 1.

# ISOMORPHISM BETWEEN TREES

- THEOREM (Castiglione, Restivo, S. 2009):
  If (w) is a circular standard word then the derivation tree of $A_w$ and the reduction tree are isomorphic.

*w= 0010010010*



# TIGHTNESS PROBLEM: UNARY CASE?

**REMARK:**

The unary case is a very special case because automata minimization can be achieved also in linear time when the alphabet has only one letter [Paige,Tarjan, Bonic, 1985]

Do there exist infinite families of binary automata for which each execution of Hopcroft's algorithm runs in O(n log n) ?

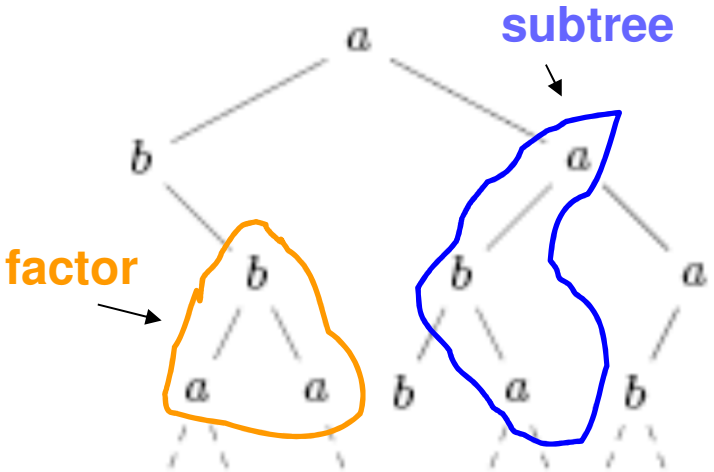
# LABELED BINARY TREES

Let $\Sigma=\{0,1\}$ (alphabet for the shape) and $A=\{a,b\}$ (alphabet for the labels) be two binary alphabets.

A binary labelled tree is a map $\tau: \Sigma^* \to A$ whose domain *dom*($\tau$) is a prefix-closed subset of $\Sigma^*$.

If *dom*(t) is finite, the tree is finite, else the tree is infinite.

EXAMPLE




Height of a finite tree: the maximal length plus 1 of the elements of *dom*(t).

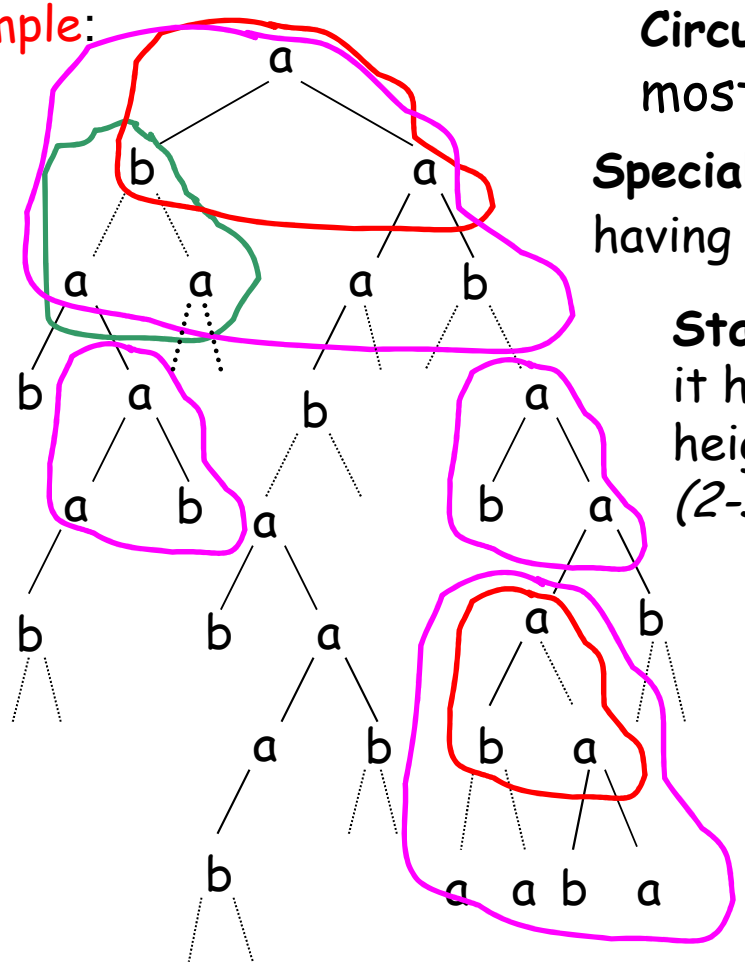
Complete infinite tree:
the domain is $\Sigma^*$.
Complete finite tree of height n:
the domain is $\Sigma^{n-1}$.

