Forward Injective Finite Automata: Exact and Random Generation of Nonisomorphic NFAs

Miguel Ferreira, Nelma Moreira(✉), and Rogério Reis

CMUP and DCC, Faculdade de Ciências da Universidade do Porto,
Rua do Campo Alegre, 1021, 4169-007 Porto, Portugal
miguelferreira108@gmail.com, {nam,rvr}@dcc.fc.up.pt

Abstract. We define the class of forward injective finite automata (FIFA) and study some of their properties. Each FIFA has a unique canonical representation up to isomorphism. Using this representation an enumeration is given and an efficient uniform random generator is presented. We provide a conversion algorithm from a nondeterministic finite automaton or regular expression into an equivalent FIFA. Finally, we present some experimental results comparing the size of FIFA with other automata.

1 Introduction

The study of the average-case complexity of determinisation of nondeterministic finite automata (NFAs) is an important research topic. In most algorithms that use NFAs, it is needed, in a way or another, to convert them to equivalent deterministic finite automata (DFAs), and that leads to an exponential blow up, in the worst-case. In practice, the feasibility of algorithms and representations, dealing with regular languages and related systems, depends primarily on their complexity on the average case, rather than the worst-case scenario which may rarely occur. The average-case analysis of operations in automata is, in general, a difficult task. One approach to this problem is to consider uniformly distributed random representations and to perform statistically significant experiments requiring most of the times nonisomorphic sampled automata. There are several uniform random generators available for nonisomorphic DFAs [1,3,4,10]. For all these generators, DFAs are considered initially connected and it is possible to order their states in a canonical way to ensure that no two different isomorphic automata are generated. However for NFAs, the problem seems unfeasible in general as for n-state NFAs the size of the automorphism group can be $n!$, and this is polynomially equivalent to testing if two NFAs are isomorphic.

Authors partially funded by CMUP (UID/MAT/00144/2013), which is funded by FCT (Portugal) with national (MCTES) and European structural funds through the programs FEDER, under the partnership agreement PT2020.

S. Konstantinidis and G. Pighizzini (Eds.): DCFS 2018, LNCS 10952, pp. 88–100, 2018.
https://doi.org/10.1007/978-3-319-94631-3_8

Recently, Héam and Joly [6] presented uniform random generators for some classes of NFAs up to isomorphism, using Monte Carlo Markov-Chains modified by the Metropolis-Hastings algorithm, to deal with isomorphic objects [8]. This involves the computation of the sizes of automorphism groups of NFAs. However, the performance of these algorithms does not seem to allow the generation of large sets of objects of acceptable size. Considering classes of NFAs for which this latter problem is polynomial, the authors obtain (randomised/heuristic) uniform random generators in polynomial time. These classes include trim NFAs with a single initial state and with states with a fixed maximal output degree.

In this paper, we follow a different path and study a class of NFAs for which it is possible the exact and uniform random generation of nonisomorphic elements. More precisely, we consider a class of initially-connected NFAs (with a single initial state) for which it is possible to define a canonical order over the set of states. Starting with the initial state, whenever new states are reached from an already visited state, we ensure that the set of transition labels are pairwise distinct. Based on a natural order of the power set of the alphabet, an ordering for the set of states is obtained. This induces a unique canonical representation for the nonisomorphic NFAs for which it is possible to obtain such an order, and which are named *forward injective finite automata* (FIFA). We also developed an algorithm that converts an NFA into an equivalent FIFA and performed some experimental tests in order to compare their relative sizes. The class of FIFA represents all regular languages, FIFA sampling is proven to be efficient to implement, and experimental results suggest that, on average, for each NFA one can find a FIFA which is not much larger. These properties convinced the authors that FIFA is a class which deserves to be studied. In particular, FIFA based sampling is a possible alternative for average complexity analysis of algorithms that have NFAs as input. The paper is organised as follows. In the next section, some notation is given and we review the canonical string representation for initially-connected DFAs. In Sect. 3, we present the new class of automata (FIFA), a canonical representation that is unique for nonisomorphic automata of this class and give an enumeration of FIFAs by states and alphabetic size. A uniform random generator for FIFA is given in Sect. 4. In Sect. 5 we adapted the determinisation to the conversion from NFAs to FIFAs, together with some experimental results in Sect. 6. Some future work is discussed in Sect. 7.

2 Preliminaries

Given two integers, m and n, let $[m, n]$ be the set $\{i \in \mathbb{Z} \mid m \leq i \ \wedge \ i \leq n\}$, and let one use the additional interval notation with its standard meaning.

