KYBERNETIKA — VOLUME *57* (2021), NUMBER 5, PAGES 856–877

MULTI-ISLAND FINITE AUTOMATA AND THEIR EVEN COMPUTATION

Dušan Kolář, Alexander Meduna and Martin Tomko

This paper discusses n-island finite automata whose transition graphs can be expressed as n-member sequences of islands $i_1, i_2, \dots, i_n$, where there is a bridge leaving i_j and entering i_{j+1} for each $1 \leq j \leq n-1$. It concentrates its attention on even computation defined as any sequence of moves during which these automata make the same number of moves in each of the islands. Under the assumption that these automata work only in an evenly computational way, the paper proves its main result stating that n-island finite automata and Rosebrugh-Wood n-parallel right-linear grammars are equivalent. Then, making use of this main result, it demonstrates that under this assumption, the language family defined by n-island finite automata is properly contained in that defined by $(n+1)$-island finite automata for all $n \geq 1$. The paper also points out that this infinite hierarchy occurs between the family of regular languages and that of context-sensitive languages. Open questions are formulated in the conclusion.

Keywords: finite automata, graph-based decomposition, regulated computation, infinite hierarchies of language families

Classification: 68Q45, 68Q42, 68Q10

1. INTRODUCTION

Over its history, the theory of computation has always systematically and intensively investigated finite automata in terms of their structures in relation to the way they work (see [1, 2, 6, 10, 11, 12, 13]). In this paper, we continue with this vivid investigation trend by using finite automata to formalize and study what this paper refers to as *even computation*, which is informally sketched next.

To give an intuitive insight into what has inspired and motivated the present study, consider any algorithm A, which works as a strictly finitary unit without using any potentially infinite auxiliary memory, such as a stack. Assume that for a natural number n, A can be decomposed into n components, i_1 through i_n, in such a way that it performs these components one by one, starting from i_1 and proceeding toward the last component i_n. If this performance consists in carrying out the same number of computational steps within each of the n components, A works *evenly* within them, and this is the computational phenomenon formalized and investigated in this paper. As already stated, this formalization is based upon the well-known notion of a finite

DOI: 10.14736/kyb-2021-5-0856

automaton, which can represent A and its even computation strictly mathematically in terms of the transition graph of the automaton. To put it somewhat more specifically, in the graph, the n components are represented by subgraphs called *islands*, which underlie the formalization of *even computation* studied in the present paper as mathematically described next.

Let us rephrase the intuitive insight into even computation sketched above in terms of graph theory. For brevity, by graphs, we automatically mean unordered labeled oriented connected graphs throughout. Let G be a graph and H be a connected subgraph of G such that H contains k bridges of G, for some $k \geq 0$ (recall that a *bridge* is an edge of a graph whose deletion increases the number of *connected components* in the graph; a *connected component* is a maximal subgraph in which every two nodes are connected by a series of edges, regardless of their orientation). H is a *k-bridge island* if it is not properly contained in any other subgraph of G containing exactly k bridges. In other words, all edges of G adjacent to vertices of H are either in H, or bridges. These notions are illustrated in Figure 1. When applying the notion of islands to directed graphs, it makes sense to talk about a bridge *entering* or *leaving* an island, depending on whether the island contains its source node or its destination node. If G contains no bridge entering H, then H is a *starting island.* If G contains no bridge leaving H, then H is a *final island.*

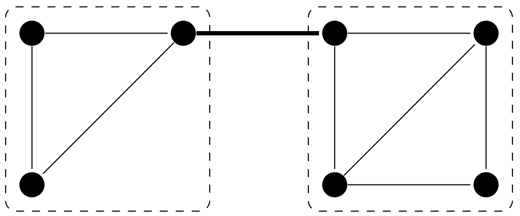

Fig. 1. The thick line in the middle of this diagram represents the only bridge of the graph—any other edge can be removed without making the graph disconnected. The two 0-bridge islands separated by this bridge are enclosed in dashed areas. The entire graph also forms a 1-bridge island.