Factor: a finite complete subtree

# Standard trees

Given a finite tree $\tau$, we denote by $\tau^\omega$ the complete infinite tree obtained by recursively concatenating $\tau$ to each node of the frontier.

Example:



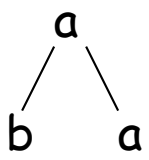

**Circular factor:** a factor of $\tau^\omega$ of height at most $h(\tau)$

**Special circular factor:** a circular factor having at least two complete extensions in $\tau^\omega$

**Standard tree:** if for each $0 \le h \le h(\tau)-2$ it has exactly one special circular factor of height $h$ and it has exactly 2 estensions *(2-special circular factor)*.

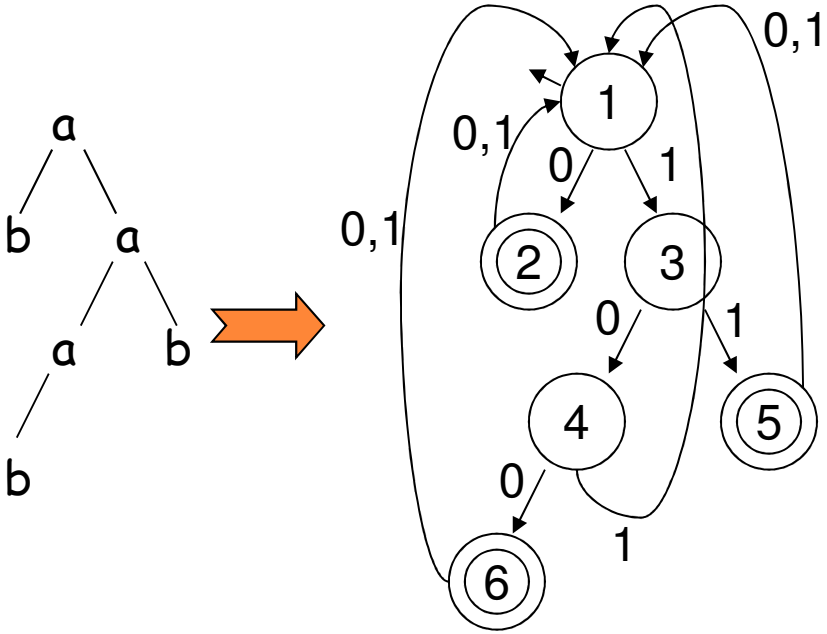
h=0 Empty tree

h=1 a is 2-special

h=2  is 2-special

# TREE-LIKE AUTOMATA

- Given a finite binary labeled tree $\tau$ we can uniquely associate a tree-like automaton $\mathbf{A}_\tau$ having $\tau$ as skeleton.

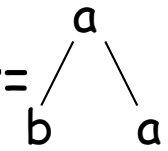



- For each missing edge we add a transition to the root of the tree.
- Moreover, the root is the initial state and the states corresponding to nodes labelled by a (resp. b) are non-final (resp. final) states.