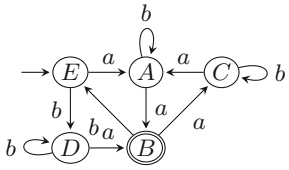
An NFA A is a tuple $\langle Q, \Sigma, \delta, q_0, F\rangle$ where Q is a finite set of states, Σ the alphabet, *i.e.*, a nonempty finite set of symbols, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, q_0 the initial state and $F \subseteq Q$ the set of final states. As usual, let $\delta(q) = \{p \mid \exists \sigma \in \Sigma,\ p \in \delta(q, \sigma)\}$. We also consider $\ell : Q \times Q \to 2^\Sigma$, such that $\ell(p, q) = \{\sigma \mid q \in \delta(p, \sigma) \wedge \sigma \in \Sigma\}$, for any $p, q \in Q$, and for $\emptyset \neq S \subseteq \Sigma$, let $\ell_p^{-1}(S) = \{q \mid \ell(p, q) = S\}$. The function ℓ allows us

to consider the amalgamation of all transitions with the same start and end states, and the value of ℓ as label. In the rest of the paper, we will consider either δ or ℓ to refer to the transition function of a given NFA. The *size* of an NFA is its number of states, $|Q| = n$. An NFA is *initially connected* (or *accessible*) if for each state $q \in Q$ there exists a sequence $(q'_i)_{i\in[0,j]}$ of states and a sequence $(\sigma_i)_{i\in[0,j-1]}$ of symbols, for some $j < |Q|$, such that $q'_{m+1} \in \delta(q'_m, \sigma_m)$ for $m \in [0, j[$, $q'_0 = q_0$ and $q'_j = q$. The transition function $\delta : Q \times \Sigma \to 2^Q$ extends naturally to $\Sigma^\star$ and to sets of states. The language accepted by A is $\mathcal{L}(A) = \{\, w \in \Sigma^\star \mid \delta(q_0, w) \cap F \neq \emptyset \,\}$. Two automata are *equivalent* if they accept the same language. Two NFAs $\langle Q, \Sigma, \delta, q_0, F\rangle$ and $\langle Q', \Sigma, \delta', q'_0, F'\rangle$ are *isomorphic* if there is a bijection $\iota : Q \to Q'$ such that $\iota(q_0) = q'_0$, for all $\sigma \in \Sigma$ and $q \in Q$, $\iota(\delta(q, \sigma)) = \delta'(\iota(q), \sigma)$, and $\iota(F) = F'$, where we naturally extend ι to sets $S \subseteq Q$. In the following, we will identify an alphabet Σ of size k with $[0, k[$ and the power set 2^Σ with $[0, 2^k[$ using a natural order for the subsets of Σ, with 0 corresponding to the emptyset and $j \in [0, 2^k[$ corresponds to a set $J \subseteq [0, k[$ if and only if $i \in J$ then the i-th bit of the binary representation of j is 1. If two automata A and B are isomorphic, we write $A \simeq B$. A *semiautomaton* (Q, Σ, δ, q_0) denotes an NFA without its final state information and is referred to as an $\text{NFA}_\emptyset$. Each semiautomaton, if $|Q| = n$, will be shared by 2^n NFAs. An NFA is *deterministic* (DFA) if $|\delta(q, \sigma)| \leq 1$, for all $(q, \sigma) \in Q \times \Sigma$ and is *complete* if the equality always holds. An initially-connected (complete) DFA is denoted by ICDFA and the corresponding semiautomaton by $\text{ICDFA}_\emptyset$.

We review here the canonical string representation for nonisomorphic ICDFAs presented by Almeida *et al.* [1]. Given an ICDFA $A = \langle Q, [0, k[, \delta, q_0, F\rangle$, a canonical order for Q can be obtained by a breath-first traversal of A, starting with the initial state q_0, and in each state considering the transitions in the natural order of Σ. Let $\varphi : Q \to [0, n[$ be the renaming of Q elements induced by that order. If we, by abuse of language, identify q and $\varphi(q)$, for $q \in Q$, a unique representation of the semiautomaton of A is $S(A) = \langle \delta(\lfloor i/k \rfloor, i \bmod k)\rangle_{i\in[0,kn[}$.

A canonical representation for A is obtained by adding a sequence $(b_i)_{i\in[0,n[}$ of n-bits such that $b_i = 1$ if state i is final, and $b_i = 0$ otherwise.

Example 1. Consider the ICDFA, $A = \langle\{A, B, C, D, E\}, \{a, b\}, \delta, E, \{B\}\rangle$, $a < b$, and φ the states renaming according to the induced order.



φ	A	B	C	D	E
	1	3	4	2	0

The canonical string $S(A)$ for the corresponding $\text{ICDFA}_\emptyset$ is $\underbrace{12}_{E}\,\underbrace{31}_{A}\,\underbrace{32}_{D}\,\underbrace{40}_{B}\,\underbrace{14}_{C}$.

Adding the information pertaining to the final states as a sequence of bits, the canonical string for A is 123132401400010.