In what follows, we automatically assume that finite automata have a single start state s and a single final state f; in addition, all their states are useful, and during a single move, they can read a string, possibly consisting of several symbols. (Of course, nondeterministic finite automata that satisfy these properties are as powerful as any finite automata that define the family of regular languages.) Let M be a finite automaton with transition graph G_M. Then, considering the properties and assumptions given above, we see that G_M can be expressed as a sequence of islands $i_1, i_2, \ldots, i_n$ for some $n \geq 1$ such that i_1 is the only starting island, which contains s, i_n is the only final island, which contains f, and there is a bridge b_j leaving i_j and entering i_{j+1} for all $1 \leq j \leq n-1$ (the case when $n = 1$ means that $G_M = i_1 = i_n$). M with G_M of this form is called an *n-island finite automaton*, and an *even computation* in this automaton is a sequence of moves during which M makes the same number of moves in each of its islands i_1

through i_n. As an example, consider the automaton M_0 in Figure 2. Understood as an ordinary finite automaton, it accepts the language $\{a\}^*\{bb\}^*\{c\}^*$. However, understood as a 3-island finite automaton (with the islands denoted by dashed areas in the diagram) and limited to even computations, it accepts the language $\{a^ib^ic^i \mid i = 2k, k \geq 0\}$.

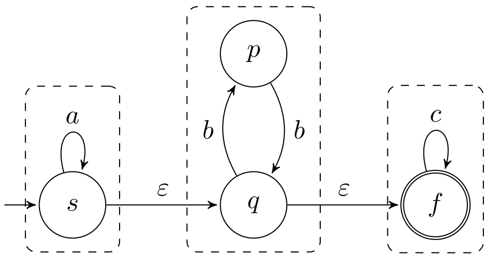

Fig. 2. The diagram of M_0.

We restrict our attention to n-island finite automata that only perform even computations. As the main result of the present paper, we demonstrate that they represent a new automata-theoretic counterpart for right-linear grammars of Rosebrugh and Wood (see [8]) because both of these language models are equivalent. Let us point out, from an utterly general viewpoint, that this new counterpart is based upon a purely graph-related restriction placed upon the way of computation, so it fundamentally differs from already existing counterparts of this kind, which are based upon a prescribed rule-regulation of pure pushdown and finite automata (see [5, 9], and a summary of these regulated automata in Chapter 15 in [7]).

Making use of the main result mentioned above, we also establish an infinite hierarchy of language families. More specifically, we prove that if the automata under consideration work only in an evenly computational way, then the language family defined by n-island finite automata is properly contained in that defined by $(n+1)$-island finite automata for all $n \geq 1$. To rephrase this result less formally, any increase in the number of islands in finite automata satisfying the above properties gives rise to an increase in their power. This hierarchy occurs between the family of regular languages and that of context-sensitive languages; it is worth pointing out that the former is defined by 1-island finite automata.

2. PRELIMINARIES

In this section, we formally define the basic notions that this paper builds on, starting with finite automata and following up with a few concepts related to directed graphs, so that we can combine these areas in the following section. For a more thorough introduction to the theory of formal languages, consult [3] or [4].

Finite automata

A *general finite automaton* (*GFA* for short) is a pentuple $M = (Q, \Sigma, R, s, f)$, where:

- Q is a finite nonempty set of *states*;
- Σ is a finite nonempty *input alphabet*, $Q \cap \Sigma = \emptyset$;
- $R \subseteq Q \times \Sigma^* \times Q$ is a finite set of *rules*; each $(p, w, q) \in R$ is written as $pw \to q$ in what follows;
- $s \in Q$ is the *start state*;
- $f \in Q$ is the *final state*.

M represents a *finite automaton* (*FA* for short) if $pw \to q \in R$ implies $\text{len}(w) \leq 1$.

Let $M = (Q, \Sigma, R, s, f)$ be a GFA. A *configuration* of M is any string in $Q\Sigma^*$. As special cases, sw is an *initial configuration* for any $w \in \Sigma^*$, and f is the unique *final configuration.*

Let px, qy be two configurations of M, where $p, q \in Q$, $x, y \in \Sigma^*$. For any rule $r \in R$, M makes a *move* from px to qy according to r, denoted as $px \vdash_M qy$ $[r]$, if and only if r is of the form $pw \to q$ for some $w \in \Sigma^*$ and $x = wy$. The subscript $_M$ is omitted whenever no confusion may arise. We denote the transitive and reflexive closure of $\vdash$ by $\vdash^*$.

The *language accepted by* M is denoted by $L(M)$ and defined as

$$L(M) = \{w \in \Sigma^* \mid sw \vdash^* f\}.$$

Languages accepted by (general) finite automata are called *regular languages*, and we denote the class of these languages by **REG**.

Let $M = (Q, \Sigma, R, s, f)$ be a GFA and let $\chi_1, \ldots, \chi_n$ be n configurations of M for some positive integer n. The sequence $c = \chi_1, \ldots, \chi_n$ is called a *successful computation of* M if and only if:

(1) $\chi_1 = sw$ for some $w \in \Sigma^*$;

(2) $\chi_i \vdash \chi_{i+1}$ for all $1 \leq i < n$;

(3) $\chi_n = f$.