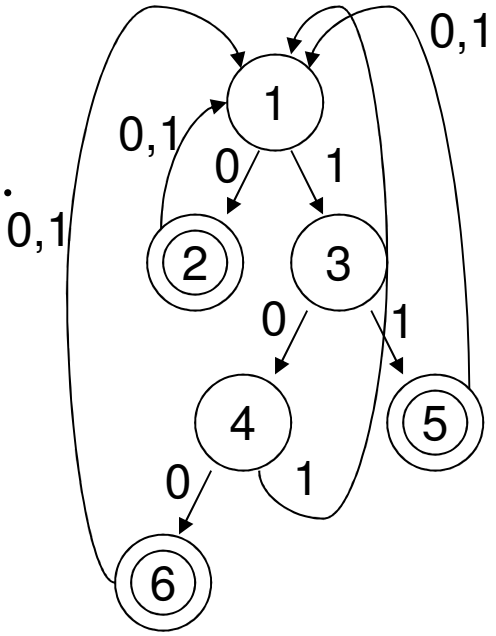
# Hopcroft's Algorithm on Standard Tree-Like Automata

*A* standard tree-like automaton *is* a tree-like automaton $A_\tau = (Q, \Sigma, i, \delta, F)$ associated to a standard tree $\tau$.

We denote by $Q_\sigma$ the set of states where the tree $\sigma$ occurs as a circular factor.

For instance if $\sigma =$ 

then $Q_\sigma = \{1,4\}$




**THEOREM (Castiglione, Restivo, S. 2010):** Let $A_\tau$ be a standard tree-like automaton. The sequence of the refinement process $\Pi_1, \Pi_2, \ldots \Pi_m$ is uniquely determined, $m = h(\tau) - 2$ and $\Pi_k = \{Q_\sigma \mid \sigma$ is a circular factor of $\tau$ with $h(\sigma) = k\}$ and $|\Pi_k| = k+1$

# Idea of the proof

The proof is by induction on k.

It uses the following result:

**Proposition:** Let $A_\tau$ be a standard tree-like automaton. Let $Q_\sigma$ and $Q_\gamma$ be classes of a partition of Q.
If $(Q_\gamma, x)$ splits $Q_\sigma$, for some $x \in \Sigma$, with $h(\gamma) = h(\sigma)$, then $\sigma$ is a 2-special circular factor of $\tau$. The resulting classes are $Q_{\sigma'}$ and $Q_{\sigma''}$, where $\sigma'$ and $\sigma''$ are the only two possible extensions of $\sigma$ in $\tau$.

We proved that at each step, the splitter in the waiting set satisfies the hypothesis of the proposition.

