Exact Generation of Minimal Acyclic Deterministic Finite Automata

Marco Almeida

Nelma Moreira

Rogério Reis

LIACC, Faculdade de Ciências, Universidade do Porto
Departamento de Ciência de Computadores
Rua do Campo Alegre, 1021/1055
4169-007 Porto, Portugal

{mfa,nam,rvr}@ncc.up.pt

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

We give a canonical representation for minimal acyclic deterministic finite automata (MADFA) with n states over an alphabet of k symbols. Using this normal form, we present a method for the exact generation of MADFAs. This method avoids a rejection phase that would be needed if a generation algorithm for a larger class of objects that contains the MADFAs were used. We give upper and lower bounds for MADFAs enumeration and some exact formulas for small values of n.

Keywords: acyclic deterministic finite automata, minimal automata, finite languages, generation, enumeration

1. Introduction

The problem of the enumeration of minimal (non-isomorphic) n-state acyclic deterministic finite automata (MADFAs) is an open problem that recently has been considered by several authors. Domaratzki *et al.* [7] presented a characterization of MADFAs, gave a lower bound and some exact calculations. Domaratzki [5, 6] obtained improved lower and upper bounds for these values. The lower bound is based upon the enumeration of certain families of MADFAs, and the upper bound is obtained by enumerating n-state initially-connected acyclic deterministic finite automata (ADFAs), where states havea topological order Π associated (*i.e.*, whenever there is a transition from a state s to a state s', $\Pi(s) < \Pi(s')$). This approach has the drawback of considering labelled automata, and thus possible isomorphic ones. Câmpeanu and Ho [4] gave a tight upper bound for the number of states of a MADFA accepting words of length less than or equal to a given integer. Liskovets [9] gave a linear recursive relation for the number of unlabelled (non-isomorphic) ADFAs, and has also enumerated initially-connected acyclic deterministic finite automata with a unique *pre-dead* state (*i.e.*, a state such that all transitions from it go to a unique absorbing state, called *dead*). As all MADFAs have this characteristic, a better upper bound is thus achieved. More recently, Callan [3] presented a canonical form for ADFAs and showed that a certain determinant of Stirling cycle numbers counts ADFAs. This canonical form is obtained observing that if we mark the visited states, starting with the initial state s_0, it is always possible to find a state whose only incident states are already marked. This induces a unique state's labelling, but it is not clear how these representations can be used in automata generation.

In this paper we give a canonical representation for MADFAs with n states over an alphabet of k symbols. Using this normal form, we present a method for the exact generation of MADFAs. This method has the advantage of avoiding a rejection phase that would be needed if a generation algorithm for a larger class of automata that contains the MADFAs were considered. Our first approach to the enumeration of MADFAs was to generate initially-connected deterministic finite automata (IDFAs) using the algorithm presented in Almeida *et al.* [11, 1, 2], and then test for non-cyclicity as well as for non-minimality. In those experiments this method was shown to be more than 20 times slower than the method described in this paper. It is also relevant to note that although Callan and Liskovets obtained, respectively, a canonical form and an exact formula for the enumeration for ADFAs, no exact generator algorithm is known for that class.

In the next section, we present basic concepts used in this paper. In Section 3 we review some characterizations of (minimal) acyclic deterministic finite automata. Based upon those characterizations, in Section 4, we present a canonical representation for MADFAs. In Section 5, we describe an algorithm for the exact generation of all MADFAs, given n and k. In Section 6, we address the problem of MADFAs enumeration (without its generation) and give exact formulae for small values of n. In Section 7, we conclude with some future work.

2. Basic Concepts and Notation

We review some basic concepts of automata theory and finite languages. For more details we refer the reader Hopcroft *et al.* [8], Yu [14] or Lothaire [10].

Let $[n, m]$ denote the set $\{i \in \mathbb{Z} \mid n \leq i \leq m\}$. In a similar way, we consider the variants $]n, m]$, $[n, m[$ and $]n, m[$. Whenever we have a finite ordered set A, and a function f on A, the expression $(f(a))_{a \in A}$ denote the values of f for increasing values of A.

Alphabets and Languages. An *alphabet* Σ is a finite set of symbols. A word over Σ is a finite sequence of symbols of Σ. The empty word is denoted by ε. The length of a word $x = \sigma_1\sigma_2 \cdots \sigma_n$, denoted by $|x|$, is n. The set $\Sigma^\star$ is the set of all words over Σ. A language L is a subset of $\Sigma^\star$. A language is finite if its cardinality is finite.

