On the Representation of Finite Automata

Marco Almeida Nelma Moreira Rogério Reis

Technical Report Series: DCC-2005-04
Version 1.4

Departamento de Ciência de Computadores – Faculdade de Ciências

&

Laboratório de Inteligência Artificial e Ciência de Computadores

Universidade do Porto

Rua do Campo Alegre, 823 4150 Porto, Portugal

Tel: +351+2+6078830 – Fax: +351+2+6003654

http://www.ncc.up.pt/fcup/DCC/Pubs/treports.html

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 [MR05a, fad] 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 Harrison [Har65], Robinson [Rob85], Harary and Palmer [HP67] and Liskovets [Lis69] amongst many others. A survey may be found in Domaratzki et al. [DKS02]. More recently, several authors examined related problems. Domaratzki et al. [DKS02] studied the enumeration of distinct languages accepted by finite automata with n states; Nicaud [Nic99], Champarnaud and Paranthoën [CP05, Par04] and Bassino and Nicaud [BN] analysed several aspects of the average behaviour of regular languages; Liskovets [Lis03] and Domaratzki [Dom04] 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 3 describes a string representation for deterministic finite automata that is unique up to isomorphism for initially connected deterministic finite automata. Section 4 presents an efficient method to generate those strings. Section 5 shows how their enumeration provides an upper bound and the exact value for the number of ICDFA's. Section 6 and Appendix A report some implementation issues and final remarks.

2 Preliminaries

We first recall some basic notions from automata theory and formal languages, that can be found in standard books [HMU00]. An alphabet Σ is a nonempty set of symbols. A string over Σ is a finite sequence of symbols of Σ. The empty string is denoted by ϵ. The set $\Sigma^\star$ is the set of all strings over Σ. A language L is a subset of $\Sigma^\star$. The density of a language L over Σ, $\rho_L(n)$, is the number of strings of length n that are in L, i.e., $\rho_L(n) = |L \cap \Sigma^n|$. A regular expression (r.e.) α over Σ represents a language $L(\alpha) \subseteq \Sigma^\star$ and is inductively defined by: $\emptyset$, ϵ and $\sigma \in \Sigma$ are a r.e., where $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ and $L(\sigma) = \{\sigma\}$; if α_1 and α_2 are r.e., $(\alpha_1 + \alpha_2)$, $(\alpha_1\alpha_2)$ and $\alpha_1^\star$ 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 $L(\alpha_1{}^\star) = L(\alpha_1)^\star$. 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, Σ is the alphabet, $\delta : Q \times \Sigma \to Q$ 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 δ extends naturally to $\Sigma^\star$: 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*[1] (ICDFA) if for each state $q \in Q$ there exists a string $x \in \Sigma^\star$ 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 $f : Q \to Q'$ 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 $L(\mathcal{A}) = \{x \in \Sigma^\star \mid \delta(q_0, x) \in F\}$. 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 of size n over an alphabet of k symbols , so we disregard the set of final states and we consider only a quadruple (Q, Σ, δ, 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 $[n]_0 = \{0, 1, \ldots, n\}$ and $[n] = \{1, \ldots, n\}$.

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 the $\text{DFA}_\emptyset$ in Fig.1 we have:

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

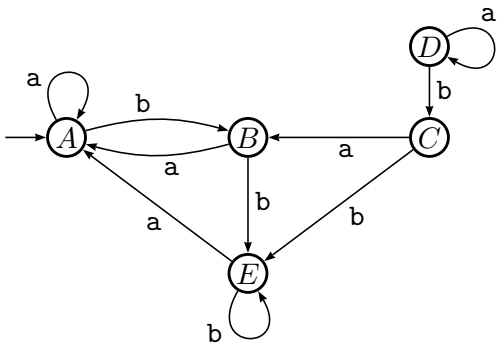


Figure 1: A DFA with no final states marked

Given a complete $\text{DFA}_\emptyset$ (Q, Σ, δ, q_0) with $|Q| = n$ and $|\Sigma| = k$ and considering a total order over Σ, the representation can be simplified by omitting the alphabetic symbols. For

[1] Also called *accessible*.

