On the transition reduction problem for finite automata

El Houcein El Abdalaoui, Mohamed Dahmoune, Djelloul Ziadi

On the transition reduction problem for finite automata

el Houcein el Abdalaoui[3], Mohamed Dahmoune[1] and Djelloul Ziadi[2]

[1] LACL, EA 4912, Université Paris-Est Créteil (UPEC), France
Mohamed.Dahmoune@u-pec.fr
[2] LITIS, Université de Rouen, France Djelloul.Ziadi@univ-rouen.fr
[3] LMRS, Université de Rouen, France elhoucein.elabdalaoui@univ-rouen.fr

Abstract. We are interested in the problem of transition reduction of nondeterministic automata. We present some results on the reduction of the automata recognizing the language $L(E_n)$ denoted by the regular expression $E_n = (1+\varepsilon)\cdot(2+\varepsilon)\cdot(3+\varepsilon)\cdots(n+\varepsilon)$. These results can be used in the general case of the transition reduction problem.

1 Introduction

Minimizing the number of states of an automaton is a subject that has been studied extensively since the 1950s, both in the deterministic case and the nondeterministic case [11,7]. However, works on the minimization of the number of transitions have appeared recently.

In 1997, J. Hromkovič *et al.* [8] have proposed an algorithm based on the concept of Common Follow Set of a regular expression, that converts a regular expression of size n into a finite state automaton with $O(n)$ states, $O(n\log n)$ transitions as lower bound and $O(n\log^2 n)$ transitions as upper bound. Muscholl *et al.* [6], showed that this algorithm can be implemented in time $O(n\log^2 n)$. In [9] Ouardi and Ziadi, based on the ZPC structure [2], gave an $O(n\log^2 n)$ algorithm to convert a weighted regular expression of size n into a weighted automaton having $O(n)$ states and $O(n\log^2 n)$ transitions. In [5], Viliam Geffert showed that every regular expression of size n over a fixed alphabet of s symbols can be converted into a nondeterministic ε-free finite state automaton with $O(sn\log n)$ transitions.

Lower bound was improved by Yuri Lifshits [15] to $\Omega(\frac{n\log^2 n}{\log\log n})$, after, Schnitger [13] improved it to $\Omega(n\log^2 n)$ transitions.

In [3], R. Cox has done an exhaustive search to find the transition minimal automata of $L(E_n)$ for $n = 1$ to 7. He has also used an heuristic approach that construct transition reduced automata for $n = 8$ to 10.

Here, we are able to produce an algorithm for which the number of transitions is minimal for $L(E_n)$ languages class, in the sense that, asymptotically, this number of transitions is equivalent to $n\log^2 n$ (see Section 6).

We mention that most of complexity results mentioned above are obtained from the study of $L(E_n)$ languages class. This class of languages corresponds

to a simple class of automata, in which, the minimization of the number of transitions is difficult and not obvious. The study of this class of languages, can also find its application in bioinformatics, since that $L(E_n)$ is exactly the set of all sub-sequences of the word $1.2.3 \ldots n$.

Our approach to reduce the number of transitions of a nondeterministic homogeneous finite state automaton is based on the decomposition of the transition table of the automaton into blocks. This decomposition is based on the concept of Common Follow Sets. From a block decomposition we construct an automaton with less transitions than the initial automaton. See Figure 1.

The main problem in our approach is to find a good block decomposition of the transition table (even the best one). In the case where this matrix is lower triangular or upper triangular, finding a minimal decomposition block leading to a minimal transition automaton, is not evident. Our study is focused on the upper triangular matrix, which corresponds to the transition table of the deterministic minimal automaton recognizing the $L(E_n)$ language. The case of lower triangular matrix can be obtained in a similar manner.

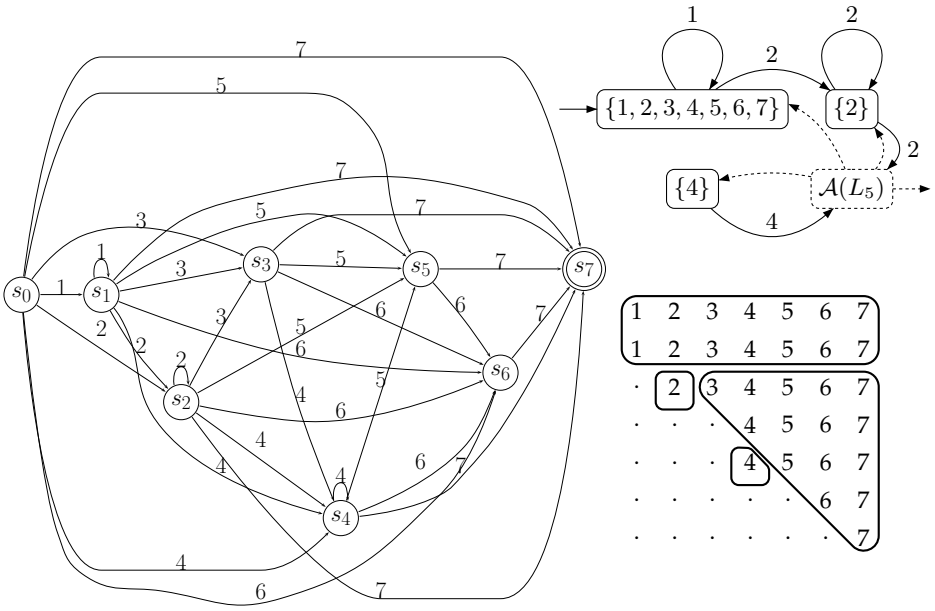


Fig. 1. The reduced automaton (at right) is obtained from a decomposition transition table (at right bottom) of the homogeneous automaton (at left). The automaton $\mathcal{A}(L_5)$ is the part of the reduced automaton which represents the triangle (in the transition table decomposition).

In this paper we present the following results: At first, in Section 3, we extend the concept of Common Follow Sets to homogeneous automata. Then, in Section 4 we introduce particular decompositions called Z-partitions associated with expressions E_n. Then, in Section 5 we introduce the notion of Z-tree

to represent any Z-partition by a binary tree. Then, we propose an algorithm of $O(n \log n)$ time complexity to generate the Z-minimal trees. We finish our study by experimental results and a last section in which we show that our algorithms construct automata with a number of transitions equivalents to $n \log^2(n)$ which is the minimal lower bound according to Schnitger.

2 Notation and terminology

We recall the basics of regular expressions, languages and finite state machines and introduce the notation that we use. Let Σ be a non-empty finite set of symbols, called alphabet. The set of all the words over Σ is denoted by Σ^*. The empty word is denoted by ε. A language over Σ is a subset of Σ^*. A finite automaton over Σ is a 5-tuple $\mathcal{A} = (Q, \Sigma, I, \delta, F)$ where Q is a set of states, I is a subset of Q whose elements are the initial states, F is a subset of Q whose elements are the final states, δ is a subset of the cartesian product $Q \times \Sigma \times Q$ whose elements are the transitions. A transition $(q, a, p) \in \delta$ goes from the head q to the tail p. A path in $\mathcal{A}$ is a sequence of transitions (q_i, a_i, q_{i+1}), $i = 1$ to n, of consecutive transitions. Its label is the word $w = a_1 a_2 \cdots a_n$. A word $w \in \Sigma^*$ is recognized by the automaton $\mathcal{A}$ if there is a path with label w such that $q_1 \in I$ and $q_{n+1} \in F$. The language recognized by the automaton $\mathcal{A}$ is the set of words that are recognized by $\mathcal{A}$. The automaton $\mathcal{A}$ is homogeneous if for all $(q, a, p),\ (q', a', p') \in \delta$, $p = p'$ implies that $a = a'$, in this case we write $h(p) = a$. The function h assigns to each non-initial state q of an homogeneous automaton the symbol that is the unique label of all the transitions having q as tail.