Deterministic Finite Automata. A *deterministic finite automaton* (DFA) $\mathcal{A}$ is a tuple $(S, \Sigma, \delta, s_0, F)$ where S is a finite set of states, Σ is the alphabet, $\delta : S \times \Sigma \to S$ is the transition function, s_0 the initial state and $F \subseteq S$ the set of final states. Let the *size of* $\mathcal{A}$ be $|S|$. We assume that the transition function is total, so we consider only *complete* DFAs. The transition function δ is inductively extended to $\Sigma^\star$, by $(\forall s \in S)\, \delta(s, \varepsilon) = s$ and $\delta(s, x\sigma) = \delta(\delta(s, x), \sigma)$.

A DFA is *initially-connected* (or *accessible*) (IDFA) if for each state $s \in S$ there exists a word $x \in \Sigma^\star$ such that $\delta(s_0, x) = s$. A DFA is *trim* if it is an IDFA and every state is *useful*, *i.e.*, $(\forall s \in S)(\exists x \in \Sigma^\star)\, \delta(s, x) \in F$.

Isomorphism. Two DFAs $(S, \Sigma, \delta, s_0, F)$ and $(S', \Sigma', \delta', s'_0, F')$ are called *isomorphic* if $|\Sigma| = |\Sigma'| = k$, there exist bijections $\Pi_1 : \Sigma \to [0, k-1]$, $\Pi_2 : \Sigma' \to [0, k-1]$ and a bijection $\iota : S \to S'$ such that $\iota(s_0) = s'_0$, $\iota(F) = F'$, and for all $\sigma \in \Sigma$ and $s \in S$, $\iota(\delta(s, \sigma)) = \delta'(\iota(s), \Pi_2^{-1}(\Pi_1(\sigma)))$.

Minimality. The *language* accepted by a DFA $\mathcal{A}$ is $L(\mathcal{A}) = \{x \in \Sigma^\star \mid \delta(s_0, x) \in F\}$. Two DFAs are *equivalent* if they accept the same language. Two isomorphic automata are equivalent (considering the bijections between the alphabets), but two non-isomorphic automata may also be equivalent. A DFA $\mathcal{A}$ is *minimal* if there is no DFA $\mathcal{A}'$, with fewer states, equivalent to $\mathcal{A}$. For obtaining a minimal DFA the notion of equivalent states is used. We say that two states s and s' are equivalent if and only if

$$(\forall x \in \Sigma^\star)\, (\delta(s, x) \in F \leftrightarrow \delta(s', x) \in F).$$

A minimal DFA has no equivalent states and is initially-connected. Minimal DFAs are unique up to isomorphism.

3. Acyclic Finite Automata and Minimality

An *acyclic deterministic finite automaton* (ADFA) is a DFA $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$ with $F \subseteq S$ and $s_0 \neq \Omega$ such that $(\forall \sigma \in \Sigma)\, \delta(\Omega, \sigma) = \Omega$ and $(\forall x \in \Sigma^\star)(\forall s \in S)\, \delta(s, x) \neq s$. The state Ω is called the *dead state*, and is the only *cyclic* state of $\mathcal{A}$. The *size of* $\mathcal{A}$ is $|S|$. We are going to consider only trim complete ADFAs, where all states but Ω are useful. It is obvious that the language of an ADFA is finite. Two states s and s' are mergeable if they are both either final or not final, and the transition function is identical. An ADFA can be minimized by merging mergeable states, thus, a minimal ADFA (MADFA) can be characterized by:

Lemma 1 ([10]) *An* ADFA $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$ *is minimal if and only if* $(\forall s, s' \in S \cup \{\Omega\})((s \in F \,\dot\vee\, s' \in F) \;\vee\; (\exists \sigma \in \Sigma)\, \delta(s, \sigma) \neq \delta(s', \sigma))$.

Every MADFA has a unique state $\pi \in S$ such that $(\forall \sigma \in \Sigma)\, \delta(\pi, \sigma) = \Omega$ and it is final. This state is called *pre-dead* and its existence is a direct consequence of

MADFAs definition and Lemma 1. Given a trim ADFA, $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$, the *rank* of a state $s \in S$, denoted $rk(s)$, is the length of the longest word $x \in \Sigma^\star$ such that $\delta(s,x) \in F$. The *rank*[a] of an ADFA $\mathcal{A}$, $rk(\mathcal{A})$, is $max\{rk(s) \mid s \in S\}$. Trivially, we have that $rk(s_0) = rk(\mathcal{A})$ and $rk(\pi) = 0$. Given a trim ADFA the rank of each state can be determined by the following algorithm:

```
for s in S
   rk(s) ← ⊥
rank(s0)

def rank(s)
  if rk(s) ≠⊥ then return rk(s)
  r ← 0
  for σ ∈ Σ
    if δ(s,σ) ≠ Ω then r ← max(r,1+rank(δ(s,σ)))
  rk(s) ← r
  return r
```

For every state $s \in S$, with $rk(s) > 0$ there exists a transition to a state with rank immediately lower than s's.