The characterisation of the canonical strings allows the enumeration, and the exact and random generation, as well as an optimal coding for ICDFAs, as defined by Lothaire [9]. Inspired by this model, we defined FIFA as NFAs having a unique breadth-first traversal order property according to the natural order of 2^{Σ}.

3 Forward Injective Finite Automata

Even if one only considers initially-connected NFAs, with a single initial state, it is not possible to extend the previous representation to NFAs because it is not possible to induce an order for the set of states from the transition function. Starting with the initial state, whenever unvisited states are reached from an already visited state, we must ensure that the set of transition labels from the current state to the newly reached states are pairwise distinct. We denote NFAs with this property as *forward injective finite automata* (FIFA).

Definition 1 *Let $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ be an initially-connected NFA and Π the bijection from 2^{Σ} to $[0, 2^k[$, induced by the order on Σ. Consider the breadth-first search traversal of A, induced by the natural order on 2^{Σ}, that starts in the initial state q_0. For each state $s \in Q$, let $S(s)$ be the set of states that are image of a transition starting from a state already visited by the BFS. The labels for the transitions departing from s to any state in $\delta(s) \setminus S(s)$ need to be unique. The automaton A is a* forward injective finite automaton *if it holds that:*

$$(\forall p, q \in \delta(s) \setminus S(s))(p \neq q \Rightarrow \ell(s,p) \neq \ell(s,q)). \tag{1}$$

This class of automata is expressive enough to recognise all regular languages, because deterministic automata trivially satisfy (1). However, not all (initially-connected) NFAs are FIFAs. Some experimental results suggest that for alphabets, of size at least 2, one can find a FIFA equivalent to an NFA that is not much larger than the NFA. These facts and the existence of a unique canonical representation show the interest in studying this class of NFAs. In particular, because of the existence of uniform random generators, one can use this model to obtain estimates of average performance of algorithms that manipulate NFAs.

3.1 A Canonical State Order for FIFAs

Given a FIFA it is possible to obtain a canonical state order φ through a breadth first traversal starting in the initial state and ordering the newly reached states according to the total order defined in 2^{Σ}. For that, one can disregard the set of final states and consider the semiautomaton $\text{FIFA}_{\emptyset}$. The canonical state order for a $\text{FIFA}_{\emptyset}$ can be computed through Algorithm 1, where Π is the bijection from 2^{Σ} to $[0, 2^k[$, *sorted* is a function that sorts integers in increasing order, and $\varphi : Q \to [0, n[$ is the computed bijection. Note that (at line 7) $\ell^{-1}_{\varphi^{-1}(s)}(\Pi^{-1}(j))$ is a single value by the injectivity of ℓ restricted to the newly seen states in a FIFA.

Algorithm 1. FIFA state order algorithm

```
1: procedure StateOrder(FIFA_∅ ⟨Q, Σ, δ, q_0⟩)
2:     φ(q_0) ← 0
3:     i ← 0; s ← 0
4:     do
5:         M ← sorted{Π(S) | ∅ ≠ S = ℓ(φ^{-1}(s), q) ∧ q ∈ Q \ φ^{-1}([0, i])}
6:         for j ∈ M do
7:             φ(ℓ^{-1}_{φ^{-1}(s)}(Π^{-1}(j))) ← i + 1
8:             i ← i + 1
9:         s ← s + 1
10:    while s < i
       return φ
```

Proposition 1 *Let $A = \langle Q, \Sigma, \delta, q_0\rangle$ be a FIFA$_\emptyset$ with n states and $k = |\Sigma|$, there is a bijection $\varphi : Q \to [0, n[$ that defines an isomorphism between A and $\langle [0, n[, \Sigma, \delta', 0\rangle$ with $\delta'(i, \sigma) = \{\varphi(s) \mid s \in \delta(\varphi^{-1}(i), \sigma)\}$, for $i \in [0, n[$ and $\sigma \in \Sigma$.*

Throughout the paper we will now consider FIFA$_\emptyset$ to have its states in their canonical order: $A = \langle [0, n[, \Sigma, \delta, 0\rangle$.

3.2 Canonical String Representation

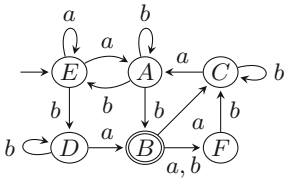
Let $A = \langle [0, n[, \Sigma, \delta, 0, F\rangle$ be a FIFA such that $\langle [0, n[, \Sigma, \delta, 0\rangle$ is a FIFA$_\emptyset$. We can represent A by the canonical representation of its FIFA$_\emptyset$ concatenated with the bitmap of the state finalities. The canonical representation of a FIFA$_\emptyset$ is defined as follows.

