

# On the Representation of Finite Automata \*

Rogério Reis  
rvr@ncc.up.pt

Nelma Moreira †  
nam@ncc.up.pt

Marco Almeida  
mfa@alunos.dcc.fc.up.pt

DCC-FC & LIACC, Universidade do Porto  
R. do Campo Alegre 823, 4150 Porto, Portugal

## Abstract

We give an unique string representation, up to isomorphism, for initially connected deterministic finite automata (ICDFA's) with  $n$  states over an alphabet of  $k$  symbols. We show how to generate all these strings for each  $n$  and  $k$ , and how its enumeration provides an alternative way to obtain the exact number of ICDFA's.

## 1 Motivation

In symbolic manipulation environments for finite automata, it is important to have an adequate representation of automata and, dependent upon their use, several representations may be available. For example, for testing if two finite automata are isomorphic objects or for (random) generation of automata, the representation must be compact and somehow canonical. In the **FAdo** project [?, ?] a canonical form is used to test if two minimal DFA's are *isomorphic* (i.e are the same up to renaming of states). In this paper we prove the correctness of that representation and show how it can be used for the exact enumeration and generation of initially connected deterministic finite automata (ICDFA). The problem of enumeration of finite automata was considered by several authors since early 1960s, in particular see Robinson [?], Harary and Palmer [?] and Liskovets [?] amongst many others. A survey may be found in Domaratzki et al. [?]. More recently, several authors examined related problems. Domaratzki et al. [?] studied the enumeration of distinct languages accepted by finite automata with  $n$  states; Nicaud [?], Champarnaud and Paranthoën [?, ?] and Bassino and Nicaud [?] analysed several aspects of the average behaviour of regular languages; Liskovets [?] and Domaratzki [?] gave (exact and asymptotic) enumerations of acyclic DFA's and of finite languages. The paper is organised as follows. In the next section, we review some basic notions and introduce some notation. Section ?? describes a string representation for deterministic finite automata that is unique up to isomorphism for initially connected deterministic finite automata. Section ?? presents an efficient method to generate those strings. Section ?? shows how their enumeration provides an upper bound and the exact value for the number of ICDFA's. Section ?? concludes with some final remarks. We address the reader attention to the longer version of this paper for some implementation issues and experimental results<sup>1</sup>.

---

\*Work partially funded by Fundação para a Ciência e Tecnologia (FCT) and Program POSI.