Directed graphs

A *directed graph* (or simply a *graph*) is a pair $G = (V, E)$ where V is a finite set of *vertices* and $E \subseteq V \times V$ is a finite set of *edges.*

An *edge-labeled directed graph* is a directed graph $G = (V, E)$ along with an *edge labeling* $W : E \to \mathbb{L}$, which assigns to each edge of G a label from a predefined set $\mathbb{L}$ of labels.

Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. G' is a *subgraph* of G if and only if $V' \subseteq V$ and $E' \subseteq E$. Note that because G' is a graph, it must also hold that $E' \subseteq V' \times V'$. Furthermore, G' is a *proper subgraph* of G if $G' \neq G$. If G' is a subgraph of G and $E' = E \cap (V' \times V')$, we say that G' is an *induced subgraph* of G (induced by the set V' of vertices). A subgraph $G' = (V', E')$ of an edge-labeled directed graph is automatically understood to have its edge labeling's domain restricted to E'.

We now proceed towards the concept of a *bridge.*

Let $G = (V, E)$ be a directed graph and let $u_0, \ldots, u_n \in V$ be $n + 1$ of its vertices for some non-negative integer n. Then the sequence $p = u_0, \ldots, u_n$ is:

a) A *path from* u_0 *to* u_n in G (or simply a *path*), if $(u_i, u_{i+1}) \in E$ or $(u_{i+1}, u_i) \in E$ for $0 \le i < n$;

b) A *directed path from* u_0 *to* u_n in G, if $(u_i, u_{i+1}) \in E$ for $0 \le i < n$.

If p is a path, then for any $0 \le i < j \le n$ we say that u_i and u_j are *connected by a path.* A graph is *connected* if any two of its vertices are connected by a path.

Let $G = (V, E)$ be a connected directed graph. An edge $e \in E$ is a *bridge*, if $(V, E \setminus \{e\})$ is not connected. We denote by $\mathcal{B}(G)$ the set of all bridges in G. A subgraph $G' = (V', E')$ of G is a *bridgeless subgraph of* G if it contains no bridges of G (that is, $\mathcal{B}(G) \cap E' = \emptyset$). In other words, a bridge is an edge in a connected graph whose removal would cause the graph to cease being connected. If the graph is not connected to begin with, the concept of a bridge is irrelevant, so we do not consider any of its edges to be bridges.

Note that a bridgeless subgraph G' of G must not contain edges that are bridges in G, but there is no requirement that the edges of G' must not be bridges in G' itself.

3. DEFINITIONS

In what follows, by the notion of a graph, we automatically understand an edge-labeled directed graph. We now have all the necessary auxiliary definitions to formally define the notion of an *island.*

Definition 3.1. Let G be a graph. A *bridgeless island in* G is a connected bridgeless subgraph I of G that is not a proper subgraph of any other connected bridgeless subgraph of G.

Next, we prove that any vertex and any edge that is not a bridge is contained in exactly one bridgeless island of its graph.

Lemma 3.2. Let $G = (V, E)$ be a connected graph. For any $u \in V$, there is exactly one bridgeless island $I = (V_I, E_I)$ of G such that $u \in V_I$.

Proof. Let $u \in V$ be any vertex of G. The single-vertex graph $(\{u\}, \emptyset)$ is clearly a connected bridgeless subgraph of G, meaning it is either an island or contained in a properly larger connected bridgeless subgraph of G. Of all these subgraphs, at least one must necessarily be maximal with regards to the “is a subgraph of” relation, meaning it is an island. In conclusion, u is necessarily contained in at least one island of G.

Now let $I_1 = (V_1, E_1)$ and $I_2 = (V_2, E_2)$ be two islands of G such that $u \in V_1$ and $u \in V_2$. Let $V_3 = V_1 \cup V_2$ and $E_3 = E_1 \cup E_2$. As $(V_1 \times V_1) \cup (V_2 \times V_2) \subseteq (V_3 \times V_3)$, it necessarily holds that $E_3 \subseteq V_3 \times V_3$, meaning that $I_3 = (V_3, E_3)$ is also a graph.

As I_1 and I_2 are both bridgeless, I_3 is also bridgeless. Finally, I_1 and I_2 are both connected. Consider any two vertices $v, w \in V_3$. Clearly, as $u \in V_1 \cap V_2$, both v and w

are connected with u by a path, either in I_1 or in I_2. By combining these two paths, we get a path between v and w, meaning that I_3 is also connected.