In Appendix A we recall the basics of asymptotic notations.

3 CFS for homogeneous automata

J. Hromkovič *et al.* [8] have given an elegant algorithm based on the notion of Common Follow Sets, to convert a regular expression of size n into a nondeterministic finite automaton having $O(n)$ states and $O(n \log^2 n)$ transitions. This notion can be easily extended to homogeneous automata.

Let $\mathcal{A} = (Q, \Sigma, \{q_0\}, \delta, F)$ be an homogeneous automaton. In order to capture the final states in the $\mathcal{A}$, we introduce a dummy state denoted by $\#$ which is not in Q. We define over Q the function $follow$ as follows:

$$follow(q) = \begin{cases} \{p \mid (q, a, p) \in \delta\} \cup \{\#\} \text{ if } q \in F, \\ \{p \mid (q, a, p) \in \delta\} \text{ otherwise.} \end{cases}$$

Let $q \in Q$ be a state in $\mathcal{A}$, we denote by $dec(q) = \{Q_1, Q_2, \cdots, Q_k\}$ (where $Q_i \subseteq follow(q)$) any decomposition of the set $follow(q)$, i.e. $follow(q) = \bigcup_{Q_i \in dec(q)} Q_i$. In the case where $dec(q)$ is a partition of the set $follow(q)$, the decomposition $dec(q)$ will be called a partition decomposition. Figure 2 provides examples of decompositions.

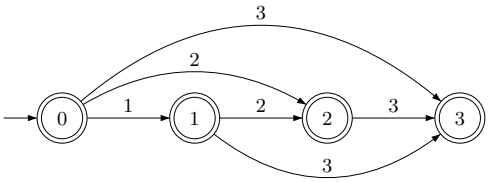


Fig. 2. We have $follow(0) = \{1, 2, 3, \#\}$. Here are three possible decompositions of $follow(0)$. The two first ones are partition decompositions. (i) $dec(0) = \{\{1, 2\}, \{3, \#\}\}$ (ii) $dec(0) = \{\{1\}, \{2\}, \{3, \#\}\}$ (iii) $dec(0) = \{\{1, 2\}, \{2, 3, \#\}\}$.

Definition 1 (Common Follow Sets System). *Let $\mathcal{A}$ be a homogeneous automaton. A CFS system for $\mathcal{A}$ is given as $S(\mathcal{A}) = (dec(q))_{q\in Q}$, where each $dec(q) \subseteq 2^Q$ is a decomposition of $follow(q)$.*

Definition 2 (CFS automaton). *Let $\mathcal{A} = (Q, \Sigma, \{q_0\}, \delta, F)$ be a homogeneous automaton and $S(\mathcal{A})$ an associated Common Follow Sets system. The Common Follow Sets automaton associated with $S(\mathcal{A})$ is defined by $\mathcal{C}_{S(\mathcal{A})} = (Q', \Sigma, I', \delta', F')$ where*

- $Q' = \bigcup_{q\in Q} dec(q)$
- $I' = dec(q_0)$
- *For $Q_1 \in Q'$, $Q_1 \in F'$ if and only if $\# \in Q_1$*
- $\delta' = \{(Q_1, a, Q_2) \mid \exists q \in Q_1 \text{ s.t. } h(q) = a \text{ and } Q_2 \in dec(q)\}$.

Theorem 1. *Let $\mathcal{A}$ be a homogeneous automaton, $S(\mathcal{A})$ be a Common Follow Sets System associated with $\mathcal{A}$ and $\mathcal{C}_{S(\mathcal{A})}$ its Common Follow Sets automaton. Then $\mathcal{C}_{S(\mathcal{A})}$ and $\mathcal{A}$ recognize the same language.*

This theorem can be proved in the same way as Theorem 5 of the paper of J. Hromkovič *et al.* [8].

To evaluate the number of transitions in the automaton $\mathcal{C}_{S(\mathcal{A})}$ we define over the states of $\mathcal{A}$ two functions,

- $a(q) = |dec(q)|$ the size of the decomposition of the set $follow(q)$
- $b(q) = |\{Q_1 \in Q' \mid q \in Q_1\}|$ the number of states in Q' that contain the state q.

Lemma 1. *The number of transitions $T_{\mathcal{C}}$ in $\mathcal{C}_{S(\mathcal{A})}$ is such that $T_{\mathcal{C}} \leq \sum_{q\in Q} a(q)b(q)$.*

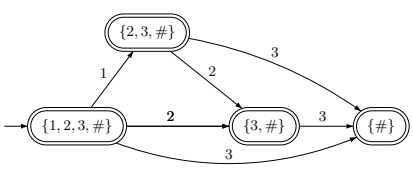
It is easy to see that if for all $p, q \in Q\backslash\{q_0\}$ such that $p \neq q$, we have $h(p) \neq h(q)$ then the equality holds. From Lemma 1, we can see that the number of transitions in a CFS automaton depends on the decomposition system. A decomposition which is not a partition will induce more transitions than a partition decomposition. Therefore in the following we are interested only in partition decompositions. As it was mentioned in the introduction our study will focus on the CFS automata associated with the family of automata $(\mathcal{A}_n)_{n\geq 1}$. The automaton $\mathcal{A}_n = (Q, \Sigma, I, \delta, F)$ is defined by:

- $\Sigma = \{1, 2, \ldots, n\}$

- $Q = \Sigma \cup \{0\}$
- $F = Q$
- $I = \{0\}$, $\delta = \{(p, q, q) \in Q \times \Sigma \times Q \mid q > p\}$.

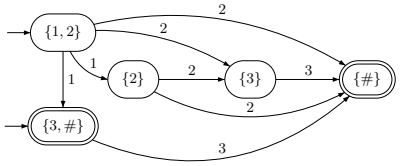
Figure 3 shows two CFS automata associated with the automaton $\mathcal{A}_3$.

In the next sections we present two algorithms that construct particular CFS systems which correspond to CFS automata with a reduced number of transitions. In the last section we give comparative and experimental results.



q	$a(q)$	$b(q)$	$a(q)b(q)$	$dec(q)$
0	1	0	0	$\{\{1, 2, 3, \#\}\}$
1	1	1	1	$\{\{2, 3, \#\}\}$
2	1	2	2	$\{\{3, \#\}\}$
3	1	3	3	$\{\{\#\}\}$

$T_{\mathcal{C}} = 6$



q	$a(q)$	$b(q)$	$a(q)b(q)$	$dec(q)$
0	2	0	0	$\{\{1, 2\}, \{3, \#\}\}$
1	2	1	2	$\{\{2\}, \{3, \#\}\}$
2	2	2	4	$\{\{3\}, \{\#\}\}$
3	1	2	2	$\{\{\#\}\}$

$T_{\mathcal{C}} = 8$

Fig. 3. Two CFS Automata constructed from the automaton $\mathcal{A}_3$ shown in Figure 2.