†Corresponding author

<sup>1</sup><http://www.dcc.fc.up.pt/Pubs/TR05/dcc-2005-04.ps.gz>

## 2 Preliminaries

We first recall some basic notions from automata theory and formal languages, that can be found in standard books [?]. An alphabet  $\Sigma$  is a nonempty set of symbols. A string over  $\Sigma$  is a finite sequence of symbols of  $\Sigma$ . The empty string is denoted by  $\epsilon$ . The set  $\Sigma^*$  is the set of all strings over  $\Sigma$ . A language  $L$  is a subset of  $\Sigma^*$ . The density of a language  $L$  over  $\Sigma$ ,  $\rho_L(n)$ , is the number of strings of length  $n$  that are in  $L$ , i.e.,  $\rho_L(n) = |L \cap \Sigma^n|$ . If  $L_1, L_2 \subseteq \Sigma^*$ ,  $L_1 L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$ . A regular expression (r.e.)  $\alpha$  over  $\Sigma$  represents a language  $L(\alpha) \subseteq \Sigma^*$  and is inductively defined by:  $\emptyset$ ,  $\epsilon$  and  $\sigma \in \Sigma$  are a r.e., where  $L(\emptyset) = \emptyset$ ,  $L(\epsilon) = \{\epsilon\}$  and  $L(\sigma) = \{\sigma\}$ ; if  $\alpha_1$  and  $\alpha_2$  are r.e.,  $(\alpha_1 + \alpha_2)$ ,  $(\alpha_1 \alpha_2)$  and  $\alpha_1^*$  are r.e., respectively with  $L((\alpha_1 + \alpha_2)) = L(\alpha_1) \cup L(\alpha_2)$ ,  $L((\alpha_1 \alpha_2)) = L(\alpha_1)L(\alpha_2)$  and  $L(\alpha_1^*) = L(\alpha_1)^*$ . In this paper, we will use regular expressions to represent descriptions of finite automata. A *deterministic finite automaton* (DFA)  $\mathcal{A}$  is a quintuple  $(Q, \Sigma, \delta, q_0, F)$  where  $Q$  is a finite set of states,  $\Sigma$  is the alphabet,  $\delta : Q \times \Sigma \rightarrow Q$  is the transition function,  $q_0$  the initial state and  $F \subseteq Q$  the set of final states. We assume that the transition function is total, so we consider only *complete* DFA's. The *size* of a DFA is the number of its states,  $|Q|$ . Normally, we are not interested in the labels of the states and we can represent them by an integer  $0 \leq i < |Q|$ . The transition function  $\delta$  extends naturally to  $\Sigma^*$ : for all  $q \in Q$ , if  $x = \epsilon$  then  $\delta(q, \epsilon) = q$ ; if  $x = y\sigma$  then  $\delta(q, x) = \delta(\delta(q, y), \sigma)$ . A DFA is *initially connected*<sup>2</sup> (ICDFA) if for each state  $q \in Q$  there exists a string  $x \in \Sigma^*$  such that  $\delta(q_0, x) = q$ . Two DFA's  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$  and  $\mathcal{A}' = (Q', \Sigma, \delta', q'_0, F')$  are called *isomorphic* (by states) if there exists a bijection  $f : Q \rightarrow Q'$  such that  $f(q_0) = q'_0$  and for all  $\sigma \in \Sigma$  and  $q \in Q$ ,  $f(\delta(q, \sigma)) = \delta'(f(q), \sigma)$ . Furthermore, for all  $q \in Q$ ,  $q \in F$  if and only if  $f(q) \in F'$ . The *language* accepted by a DFA  $\mathcal{A}$  is  $L(\mathcal{A}) = \{x \in \Sigma^* \mid \delta(q_0, x) \in F\}$ . Two DFA are *equivalent* if they accept the same language. Obviously, two isomorphic automata are equivalent, but two non-isomorphic automata may be equivalent. A DFA  $\mathcal{A}$  is *minimal* if there is no DFA  $\mathcal{A}'$  with fewer states equivalent to  $\mathcal{A}$ . Trivially a minimal DFA is an ICDFA. Minimal DFA's are unique up to isomorphism. We are mainly concerned with the representation of the transition function of DFA's, so we disregard the set of final states and we consider only a quadruple  $(Q, \Sigma, \delta, q_0)$  called the *structure* of an automaton and referred as  $\text{DFA}_\emptyset$ . For each of our representations, there will be  $2^n$  DFA's. We denote by  $\text{ICDFA}_\emptyset$  the structure of an ICDFA. We consider that any integer variable has always a nonnegative value (if not otherwise stated). Let  $[n]_0 = \{0, 1, \dots, n\}$  and  $[n] = \{1, \dots, n\}$ .

## 3 Representations towards a normal form

The method used to represent a DFA has a significative role in the amount of computer work needed to manipulate that information, and can give an important insight about this set of objects, both in its characterisation and enumeration. Let us disregard the set of *final states* of a DFA. A *naive* representation of a  $\text{DFA}_\emptyset$  can be obtained by the enumeration of its states and for each state a list of its transitions for each symbol. For

---

<sup>2</sup>Also called *accessible*.

the DFA<sub>∅</sub> in Fig.?? we have:

$$[[A(a : A, b : B)], [B(a : A, b : E)], [C(a : B, b : E)], [D(a : D, b : C)], [E(a : A, b : E)]] \quad (1)$$