Lemma 2. *Let* $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$ *be an* ADFA*, then*

$$(\forall s \in S)(rk(s) \neq 0 \Rightarrow (\exists \sigma \in \Sigma)\ rk(\delta(s,\sigma)) = rk(s) - 1).$$

The above considerations lead to a optimized minimization algorithm for ADFAs. Consider a total ordering in Σ and for each rank a total order $\prec$ in S. We denote $R_l = \{s \in S \mid rk(s) = l\}$ and $n_l = |R_l|$, for $l \in [0, rk(\mathcal{A})]$. The minimization algorithm for trim ADFAs described below is based on the one presented in Lothaire [10] (page 33):

```
L ← ∅
for l ∈ [0,rk(A)]
  for (s',s'') ∈ R_l^2 and s' ≺ s''
    if (Rnm(L,δ(s',σ)) = Rnm(L,δ(s'',σ)))_{σ∈Σ} ∧ (s' ∈ F ↔ s'' ∈ F)
    then
      L ← L ∪{(s'',s')}
      delete(s'')

def Rnm(L,s)
  if (∃s' ∈ S) (s,s') ∈ L then return s' else return s
```

[a] Also called the *diameter* of $\mathcal{A}$.

Note that if two states are mergeable, than they must be in the same rank. By Lemma 1, they must be both final or non final, and have the same value of the transition function. By proceeding in increasing rank order and knowing that all transitions from a state have lower rank states as targets, the correctness of the algorithm is ensured.

4. Normal Form for MADFAs

Based upon the minimization algorithm described in the last section, we are going to characterize a canonical representation for MADFAs.

Let $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$ be a MADFA with $k = |\Sigma|$ and $n = |S| \geq 2$. Consider a total order over Σ and let $\Pi : \Sigma \longrightarrow [0, k[$ be the bijection induced by that order. For each state $s \in S$, let its *representation* be a $(k+1)$-tuple $\Delta(s) = (\varphi(\delta(s, \Pi^{-1}(0))), \ldots, \varphi(\delta(s, \Pi^{-1}(k-1))), f)$, where the first k values represent the transitions from state s and the last value, f, is 1 if $s \in F$ or 0, otherwise. If the last value is omitted we denote the representation by $\hat{\Delta}(s)$. The function φ will assign a number to each state and is defined as follows. All MADFAs have a dead state Ω and a pre-dead state π. Let $\varphi(\Omega) = 0$ and $\varphi(\pi) = 1$. Thus, the representation of Ω and π are $(0^k, 0)$, and $(0^k, 1)$, respectively. We can continue this process considering the states by increasing rank order, and in each rank we number the states by lexicographic order over their transition representations. It is important to note that transitions from a given state can only refer to states of a lower rank, and thus already numbered. Formally, the assignment of state numbers, φ, can be described by the following simple algorithm:

```
φ(Ω) ← 0
φ(π) ← 1
i ← 2
for l in ]0, rk(A)]
  for s ∈ R_l by lexicographic order over Δ(s)
    φ(s) ← i
    i ← i + 1
```

For example, considering the MADFA of Figure 1 ($n = 7$ and $k = 3$), its canonical representation can be constructed as follows:

rank	state	$\varphi(state)$	Δ(state)			
	Ω	0	0	0	0	0
0	π	1	0	0	0	1
1	s_5	2	1	1	1	0
2	s_4	3	2	1	1	0
3	s_3	4	2	3	2	0
3	s_2	5	3	3	0	0
4	s_1	6	4	0	0	0
5	s_0	7	5	6	6	0

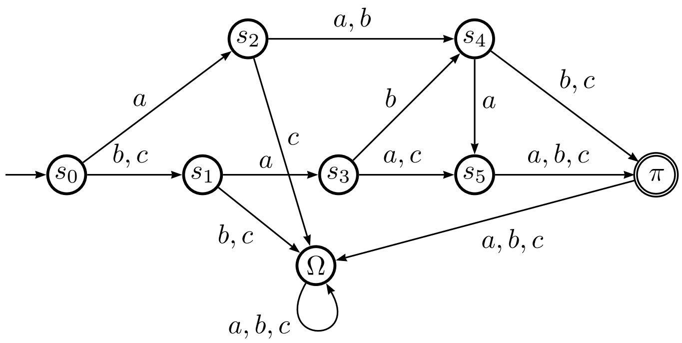


Fig. 1. An example of a MADFA that can be described by the canonical representation $[[0, 0, 0, 0], [0, 0, 0, 1], [1, 1, 1, 0], [2, 1, 1, 0], [2, 3, 2, 0], [3, 3, 0, 0], [4, 0, 0, 0], [5, 6, 6, 0]]$.

The following three theorems guarantee that this representation is indeed a canonical representation for MADFAs.