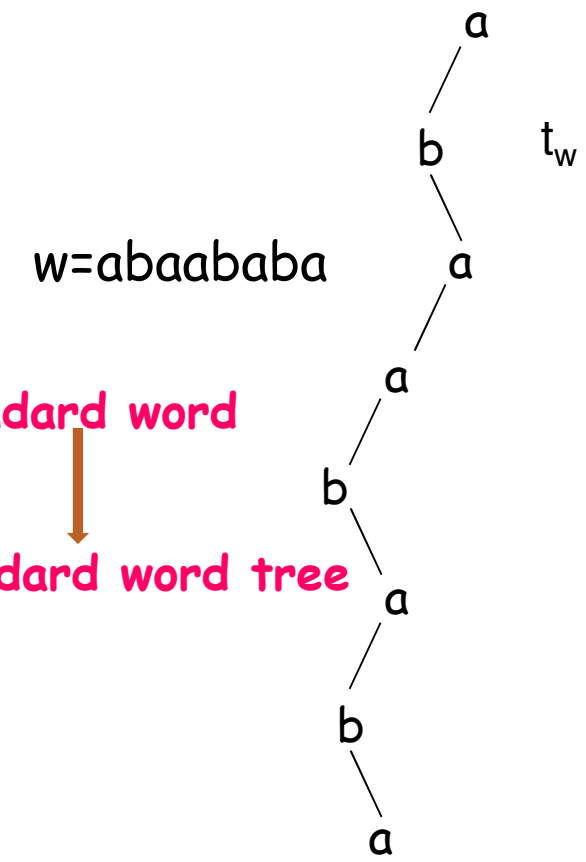
# Word Trees and word automata

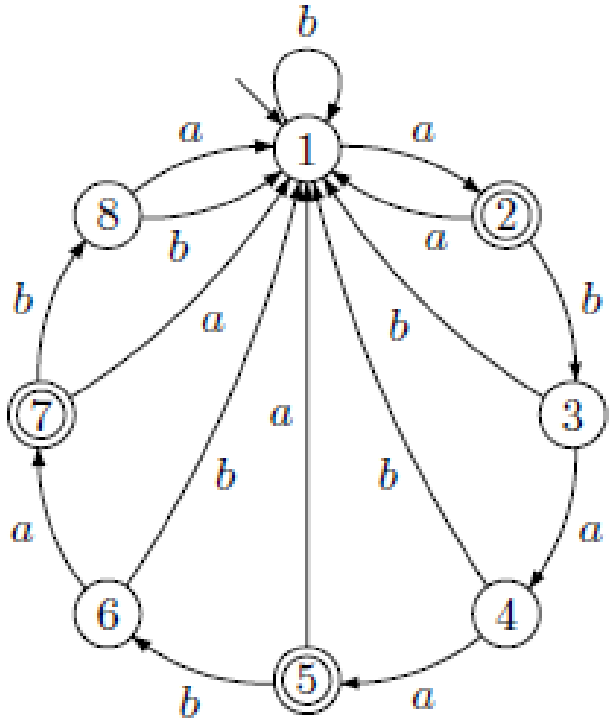
Given a word w, a word tree $t_w$ is a binary labeled tree in which the alphabet of the shape and the labels coincide. Moreover, $dom(t_w)$ is the set of prefixes of w. The word automaton $A_w$ is the tree-like automaton associated to $t_w$

w=abaababa

standard word

standard word tree



# Tightness of Hopcroft's algorithm

We consider the executions of Hopcroft's algorithm on binary word automata.

THEOREM (Castiglione, Restivo, S. 2010): There exists an infinite family of binary automata for which each execution of Hopcroft's algorithm runs in time $\Theta(n \log n)$.

The bound is tight on word automata associated to standard word trees constructed with Fibonacci words.

# In the proof

We prove and use the following:

**Proposition:** Let $\tau_w$ be a standard word tree.
Each execution of Hopcroft's algorithm on $A\tau_w$ has time complexity $C_H$ satisfying

$$\sum_{\sigma \in sp(\tau w)} \min(|Q_{\sigma'}|, |Q_{\sigma''}|) + n - 1 \leq C_H(A\tau_w) \leq 2 \times \sum_{\sigma \in sp(\tau w)} \min(|Q_{\sigma'}|, |Q_{\sigma''}|)$$

where with $sp(\tau_w)$ we denote the set of 2-special circular factor of $\tau_w$.

REMARK: If we consider the circular standard word $w=a^nb$, all the executions run in time $\Theta(n)$. If word automata associated to fibonacci words, the time complexity is $\Theta(n \log n)$.

# In the proof

We prove and use the following:

**Proposition:** Let $\tau_w$ be a standard word tree.
Each execution of Hopcroft's algorithm on $A\tau_w$ has time complexity $C_H$ satisfying

$$\sum_{\sigma \in sp(\tau w)} \min(|Q_{\sigma'}|, |Q_{\sigma''}|) + n - 1 \leq C_H(A\tau_w) \leq 2 \times \sum_{\sigma \in sp(} \min(|Q_{\sigma'}|, |Q_{\sigma''}|)$$

where with $sp(\tau_w)$ we denote the set of 2- f $\tau_w$.

REM the circular ll the
exe ). If word aut onacci
wor is $\Theta(n \log n)$.

In case of standard words, time complexity does not depend from implementation of the waiting set!

For de Brujin words, a stack implementation runs in $\Theta(n)$, a queue implementation in $\Theta(n \log n)$.

# SOME RESULTS ON AVERAGE

[David 2012]

- For the uniform distribution on complete deterministic automata, the average time complexity of Moore's state minimization algorithm is O(nloglogn).
- There is a family of implementations of Hopcroft's state minimization algorithm are always faster than Moore's algorithm.

[Bassino, David, Sportiello 2012]

- For the uniform distribution on complete deterministic accessible automata, there exists a family of implementations of Hopcroft's state minimization algorithm whose average complexity is O(n log log n).