4 $\mathcal{A}_n$ Reduction Algorithm

The notion of *Z-partition* is nowhere introduced formally. In the following we are interested in reducing the number of transitions in the automaton $\mathcal{A}_n$. The following algorithm computes particular CFS systems $S(\mathcal{A}_n)$ that provide $\mathcal{C}_{S(\mathcal{A}_n)}$ automata with small number of transitions and having $n + 1$ states. Let $E_n = (1 + \varepsilon) \cdot (2 + \varepsilon) \cdot (3 + \varepsilon) \cdots (n + \varepsilon)$ be a regular expression, it is easy to see that the language denoted by the expression E_n is exactly the language recognized by the automaton $\mathcal{A}_n$. We have:

Proposition 1. *Each transition minimal automaton that recognizes $L(E_n)$ has exactly $n + 1$ states.*

This proposition can be proved using properties of the universal automaton [10] of $L(E_n)$.

For fixed n, a CFS system produced by the following algorithm will be denoted by $Z(\mathcal{A}_n)$. The set of all $Z(A_n)$ will be denoted $CFSZ(n)$. Our aim in this section is to compute all minimal decompositions in $CFSZ(n)$.

Algorithm 1 CFSPartitions(n)

Require: $n \in \mathbb{N}$
Ensure: $Z(\mathcal{A}_n)$
1: $Q \leftarrow \{0, 1, 2, 3, 4, \ldots, n\}$
2: **for** i = 0 to n **do**
3: $follow(i) \leftarrow \{j \in Q | j > i\} \cup \{\#\}$
4: $dec(i) \leftarrow \phi$
5: $Q_i \leftarrow \phi$
6: **end for**
7: **for** i = 0 to n **do**
8: Choose j in Q
9: $Q \leftarrow Q \backslash \{j\}$
10: $Q_j \leftarrow follow(j)$
11: **for all** $k \in Q$ **do**
12: **if** ($Q_j \subseteq follow(k)$) **then**
13: $dec(k) \leftarrow dec(k) \cup \{Q_j\}$
14: $follow(k) \leftarrow follow(k) \backslash Q_j$
15: **end if**
16: **end for**
17: **end for**

Proposition 2. *The number of all $Z(\mathcal{A}_n)$ CFS partition systems is the n^{th} Catalan number: $|CFSZ(n)| = \frac{1}{n+1}\binom{2n}{n}$.*

The successive choice of values of j (line 8) leads to a permutation of size n. So, each CFS partition system $Z(\mathcal{A}_n)$ can be associated with at least one permutation of size n.

The following algorithm is a recursive version of Algorithm 1. Its first call is done by RecursiveDecomposition$(0, n)$. Without loss of generality we associate in this last algorithm the dummy state $\#$ to the number $n + 1$. At each call, Algorithm 2 constructs one block from the transition matrix M, for the call RecursiveDecomposition(n_1, n_2) and the choice of j (line 2), it produces the block B_j which is the submatrix $M[j..n_1; j..n_2]$.

Algorithm 2 RecursiveDecomposition (n_1, n_2)

Require: $n_1, n_2 \in \mathbb{N}$
Ensure: $Z(\mathcal{A}_n)$ when $n_1 = 0$ and $n_2 = n$
if $n_1 \leq n_2$ **then**
  Choose an integer j between n_1 and n_2, $j \in \{n_1, \ldots, n_2\}$
  $Q_j \leftarrow \{j+1, \ldots, n_2+1\}$
  for $k = n_1$ to j **do**
    $dec(k) \leftarrow dec(k) \cup \{Q_j\}$
  end for
  $B_j = M[j..n_1; j..n_2]$
  RecursiveDecomposition$(n_1, j-1)$
  RecursiveDecomposition$(j+1, n_2)$
end if

Example 1. In this example we shows the CFS partition systems associated with permutations $(0, 2, 1, 3)$ and permutation $(1, 0, 2, 3)$.

$dec(0) = \{ \{ 1 , 2 , 3 , \# \} \}$ 0 : $\boxed{1}\boxed{2}\boxed{3}\boxed{\#}$
$dec(1) = \{ \{ 2 \} , \{ 3 , \# \} \}$ 1 : $\boxed{2}\boxed{3}\boxed{\#}$
$dec(2) = \{ \{ 3 , \# \} \}$ 2 : $\boxed{3}\boxed{\#}$
$dec(3) = \{ \{ \# \} \}$ 3 : $\boxed{\#}$

$dec(0) = \{ \{ 1 \} , \{ 2 , 3 , \# \} \}$ 0 : $\boxed{1}\boxed{2}\boxed{3}\boxed{\#}$
$dec(1) = \{ \{ 2 , 3 , \# \} \}$ 1 : $\boxed{2}\boxed{3}\boxed{\#}$
$dec(2) = \{ \{ 3 , \# \} \}$ 2 : $\boxed{3}\boxed{\#}$
$dec(3) = \{ \{ \# \} \}$ 3 : $\boxed{\#}$

For permutation $(0, 2, 1, 3)$ the first block B_0 is $\boxed{1}\boxed{2}\boxed{3}\boxed{\#}$, the second block B_2 is $\boxed{\begin{matrix}3 & \# \\ 3 & \#\end{matrix}}$, the third block B_1 is $\boxed{2}$ and $B_3 = \boxed{\#}$ is the last one.

Proposition 3. *The computation of all minimal partition system $Z(\mathcal{A}_n)$ in $CFSZ(n)$ can be done in time $O(n!)$.*

Proof. This can be done by calling the nondeterministic Algorithm 1 or 2 for each possible execution.

Remark 1. By the use of the dynamic programming, we can improve the exponential brute force method to a polynomial algorithm as shown in Algorithm **??**.

In the following sections we will introduce our second algorithm which is based on trees. It computes efficiently the reduced $Z(\mathcal{A}_n)$ systems.

5 Tree based reduction

A *binary tree* is a structure defined on a finite set of nodes that either contains no nodes, or is made of three disjoint sets of nodes:

- a root node
- a binary tree called its left subtree
- a binary tree called its right subtree.

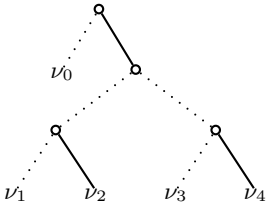
The binary tree that contains no nodes is called the empty tree. If the left subtree is non-empty, its root is called the left child of the root of the entire tree. Likewise, the root of a non-empty right subtree is the right child of the root of the entire tree. Therefore, in a *full binary tree* each node is either a leaf or has degree exactly 2, there is no degree-1 nodes. In the following we call a n-tree a full binary tree with n leaves. There is a unique n-tree for $n = 0$ to 2.

Let t be a n-tree and let π be a path in t. The *left weight* (resp. *right weight*) a_π (resp. b_π) is defined as the number of left (resp. right) edges in the path π. The *length* of π denoted by $l_\pi = a_\pi + b_\pi$ is the length of the path π. Denote by $w_\pi = a_\pi b_\pi$ the *weight* of π. The *cost* c_π of π is the sum of its weight and its length. So we have $c_\pi = w_\pi + l_\pi$.

Let ν be a node in t. Denote by π_ν the path from the node ν to the root of t. Denote by ν_l (resp. ν_r) the left child of ν (resp. the right child of ν). Denote by f_ν the father[4] of ν. If π is a path from the node ν to the root of t then we denote by f_π the path from the node f_ν to the root of t. We also associate a_{π_ν}, b_{π_ν}, w_{π_ν}, l_{π_ν} and c_{π_ν} to the node ν and we denote them respectively by a_ν, b_ν, w_ν, l_ν and c_ν. The set of leaves of a tree t will be denoted by L_t. The weight $w(t)$ of the tree t is defined as the sum of the weight of its leaves, that is $w(t) = \sum_{\nu \in L_t} w_\nu$.