Definition 2 *Given a FIFA$_\emptyset$ $\langle [0, n[, \Sigma, \delta, 0\rangle$ with $|\Sigma| = k$, its canonical representation is a sequence $(r_i)_{i\in[0,n[}$ such that for each state i,*

$$r_i = s_{i,1}s_{i,2}\dots s_{i,m_i}u_{i,1}\dots u_{i,\overline{m}_i},$$

*and where m_i is the number of previously seen states, $\overline{m}_i$ is the number of newly seen states, $s_{i,j} = \Pi(\ell(i, j-1))$ for $j \in [1, m_i]$, and $u_{i,j} = \Pi(\ell(i, m_i + j - 1))$ for $j \in [1, \overline{m}_i]$. This means that, for each state i, $s_{i,j}$ correspond to the sets of transitions to states already seen (*back transitions*) and $u_{i,j}$ correspond to the sets of transitions to newly seen states from state i (*forward transitions*).*

Example 2. Consider the following FIFA on the left.



φ	A	B	C	D	E	F
	1	3	4	2	0	5

Let $\Pi(\emptyset) = 0$, $\Pi(\{a\}) = 1$, $\Pi(\{b\}) = 2$ and $\Pi(\{a, b\}) = 3$. The state renaming according to the induced order on the states is given above. The canonical string $(r_i)_{i\in[0,5]}$ for the corresponding $\text{FIFA}_\emptyset$ is

$$\underbrace{[1][1,2]}_{E}\underbrace{[2,2,0][2]}_{A}\underbrace{[0,0,2,1][]}_{D}\underbrace{[0,0,0,0][1,3]}_{B}\underbrace{[0,1,0,0,2,0][]}_{C}\underbrace{[0,0,0,0,2,0][]}_{F},$$

where the transitions for each state are as indicated. The FIFA can be represented by its semiautomaton canonical string with the state finalities appended. Thus, this FIFA canonical string is

$$[1][1,2][2,2,0][2][0,0,2,1][][0,0,0,0][1,3][0,1,0,0,2,0][][0,0,0,0,2,0][][0,0,0,1,0,0].$$

Lemma 3. *Let $A = \langle [0, n[, \Sigma, \delta, 0\rangle$ be a $\text{FIFA}_\emptyset$ with $k = |\Sigma|$. Let $(r_i)_{i\in[0,n[}$ with $r_i = s_{i,1}s_{i,2}\cdots s_{i,m_i}u_{i,1}u_{i,2}\cdots u_{i,\overline{m}_i}$ be the canonical representation for A as given above. Then the following rules are satisfied:*

$$s_{i,j} \in [0, 2^k[, \qquad \forall i \in [0, n[, \forall j \in [1, m_i], \tag{F1}$$

$$u_{i,j} \in [1, 2^k[, \qquad \forall i \in [0, n[, \forall j \in [1, \overline{m}_i], \tag{F2}$$

$$j < l \Rightarrow u_{i,j} < u_{i,l}, \qquad \forall i \in [0, n[, \forall j, l \in [1, \overline{m}_i], \tag{F3}$$

$$m_0 = 1, \tag{F4}$$

$$m_i = m_{i-1} + \overline{m}_{i-1}, \qquad \forall i \in [1, n[, \tag{F5}$$

$$i < m_i \leq n, \qquad \forall i \in [1, n[, \tag{F6}$$

$$\overline{m}_{n-1} = 0. \tag{F7}$$

Proof. The rule (F1) describes how transitions to previously seen states are represented, i.e. $s_{i,j} = \Pi(\ell(i, j-1))$, possibly with $s_{i,j} = 0$. The rule (F2) considers the transitions from state i to states $t_{i,j} = m_i + j - 1$ visited for the first time in i, which implies that $\ell(i, t_{i,j}) \neq \emptyset$. Consequently, $u_{i,j} \in [1, 2^k[$. For representation purposes, rule (F3) states that the set of states visited for the first time is represented with its transitions sorted in ascending order. This is a representation choice that ensures that all $u_{i,j}$ are distinct. Rule (F4) is obvious as one starts at state 0 and thus 0 is the only seen state. Rule (F5) is a direct consequence of the definition of m_i in (2), and implies that $m_i = 1 + \sum_{j=0}^{i-1} \overline{m}_j$ for $i \in [1, n[$. Rules (F6) and (F7) ensures that all states are seen, and that the $\text{FIFA}_\emptyset$ is initially connected. It is a consequence of the definition of φ and also ensures that a state must be seen before its representation is given. Rule (F6) implies that $m_{n-1} = n$. By contradiction, suppose that there is i such that $m_i \leq i$, for $i \in [1, n[$. Then, there exist $1 \leq j \leq i$ such that j is not accessible from 0 in paths that use states only in $[0, i[$. But that contradicts φ definition. □

Lemma 4. *Every string $(r_i)_{i\in[0,n[}$ satisfying rules (F1)–(F7) represents a $\text{FIFA}_\emptyset$ with states $[0, n[$ over an alphabet of k symbols.*

Proof. There is at least one transition reaching each state in $[1, n[$ and there is at least one transition from the state 0 to state 1. The transition function is defined

by $\ell(i, j-1) = \Pi^{-1}(s_{i,j})$ for $i \in [0, n[$ and $j \in [1, m_i]$, and $\ell(i, m_i + j - 1) = \Pi^{-1}(u_{i,j})$ for $i \in [0, n[$ and $j \in [1, \overline{m}_i]$. The proof that the FIFA$_\emptyset$ is initially connected is analogous to the one in Lemma 3. □

From these lemmas the following theorem holds.