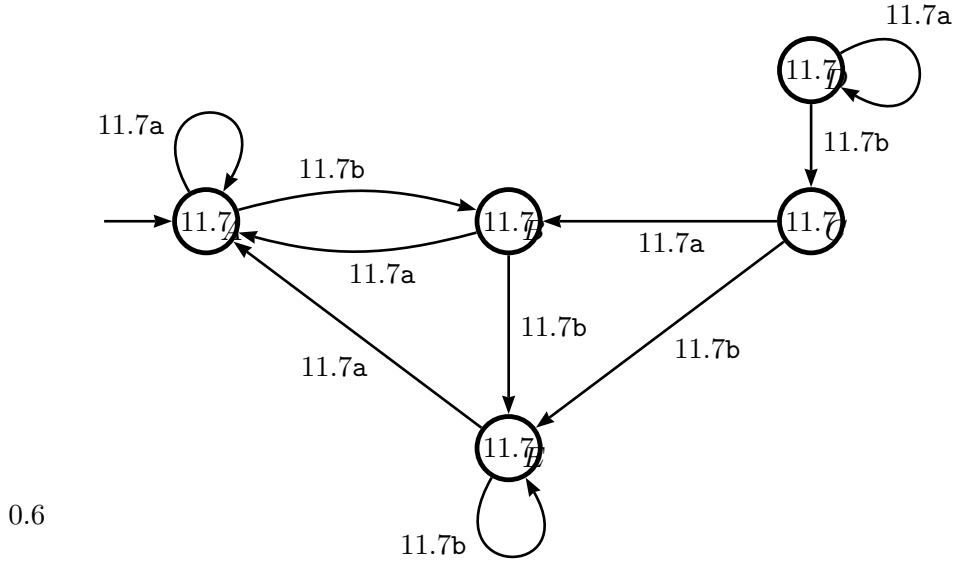


Figure 1: A DFA with no final states marked

Given a complete DFA<sub>∅</sub> (Q, Σ, δ, q<sub>0</sub>) with |Q| = n and |Σ| = k and considering a total order over Σ, the representation can be simplified by omitting the alphabetic symbols. For our example, we would have

$$[[A(A, B)], [B(A, E)], [C(B, E)], [D(D, C)], [E(A, E)]] \quad (2)$$

The labels chosen for the states have a standard order (in the example, the alphabetic order). We can simplify the representation a bit if we use that order to identify the states, and because we are representing complete DFA<sub>∅</sub>'s we can drop the inner tuples as well. We obtain

$$[0, 1, 0, 4, 1, 4, 3, 2, 0, 4] \quad (3)$$

Because this representation depends on the order we label the states, we have more than one representation for each DFA<sub>∅</sub>. Can we have a canonical order for the set of the states? Let the first state be the initial state q<sub>0</sub> of the automaton, the second state the first one to be referred (excepting q<sub>0</sub>) by a transition from q<sub>0</sub>, the third state the next referred in transitions from one of the first two states, and so on... For the DFA<sub>∅</sub> in the example, this method induces an unique order for the first three states (A, B, E), but then we can arbitrate an order for the remaining states (C, D). Two different representations are thus admissible:

$$[0, 1, 0, 2, 0, 2, 3, 4, 1, 2] \text{ and } [0, 1, 0, 2, 0, 2, 1, 2, 4, 3] \quad (4)$$

If we restrict this representation to IC DFA<sub>∅</sub>'s, then this representation is unique and defines an order over the set of its states. In the example, the DFA<sub>∅</sub> restricted to the

set of states  $\{A, B, E\}$  is represented by  $[0, 1, 0, 2, 0, 2]$ . Let  $\Sigma = \{\sigma_i \mid i < k\}$ , with  $\sigma_0 < \sigma_1 < \dots < \sigma_{k-1}$ . Given an IC DFA $_{\emptyset}$   $(Q, \Sigma, \delta, q_0)$  with  $|Q| = n$ , the representing string is of the form  $[(S_i)_{i < kn}]$  with  $S_i \in [n-1]_0$  and  $S_i = \delta(\lfloor i/k \rfloor, \sigma_{i \bmod k})$ .