Theorem 3. *Let $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$ be a* MADFA *with $rk(\mathcal{A}) = d$, $n = |S|$ and $k = |\Sigma|$. Let $(s_i)_{i\in[0,(k+1)(n+1)[}$, with $s_i \in [0, n[$, be the string representation of $\mathcal{A}$ as above. Let $(r_l)_{l\in[0,d]}$ be the sequence of the first states of each rank in $(s_i)_i$, and let $(f_i)_{i\in[1,n[}$ be the sequence of the positions in $(s_i)_i$ of the first occurrence of each $i \in [1, n[$. Then*

$$s_0 = \cdots = s_k = \cdots = s_{2k} = 0 \ \wedge \ s_{2k+1} = 1 \tag{N0}$$

$$(\forall i \in [0, n])\, s_{(k+1)i+k} \in \{0, 1\} \tag{N1}$$

$$r_0 = 1 \ \wedge \ r_1 = 2 \ \wedge \ r_d = n \ \wedge \ (\forall l \in [0, d[)\, r_l < r_{l+1} \tag{N2}$$

$$\begin{aligned} &((\forall i \in [1, n[)\, s_{f_i} = i \ \wedge \\ &(\forall j \in [0, n])(\forall m \in [0, k[)\, ((k + 1)j + m < f_i \ \Rightarrow \ s_{(k+1)j+m} \neq i)) \end{aligned} \tag{N3}$$

$$(\forall l \in [0, d[)(\forall i \in [r_l, r_{l+1}[)\, kr_{l+1} + 1 \leq f_i \tag{N4}$$

$$(\forall l \in [0, d])(\forall i \in [r_l, r_{l+1}[)(\exists m \in [0, k[)\ s_{(k+1)i+m} \in [r_{l-1}, r_l[\tag{N5}$$

$$(\forall l \in [0, d[)(\forall i \in [r_l, r_{l+1} - 1[)\, (s_{(k+1)i+m})_{m\in[0,k]} < (s_{(k+1)(i+1)+m})_{m\in[0,k]} \tag{N6}$$

Proof. The condition N0 is obvious from the definition of MADFAs and the uniqueness of the pre-dead state. The condition N1 states that the last symbol of each state representation indicates if the state is final or not. The condition N2 ensures that states are numbered by increasing rank order. The condition N3 defines the sequence $(f_i)_{i\in[1,n[}$, and ensures that $\mathcal{A}$ is initially connected. The condition N4 is a direct consequence of the rank definition, i.e., a state can only refer a state of a lower rank. The condition N5 follows from Lemma 2. Finally, in condition N6, $<$

denotes the lexicographic order which is imposed by the state numbers and the way the representation is constructed. □

We note that the above conditions N0–N6 could be expressed using directly the string representations $\Delta(i)$ and the sets of states in each rank, $(R_l)_l$. For instance, the condition N5 could be $(\forall l \in [0,d])(\forall i \in R_l[)(\exists m \in \hat{\Delta}(i))\; m \in R_{l-1}$. The adopted notation enforces the possible treatment of the sets of canonical representations as formal languages.

Given a representation $(s_i)_{i\in[0,(k+1)(n+1)[}$ verifying conditions N0–N6 it is possible to determine the rank d and the sets of states in each rank, R_l for $l \in [0,d]$. Let $\max\hat{\Delta}(i)$ be the largest value in the representation of a state $i \in [0,n]$. Assuming $R_0 = \{1\}$ we have

$$R_l = \{i \mid \max\hat{\Delta}(i) \in R_{l-1}\}, \qquad l \in [1,d],$$

where d is determined considering that $(R_l)_l$ is a (ordered) partition of $[1,n]$. Analogously, it is possible to determine the sequence $(r_l)_l$ refered in Theorem 3. Thus, in the following theorems we assume that this sequence was previously calculated from $(s_i)_i$.

Theorem 4. *Let $(s_i)_{i\in[0,(k+1)(n+1)[}$ with $s_i \in [0,n[$ be a string that satisfies conditions N0–N6, then the corresponding automaton is a* MADFA *with n states and an alphabet of k symbols.*

Proof. From the string $(s_i)_{i\in[0,(k+1)(n+1)[}$ we can obtain a DFA with an alphabet of k symbols and n states. By conditions N0, N2 and N4 it must be acyclic. By conditions N3 and N5 it must be trim. That it is minimal is a direct consequence of Lemma 1 and condition N6. □

Theorem 5. *Let $(s_i)_{i\in[0,(k+1)(n+1)[}$ and $(s'_i)_{i\in[0,(k+1)(n+1)[}$ be two distinct strings satisfying conditions N0–N6. Then they correspond to distinct* MADFA*s.*