Thus, I_1, I_2 and I_3 are all connected bridgeless subgraphs of G. I_1 and I_2 are also subgraphs of I_3, but they are also islands in G, meaning that they cannot be proper subgraphs of I_3, implying $I_1 = I_2 = I_3$. □

Lemma 3.3. Let $G = (V, E)$ be a connected graph. For any $e \in E \setminus \mathcal{B}(G)$[1], there is exactly one bridgeless island $I = (V_I, E_I)$ of G such that $e \in E_I$.

Proof. Let $e = (u, v) \in E \setminus \mathcal{B}(G)$ be any edge of G that is not a bridge. The graph $(\{u, v\}, \{(u, v)\})$ is clearly a connected bridgeless subgraph of G, meaning that it is an island of G or is contained in an island of G, which we can prove analogously to the proof of Lemma 3.2 and which leads to the conclusion that e is contained in at least one island of G.

Now let $I_1 = (V_1, E_1)$ and $I_2 = (V_2, E_2)$ be two islands of G such that $e \in E_1$ and $e \in E_2$. Note that it is necessarily the case that $u, v \in V_1 \cap V_2$. Also let $V_3 = V_1 \cup V_2$ and $E_3 = E_1 \cup E_2$. Analogously with the proof of Lemma 3.2, we can prove that $I_3 = (V_3, E_3)$ is a connected bridgeless subgraph of G, and as I_1 and I_2 are both subgraphs of I_3 and islands of G, it must hold that $I_1 = I_2 = I_3$. □

Islands in finite automata

To establish the connection between finite automata and graphs, let us first recall the notion of a *transition graph*:

Definition 3.4. Let $M = (Q, \Sigma, R, s, f)$ be a GFA and let $W : Q \times Q \to 2^{\Sigma^*}$ be a function defined for any $(p, q) \in Q \times Q$ as $W(p, q) = \{w \in \Sigma^* \mid (pw \to q) \in R\}$. The *transition graph* of M, denoted by G_M, is the edge-labeled directed graph $G_M = (Q, E)$ with edge-labeling $W_{|E}$, where $E = \{(p, q) \in Q \times Q \mid W(p, q) \neq \emptyset\}$ and the function $W_{|E}$ is W with its domain restricted to E.

If $G = (V, E)$ is the transition graph G_M for some GFA M, we can denote the island that contains the vertex $u \in V$ by I_u. Thus, we denote the island that contains the start state by I_s and the island that contains the final state by I_f.

For a GFA $M = (Q, \Sigma, R, s, f)$ and its corresponding transition graph G_M, we say that a state $q \in Q$ is *useful* if there exists a directed path p from s to f in the transition graph such that q occurs in p; otherwise, q is *useless*. Note that s and f are always useful unless $L(M) = \emptyset$ (if $L(M) = \emptyset$, M has no useful states). Also note that useful states are exactly those states that are used in some successful computation, meaning that we can remove the useless states and all associated edges without changing the accepted language.

To simplify a few statements about the islands of a GFA, let us introduce the notion of an *island graph*:

[1]Any $e \in \mathcal{B}(G)$ is a bridge, meaning it is not contained in any bridgeless island by definition.

Definition 3.5. Let $G = (V, E)$ be a directed graph, S be the set of its islands, and $\mathcal{B}(G)$ be the set of its bridges. For any node $u \in V$, $I_u \in S$ denotes the island of G containing u. The *island graph* of G is the graph $\mathcal{I}(G) = (S, \Gamma)$, where $\Gamma = \{(I_u, I_v) \mid (u, v) \in \mathcal{B}(G)\}$.

Furthermore, let $p = u_0, \dots, u_n$ be a path in G. then its *corresponding path in* $\mathcal{I}(G)$ is the unique path $\mathcal{I}(p) = I_0, \dots, I_m$, $1 \leq m \leq n$ which can be constructed as follows:

1. Let I_0 be the island containing u_0;
2. For each subsequent node u_i, if the island I containing u_i is different from the island containing u_{i-1}, append I to the end of the sequence constructed so far;
3. After u_n has been processed, the sequence is finished, clearly ending with the island I_m containing u_n.

Note that the definition of Γ encounters no duplicates as by definition there can be no more than a single bridge connecting any two islands. There is therefore a bijection $\psi : \mathcal{B}(G) \leftrightarrow \Gamma$ defined as $\psi(u, v) = (I_u, I_v)$.