**Lemma 1.** *Let  $[(S_i)_{i < kn}]$  be a representation of a complete IC DFA $_{\emptyset}$   $(Q, \Sigma, \delta, q_0)$  with  $|Q| = n$  and  $|\Sigma| = k$ , then:*

$$(\forall m > 1)(\forall i)(S_i = m \Rightarrow ((\exists j < i) S_j = m - 1)) \quad (\mathbf{R1})$$

$$(\forall m \in [n-1])(\exists j < km) S_j = m \quad (\mathbf{R2})$$

*Proof.* The condition **R1** establishes that a state label (greater than 1) can only occur after the occurrence of its predecessors. This is a direct consequence of the way we defined the representing string. Suppose **R2** does not verify, thus there exists a state  $m$  that does not occur in the first  $km$  symbols of the string (the  $m$  first state descriptions). Because the automaton is initially connected there must be a sequence of states  $(m_i)_{i \leq l}$  and symbols  $(\sigma_i)_{i \leq l}$  such that  $m_0 = 0$ ,  $m_l = m$  and  $\delta(m_i, \sigma_i) = m_{i+1}$  for  $i < l$ . We must have  $0 < m < m_{l-1}$  because  $m$  appears in the  $m_{l-1}$  description and we supposed no occurrences of  $m$  in the first  $m$  state descriptions. There must exist  $l' < l$  such that  $m_{l'-1} < m < m_{l'}$ , implying that  $m_{l'} \in \{S_i \mid i < km\}$ . This contradicts **R1** because we are supposing that  $m \notin \{S_i \mid i < km\}$  and  $m < m_{l'}$ . Thus **R2** is verified.  $\square$

Note that the conditions **R1** and **R2** are independent. For  $k = 2$  and  $n = 3$ , the string  $[2, 1, 0, 0, 1, 0]$  satisfies **R2** but not **R1**, and the opposite occurs for the string  $[0, 0, 1, 1, 0, 2]$ .

**Lemma 2.** *Every string  $[(S_i)_{i < kn}]$  with  $S_i \in [n-1]_0$  satisfying **R1** and **R2** represents a complete IC DFA $_{\emptyset}$  with  $n$  states over an alphabet of  $k$  symbols.*

*Proof.* Let  $[(S_i)_{i < kn}]$  be a string in the referred conditions, and consider the associated automaton  $\mathcal{A}$  using the string symbols as labels for the corresponding states. By its construction,  $\mathcal{A}$  is a DFA $_{\emptyset}$ . We only need to prove that it is initially connected. Let  $m$  be a state of the automaton. A proof that  $m$  is reachable from the initial state 0 can be done by induction on  $m$ . If  $m = 0$  there is nothing to prove. If  $m = 1$  then, by **R2**, 1 must occur in the description of state 0, making state 1 reachable from state 0. Let us suppose that every state  $m' < m$  is reachable from state 0 and prove that state  $m$  is reachable too. By **R2**,  $m$  occurs at least once before position  $km$ , say in position  $km' + i$  with  $m' < m$  and  $i < k$ . Then for some symbol  $\sigma$ ,  $\delta(m', \sigma) = m$ . By induction hypothesis, state  $m'$  is reachable from state 0, thus state  $m$  is reachable too and the automaton is initially connected. Now consider the string representation obtained for  $\mathcal{A}$ ,  $[(S'_i)_{i < kn}]$ . By Lemma ?? it satisfies **R1** and **R2**. It is easy to see that this representation is the same as  $[(S_i)_{i < kn}]$ . By **R1**,  $S_0 = S'_0$ . Suppose that  $(\forall i < j)(S_i = S'_i)$ . Now we prove that  $S'_j = S_j$ . By **R1**, either  $S_j \in \{S_i \mid i < j\}$  or  $S_j = \max\{S_i \mid i < j\} + 1$ . In the first case, there exists  $l < j$  such that  $S_j = S_l$  and, by induction hypothesis,  $S_l = S'_l$ , thus