Theorem 5. *For each $n > 0$ and $k > 0$, there is a one-to-one mapping from sequences $(r_i)_{i\in[0,n[}$ satisfying rules* (F1)–(F7) and nonisomorphic FIFA$_\emptyset$s with n states over an alphabet of k symbols.

The canonical form for FIFA$_\emptyset$ is not a simple extension of the one for ICDFA$_\emptyset$s reviewed in Sect. 2 for several reasons. One needs to consider instead of the alphabet its power set, there are no restrictions for transitions to already seen states, and transitions to newly seen states must have different labels. However, the first occurrences of each state satisfy exactly the same rules (over an alphabet of 2^k symbols) observed in the canonical representation of ICDFA$_\emptyset$s. This will be made evident in the next section.

3.3 Counting FIFAs

Our aim is to enumerate (exactly generate) and count all the nonisomorphic FIFA$_\emptyset$ with n states and k symbols. This will also allow us to obtain a uniform random generator for the class of FIFAs. Let $\Psi : [0, n[\times[1, 2^k[\to [0, n(2^k - 1)[$, be defined by $\Psi(i, j) = i(2^k - 1) + j - 1$. The mapping Ψ is a bijection with $\Psi^{-1}(p) = \left(\lfloor p/(2^k - 1)\rfloor, (p \bmod (2^k - 1)) + 1\right)$. Let $(r_i)_{i\in[0,n[}$ be a sequence satisfying rules (F1)–(F7), thus, representing a FIFA$_\emptyset$. Let us denote by *flag of a state* $t \in [1, n[$ the pair $(i, u_{i,j})$, occurring in state i, such that $t = m_i + j - 1$ (and $\ell(i, t) = \Pi^{-1}(u_{i,j})$). According to (F3), if in a state i two different flags $(i, u_{i,j})$ and $(i, u_{i,l})$ occur, we know that $j < l \Rightarrow u_{i,j} < u_{i,l}$. For the sake of readability, given $t \in [1, n[$, we denote by (i_t, u_t) its flag, and let $\Psi(i_t, u_t) = f_t$. Then, by (F3), one has

$$(\forall t \in [2, n[)(i_t = i_{t-1} \Rightarrow u_t > u_{t-1}) \vee (i_t > i_{t-1}),$$

which implies

$$(\forall t \in [2, n[)(f_t > f_{t-1}), \tag{G1}$$

$$(\forall t \in [1, n[)(f_t < t(2^k - 1)). \tag{G2}$$

Rules (G1)–(G2) are satisfied by the positions of the first occurrence of a state in the canonical strings for ICDFA$_\emptyset$s, considering k instead of $2^k - 1$ in rule (G2). The following theorem computes the number of allowed sequences of flags.

Proposition 2 (Theorem 6 of [1]) *Given $k > 0$ and $n > 0$, the number of sequences $(f_t)_{t\in[1,n[}$, $F_{2^k-1,n}$, is given by:*

$$F_{2^k-1,n} = \sum_{f_1=0}^{2^k-1-1} \sum_{f_2=f_1+1}^{2(2^k-1)-1} \cdots \sum_{f_{n-1}=f_{n-2}+1}^{(2^k-1)(n-1)-1} 1 = C_n^{(2^k-1)},$$

where $C_n^{(2^k-1)} = \binom{n(2^k-1)}{n} \frac{1}{(2^k-2)n+1}$ are the generalized Fuss-Catalan numbers.

Example 6. For the FIFA of Example 2, $((0,1),(0,2),(1,2),(3,1),(3,3))$ is the sequence of flags and $(f_t)_{t\in[1,5]} = (0,1,4,9,11)$.

Given a sequence of flags $(f_t)_{t\in[1,n[}$, the set of possible canonical strings that represent FIFA$_\emptyset$s can be easily enumerated: each state i has unconstrained transitions for states already seen (m_i) and has the transitions to new states given by the flags occurring in its description (forward transitions).