our example, we would have

$$[[A\ (A,B)],[B\ (A,E)],[C\ (B,E)],[D\ (D,C)],[E\ (A,E)]]. \tag{2}$$

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 $\mathrm{DFA}_\emptyset$'s we can drop the inner tuples as well. We obtain

$$[0,1,0,4,1,4,3,2,0,4]. \tag{3}$$

Because this representation depends on the order we label the states, we have more than one representation for each $\mathrm{DFA}_\emptyset$. Can we have a canonical order for the set of the states? Let the first state be the initial state q_0 of the automaton, the second state the first one to be referred (excepting q_0) by a transition from q_0, the third state the next referred in transitions from one of the first two states, and so on... For the $\mathrm{DFA}_\emptyset$ 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:

$$[0,1,0,2,0,2,3,4,1,2] \text{ and } [0,1,0,2,0,2,1,2,4,3]. \tag{4}$$

If we restrict this representation to $\mathrm{ICDFA}_\emptyset$'s, then this representation is unique and defines an order over the set of its states. In the example, the $\mathrm{DFA}_\emptyset$ 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 $\sigma_0 < \sigma_1 < \cdots < \sigma_{k-1}$. Given an $\mathrm{ICDFA}_\emptyset$ (Q, Σ, δ, q_0) with $|Q| = n$, the representing string is of the form $[(S_i)_{i<kn}]$ with $S_i \in [n-1]_0$ and $S_i = \delta(\lfloor i/k \rfloor, \sigma_{i \bmod k})$. In Figure 2, we present an algorithm for obtain these string representation.

```
1  uniqueStr {
2    S = []
3    Ord(q_0) = 0
4    i = j = 0
5    while i ≤ j :
6      for l in [k − 1]_0:
7        if Ord(δ(Ord^{-1}(i),σ_l)) not defined then
8          j = j + 1
9          Ord(δ(Ord^{-1}(i),σ_l)) = j
10       S = S + [Ord(δ(Ord^{-1}(i),σ_l))]
11     i = i + 1
12   return S
13 }
```

Figure 2: Obtaining the string representation of an $\mathrm{ICDFA}_\emptyset$.

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

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

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

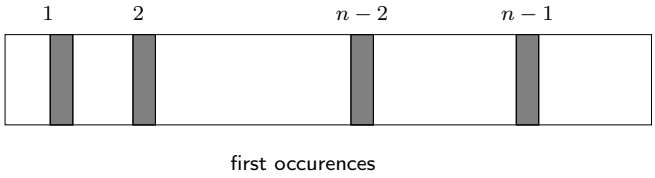


Figure 3: **R1** states that first refferences to each state occur sequencially.

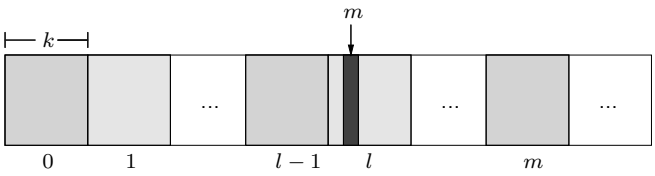


Figure 4: **R2** ensures that before the appearance of the set of transitions from a given state at least a reference to that state must appear in the string.

Proof. (of Lemma 1) 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. □

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* ICDFA$_\emptyset$ *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 deterministic complete finite automaton structure. 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 σ, $\delta(m', \sigma) = m$. 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 1 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

$$\begin{aligned} S_j &= \delta(\lfloor j/k \rfloor, \sigma_{j \bmod k}) \\ &= S_l \\ &= \delta(\lfloor l/k \rfloor, \sigma_{l \bmod k}) \\ &= S'_l \\ &= S'_j. \end{aligned}$$

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

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

□

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* ICDFA$_\emptyset$*'s with n states, over an alphabet Σ of size k.*

Proof. Let (Q, Σ, δ, q_0) and $(Q', \Sigma, \delta', q'_0)$ be two ICDFA$_\emptyset$'s and $[(S_i)_{i<kn}]$ and $[(S'_i)_{i<kn}]$ their representing strings. By Lemma 1, these strings satisfy **R1** and **R2**. Suppose that $f : Q \longrightarrow Q'$ is an isomorphism between the ICDFA$_\emptyset$'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** $S'_j = \max\{S_i \mid i < j\} + 1 = S_j$.