Proof. Let $(s_i)_{i\in[0,(k+1)(n+1)[}$ and $(s'_i)_{i\in[0,(k+1)(n+1)[}$ be two distinct strings in the conditions required. Let $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$ and $\mathcal{A}' = (S' \cup \{\Omega'\}, \Sigma, \delta', s'_0, F')$ be the correspondent MADFAs, with π and π' their pre-dead states, respectively. The first two tuples of the two strings are the same, by condition N0. Let j, for $j \in [2,n]$, be the first tuple where the two strings differ, and let $(s_{(k+1)j+m})_{m\in[0,k]} < (s'_{(k+1)j+m})_{m\in[0,k]}$. Suppose that there exists a bijection $\psi : S \to S'$ that defines an isomorphism between $\mathcal{A}$ and $\mathcal{A}'$, then

(1) $\psi(\Omega) = \Omega'$, because Ω and Ω' are the unique cyclic states.
(2) $\psi(\pi) = \pi'$, because they are the unique states in each automaton, such that the dead state is the target of all its transitions.
(3) let $\Delta(i)$ and $\Delta'(i)$ denote the representation of state $i \in [2,n]$ (transitions and finality) of $\mathcal{A}$ and $\mathcal{A}'$, respectively; then, $(\forall i < j)\; \psi(\Delta(i)) = \Delta'(i)$

The values of both strings $(s_{(k+1)j+m})_{m\in[0,k]}$ and $(s'_{(k+1)j+m})_{m\in[0,k]}$ are lower than j, by condition N4. Thus, $\psi(\Delta(j)) \neq \Delta'(j)$. Moreover there cannot exist $j' > j$ such that $\psi(\Delta(j)) = \Delta'(j')$, because such a tuple would be lexicographically smaller than the tuple j (and that would contradict condition N6). Therefore, such an isomorphism cannot exist, and thus the two automata are non-isomorphic. □

We obtain an equivalent normal form if we omit the representation of the dead and the pre-dead states. Their explicit inclusion only simplifies the presentation of the algorithms in the next sections.

5. Exact Generation of MADFAs

In this section, we present a method to generate all MADFAs, given n and k. For each MADFA, its state representations are generated lexicographically according to the conditions N0-N6 of Theorem 3. The algorithm traverses the search tree, backtracking in its way, and generates all possible representations.

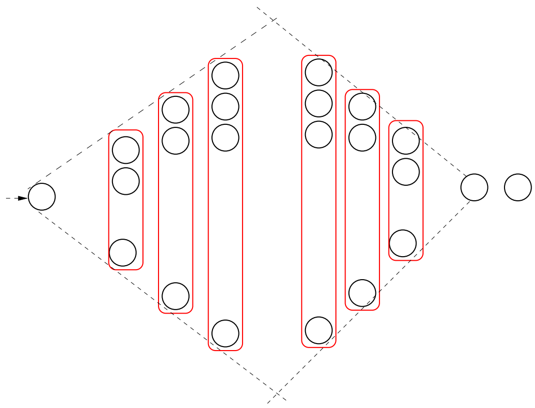


Fig. 2. Constraints on the size of ranks, where each rank (except the last and the first) is represented by a rectangle.

Let $NextState(k, l, c, r, r', D, m)$ be a function that returns the first $(k+1)$-tuple $\alpha = (\alpha_0, \ldots, \alpha_k)$ that lexicographically succeeds tuple l, and that satisfies the following constraints:

(1) $(\forall i \in [0, c[)\, \alpha_i \in [0, r[$
(2) $\alpha_c \in [r, r']$
(3) $(\forall i \in [c+1, k[)\, \alpha_i \in [0, r']$
(4) $\alpha_k \in \{0, 1\}$
(5) $m = 1 \;\Rightarrow\; \{\alpha_i \mid i \in [0, k[\,\} \cap D \neq \emptyset$
(6) $m = 2 \;\Rightarrow\; \{\alpha_i \mid i \in [0, k[\,\} \subseteq D$ and $|\{\alpha_i \mid i \in [0, k[\,\}| = k$.

If the above conditions cannot be satisfied, the function returns $\bot$. If $l = \bot$, it returns the first tuple that satisfies the conditions. The parameters r and r' are the first and last states in the previous rank, respectively, and the parameter c is the position of the first state to refer to a state of the previous rank (cf., constraints (1)-(3)). The parameter D is the set of *dangling* states not yet referred, *i.e.* not initially-accessible, and, that depending on the *mode* m, should or should not be *connected* in the new tuple α.