The number of canonical strings with a given sequence of flags is given by

$$\prod_{i\in[0,n[} (2^k)^{m_i}. \tag{2}$$

Theorem 7. *The total number of FIFA$_\emptyset$s with n states over a k-ary alphabet is*

$$b_{k,n} = \sum_{f_1=0}^{2^k-1-1} \sum_{f_2=f_1+1}^{2(2^k-1)-1} \cdots \sum_{f_{n-1}=f_{n-2}+1}^{(2^k-1)(n-1)-1} \prod_{i\in[0,n[} (2^k)^{m_i},$$

where $m_i = 1 + \sum_{j=0}^{i-1} \overline{m}_j$ *and* $\overline{m}_j = |\{f_t \mid i_t = j\}|$ *for* $i \in [0,n[$ *and* $j \in [1,n[$.

This can be adapted for the exact generation/enumeration of all canonical representations. Each FIFA$_\emptyset$ corresponds to a number between 1 and $b_{k,n}$. In Table 1 we present the values of $b_{k,n}$ for $n \in [2,7]$ and $k \in [2,3]$. An equivalent recursive definition for $b_{k,n}$ is given in the next section for uniform random generation.

Table 1. Values of $b_{k,n}$

n	$k=2$	$k=3$
2	192	3584
3	86016	56885248
4	321912832	32236950781952
5	10382009696256	738091318939425439744
6	3073719939819896832	733871593861464877408622477312
7	871581830440515993285427 2	326867227491799792314941447869937011916 80

Corollary 8. *The number of nonisomorphic FIFAs with n states and k alphabetic symbols is $B_{k,n} = b_{k,n}2^n$.*

4 Uniform Random Generation

The canonical representation for FIFAs allows an easy uniform random generation for this class of automata. Given the number of flags occurring in a prefix of a canonical string we count the number of valid suffixes. To count the number of automata with a given prefix a recursive counting formula for FIFA$_\emptyset$ is needed.

With these partial values, we can reconstruct any $\text{FIFA}_\emptyset$ by knowing its number, which varies from 1 to $b_{k,n}$. The process of uniform randomly generating a FIFA consists, thus, in four steps: creation of a table with partial counts for each prefix; uniformly sample a number between 1 and $b_{k,n}$; construct the $\text{FIFA}_\emptyset$ representation using the table; random generation of values from 0 to $2^n - 1$ for the state finalities and return the FIFA. Let m be the number of already seen states for the state i of a canonical string of a $\text{FIFA}_\emptyset$. We count the number $N_{m,i}$ of $\text{FIFA}_\emptyset$s for each value of m and i. This gives us the following recursive formula for fixed n and k:

$$\begin{aligned} &N_{m,i} = (2^k)^m \sum_{j=0}^{n-m} \binom{2^k-1}{j} N_{m+j,i+1}, && m \in [1,n], i \in [0,m[, \\ &N_{m,i} = 0, && m \in [1,n], i \notin [0,m[, \\ &N_{n,n} = 1. \end{aligned}$$

Proposition 3 *$b_{k,n} = N_{1,0}$, for all $k \geq 1$ and $n \geq 1$.*

Proof. Immediate consequence of the canonical representation and Theorem 7. □

Algorithm 2. Random $\text{FIFA}_\emptyset$ algorithm.

```
1: procedure RANDOMFIFA(n, k)
2:     r ← Random(0, N_{1,0} − 1)                          ▷ number of the FIFA_∅
3:     m_0 ← 1
4:     for q ∈ [0, n[ do                                   ▷ reconstruct FIFA_∅ flags
5:         m̄_q ← 0
6:         while N_{m_q + m̄_q, q} ≤ r do
7:             m̄_q ← m̄_q + 1
8:         m_{q+1} ← m_q + m̄_q
9:     b ← r mod (2^k)^{Σ_{i=0}^{n-1} m_i}                 ▷ number representing back transitions
10:    f ← r/(2^k)^{Σ_{i=0}^{n-1} m_i}                     ▷ number representing forward transitions
11:    for q ∈ [0, n[ do
12:        for p ∈ [1, m_q] do                             ▷ reconstruct back transitions
13:            s_{q,p} ← b mod 2^k
14:            b ← b/2^k
15:        if m̄_q ≠ 0 then
16:            c ← C(2^k − 1, m̄_q)
17:            t ← f mod c
18:            f ← f/c
19:            for p ∈ [1, m̄_q] do
20:                u_{q,p} ← t mod (2^k − 1)
21:                t ← ⌊t/(2^k − 1)⌋
       return (s_{i,1} s_{i,2} ··· s_{i,m_i} u_{i,1} u_{i,2} ··· u_{i,m_i})_{i∈[0,n[}
```

Proposition 4 *Algorithm 2 presents a uniform random generator for a $\text{FIFA}_\emptyset$ with n states and k symbols.*