Analogously, by **R1**, in the second case we have that

$$S'_j = \max\{S_i \mid i < j\} + 1 = S_j$$

$\square$

**Theorem 1.** *There is a one-to-one mapping between strings  $[(S_i)_{i < kn}]$  with  $S_i \in [n - 1]_0$  satisfying **R1** and **R2**, and the non-isomorphic IC DFA $_{\emptyset}$ 's with  $n$  states, over an alphabet  $\Sigma$  of size  $k$ .*

*Proof.* Let  $(Q, \Sigma, \delta, q_0)$  and  $(Q', \Sigma, \delta', q'_0)$  be two IC DFA $_{\emptyset}$ 's and  $[(S_i)_{i < kn}]$  and  $[(S'_i)_{i < kn}]$  their representing strings. By Lemma ??, these strings satisfy **R1** and **R2**. Suppose that  $f : Q \rightarrow Q'$  is an isomorphism between the IC DFA $_{\emptyset}$ 's. Then  $0 = q_0$  and  $f(q_0) = q'_0 = 0$ . Either  $S_0 = \delta(q_0, \sigma_0) = q_0 = 0$  or  $S_0 = \delta(s_0, \sigma_0) = 1$  (by **R1**).

- i) If  $S_0 = 0$  then  $f(q_0) = f(\delta(q_0, \sigma_0)) = \delta'(q'_0, \sigma_0) = S'_0 = 0$ , because  $\delta(q_0, \sigma_0) = q_0$  implies  $\delta'(f(q_0), \sigma_0) = f(q_0)$ .
- ii) If  $S_0 = 1$  then  $f(1) = \delta'(q'_0, \sigma_0) = S'_0 \neq 0$ , thus  $S'_0 = 1$ , again by **R1**.

Supposing that  $(\forall i < j)(S_i = S'_i \wedge f(S_i) = S'_i)$  we need to prove that  $S_j = S'_j \wedge f(S_j) = S'_j$ . Trivially we have  $\{S_i \mid i < j\} = \{S'_i \mid i < j\}$ . We know that  $S_j = \delta(\lfloor j/k \rfloor, \sigma_{j \bmod k})$ , and by **R2** there exists  $l < j$  such that  $\lfloor j/k \rfloor = S_l$  thus  $f(\lfloor j/k \rfloor) = f(S_l) = S_l = S'_l = \lfloor j/k \rfloor$  by induction hypothesis. We have

$$S'_j = \delta'(\lfloor j/k \rfloor, \sigma_{j \bmod k}) = \delta'(f(\lfloor j/k \rfloor), \sigma_{j \bmod k}) = f(\delta(\lfloor j/k \rfloor, \sigma_{j \bmod k})) = f(S_j).$$

By **R1**, either  $S_j \in \{S_i \mid i < j\}$  or  $S_j = \max\{S_i \mid i < j\} + 1$ .

- i) If  $S_j \in \{S_i \mid i < j\}$  then there exists  $l < j$  such that  $S_j = S_l$  and  $S_l = S'_l$ . Then

$$\begin{aligned} \delta(\lfloor j/k \rfloor, \sigma_{j \bmod k}) = \delta(\lfloor l/k \rfloor, \sigma_{l \bmod k}) &\Rightarrow f(\delta(\lfloor j/k \rfloor, \sigma_{j \bmod k})) = f(\delta(\lfloor l/k \rfloor, \sigma_{l \bmod k})) \\ \Leftrightarrow \delta'(\lfloor j/k \rfloor, \sigma_{j \bmod k}) = \delta'(\lfloor l/k \rfloor, \sigma_{l \bmod k}) \end{aligned}$$

Thus  $S_j = S_l$  implies  $S'_j = S'_l$ , and so  $S'_j = S_j$ .

- ii) If  $S_j = \max\{S_i \mid i < j\} + 1$  then  $S'_j \notin \{S_i \mid i < j\}$  because if there exists a  $l < j$  such that  $S'_l = S'_j$  by the same reason as before  $S_j \in \{S_i \mid i < j\}$ . Thus, by **R1**  $S'_j = \max\{S_i \mid i < j\} + 1 = S_j$ .