# What about other minimization algorithms?

- Word automata associated to Fibonacci words are a challenging class for other minimization algorithms.

# Brzozowski's method

Given the automaton $A = (Q,\Sigma,\delta,q_0,F)$ recognizing the language L

- With $d(A)$ we denote the deterministic automaton equivalent to $A$ (obtained by the subset construction).
- With $r(A)$ we denote the reverse of $A$.

The minimal automaton recognizing L is $d(r(d(r(A))))$

# Time complexity of Brzozowski's algorithm

- Since the reverse of an automaton can be computed in linear time with respect to the number of states and transictions, the critical part is the accessible subset construction.
- Time complexity is exponential in the worst case.
- The sum of the size of all accessible states in the subset construction $d(r(\mathcal{A}))$, is a lower bound of the time complexity of the accessible subset construction and then a lower bound for the algorithm.

# Brzozowski's algorithm and word automata

- [Castiglione, Nicaud, S. 2011]
  Brzozowski's algorithm has a $\Omega(n^2)$ time complexity for word automata constructed by Fibonacci words.

# Other results

- De Felice, Nicaud: Brzozowski Algorithm is Generically Super-Polynomial for Deterministic Automata, 2013.
- Brzozowski, Tamm: Minimal Nondeterministic Finite Automata and Atoms of Regular Languages, 2013
- Brzozowski, Tamm: Quotient Complexities of Atoms of Regular Languages, 2012.
- Vazquez de Parga, García, López: A polynomial double reversal minimization algorithm for deterministic finite automata, 2013
- Vazquez de Parga, García, López: DFA minimization: from Brzozowski to Hopcroft, 2013

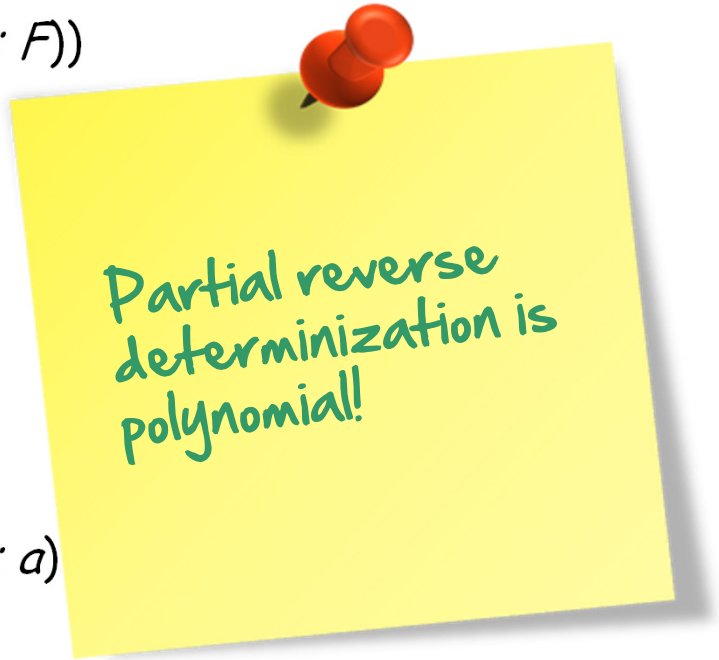
# What about the variants?

- We consider the recent variant proposed by [Garcia, Lopez, Vazquez de Parga, 2013]

# Partial reverse determinization

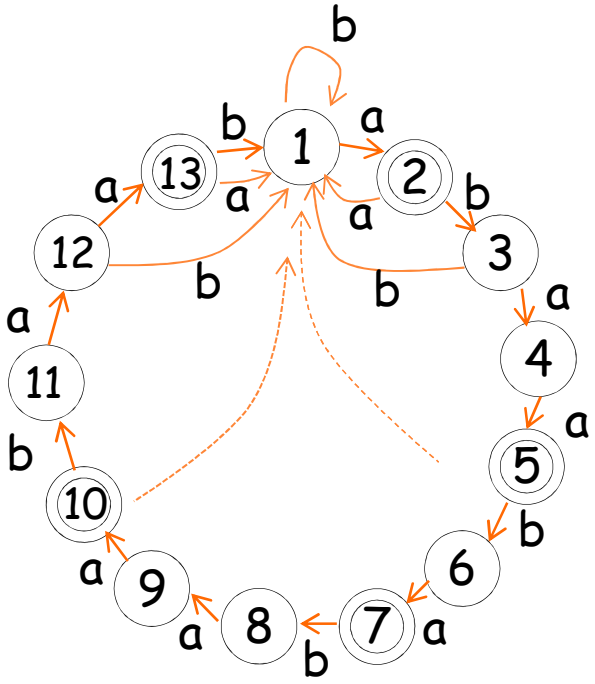
Minimization by PRD ($A = (Q; \Sigma; \delta; q_0; F)$)