The algorithm is described as follows:

```
1   F ← ((0^k, 0), (0^k, 1))
2   NewRank(n, k, F, 1, 1, {1})
3
4   def EvalMode(n, k, F, D)
5     if |F| = n then
6       if |D| = 1 then output F
7       return −1
8     if |D| ≤ (k − 1)(n − |F| − 1) then return 0
9     if |D| < (k − 1)(n − |F|) + 1 then return 1
10    else return 2
11
12  def NewRank(n, k, F, r, r', D)
13    if (m = evalMode(n, k, F, D)) ≠ −1 then
14      for c ∈ (k − 1, ..., 0)
15        l ← ⊥
16        while (l ← NextState(k, l, c, r, r', D, m)) ≠ ⊥
17          SameRank(n, k, F + l, c, r, r', (D \ {l_i | i < k}) ∪ {|F|}, l)
18
19  def SameRank(n, k, F, c, r, r', D, l)
20    if (m = evalMode(n, k, F, D)) ≠ −1 then
21      for c' ∈ (c, ..., 0)
22        while (l ← NextState(k, l, c', r, r', D, m)) ≠ ⊥
23          SameRank(n, k, F + l, c', r, r', (D \ {l_i | i < k}) ∪ {|F|}, l)
24      NewRank(n, k, F, r' + 1, |F| − 1, D)
```

The following claims ensure the correctness of the algorithm.

Claim 1. Every generated sequence F satisfies conditions N0-N6.

Claim 2. For each n and k all legal strings are generated.

Proof. Claims 1. and 2. (Sketch) Considering the algorithm above and the existence of the function $NextState$, we have,

- The condition N0 is guaranteed by line 1.
- The condition N1 is a direct consequence of constraint (4).

- By the constraints (1)-(3), each new state generated by $NextState$ belongs to the rank immediately after the rank of state r. Moreover, the functions $NewRank$ and *SameRank* constraint the states to be generated by rank increasing order (and unit steps). This guarantees condition N2.
- The function $EvalMode$ and the constraints (5)-(6), ensure that conditions N3 (ADFAinitially connect) and N4 (rank order) are fulfilled. The conditions stated in lines $8 - 10$ correspond to a pruning of the state's representations search tree, and is illustrated in Figure 2. If the number of *dangling* states ($|D|$) is equal to $(k-1)(n-|F|)+1$, *i.e.*, equals all possible transitions left then all transitions from the states to be created must refer *dangling* those states ($m = 2$). If $(k-1)(n-|F|-1) < |D| < (k-1)(n-|F|)+1$ then at least one of those states must be referred ($m = 1$). If $|D| \leq (k-1)(n-|F|-1)$ none of those *dangling* states needs to be referred. The set of dangling states is updated in each recursive call to the functions $NewRank$ and $SameRank$ (lines $17, 23-24$).
- The condition N5 is a direct consequence of constraint (2).
- The condition N6 is ensured because *NextState* generates the tuples in lexicographic order, and the way the parameter c takes values (lines 14 and 21).

□

The algorithm was implemented in `Python` [13]. In Table 5 the number of MADFAs for some small values of n and k is summarized. For $k = 2$ and $n \leq 6$, those values were already presented by Domaratzki *et al.* [7] and by Liskovets [9].

	$k=2$		$k=3$	
n	MADFAs	Time (s)	MADFAs	Time (s)
2	6	0.012	14	0.015
3	60	0.015	532	0.019
4	900	0.026	42644	0.579
5	18480	0.026	6011320	95.034
6	487560	7.240	1330452032	24481.959
7	15824880	243.873		
8	612504240	9695.755		
9	27619664640	457881.581		
	$k=4$		$k=5$	
n	MADFAs	Time (s)	MADFAs	Time (s)
2	30	0.017	62	0.017
3	3900	0.061	26164	0.307
4	1460700	17.965	43023908	507.296
5	1220162880	16683.977		

Table 1. Number of MADFAs for small values of n and k, and performance times for its generation (AMD Athlon 64 at 2.5MHz).

6. Towards an Exact Enumeration of MADFAs

In this section we address some issues regarding the enumeration of MADFAs and therefore of finite languages.

6.1. *Counting* MADFA*s by Ranks*

It is easy to count the number of MADFAs over an alphabet of k symbols and with rank d. Each MADFA represents a distinct finite language where the largest word has length d, and all languages in these conditions are represented by a MADFA of rank d. The number of words of length i is k^i, for $i \in [0, d]$, and thus we have,

$$R_k(d) = \left(\prod_{i=0}^{d-1} 2^{k^i} \right) (2^{k^d} - 1).$$

6.2. *Counting* MADFA*s for n and k*

The number of finite languages represented by MADFAs with n states over an alphabet of k symbols, $M_k(n)$, would be obtained if we could count its canonical representations. So far, however, we were only able to obtain $M_k(n)$ for small values of n and assuming that we know the possible distributions of states by ranks. Let $\mathcal{A} = (S \cup \{\Omega\}, \Sigma, \delta, s_0, F)$ be a MADFA and $rk(\mathcal{A}) = d$.

Let $(n_l)_{l\in[1,d]}$ be the sequence of the number of states in each rank. The number of these sequences is at most 2^{n-3}, for $n > 2$, as they correspond to the integer compositions of size $n-2$. For each sequence$(n_l)_{l\in[0,d]}$, let $(m_f)_{f\in[1,d]}$ be the number of *dangling* states that are target of transitions from a state of a previous rank, for the first time.