Conversely, by Lemma ??, we have that each string represents a IC DFA $_{\emptyset}$  up to a compatible renaming of states, i.e., if two IC DFA $_{\emptyset}$ 's are represented by the same string, that representation defines an isomorphism between them.  $\square$

These string representations lead to a normal representation for IC DFA $_{\emptyset}$ 's. For each of them, if we add a sequence of *final states*, we obtain a **normal form** for IC DFA's.

## 4 Generating automata

Normal representations for IC DFA $_{\emptyset}$ 's (as presented above) can be used as compact computer representations for that kind of objects, but even though rules **R1** and **R2** are quite simple, it is not evident how to write an enumerative algorithm in an efficient way. In a string representing an IC DFA $_{\emptyset}$  with  $n$  states over an alphabet of  $k$  symbols,  $[(S_i)_{i < kn}]$ , let  $(f_j)_{0 < j < n}$  be the sequence of indexes of the first occurrence of each state label  $j$ . That those indexes exist is a direct consequence of the way the string is constructed. Now

consider  $b_1 = f_1$ ,  $b_j = f_j - f_{j-1}$ , for  $2 \leq j \leq n-1$  and  $b_n = kn - f_{n-1} + 1$ . Note that  $\sum_{l=1}^j b_l = f_j$ , for  $2 \leq j \leq n-1$ .

Note that

$$\sum_{l=1}^j b_l = f_j, \text{ for } j \in [n-1].$$

It is easy to see that

1. Rule **R1** simply states that

$$(\forall 2 \leq j \leq n-1)(b_j > 0). \quad (\mathbf{G1})$$

2. Rule **R2** establishes that

$$(\forall m \in [n-1])(f_m < km). \quad (\mathbf{G2})$$

To generate all the automata, for each allowed sequence of  $(b_j)_{0 < j < n}$  we can generate all the remaining symbols  $S_i$  (those with  $i \notin \{f_j \mid 0 < j < n\}$ ) according to the following rules:

$$i < b_1 \Rightarrow S_i = 0; \quad (\mathbf{G3})$$

$$(\forall j \in [n-2])(f_j < i < f_{j+1} \Rightarrow S_i \in [j]_0); \quad (\mathbf{G4})$$

$$i > f_{n-1} \Rightarrow S_i \in [n-1]_0. \quad (\mathbf{G5})$$

## 5 Enumeration of IC DFA's

In this section we obtain a formula  $B_k(n)$  for the number of strings  $[(S_i)_{i < kn}]$  representing IC DFA $_{\emptyset}$ 's with  $n$  states over an alphabet of  $k$  symbols. Although it is already known a formula for the number of non-isomorphic IC DFA $_{\emptyset}$ 's, we think that our method is new. Liskovets [?] and, independently, Robinson [?] gave for that number the formula  $H_k(n) = \frac{h_k(n)}{(n-1)!}$  where  $h_k(1) = 1$  and for  $n > 1$

$$h_k(n) = n^{kn} - \sum_{1 \leq j < n} \binom{n-1}{j-1} n^{k(n-j)} h_k(j) \quad (5)$$

Note that  $n^{kn}$  is the number of transition functions, from which we subtract the number of them that have  $n-1, n-2, \dots, 1$  states not accessible from the initial state. And then, we may divide by  $(n-1)!$ , as the names of the remaining states (except the initial) are irrelevant. Reciprocally, the formula we will derive ( $B_k(n)$ ) is a direct positive summation.