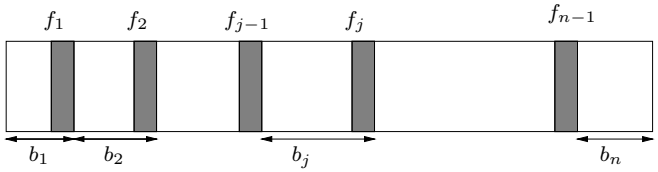
Conversely, by Lemma 2, we have that each string represents a ICDFA$_\emptyset$ up to a compatible renaming of states, i.e., if two ICDFA$_\emptyset$'s are represented by the same string, that representation defines a isomorphism between them. □

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

4 Generating automata

Normal representations for ICDFA$_\emptyset$'s (as presented above) can be used as compact computer representations for that kind of objects, but even though rules **R1** and **R2** are quit simple, it is not evident how to write an enumerative algorithm in an efficient way. In a string representing an ICDFA$_\emptyset$ 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

$$\begin{aligned} b_1 &= f_1 + 1, && (5)\\ b_j &= f_j - f_{j-1} \text{ , for } 2 \leq j \leq n-1 && (6)\\ b_n &= kn - f_{n-1}. && (7) \end{aligned}$$



Note that

$$\sum_{l=1}^{j} b_l = f_j + 1 \text{ , for } j \in [n-1]. \quad (8)$$

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 ICDFA's

In this section we obtain a formula $B_k(n)$ for the number of strings $[(S_i)_{i<kn}]$ representing ICDFA$_\emptyset$'s with n states over an alphabet of k symbols. Although it is already known a formula for the number of non-isomorphic ICDFA$_\emptyset$'s, we think that our method is new. Liskovets [Lis69] and, independently, Robinson [Rob85] gave for that number the formula $H_k(n) = \frac{h_k(n)}{(n-1)!}$ where $h_k(1) = 1$ and for $n > 1$

$$h_k(n) = n^{kn} - \sum_{1 \leq j < n} \binom{n-1}{j-1} n^{k(n-j)} h_k(j) \quad (9)$$

Note that n^{kn} is the number of transition functions, from which we subtract the number of them that have $n-1$, $n-2$,...,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 $c > 0$,

$$A_c = L(0^\star + \sum_{i=1}^{c-1} 0^\star \prod_{j=1}^{i} j(0 + \cdots + j)^\star). \tag{10}$$

These languages belong to a family of languages L_c presented by Moreira and Reis [MR05b] and that represent partitions of $[n]$ with no more than $c \geq 1$ parts, i.e.,

$$L_c = L(\sum_{i=1}^{c} \prod_{j=1}^{i} j(1 + \cdots + j)^\star). \tag{11}$$

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 [BN] 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

$$\bigwedge_{m=1}^{n-1} \bigvee_{j=0}^{km-1} S_j = m. \tag{12}$$

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

$$\bigcap_{m=1}^{n-1} \sum_{j=0}^{km-2} (0 + \cdots + (m-1))^j m (0 + \cdots + (n-1))^{kn-j-1}, \tag{13}$$

where we extended the operators of regular expressions to intersection.

Now in order to simultaneously satisfy rules **R1** and **R2**, in formula (13), the first occurence 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 4. Given these positions and considering the correspondent sequence $(b_j)_{0<j<n}$ we obtain the regular expression:

$$\left(\prod_{j=1}^{n-1} (0 + \ldots + (j-1))^{b_j - 1} j \right) (0 + \cdots + (n-1))^{b_n - 1},$$

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

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

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

$$\prod_{j=1}^{n} j^{b_j - 1}, \tag{14}$$

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*

$$B_k(n) = \sum_{b_1=1}^{k} \sum_{b_2=1}^{2k-b_1} \sum_{b_3=1}^{3k-b_1-b_2} \cdots \sum_{b_{n-1}=1}^{k(n-1)-\sum_{l=1}^{n-2} b_l} \prod_{j=1}^{n} j^{b_j - 1}. \tag{15}$$