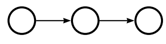
We are going to analyse the possible configurations for $n \in [2, 5]$, using the Principle of Inclusion and Exclusion. It is important to note that some configurations are not allowed for small values of k.

For $n = 2$, the state s_0 can be final or not, and the number of possible transition functions is 2^k, excluding the one where all transitions have as target the dead state. We have

$$M_k(2) = 2(2^k - 1).$$

In the following diagrams the dead state is omitted. For $n = 3$, we only have to consider one configuration:

$$(n_l)_{l\in[0,2]} = (1, 1, 1) \quad (m_f)_{f\in[1,2]} = (1, 1).$$

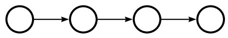
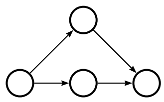


Then,

$$M_k(3) = 2^2(3^k - 2^k)(2^k - 1).$$

Note that now there are 3^k different transitions from the state s_0, but 2^k have the dead state as unique target.

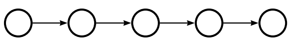
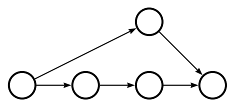
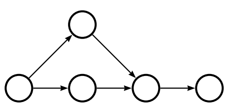
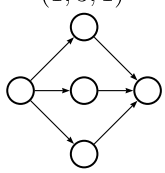
For $n = 4$ (and $k > 1$), we have two configurations $(n_l)_{l\in[0,3]}$ and $(n_l)_{l\in[0,3[}$, each one with a possible sequence $(m_f)_f$:

d	$(n_l)_{l\in[0,d]}$	$(m_f)_{f\in[1,d]}$
3	$(1,1,1,1)$ 	$(1,1,1)$
2	$(1,2,1)$ 	$(1,2)$

Then,

$$M_k(4) = 2^3(4^k - 3^k)(3^k - 2^k)(2^k - 1) + 2(4^k - 3^k 2 + 2^k)\binom{2(2^k - 1)}{2}$$

For $n = 5$, we have four possible configurations for $(n_l)_l$:

d	$(n_l)_l$	$(m_f)_f$	d	$(n_l)_l$	$(m_f)_f$
5	$(1,1,1,1,1)$ 	$(1,1,1,1)$	4	$(1,1,2,1)$ 	$(1,1,2)$
4	$(1,2,1,1)$ 	$(1,1,2)$ $(1,2,1)$	3	$(1,3,1)$ 	$(3,1)$

The last configuration is only possible for $k > 2$. Then,

$$\begin{aligned} M_k(5) &= 2^4(5^k - 4^k)(4^k - 3^k)(3^k - 2^k)(2^k - 1) \\ &\quad + 2^2(5^k - 4^k)(4^k - 3^k 2 + 2^k)\binom{2(2^k-1)}{2} \\ &\quad + 2^3(5^k - 4^k 2 + 3^k)(3^k - 2^k)\binom{2(2^k-1)}{2} \\ &\quad + 2^2(5^k - 4^k 2 + 3^k)\binom{2(3^k-2^k)}{2}(2^k - 1) \\ &\quad + 2(5^k - 4^k 3 + 3^k 3 - 2^k)\binom{2(2^k-1)}{3}. \end{aligned}$$

In a similar way, we can obtain the formulae for higher values[b] of n. But the interaction between the sequences $(n_l)_l$ and $(m_f)_f$ is not so straightforward to count and the above general approach leads to double counting. It provides, however, an upper bound for $M_k(n)$.

We now consider some lower bounds for $M_k(n)$. Any family of MADFAs which canonical representation can be expressed by a (unambiguous) regular expression or context-free grammar can be easily enumerated using generating functions (see Sedgewick and Flagolet [12]).

For each n and k, consider the set of MADFAs with one state per rank. The normal forms of these automata, omiting the values for the pre-dead and dead states, can be characterized by the following regular expression, where concatenation is taken as a product and disjunction as a sum:

$$\prod_{i=1}^{n-1} \sum_{c=0}^{k-1} (0 + \cdots + i)^{k-c-1} i (0 + \cdots + (i-1))^c (0+1)$$

Counting the number of strings represented by this regular expression, for each n and k, we have:

$$\prod_{i=1}^{n-1} \sum_{c=0}^{k-1} 2(i+1)^{k-c-1} i^c = 2^{(n-1)} \prod_{i \in [1,n[} ((i+1)^k - i^k)$$

So we obtain the following lower bound:

$$M_k(n) \geq 2^{(n-1)} \prod_{i \in [1,n[} ((i+1)^k - i^k),$$

where equality holds for $n = 2$ and $n = 3$.