Proposition 4. *Each $Z(\mathcal{A}_n)$ partition system corresponds to a unique n-tree.*

Proof. The idea is that if we follow the execution trace of the recursive Algorithm 2 we can see that it corresponds to a binary tree whose weight is the number of transitions of the reduced automaton. And by induction we can prove that for each state q we have $a_{\nu_q} = a(q)$ and $b_{\nu_q} = b(q)$. See Figure 4.



ν	a_ν	b_ν	l_ν	w_ν	c_ν
ν_0	1	0	1	0	1
ν_1	2	1	3	2	5
ν_2	1	2	3	2	5
ν_3	1	2	3	3	6
ν_4	0	3	3	0	3

$follow(0)$:	1 2 3 4
$follow(1)$:	2 3 4
$follow(2)$:	3 4
$follow(3)$:	4

$dec(0) : \{1, 2, 3, 4\}$
$dec(1) : \{2\}\ \{3, 4\}$
$dec(2) : \{3, 4\}$
$dec(3) : \{4\}$

Fig. 4. A full binary 5-tree and an associated $Z(\mathcal{A}_5)$ partition (left edges are represented by dotted lines and right edges with solid lines).

So, finding a minimal $Z(\mathcal{A}_n)$ partition system is reduced to finding a n-tree having minimal weight. Let us denote it by Z-tree of rank n.

Let $Split(t)$ be the function that returns the tree obtained from t by replacing a leaf having minimal cost in t by the unique 2-tree. See Figure 5.

Proposition 5. *The set of Z-trees can be computed inductively as follows:*

[4] The first ancestor.

- 1-*tree is the* Z*-tree of rank one*
- *if* t *is a* Z*-tree (of rank* i*) then* $Split(t)$ *is a* Z*-tree (of rank* $i+1$*).*

Proof. Let t_n be a Z-tree of rank n. To get a tree t_{n+1} of rank $n+1$ from t_n we have to split a leaf μ. The weight of t_{n+1} is:

$$\begin{aligned} w(t_{n+1}) &= \sum_{\nu \in L_{t_{n+1}}} w_\nu \\ &= (\sum_{\nu \in L_{t_n}} w_\nu) - w_\mu + w_{left-child(\mu)} + w_{right-child(\mu)} \\ &= w(t_n) - a_\mu b_\mu + (a_\mu + 1)b_\mu + a_\mu(b_\mu + 1) \\ &= w(t_n) + c_\mu \end{aligned}$$

If μ is the leaf of t_n which have the minimal coast, then, the tree t_{n+1} will have minimal weight.

So, this inductive construction allows us to have the minimal weight tree. The difference of weights between two consecutive minimal trees is exactly the cost of the split leaf. All Z-trees of rank less than n, can be generated by the following Algorithm 3.

Algorithm 3 MinZtree (n)

Require: $n \in \mathbb{N}$
Ensure: Z-tree of rank less than n
1: $t \leftarrow$ 1-tree
2: **for** i = 1 to n **do**
3: $\quad t \leftarrow Split(t)$
4: **end for**

Theorem 2. *Algorithm 3 computes one* Z*-tree of rank* i *for all* $i = 1$ *to* n *in* $O(n \log n)$ *time.*

Proof. At each step of this algorithm we look for a minimal cost leaf and then we split it. We can maintain the costs of the leaves in a dynamic structure which allow us a logarithmic time search for the minimal cost leaf and also a logarithmic time insertion of the two leaves obtained from the split function.

It is clear that for a given n, there may exist several Z-trees of rank n.

In the following we introduce a subclass of full binary trees (called P-trees), for which the Z-trees are unique. We do that in order to study the size-complexity of the reduced automata (the number of transitions).

5.1 P-Trees

We denote by M_t the set of leaves having minimal cost in t, that is: $M_t = \arg\min_{\nu \in L_t} c_\nu$. The function $SplitAll(t)$ returns the tree obtained from t by replacing every leaf in M_t by the unique 2-tree. See Figure 5.

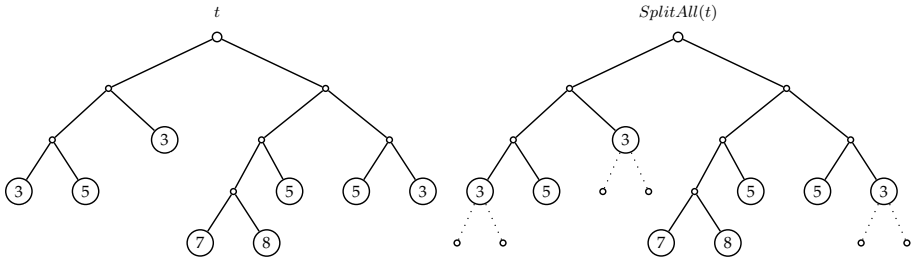


Fig. 5. A 8-tree t with a $Split$ tree and its $SplitAll$ tree. Values in the nodes are costs.

Definition 3. *The class $(t_n)_{n>0}$ of P-trees is defined inductively as follows:*

- $t_1 = 1$*-tree and*
- $t_{(n+1)} = SplitAll(t_n)$.

See Figure 6.

Remark 2. Notice that if $\nu \in M_{t_n}$ then $c_\nu = (n-1)$. This can be established by induction on n. Therefore, to get $t_{(n+1)}$ from t_n we split leaves of cost $(n-1)$.

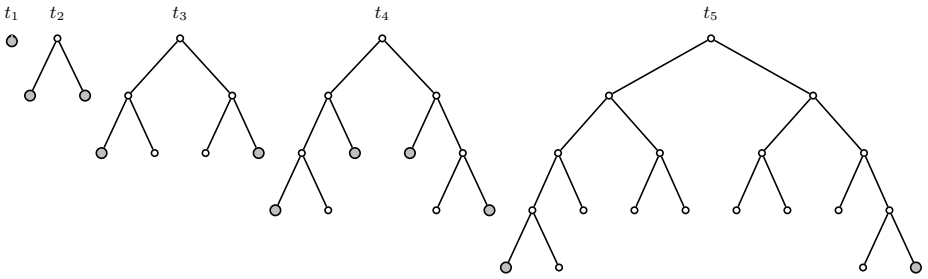


Fig. 6. The first four P-trees. Leaves with big circle have minimal cost.

5.2 Eratosthenes-Pascal's Triangle

The Eratosthenes-Pascal's Triangle is constructed from Pascal's Triangle as follows: we interleave each column k of Pascal's Triangle with $(k-1)$ zeros. The element of the Eratosthenes-Pascal's Triangle at the n^{th} row and the k^{th} column is denoted by T_n^k with $n \geq 1$ and $k \geq 1$.

	1	2	3	4	5	6	7	8	9	10	⋯
1	1										
2	1	1									
3	1	0	1								
4	1	2	0	1							
5	1	0	0	0	1						
6	1	3	3	0	0	1					
7	1	0	0	0	0	0	1				
8	1	4	0	4	0	0	0	1			
9	1	0	6	0	0	0	0	0	1		
10	1	5	0	0	5	0	0	0	0	1	
11	1	0	0	0	0	0	0	0	0	0	1

Let n be a natural number. We denote by D_n the set of divisors of n.