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

The above formula can also be rewritten using the sequence $(f_i)_{0<i<n}$, and considering $f_n = kn$:

$$B_k(n) = \sum_{f_1=0}^{k-1} \sum_{f_2=f_1+1}^{2k-1} \sum_{f_3=f_2+1}^{3k-1} \cdots \sum_{f_{n-1}=f_{n-2}+1}^{k(n-1)-1} \prod_{i=2}^{n} i^{f_i - f_{i-1} - 1}.$$

Corollary 1. *The number of non-isomorphic* ICDFA*'s with n states over an alphabet of k symbols is*

$$2^n B_k(n). \tag{16}$$

Proof. By Theorems 1 and 2 and considering the possible sets of final states. □

6 Conclusion

The method described in Section 4 was implemented in `Python` [pyt] and used to generate all ICDFA$_\emptyset$'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 ICDFA$_\emptyset$'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.

The formula $B_k(n)$ was also implemented and its values where computed for $k = 1..10$ and $n = 1..9$. Those values are presented in Appendix A. The sequences $B_2(n)$ and $B_3(n)$ appear in Sloane [Slo03] as **A082165** and **A065756**, respectively. If an ICDFA 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 [Dom04] only for finite languages. Although the formula $B_k(n)$ is quite similar to the one obtained in [Dom04] for an upper bound of the number of finite languages, the meaning of the parameters (b_j) are not directely related.

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.

[Har65] M. A. Harrison. A census of finite automata. *Canad. J. Math.*, 17:100–113, 1965.

[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.

[pyt] Python language website. http://python.org.

[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.

A Experimental Results

In this appendix we present the number of ICDFA's non-isomorphic without final states for $n = 1..9$ states and $k = 2..10$ alphabetic symbols.

k= 2 n	
1	1
2	12
3	216
4	5248
5	160675
6	5931540
7	256182290
8	12665445248
9	705068085303

k= 3 n	
1	1
2	56
3	7965
4	2128064
5	914929500
6	576689214816
7	500750172337212
8	572879126392178688
9	835007874759393878655

k=4 n	
1	1
2	240
3	243000
4	642959360
5	3508208993750
6	34253071111894176
7	544271118689873008532
8	13147735690099619023732736
9	458677874292647947600097994111

k=5 n	
1	1
2	992
3	6903873
4	175483321344
5	11826519415721875
6	1744085190146957291232
7	494949686355427145872161111
8	246491144450280856073240885624832
9	200977948941552280610264305518977871090

k=6 n	
1	1
2	4032
3	190505196
4	46086910722048
5	38056697263376203125
6	8412194318600644571322 4896
7	42311779474985218950200641090 5462
8	43107988409138813783150335301212915630 08
9	81510780531114326278646228956855976801744959908

k=7 n	
1	1
2	16256
3	5192233245
4	11921614605697024
5	1203158945418522832812 50
6	39760630290347678869359335109 12
7	35352134880615199574345580083298157131 4
8	73484638707005629827978811367001966356732051456
9	32134987099884609628834726023582411808822980002131697574

k=8 n	
1	1
2	65280
3	140764942800
4	3065045074098257920
5	37774648436758551936718 7500
6	18646311089801204325486161799337 2672
7	29279032751153335518638081828541936916513 4504
8	12405178593678541407417860030685556146529447406647377 92
9	12533845162122187320986901745839566315023480777415952875118142242

k=9 n	
1	1
2	261632
3	3807455329593
4	786050986901533097984
5	11826944437401392213967597656 25
6	87174774177655261106696069206610619540 48
7	24166320989316602931123570944929684848900715003 8885
8	20862781312540752296309668431262192459252081308963680368459776
9	48685620547821011542400089049693743352890406293621927191606374683842353 31

k=10 n	
1	1
2	1047552
3	102881965757076
4	201378988990926052917248
5	36987713763758090743237756542968 75
6	40705662003140936498269017579631064087700 7872
7	19919542529963785985915910443133372795968790579034086 0554
8	35035077358953741660493447152751013683551167125420054867666470227148 8
9	18880963360320663330992680074514720259464695005177220879245815882004727092412348 33