First, let us consider the set of strings  $[(S_i)_{i < kn}]$  with  $S_i \in [n-1]_0$  and satisfying only rule **R1**. The number of these strings gives an upper bound for  $B_k(n)$ . This set can be given by  $A_n \cap [n-1]_0^{kn}$ , where for  $c > 0$ ,

$$A_c = L(0^* + \sum_{i=1}^{c-1} 0^* \prod_{j=1}^i j(0 + \dots + j)^*). \quad (6)$$

These languages belong to a family of languages  $L_c$  presented by Moreira and Reis [?] and that represent partitions of  $[n]$  with no more than  $c \geq 1$  parts, i.e.,

$$L_c = L\left(\sum_{i=1}^c \prod_{j=1}^i j(1 + \dots + j)^*\right). \quad (7)$$

We have that  $\rho_{A_c}(n) = \rho_{L_c}(n+1)$  and that  $\rho_{L_c}(n) = \sum_{i=1}^c S(n, i)$ , where  $S(n, i)$  are Stirling numbers of second kind. So we get that the number of strings of length  $kn$  that are in  $A_n$ , is  $\rho_{A_n}(kn) = \sum_{i=1}^n S(kn+1, i)$ . We have the proposition,

**Proposition 1.** For all  $n, k \geq 1$ ,  $B_k(n) \leq \sum_{i=1}^n S(kn+1, i)$ .

For  $n = 3$  and  $k = 2$ ,  $B_2(3) \leq 365$ . For  $k = 2$ , Bassino and Nicaud [?] presented a better upper bound, namely that  $B_2(n) \leq nS(2n, n)$ .

Now let us consider only the rule **R2**. This rule can be formulated as

$$\bigwedge_{m=1}^{n-1} \bigvee_{j=0}^{km-1} S_j = m. \quad (8)$$

From this formula it is easy to see that the strings  $[(S_i)_{i < kn}]$  with  $S_i \in [n-1]_0$  and satisfying only rule **R2** can be represented by the regular expression

$$\bigcap_{m=1}^{n-1} \sum_{j=0}^{km-2} (0 + \dots + (m-1))^j m (0 + \dots + (n-1))^{kn-j-1}, \quad (9)$$

where we extended the operators of regular expressions to intersection.

Now in order to simultaneously satisfy rules **R1** and **R2**, in formula (??), the first occurrence of  $m$  must precede the one of  $m-1$ , for  $2 \leq m \leq n-1$ . These positions are exactly the sequence  $(f_j)_{0 < j < n}$  defined in Section ???. Given these positions and considering the correspondent sequence  $(b_j)_{0 < j < n}$  we obtain the regular expression:

$$\left( \prod_{j=1}^{n-1} (0 + \dots + (j-1))^{b_{j-1} j} \right) (0 + \dots + (n-1))^{b_{n-1}},$$

and we must consider the possible values of  $(b_j)_{0 < j < n}$ , constrained to **G1** and **G2**:

$$\sum_{b_1=1}^k \sum_{b_2=1}^{2k-b_1} \dots \sum_{b_{n-1}=1}^{k(n-1) - \sum_{l=1}^{n-2} b_l} \left( \prod_{j=1}^{n-1} (0 + \dots + (j-1))^{b_{j-1} j} \right) (0 + \dots + (n-1))^{b_{n-1}}$$

For  $n = 3$  and  $k = 2$  we have

$$(01 + 1(0+1))((0+1)2 + 2(0+1+2))(0+1+2)^2 + 12(0+1+2)^4,$$

and the number of these strings is  $(1+2)((2+3)3^2) + 3^4 = 216$ .

For each sequence  $(b_j)_{0 < j < n}$  the number of strings  $[(S_i)_{i < kn}]$  with  $S_i \in [n-1]_0$  and satisfying **R1** and **R2** is

$$\prod_{j=1}^n j^{b_{j-1}}, \quad (10)$$

a direct consequence of rules **G3**, **G4** and **G5**. And then we must take the sums over all  $b_j$  constrained to rules **G1** and **G2**.