1. $\Pi \leftarrow \{F; Q \setminus F\}$
2. $S = F$
3. $L = \emptyset$
4. **for all** $a \in \Sigma$ **do**
5. $L \leftarrow (S; a)$
6. **while** $L \neq \emptyset$ **do**
7. *choose and delete a pair* $(S; a)$
8. **for all** $B \in \Pi$ **do**
9. **if** $B$ **is split by** $(S; a)$ **then**
10. $B' \leftarrow \delta^{-1}{}_a(S) \cap B$
11. $B'' \leftarrow B \setminus \delta^{-1}{}_a(S)$
12. $\Pi \leftarrow \Pi \setminus B \cup \{B' \cup B''\}$
13. **for all** $b \in \Sigma$ **do**
14. $L \leftarrow L \cup \{(\delta^{-1}{}_a(S), b)\}$



# PRD ALGORITHM ON WORD AUTOMATA

- We consider the word automaton associated to fibonacci word $f_n$
- Example $f_6$= abaababaabaab

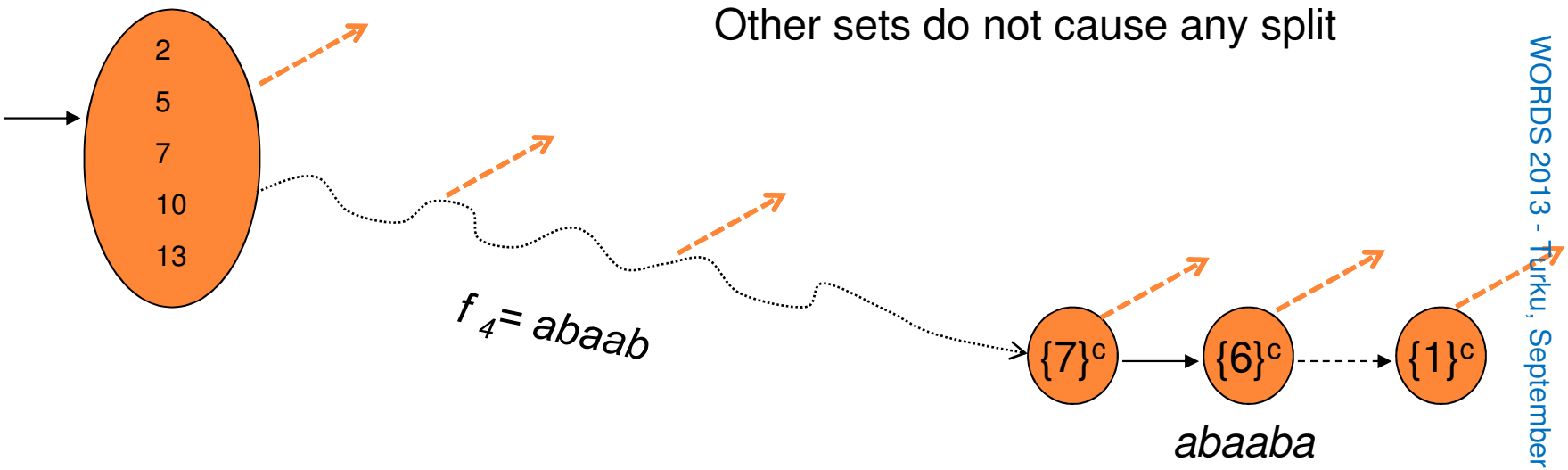



In order to find the time complexity we have to consider the set of states going into L