Proof. (Sketch) Given an arbitrary integer r representing a $\text{FIFA}_\emptyset$ we determine the values $\overline{m}_i$ for each $i \in [0, n[$, using the precalculated table $N_{m,i}$. For each state i and m previously seen states, $\overline{m}_i$ is the first value such that $N_{m+\overline{m}_i,i} > r$, that is, $N_{m+\overline{m}_i-1,i} \leq r < N_{m+\overline{m}_i,i}$ (lines 4–8). Then, we count the number of available back transitions in each state i using m_i which can be determined by $\overline{m}_i$ (by F5). The total number of possible back transitions is $\sum_{i=0}^{n-1} m_i$. With this number (lines 9 and 10) one obtains the integers that represent all the $s_{i,j}$ and the $u_{i,j}$, respectively. For a given $i \in [0, n[$, in lines 12–14 we calculate $s_{i,j}$ for $j \in [1, m_i]$. If state i has forward transitions, their values are computed in lines 16–21. □

To obtain a random FIFA from the $\text{FIFA}_\emptyset$ we can generate a random number from $[0, 2^n[$ and reconstruct the state finalities according to the corresponding choice. Using dynamic programming techniques it is possible to generate a table indexed by the values of m and i ($N_{m,i}$) with time complexity $O(n^3 \log((2^k)^{n^2})) = O(n^5 k)$. The amount of memory used is $O(n^4 k)$, and this is a limiting factor for the dimension of the $\text{FIFA}_\emptyset$ being generated. This is justified by the huge number of $\text{FIFA}_\emptyset$s for a given n and k. For example, $b_{2,100}$ is greater than 10^{11531}. In Table 2 we present the execution times for the generation of 10000 FIFAs for $n \in \{1, 20, 30, 50, 75, 100\}$ and $k \in \{1, 2, 3, 4\}$, using Python 2.7 interpreted by Pypy, with a Intel Xeon CPU X5550 at 2.67 GHz. Comparing with some experiments presented by Héam and Joly [6, Table 1], these times correspond, approximately, to the generation of a single NFA.

Table 2. Execution times for the generation of 10000 random FIFA.

Times	$k = 1$	$k = 2$	$k = 3$	$k = 4$
$n = 10$	0.77 s	1.05 s	0.95 s	8.59 s
$n = 20$	1.06 s	2.33 s	3.13 s	3.96 s
$n = 30$	1.15 s	5.01 s	7.38 s	9.52 s
$n = 50$	2.84 s	16.86 s	26.64 s	40.43 s
$n = 75$	7.11 s	47.62 s	71.92 s	91.70 s
$n = 100$	15.86 s	100.25 s	156.24 s	202.41 s

5 Converting an NFA into a FIFA

In this section we discuss a process of converting an arbitrary NFA to a FIFA and the asymptotic time complexity bounds of such a procedure. The algorithm has an NFA as input and outputs an equivalent FIFA. It is based on the subset construction for NFA determinisation, with addition of an heuristic that attempts to create back transitions whenever possible. This gives us a FIFA that is not only forward injective but also forward deterministic. It may be also possible to add an heuristic for nondeterministic forward injective transitions or to have

other procedures that are not based on the subset construction. However this one had a good performance in our experiments.

Proposition 5 *The procedure* NfaToFifa *in Algorithm 3 computes a FIFA equivalent to a given NFA.*

This algorithm has time complexity $O(|\Sigma|2^{2|Q|})$, which is justified by $|Q'|$ having space complexity $O(2^{|Q|})$ due to the determinisation based algorithm. It is an open problem whether these bounds are tight for this algorithm.

Algorithm 3. An NFA to FIFA algorithm

```
1: procedure NfaToFifa(NFA ⟨Q, Σ, δ, I, F⟩)
2:     Q' ← {I}
3:     w ← {I}                                          ▷ states to be processed
4:     while w ≠ ∅ do
5:         S ← Pop(w)                                   ▷ popping S from w
6:         for σ ∈ Σ do
7:             P ← ∅
8:             δ'(S, σ) ← ∅
9:             for s ∈ S do
10:                P ← P ∪ δ(s, σ)
11:            for R ∈ sortedBySizeDesc(Q') do
12:                if R ⊆ P then                        ▷ nondeterministic back transitions
13:                    δ'(S, σ) ← δ'(S, σ) ∪ {R}
14:                    P ← P \ R
15:            if P ≠ ∅ then                            ▷ forward transition
16:                Q' ← Q' ∪ {P}
17:                w ← w ∪ {P}
18:                δ'(S, σ) ← δ'(S, σ) ∪ {P}
19:    return FIFA ⟨Q', Σ, δ', I, {S ∩ F ≠ ∅, S ∈ Q'}⟩
```

6 Experimental Results

The algorithm defined in the previous section was implemented within the FAdo package [5]. We performed some experiments to compare the sizes of the input and output automata. The input NFAs were obtained from uniform random generated regular expressions, for a fixed (standard) grammar, of a given syntactic tree size m over an alphabet of k symbols. The conversion method used was the partial derivative automata [2]. For each m and k, 10000 random regular expressions were generated to ensure a 95% confidence level within a 1% error margin [7, pp. 38–41]. For each sample, we calculated the minimal, the average and the maximum sizes of the obtained automata. For each partial derivative automaton (PD) we applied the algorithm NFAtoFIFA and obtained a FIFA (FIFA). We also computed the DFA obtained by determinisation of PD, by the usual subset construction, (DT), and the minimal DFA (MIN). Results for

$m \in \{50, 100, 250, 500\}$ and $k \in \{2, 3, 10\}$ are presented in Table 3. In general the FIFA computed is not much larger than the PD, although the determinised automata can be significantly larger.