Proposition 6.

$$T_n^k = \begin{cases} \binom{\frac{n}{k} + k - 2}{k-1} & \text{if } k \in D_n \\ 0 & \text{otherwise.} \end{cases}$$

Proof. The element of the Pascal's Triangle at the n^{th} row and the j^{th} column is $\binom{i-1}{j-1}$. This element is moved in the Eratosthenes-Pascal's Triangle to the row $r = (i - j + 1)j$ in the same column j. We have then

$$T^j_{(i-j+1)j} = \binom{i-1}{j-1} \text{ Thus } T_r^j = \binom{\frac{r}{j} + j - 2}{j-1}$$

Let S_n be the sum of the n^{th} row in the Eratosthenes-Pascal's Triangle.

$$S_n = \sum_{k=1}^{n} T_n^k = \sum_{k \in D_n} \binom{\frac{n}{k} + k - 2}{k-1}$$

5.3 P-trees and Eratosthenes-Pascal's Triangle

In this section we will describe the link between the Eratosthenes-Pascal's Triangle and the set of P-trees.

Theorem 3. *The sum of the elements of the n^{th} row of Eratosthenes-Pascal's Triangle's, S_n, is exactly $|M_{t_n}|$ the number of leaves of minimal cost in the P-tree t_n.*

To prove this theorem we introduce a family F_n which is in bijection with both the Eratosthenes-Pascal's Triangle's rows and with P-trees. We focus on the following question: *Given a natural number n what are all the possible paths that have $(n-1)$ as cost?* To answer this question, we define $F_{(n-1)}$ as the set of all paths of cost $(n-1)$:

$$F_{(n-1)} = \{\pi \mid c_\pi = (n-1)\}$$

Lemma 2. *For all $n \geq 1$, $|F_{(n-1)}| = S_n$.*

Proof. Let $F^i_{(n-1)}$ for $0 \leq i \leq (n-1)$ be the set of paths of $F_{(n-1)}$ having i left edges. We have $F_{(n-1)} = \bigcup_{i=0}^{(n-1)} F^i_{(n-1)}$ where

$$\begin{aligned} F^i_{(n-1)} &= \{\pi \in F_{(n-1)} \mid a_\pi = i\} \\ &= \{\pi \mid (a_\pi b_\pi + a_\pi + b_\pi = (n-1)) \wedge (a_\pi = i)\} \\ &= \{\pi \mid (b_\pi = \frac{n}{i+1} - 1) \wedge (a_\pi = i)\} \end{aligned}$$

$$\text{So, we get } |F^i_{(n-1)}| = \begin{cases} \binom{\frac{n}{i+1} + i - 1}{i} & \text{if } (i+1) \in D_n \\ 0 & \text{otherwise.} \end{cases}$$

This corresponds to the different ways to arrange i left edges in a path of length $\frac{n}{i+1} + i - 1$. Let $k = i + 1$ then

$$|F^{k-1}_{(n-1)}| = \begin{cases} \binom{\frac{n}{k} + k - 2}{k-1} & \text{if } k \in D_n \\ 0 & \text{otherwise.} \end{cases}$$

Thus $|F^{k-1}_{(n-1)}| = T^k_n$. That is for $0 \leq i < n$ we get $|F^i_{(n-1)}| = T^{i+1}_n$. Finally

$$|F_{(n-1)}| = \sum_{i=0}^{(n-1)} |F^i_{(n-1)}| = \sum_{i=0}^{(n-1)} T^{i+1}_n = \sum_{i=1}^{n} T^i_n = S_n$$

Therefore, we can associate the n^{th} row of the Eratosthenes-Pascal's Triangle to $F_{(n-1)}$.

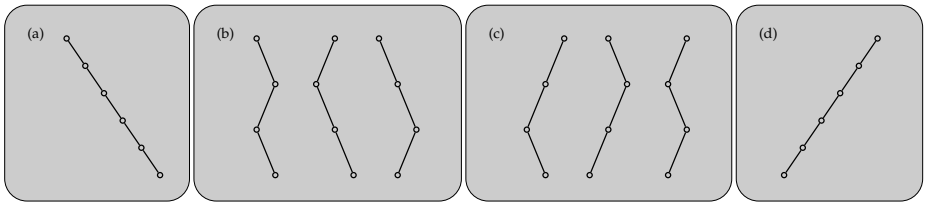


Fig. 7. All paths of cost 5. (a) The unique path of cost 5 with no left edges which corresponds to $\binom{5}{0} = 1$. (b) The three paths of cost 5 with one left edges which corresponds to $\binom{3}{1} = 3$. (c) The three paths of cost 5 with two left edges which corresponds to $\binom{3}{2} = 3$. (d) The unique path of cost 5 with five left edges which corresponds to $\binom{5}{5} = 1$. There is no paths of cost 5 with three or four left edges that correspond to zeros in the 6^{th} row of the Eratosthenes-Pascal's Triangle.

Lemma 3. *For all $n \geq 1$, $|F_{(n-1)}| = |M_{t_n}|$.*

Proof. We will show that each set F_n is associated with the P-tree t_n. We proceed by induction. Assume that for all $k \leq n$, the P-tree t_n contains all paths of cost less or equal to n, that is, for all $k \leq n$, F_k is in the P-tree t_n. From this hypothesis we should show that $t_{(n+1)} = SplitAll(t_n)$ contains all the paths of cost less or equal to $(n+1)$. Of course, $t_{(n+1)}$ contains all the paths of cost less or equal to n because the tree $t_{(n+1)}$ is obtained from t_n. However, *does $t_{(n+1)}$ contain all paths of cost equal to $(n+1)$?* To answer this question, we proceed by absurd. Let π be a path with cost $c_\pi = (n+1)$ which is not in $t_{(n+1)}$. It is clear that the cost of the father of π is:

$$c_{f_\pi} = \begin{cases} (a_\pi - 1)b_\pi + (a_\pi - 1) + b_\pi & \text{if } \pi \text{ is a left child,} \\ a_\pi(b_\pi - 1) + a_\pi + (b_\pi - 1) & \text{if } \pi \text{ is a right child.} \end{cases}$$

As $c_{f_\pi} < c_\pi$ then $c_{f_\pi} \leq n$. From this we deduce that when $c_{f_\pi} < n$, the father f_π of π was split by the $SplitAll()$ function within an earlier or it will be split in the current tree t_n in the case where $c_{f_\pi} = n$ (see Remark 2). In both cases, the path π is necessarily in $t_{(n+1)}$.

From the two last lemmas, we construct a bijection between the P-trees and the Eratosthenes-Pascal's Triangle's rows, and we claim following corollary:

Corollary 1. *Let t_n be a P-tree. Then $|M_{t_n}| = |F_{(n-1)}| = S_n$.*

From Corollary 1, we have $S_n = \sum_{k|n} \binom{\frac{n}{k} + k - 2}{k - 1}$.

Proposition 7. *Let t_n be a P-tree. Its size $s(t_n) = |L_{t_n}|$ (the number of leaves), and its weight $w(t_n)$ are: $s(t_n) = 1 + \sum_{i=1}^{n-1} S_i$ and $w(t_n) = \sum_{i=2}^{n} (i-2)S_{i-1}$.*

Proof. From Corollary 1 and Remark 2, we have, the size of the tree t_n is the sum of the size of the tree t_{n-1} and the number of all split minimal leaves, that is, $s(t_n) = s(t_{n-1}) + |M_{t_{n-1}}|$. We have also, the weight of the tree t_n is the sum of the weight of the tree t_{n-1} and the costs of all split minimal leaves, that is, $w(t_n) = w(t_{n-1}) + (n-2)|M_{t_{n-1}}|$.

From Proposition 4, the P-tree t_n (which is a Z-tree) corresponds to a $Z(\mathcal{A}_{s(t_n)})$ partition system. The CFS automaton associated with $Z(\mathcal{A}_{s(t_n)})$ has $w(t_n)$ transitions.