# PRD ON WORD AUTOMATA

If n is even, if we consider a set of $\delta^{-1}$ operations by using the letters of $f_{n-2}$, we obtain sets that are complement of singletons

**Ex. $f_6$ = abaababaabaab**

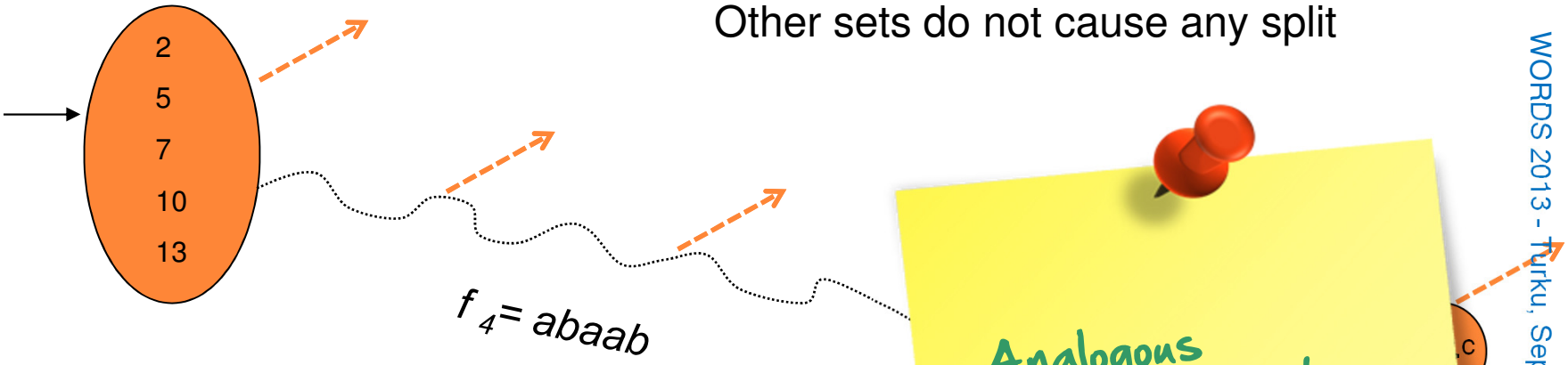



- At each step only a subset causes a split and then goes into the waiting set L.
- The subsets $\{F_{n-1}-1\}^c$, ...,$\{1\}^c$ are obtained as inverse image by $\delta$ and cause a split.
- The waiting set contains $F_{n-1}$-1 sets of cardinality $F_n$-1

# PRD ON WORD AUTOMATA

If n is even, if we consider a set of $\delta^{-1}$ operations by using the letters of $f_{n-2}$, we obtain sets that are complement of singletons

**Ex. $f_6$ = abaababaabaab**




Other sets do not cause any split

Analogous reasoning can be done when n is odd.

- At each step only a subset causes a split and … set L.
- The subsets $\{F_{n-1}-1\}^c$, …,$\{1\}^c$ are obtained as i… cause a split.
- The waiting set contains $F_{n-1}$-1 sets of cardinality $F_n$-1

# TIME COMPLEXITY FOR STANDARD WORD AUTOMATA

- THEOREM: Let $A_w$ the word automaton associated to a circular standard word. The minimal automaton is obtained by PRD algorithm by a sequence of $\delta^{-1}_{b_1}$ , $\delta^{-1}_{b_2}$ , …, $\delta^{-1}_{b_{|w|-2}}$ where $b_1b_2…b_{|w|-2}$ is the prefix of length $|w|-2$ of w.

- THEOREM: Let $A_{fn}$ the word automaton associated to the n-th Fibonacci word $f_n$ with $F_n$ states. Then

$$C_{PRD}(F_n)= \Theta(F_n^2)$$

They represent the worst case of the algorithm!

# Problems and perspectives

- Are Fibonacci word automata challenging for the minimization process? Is there any better minimization algorithm for DFA?

- Characterizing the words for which the correspondent word automata are always difficult to be minimized.

# Thanks for Your attention...!!!