Table 3. State complexities of automata where m, syntactic size of RE; k, size of alphabet; PD, size of partial derivative NFA; DT size of DFA from PD; MIN, size of minimal DFA; FIFA, size of FIFA from PD.

m, k	Type	min	avg	max
50, 2	FIFA	3	10.1684	25
	DT	3	10.1338	62
	MIN	1	5.0762	51
	PD	3	10.6904	19
100, 2	FIFA	3	19.2113	46
	DT	3	19.7193	158
	MIN	1	6.2814	116
	PD	9	20.0239	30
250, 2	FIFA	9	48.3731	107
	DT	12	185.6424	1120
	MIN	1	7.0256	630
	PD	34	47.998	66
500, 2	FIFA	35	99.8889	189
	DT	13	186.1518	6451
	MIN	1	6.8369	745
	PD	72	94.6422	124
50, 3	FIFA	3	12.4948	25
	DT	3	13.9339	56
	MIN	1	9.1225	41
	PD	4	11.6522	19
100, 3	FIFA	5	24.3173	45
	DT	5	34.289	166
	MIN	1	18.2094	148
	PD	11	21.8518	32
250, 3	FIFA	23	60.7792	110
	DT	12	185.6424	1586
	MIN	1	59.0185	988
	PD	34	52.5888	70
500, 3	FIFA	42	122.2247	198
	DT	37	1143.2134	11687
	MIN	1	92.8985	2343
	PD	79	103.6871	126
50, 10	FIFA	6	14.1252	24
	DT	7	15.3764	33
	MIN	1	13.8138	30
	PD	6	13.3556	21
100, 10	FIFA	14	27.4189	43
	DT	15	32.5608	66
	MIN	1	28.7841	58
	PD	13	25.2294	36
250, 10	FIFA	44	67.8454	99
	DT	55	102.6683	329
	MIN	1	88.8932	253
	PD	41	60.8428	78
500, 10	FIFA	102	135.1439	170
	DT	128	277.5053	1122
	MIN	1	237.4012	869
	PD	94	120.0465	150

7 Conclusions

We presented a class of initially-connected NFAs with a single initial state for which it is possible to test isomorphism in polynomial time. The definition of these automata (FIFA) is based on the possibility to order the set of states in a breath-first search traversal from the initial state. The uniform random generator can efficiently sample datasets to test the performance of algorithms dealing with NFAs. Moreover one can extend it with a parameter for the density of transitions in order to avoid the high frequency of NFAs recognising the universal language. The class of FIFAs can be further studied in its own. One can obtain asymptotic bounds for the number of FIFAs of given n and k. Upper bounds for the size of the minimal FIFAs for a given language will be of major interest. Operational state complexity with respect to this class of automata is also an open problem.

References

1. Almeida, M., Moreira, N., Reis, R.: Enumeration and generation with a string automata representation. Theor. Comput. Sci. **387**(2), 93–102 (2007)
2. Antimirov, V.M.: Partial derivatives of regular expressions and finite automaton constructions. Theor. Comput. Sci. **155**(2), 291–319 (1996)
3. Bassino, F., Nicaud, C.: Enumeration and random generation of accessible automata. Theor. Comput. Sci. **381**(1–3), 86–104 (2007)
4. Champarnaud, J.M., Paranthoën, T.: Random generation of DFAs. Theor. Comput. Sci. **330**(2), 221–235 (2005)

5. FAdo, P.: FAdo: tools for formal languages manipulation. http://fado.dcc.up.pt. Accessed 13 2018
6. Héam, P.-C., Joly, J.-L.: On the uniform random generation of non deterministic automata up to isomorphism. In: Drewes, F. (ed.) CIAA 2015. LNCS, vol. 9223, pp. 140–152. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22360-5_12
7. Knuth, D.E.: The Art of Computer Programming. Seminumerical Algorithms, vol. 2, 2nd edn. Addison Wesley, Boston (1981)
8. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov Chain and Mixing Times. American Mathematical Society (2008). http://pages.uoregon.edu/dlevin/MARKOV/markovmixing.pdf
9. Lothaire, M.: Applied Combinatorics on Words, vol. 105. Encyclopedia of Mathematics and its Applications. Cambridge University Press, New York (2005)
10. Nicaud, C.: Random deterministic automata. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014, Part I. LNCS, vol. 8634, pp. 5–23. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44522-8_2