With our construction the CFS automaton associated with a $Z(\mathcal{A}_n)$ CFS partition system may contain several initial states. However, in order to compare the number of minimal automata and their number of transitions, with those obtained by R. Cox [3] (seen Table 1), we must restrict our study to CFS partition systems leading to a unique initial state automata. It is easy to verify that in this case a P-tree t_n has $s(t_n) = 1 + \sum_{i=1}^{n-1} \frac{S_{i+2}}{2}$ and $w(t_n) = \sum_{i=1}^{n} \frac{iS_{i+1}}{2}$.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14
(i)	1	3	6	9	13	18	23	≤ 28	≤ 34	≤ 41	?	?	?	?
(ii)	1	1	2	1	1	4	6	≥ 1	≥ 1	≥ 1	?	?	?	?
(iii)	1	3	6	9	13	18	23	28	33	39	46	53	60	67
(iv)	1	1	2	1	1	4	6	4	1	1	5	10	10	5

Table 1. Comparison table. (i) Minimal transition number estimated by R. Cox [3] (ii) Number of minimal automata estimated by R. Cox [3] (iii) Number of transitions in our reduced automaton (iv) Number of reduced automata estimated by our approach.

6 Asymptote behavior of the number of transitions

Appendix A contains the basics of asymptotic notations. In this section we shall establish one of our main result which concern the asymptotically result on the behavior of the number of transitions $w(t_n)$, where $s(t_n)$ is the number of states. Namely, we will show that the weight of our automata is asymptotically equivalent to $s(t_n)log^2 s(t_n)$ up to constant which means that the number of transitions is minimal in the sense that we reach the lower bounded of Shnitger [13]. Indeed, we have

Theorem 4. $\log^2(4)\ w(t_n) \sim s(t_n)\log^2 s(t_n)$.

As a consequence we obtain the following

Corollary 2. *For a large n, we have*

$$\omega(t_n) < s(t_n)\log^2 s(t_n).$$

It is also easy to deduce the following

Corollary 3. $\dfrac{s(t_n)\log^2 s(t_n)}{\log\log\ s(t_n)} = o\big(w(t_n)\big)$.

Before starting the proof of Theorem 4. Observe that according to the Corollary 1 combined with Proposition 7, one may consider that $\omega(t_n)$ and $s(t_n)$ are given by

$$s(t_n) = 1 + \sum_{1}^{n-1} S_i \ \text{ and } \ \omega(t_n) = \sum_{i=1}^{n-2} iS_{i+1}.$$

As usual in the number theory, any arithmetical function $f\ :\ \mathbb{Z} \longrightarrow \mathbb{R}$ can be extended to the real line by putting, for any $x \in \mathbb{R}$, $f(x) = f(\lfloor x \rfloor)$. Therefore, for any $x \geq 2$, we have

$$s(t_x) = 1 + \sum_{1}^{\lfloor x \rfloor -1} S_i \ \text{ and } \ \omega(t_x) = \sum_{i=1}^{\lfloor x \rfloor -2} iS_{i+1}.$$

We recall that the classical arithmetical function $\pi(x)$ denote the number of primes not exceeding x. We shall need also the following classical identity due to Abel

Theorem 5 (Abel's identity, [1]). *For any arithmetical function $a(n)$ let*

$$A(x) = \sum_{n \leq x} a(n)$$

where $A(x) = 0$ if $x < 1$. Assume f has a continuous derivative on the interval $[y, x]$, where $0 < y < x$. Then we have

$$\sum_{y < n \leq x} a(n)f(n) = A(x)f(x) - A(y)f(y) - \int_y^x A(u)f'(u)du.. \tag{1}$$

We deduce easily from the Abel's identity the following lemma

Lemma 4. *For any integer $n \geq 1$, we have*

$$\omega(t_n) = (n-2)\widetilde{s}(t_n) - 3 - \int_2^{n-2} \widetilde{s}(t_u)du,$$

where $\widetilde{s}(t_u) = s(t_u) - 1$.

We need to estimate $\omega(t_n)$ with respect to $s(t_n)$. For that, we shall need the following weaker form of the Prime Number Theorem (WPNT for short) due to Cheyshev

Theorem 6 ([1]). *For every integer $n \geq 2$ we have*

$$\frac{1}{6}\frac{n}{\log(n)} < \pi(n) < 6\frac{n}{\log(n)}. \tag{2}$$

We deduce from the WPNT the following crucial proposition

Proposition 8. *For all $u \geq 4$ we have*

$$\widetilde{s}(t_u) \geq \frac{1}{3}\frac{u-2}{\log(u-2)}. \tag{3}$$

Proof. We can show that for any prime number $p \geq 2$, we have $S_p = 2$. Hence

$$\widetilde{s}(t_u) = \sum_{i \leq u} S_i \geq 2\pi(u-1).$$

Consequently, by WPNT we get

$$\widetilde{s}(t_u) \geq \frac{1}{3}\frac{[u]-1}{\log([u]-1)}.$$

But the function $x \in [3, +\infty[\mapsto \frac{x}{\log(x)}$ is increasing function. It follows that we have, For all $u \geq 3$,

$$\widetilde{s}(t_u) \geq \frac{1}{3}\frac{u-2}{\log(u-2)},$$

Which achieve the proof of the proposition.

For any $x \geq 2$ and any positive integer n, let

$$\mathrm{Li}_n(x) = \int_2^x \frac{dt}{\log^n(t)}.$$

Let us summarize in the following proposition a classical well-known results on $Li_n(x)$ that we shall used.

Proposition 9. *For every $x \geq 2$ and integer $n \geq 1$, we have*

$$\mathrm{Li}_1(x) = \frac{x}{\log(x)} + \mathrm{Li}_2(x) - \frac{2}{\log(2)}, \tag{4}$$

$$\mathrm{Li}_n(x) = O\big(\frac{x}{\log^n(x)}\big). \tag{5}$$

Now, we are able to formulate our key estimation of $\omega(t_n)$ with respect to $s(t_n)$ in the following proposition.

Proposition 10. $\limsup\big(\frac{\omega(t_n)}{(n-2)\widetilde{s}(t_n)}\big) \leq 1.$

Proof. Applying Lemma 4 we have, for any $n \geq 3$,

$$\frac{\omega(t_n)}{(n-2)\widetilde{s}(t_n)} = 1 - \frac{3}{(n-2)\widetilde{s}(t_n)} - \frac{1}{(n-2)\widetilde{s}(t_n)}\int_2^{n-2} \widetilde{s}(t_u)du.$$

Therefore, for any $n \geq 5$, we have

$$\frac{\omega(t_n)}{(n-2)\widetilde{s}(t_n)} \leq 1 - \frac{3}{(n-2)\widetilde{s}(t_n)}$$

From Proposition 8 we deduce, that for any $n \geq 5$ we have

$$\frac{3}{(n-2)\widetilde{s}(t_n)} \leq \frac{1}{\log(n-2)}$$

Hence, by letting n goes to ∞ we obtain

$$\frac{3}{(n-2)\widetilde{s}(t_n)} \xrightarrow[n\to\infty]{} 0.$$

Which implies that

$$\limsup\big(\frac{\omega(t_n)}{(n-2)\widetilde{s}(t_n)}\big) \leq 1,$$

and this finish the proof of the proposition.

We are reduced to compare the sequences $(n-2)s(t_n)$ and $s(t_n)\log^2 s(t_n)$. For that we shall estimate $s(t_x)$. Precisely, we argue that we have the following

Theorem 7. *For a large $x>0$ we have*

$$x^{\frac{3}{4}}\frac{4^{\sqrt{x}-1}}{\sqrt{\pi}}\leq s(t_x)\leq x^{\frac{3}{4}}\log(x)\frac{4^{\sqrt{x}-1}}{\sqrt{\pi}}.$$

The proof of Theorem 7 will be given later. For instance, using Theorem 7 holds we shall extended Proposition 10 as follows.