This family of automata coincides with the family $S_{n+1,k}$ introduced by Domaratzki [6], which size provided a lower bound for $M_k(n)$. We note that our normal form allows much shorter proofs. For $k = 2$ and $n \geq 4$, Domaratzki [6] improved the previous lower bound by presenting another family of MADFAs, $T(n)$. Although

[b] The formula for $n = 6$ is a page long and cumbersome.

it is not hard to obtain a regular expression for the canonical representation of this family, it is too long to present its construction here.

6.3. *Estimates of the Number of States per Rank*

The possible distributions of states per ranks are an important issue towards the enumeration of MADFAs. As was pointed out by Liskovets [9], the number of states in rank 1 must be at most $2(2^k - 1)$. Let $d \geq 1$ be the rank of a MADFA and let n_l for $l \in [1, d]$ be the number of states in each rank, with $n_{-1} = 1$, $n_0 = 1$, and $n_d = 1$. Because the MADFA must be initially-connected and in rank $d - i$ the maximum number of states is k^i, for $i \in [1, d[$, we have the following recurrence:

$$n_{d-i} \leq k n_{d-(i-1)} + \sum_{j=1}^{i-1} (k^j - n_{d-j}), \qquad i \in [1, d[.$$

In the other hand, for each state in each rank there exists a transition to a state in the previous rank, thus we have

$$n_i \leq 2((\sum_{j=-1}^{i-1} n_j)^k - (\sum_{j=-1}^{i-2} n_j)^k), \qquad i \in [1, d[. \tag{7}$$

The inequality (7) was also derived by Câmpeanu and Ho [4]. They presented a closed formula for the recurrence obtained considering the equality in (7) and, with that, obtained upper bounds for the number of states of MADFAs accepting words of length at most d and with k symbols.

7. Conclusions

We presented a canonical representation for minimal acyclic deterministic finite automata with n states and k symbols and a method for their exact generation. The study of the combinatorial properties of this canonical representation can contribute to obtain a formula for their enumeration, or at least more upper and lower bounds. In particular, a characterization in terms of context-free languages, if it exists, would be helpful. We also plan to study the possibility of use this canonical representation for a uniform random generator for MADFAs. It is possible to extend this canonical representation for ADFAs by inducing a canonical order over mergeable states in the same rank. Further extension of this representation to cyclic minimal automata seems impossible as for those automata there is no notion of rank.

8. Acknowledgements

We are most grateful to Valery Liskovets that kindly posed several interesting questions related to the number of MADFAs and its distributions by number of final states. We thank also the anonymous referees for their valuable comments that helped to improve the paper.

This work was partially funded by Fundação para a Ciência e Tecnologia (FCT) and Program POSI, and by project ASA (PTDC/MAT/65481/2006). Marco Almeida is funded by FCT grant SFRH/BD/27726/2006.

References

[1] M. Almeida, N. Moreira, and R. Reis. Aspects of enumeration and generation with a string automata representation. In H. Leung and G.Pighizzini, editors, *Proc. of DCFS'06*, pages 58–69, Las Cruces, New Mexico, 2006. NMSU.

[2] M. Almeida, N. Moreira, and R. Reis. Enumeration and generation with a string automata representation. *Theoretical Computer Science*, 387(2):93–102, 2007.

[3] D. Callan. A determinant of Stirling cycle numbers counts unlabeled acyclic single-source automata. Department of Statistics, University of Wisconsin-Madison, 2007.

[4] C. Câmpeanu and W. H. Ho. The maximum state complexity for finite languages. *J. of Automata, Languages and Combinatorics*, 9(2-3):189–202, 2004.

[5] M. Domaratzki. Combinatorial interpretations of a generalization of the Genocchi numbers. *Journal of Integer Sequences*, 7(04.3.6), 2004.

[6] M. Domaratzki. Improved bounds on the number of automata accepting finite languages. *International Journal of Foundations of Computer Science*, 15(1):143–161, 2004.

[7] M. Domaratzki, D. Kisman, and J. Shallit. On the number of distinct languages accepted by finite automata with n states. *J. of Automata, Languages and Combinatorics*, 7(4):469–486, 2002.

[8] J. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 2000.

[9] V. A. Liskovets. Exact enumeration of acyclic deterministic automata. *Discrete Applied Mathematics*, 154(3):537–551, March 2006.

[10] M. Lothaire. Algorithms on words. In *Applied Combinatorics on Words*, chapter 1. Cambridge University Press, 2005.

[11] R. Reis, N. Moreira, and M. Almeida. On the representation of finite automata. In C. Mereghetti, B. Palano, G. Pighizzini, and D.Wotschke, editors, *Proc. of DCFS'05*, pages 269–276, Como, Italy, 2005.

[12] R. Sedgewick and P. Flajolet. *Analysis of Algorithms*. Addison Wesley, 1996.

[13] G. van Rossum. *Python Library Reference*, 2.4.2 edition, 2005.

[14] S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 1. Springer-Verlag, 1997.