**Theorem 2.** *We have*

$$B_k(n) = \sum_{b_1=1}^k \sum_{b_2=1}^{2k-b_1} \sum_{b_3=1}^{3k-b_1-b_2} \cdots \sum_{b_{n-1}=1}^{k(n-1)-\sum_{i=1}^{n-2} b_i} \prod_{j=1}^n j^{b_j-1}. \quad (11)$$

*Proof.* It is an immediate consequence of rules **G1** to **G5**. □

**Corollary 1.** *The number of non-isomorphic IC DFA's with  $n$  states over an alphabet of  $k$  symbols is  $2^n B_k(n)$ .*

*Proof.* By Theorems ?? and ?? and considering the possible sets of final states. □

## 6 Conclusion

The method described in Section ?? was implemented and used to generate all IC DFA $_{\emptyset}$ 's for  $k = 2$  and  $n < 10$ , and  $k = 3$  and  $n < 7$ . The time complexity of the program is linear in the number of automata and took about a week to generate all the referred IC DFA $_{\emptyset}$ 's, in a PPC G4 1.5MHz.

One of the advantage of this method is that only the allowed strings are computed so it is not a *generate-and-test* algorithm and because automata are generated in lexicographic order it is easy to generate them as needed for consumption by another algorithm.

If an IC DFA with  $n$  states accepts a finite language then there exists a topological order of its states such that  $\delta(i, \sigma) > i$ , for all  $i < n - 1$  and  $\sigma \in \Sigma$ . But the order we used for string representations is not a topological order. So we can not determine directly from the string if the accepted language is finite, as was done by Domaratzki [?] only for finite languages. Although the formula  $B_k(n)$  is quite similar to the one obtained in [?] for an upper bound of the number of finite languages, the meaning of the parameters ( $b_j$ ) are not directly related.

## 7 Acknowledgements

We thank the anonymous referees for their comments that helped to improve this paper.

## References

- [BN] Frédérique Bassino and Cyril Nicaud. Enumeration of complete accessible deterministic automata over a 2-letter alphabet. Submitted.
- [CP05] J.-M. Champarnaud and T. Paranthoën. Random generation of DFAs. *Theoretical Computer Science*, 330(2):221–235, 2005.
- [DKS02] Michael Domaratzki, Derek Kisman, and Jeffrey Shallit. On the number of distinct languages accepted by finite automata with  $n$  states. *Journal of Automata, Languages and Combinatorics*, 7(4):469–486, 2002.
- [Dom04] Michael Domaratzki. Combinatorial interpretations of a generalization of the Genocchi numbers. *Journal of Integer Sequences*, 7(04.3.6), 2004.



- [fad] FAdo: tools for formal languages manipulation. <http://www.ncc.up.pt/fado>.
- [HMU00] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 2000.
- [HP67] F. Harary and E. M. Palmer. Enumeration of finite automata. *Information and Control*, 10:499–508, 1967.
- [Lis69] Valery Liskovets. The number of inittially connected automata. *Kibernetika*, 3:16–19, 1969. (in Russian; Engl. transl: *Cybernetics*, 4 (1969), 259-262).
- [Lis03] Valery Liskovets. Exact enumeration of acyclic automata. In *Proc. 15th Conf. "Formal Power Series and Algebr. Combin. (FPSAC'03)*, 2003.
- [MR05a] Nelma Moreira and Rogério Reis. Interactive manipulation of regular objects with FAdo. In *Proceedings of 2005 Innovation and Technology in Computer Science Education (ITiCSE 2005)*. ACM, 2005.
- [MR05b] Nelma Moreira and Rogério Reis. On the density of languages representing finite set partitions. *Journal of Integer Sequences*, 8(05.2.8), 2005.
- [Nic99] Cyril Nicaud. Average state complexity of operations on unary automata. In M. Kurylowski, L. Pacholski, and T. Wierzbicki, editors, *Proc. 24th Symposium, Mathematical Foundations of Computer Science*, volume 1672 of *Lecture Notes on Computer Science*, pages 231–240. Springer-Verlag, 1999.
- [Par04] T. Paranthoën. *Génération aléatoire et structure des automates à états finis*. PhD thesis, Université de Rouen, 2004.
- [Rob85] R. W. Robinson. Counting strongly connected finite automata. In *Graph Theory with Applications to Algorithms and Computer Science*, pages 671–685. Wiley, 1985.
- [Slo03] N.J.A. Sloane. The On-line Encyclopedia of Integer Sequences, 2003. <http://www.research.att.com/~njas/sequences>.