Proposition 11. *The sequences $\omega(t_n)$ and $(n-2)s(t_n)$ two sequences be equivalent. That is,*

$$\frac{\omega(t_n)}{(n-2)s(t_n)}\xrightarrow[n\to\infty]{}1.$$

Proof. By Lemma 4, write

$$\frac{\omega(t_n)}{(n-2)s(t_n)}=1-\frac{3}{(n-2)s(t_n)}-\frac{1}{(n-2)s(t_n)}\int_2^{n-2}s(t_x)dx.$$

Let $\varepsilon>0$ and x sufficiently large. Then, by Theorem 7, for a large x, we have

$$x^{\frac{3}{4}}\frac{4^{\sqrt{x}-1}}{\sqrt{\pi}}\leq s(t_x)\leq x^{\frac{3}{4}}\log(x)\frac{4^{\sqrt{x}-1}}{\sqrt{\pi}}. \tag{6}$$

But

$$\begin{aligned}\int_2^{n-2}4^{\sqrt{x}}dx &\overset{u=\sqrt{x}}{=} \int_{\sqrt{2}}^{\sqrt{n-2}}2\,u\,4^u du\\ &= \Big[\frac{2u}{\log(4)}4^u\Big]_{\sqrt{2}}^{\sqrt{n-2}}-\Big[\frac{2}{\log^2(4)}4^u\Big]_{\sqrt{2}}^{\sqrt{n-2}}\\ &= \frac{2\sqrt{n-2}\,.4^{\sqrt{n-2}}}{\log(4)}-\frac{2\sqrt{2}\,.4^{\sqrt{2}}}{\log(4)}-\frac{2\,.4^{\sqrt{n-2}}}{\log^2(4)}+\frac{2\,.4^{\sqrt{2}}}{\log^2(4)}. \end{aligned}\tag{7}$$

Since $\frac{1}{(n-2)s(t_n)}$ vanishes at the infinity and $\widetilde{s}(t_n)$ is equivalent to $s(t_n)$ we may assume that (6) holds starting from 2 and Theorem 7 is valid for $\widetilde{s}(t_n)$. Therefore

$$\begin{aligned}\frac{1}{(n-2)\widetilde{s}(t_n)}\int_2^{n-2}\widetilde{s}(t_x)dx &\leq \frac{1}{(n-2)(n-2)^{\frac{3}{4}}\,4^{\sqrt{n}}}\int_2^{n-2}x^{\frac{3}{4}}\log(x)\,4^{\sqrt{x}}dx\\ &\leq \frac{(n-2)^{\frac{3}{4}}\log(n-2)}{(n-2)(n-2)^{\frac{3}{4}}\,4^{\sqrt{n}}}\int_2^{n-2}4^{\sqrt{x}}dx\\ &\leq \frac{\log(n-2)}{(n-2)4^{\sqrt{n}}}\int_2^{n-2}4^{\sqrt{x}}dx. \end{aligned}\tag{8}$$

From (7) combined with (8) it follows that

$$\frac{1}{(n-2)\widetilde{s}(t_n)}\int_2^{n-2}\widetilde{s}(t_x)dx \leq$$
$$\frac{2\sqrt{n-2}\,.4^{\sqrt{n-2}}}{\log(4)}\times\frac{\log(n-2)}{(n-2)4^{\sqrt{n}}}+\frac{2\,.4^{\sqrt{2}}}{\log^2(4)}\times\frac{\log(n-2)}{(n-2)4^{\sqrt{n}}}\xrightarrow[n\to\infty]{}0.$$

We conclude that

$$\frac{\omega(t_n)}{(n-2)\widetilde{s}(t_n)}\xrightarrow[n\to\infty]{}1.$$

which proves the proposition.

It remains to prove Theorem 7. For that we shall need the following classical lemma. The proof of it can be found in [4]. Nevertheless we include the proof for the sake of completeness.

Lemma 5. $\binom{2n}{n}\sim\frac{4^n}{\sqrt{\pi}\sqrt{n}}$.

Proof. By Stirling formula we have

$$n!\sim n^ne^{-n}\sqrt{2\pi n}$$

Hence

$$\binom{2n}{n}=\frac{2n!}{(n!)^2}\sim\frac{4^n}{\sqrt{\pi}\sqrt{n}}.$$

This finishes the proof of the lemma.

Proof (Proof of Theorem 7). For $x\geq 2$, Write

$$\begin{aligned}s(t_x)&=\sum_{n\leq x}\sum_{d|n}\binom{\frac{n}{d}+d-2}{d-1}\\&=\sum_{dq\leq x}\binom{q+d-2}{d-1}\\&=\sum_{d=1}^{\lfloor x\rfloor}\sum_{q=1}^{\lfloor\frac{x}{d}\rfloor}\binom{q+d-2}{d-1}\end{aligned}$$

From this we see that

$$\begin{aligned}s(t_x)&\leq\sum_{d=1}^{\lfloor x\rfloor}\lfloor\frac{x}{d}\rfloor\binom{2(\lfloor x\rfloor-1)}{\lfloor x\rfloor-1}\\&\leq x\log(x)\binom{2(\lfloor x\rfloor-1)}{\lfloor x\rfloor-1},\end{aligned}\tag{9}$$

and

$$s(t_x) \geq \lfloor x \rfloor \binom{2(\lfloor x \rfloor - 1)}{\lfloor x \rfloor - 1}, \tag{10}$$

Using the relation $\lfloor x \rfloor = x + O(1)$ combined with (9) and (10), we obtain

$$x\binom{2(\lfloor x \rfloor - 1)}{\lfloor x \rfloor - 1} \leq s(t_x) \leq x \log(x) \binom{2(\lfloor x \rfloor - 1)}{\lfloor x \rfloor - 1}.$$

By Lemma 5, this gives

$$x^{\frac{3}{4}} \frac{4^{\sqrt{x}-1}}{\sqrt{\pi}} \leq s(t_x) \leq x^{\frac{3}{4}} \log(x) \frac{4^{\sqrt{x}-1}}{\sqrt{\pi}},$$

which proves the theorem.

Now we are able to give the proof of Theorem 4.

Proof (of Theorem 4). By Proposition 11, it is sufficient to show that

$$s(t_n) \log^2(s(t_n)) \sim (n-2) s(t_n).$$

For that, observe that we have

$$\frac{s(t_n) \log^2(s(t_n))}{(n-2) s(t_n)} = \frac{\log^2(s(t_n))}{n-2}$$

Applying Theorem 3, we deduce that

$$\log(s(t_n)) \sim \log(4)\ \sqrt{n}.$$

Whence

$$\log^2(s(t_n)) \sim \log^2(4)\ n.$$

Hence

$$\frac{\log^2(s(t_n))}{(n-2)} \xrightarrow[n \to \infty]{} \log^2(4).$$

We deduce that

$$\frac{\omega(t_n)}{s(t_n) \log^2 s(t_n)} \xrightarrow[n \to \infty]{} \frac{1}{\log^2(4)} < 1.$$

This finishes the proof of the theorem.

7 Conclusion

In this paper we show how binary trees can be used to design a fast algorithm for computing an automaton with a reduced[5] number of transitions recognizing the language $L(E_n)$. We have verify that our algorithm gives the minimal number of transitions for $n = 1$ to 7 (see Table 1) and we have shown that our reduction is asymptotically a minimization. Hence, we conjecture that Algorithm 3 computes the minimal transition automaton.

[5] Asymptotically minimal.

Acknowledgments. Special thanks to Saïd Abdeddaïm, Alexis Bès, Patrick Cégielski, Jean-Marc Champarnaud and Yuri Matiyasevich.

References

1. Tom M. Apostol. *Introduction to analytic number theory*. Springer-Verlag, New York-Heidelberg, 1976.
2. Jean-Marc Champarnaud, Faissal Ouardi, and Djelloul Ziadi. An efficient computation of the equation k-automaton of a regular k-expression. *IOS Press, Amsterdam, The Netherlands, Fundam. Inf*, 90(1-2):1–16, 2009.
3. Russ Cox. Minimal number of edges in e-free non-deterministic finite automata (nfa) for regular expression $(1+\varepsilon)\cdot(2+\varepsilon)\cdot(3+\varepsilon)\cdots(n+\varepsilon)$. *The On-Line Encyclopedia of Integer Sequences, Sequence A129403*.
4. Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.
5. Viliam Geffert. Translation of binary regular expressions into nondeterministic ϵ-free automata with transitions. *J. Comput. Syst. Sci.*, 66(3):451–472, 2003.
6. Christian Hagenah and Anca Muscholl. Computing epsilon-free nfa from regular expressions in $O(n\ log^2(n))$ time. pages 277–285, 1998.
7. J. E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. *In* Theory of machines and computation*, Pages 189-196. Academic Press, New York, 1971*.
8. Juraj Hromkovic, Sebastian Seibert, and Thomas Wilke. Translating regular expressions into small epsilon-free nondeterministic finite automata. In *STACS '97: Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science, Springer-Verlag, London, UK*, pages 55–66, 1997.
9. Ahmed Khorsi, Faissal Ouardi, and Djelloul Ziadi. Fast equation automaton computation. *J. of Discrete Algorithms, Elsevier Science Publishers B. V, Amsterdam, The Netherlands*, 6(3):433–448, 2008.
10. Sylvain Lombardy and Jacques Sakarovitch. The universal automaton. In *Logic and Automata*, pages 457–504, 2008.
11. E.-F. Moore. Gedanken-experiments on sequentiel machines. *In* Automata studies*, Volume 34 of* Annals of mathematics studies*, pages 129-153. Princeton University Press, Princeton, N. J., 1956*.
12. Harold G. Diamond Paul T. Bateman;. A hundred years of prime numbers. Amer. Math. Monthly*, 103 (1996), no. 9, 729741.*
13. Georg Schnitger. Regular expressions and nfas without ϵ-transitions. In *STACS*, pages 432–443, 2006.
14. E. C. Titchmarsh. *The theory of the Riemann zeta-function*. The Clarendon Press, Oxford University Press, New York, 1986.
15. Lifshits Yuri. A lower bound on the size of e-free nfa corresponding to a regular expression. *Inf. Process. Lett, Elsevier North-Holland, Inc, Amsterdam, The Netherlands*, 85(6):293–299, 2003.

A Asymptotic notations

Following [4], we employ the standard asymptotic notation called Bachmann-Landau notation as follows. Let $\mathbb{S}$ be a set and $s_0 \in \mathbb{S}$ a particular element of $\mathbb{S}$. We assume a notion of neighbourhood to exist on $\mathbb{S}$. Examples are $\mathbb{S} =$

$\mathbb{Z}_{>0} \bigcup \{+\infty\}$ with $s_0 = +\infty$, $\mathbb{S} = \mathbb{R}$ with s_0 any point in $\mathbb{R}$; $\mathbb{S} = \mathbb{C}$ or a subset of $\mathbb{C}$ with $s_0 = 0$, and so on. Two functions f and g from $\mathbb{S} \setminus \{s_0\}$ to $\mathbb{R}$ or $\mathbb{C}$ are given.

- O-notation: write
$$f(s) \overset{s \to s_0}{=} O(g(s)),$$
if the ratio $\dfrac{f(s)}{g(s)}$ stays bounded as $s \to s_0$ in $\mathbb{S}$. In other words, there exists a neighborhood V of s_0 and a constant $C > 0$ such that
$$|f(s)| < C|g(s)|, \;\; s \in V, s \neq s_0.$$
One also says that "f is of order at most g", or "f is big-Oh of g"(as s tends to s_0).
- o-notation: write
$$f(s) \overset{s \to s_0}{=} o(g(s)),$$
if the ratio $\dfrac{f(s)}{g(s)}$ tends to 0 as $s \to s_0$ in $\mathbb{S}$. In other words, for any (arbitrarily small) $\varepsilon > 0$, there exists a neighborhood $\mathcal{V}_\varepsilon$ of s_0 (depending on ε), such that
$$|f(s)| < \varepsilon |g(s)|, \qquad s \in \mathcal{V}_\varepsilon, s \neq s_0.$$
One also says that "f is of order smaller than g, or f is little-oh of g" (as s tends to s_0).
- $\sim$-notation: write
$$f(s) \overset{s \to s_0}{\sim} g(s),$$
if the ratio $\dfrac{f(s)}{g(s)}$ tends to 1 as $s \to s_0$ in $\mathbb{S}$. One also says that "f and g are asymptotically equivalent" (as s tends to s_0).
- Ω-notation: write
$$f(s) \overset{s \to s_0}{=} \Omega(g(s)),$$
if the ratio $\dfrac{f(s)}{g(s)}$ stays bounded from below in modulus by a non-zero quantity, as $s \to s_0$ in $\mathbb{S}$. Which means that there exists $k > 0$ and a neighborhood $\mathcal{V}$ of s_0, such that
$$f(s) \geq k.g(s), \;\; s \in \mathcal{V}.$$
One then says that f is of order at least g.
- θ-notation: if $f(s) = O(g(s))$ and $f(s) = \Omega(g(s))$, write
$$f(s) \overset{s \to s_0}{=} \theta(g(s)).$$
This implies that there exits $k, C > 0$ and a neighborhood $\mathcal{V}$ of s_0, such that
$$k.g(s) \leq f(s) \leq C.g(s), \;\; s \in \mathcal{V}.$$
One then says that f is of order exactly g.

At this point we are able to make a parallel between the history of our contribution and the history of the famous Prime Number Theorem (PNT) which we shall use later in its weaker form. The PNT Theorem concerns the asymptotic behavior of the prime-counting function $\pi(x) = |\{p \leq x, p \text{ prime}\}|$. Using asymptotic notation the PNT can be restated as

$$\pi(x) \sim \frac{x}{\ln x}.$$

The behavior of $\pi(x)$ has been the object of intense study by many celebrated mathematicians ever since the eighteenth centry. Inspection of tables of primes led Gauss (1792) and Legendre (1798) to conjecture the PNT. In 1808 Legendre published the formula $\pi(x) = x/(logx + A(x))$, where $A(x)$ tends to a constant $B = -1.08366$ as $x \longrightarrow +\infty$, which means that π is $\Omega(x/log(x))$.
According to Bateman and Diamond [12], The first person to establish the true order of $\pi(x)$ was P. L. Chebyshev. Indeed, in two papers from 1848 and 1850, Chebychev prove that $\pi(x)$ is $\theta(x/log(x)$. This result is known in nowadays as Chebychev Theorem.
Finally, in 1896 the PNT was first proved by Hadamard and de la Vallée Poussin. Their proofs were long and intricate. A simplified modern presentation is given on pages 41-47 of Titchmarsh's book on the Riemann Zeta function [14].