It is easy to see that $\mathcal{I}(G)$ contains no cycles, because we would then be able to remove any edge on any cycle without making the resulting graph disconnected, which contradicts the fact that all edges of $\mathcal{I}(G)$ are bridges in G. For a connected graph G, $\mathcal{I}(G)$ is also connected, meaning it can be thought of as a tree, leading to an easy conclusion that if G has n bridges for some integer $n \geq 0$, it has $n + 1$ islands.

Theorem 3.6. Let $M = (Q, \Sigma, R, s, f)$ be a GFA with n bridges such that every state in Q is useful. Let G_M be the transition graph of M, and let $\mathcal{I}(G_M)$ be the corresponding island graph. Finally, let $w_1, w_2 \in L(M)$ be any two words accepted by M, and let the paths p_1 and p_2 represent in G_M the successful computations $sw_1 \vdash^* f$ and $sw_2 \vdash^* f$, respectively. Then the paths $\mathcal{I}(p_1)$ and $\mathcal{I}(p_2)$ in $\mathcal{I}(G_M)$ corresponding to p_1 and p_2, respectively, are equal, that is,

$$\mathcal{I}(p_1) = \mathcal{I}(p_2).$$

In other words, all islands of M can be ordered as $I_0, \dots, I_n$, such that every successful computation of M visits them exactly in this order and never returns to any island after once leaving it.

Proof. Let M be a GFA with no useless states. Recall that $\mathcal{I}(G_M)$ for a GFA M with no useless states can be thought of as a tree. In such a graph, there is a unique path between any two nodes. Any path corresponding to a successful computation of M necessarily starts in I_s and ends in I_f, and in between corresponds to the unique path between them. Furthermore, as M contains no useless states, every island of M occurs on some path from I_s to I_f, and as we have just established, there is exactly one such path, and all islands of M occur on it.

In conclusion, any two paths in G_M corresponding to a successful computation of M are represented in $\mathcal{I}(G_M)$ by the unique path $I_0, \dots, I_n$ connecting $I_s = I_0$ and $I_f = I_n$. This unique path defines the order $I_0, \dots, I_n$ of the islands of M. $\square$

Informally speaking, this theorem says that the island graph of the transition graph of any GFA with no useless states is of the following form:

$$I_0 \longrightarrow I_1 \longrightarrow \cdots \longrightarrow I_{n-1} \longrightarrow I_n.$$

This also means that we can always denote its islands as $I_0, \dots, I_n$, where n is the number of bridges.

Definition 3.7. Let $G = (V, E)$ be a connected graph and let k be a non-negative integer. A *k-bridge island* of G is a connected subgraph I of G that contains exactly k edges that are bridges in G and is not properly contained in any other such subgraph.

In other words, any connected subgraph H of G which contains I as a proper subgraph contains at least $k + 1$ bridges of G. When the number k of bridges can be omitted, we often refer to k-bridge islands simply as *islands.*

Note that for $k = 0$, this definition corresponds to Definition 3.1, so the term *bridgeless island* is synonymous with the term 0-*bridge island.* The following theorem demonstrates how k-bridge islands are formed from bridgeless islands.

Theorem 3.8. Let G be a connected graph with n bridgeless islands denoted by $I_1, \dots, I_n$, k be an integer satisfying $0 \le k < n$, I be a k-bridge island of G, and $I_j = (V_j, E_j)$ denote each bridgeless island. Then, statements I. and II., given next, hold true.

I. There is a set $S = \{j_0, \dots, j_k\}$ of $k + 1$ indices, $1 \le j_i \le n$ for each j_i, such that $I = (V, E)$, where $V = \bigcup_{j \in S} V_j$ and $E = (\bigcup_{j \in S} E_j) \cup (\mathcal{B}(G) \cap V \times V)$;

II. For any such set S of $k + 1$ indices, the graph I constructed in this way is either disconnected or a k-bridge island in G.

Less formally, a k-bridge island I in a graph G consists precisely of the union of $k+1$ bridgeless islands of G and the k bridges connecting them. Conversely, any such union is either disconnected or a k-bridge island.

Proof. We prove I. in detail. Concerning II., we only give a sketch of its proof.

Proof of I. (a rigorous version). We establish I. by complete induction on $k \ge 0$.

Basis. For $k = 0$, the theorem says that any bridgeless island of G consists of exactly one bridgeless island of G, which is obviously true: if the particular island is I_j for some j, simply set $S = \{j\}$. Conversely, an island constructed from a singleton set of indices $\{j\}$ just corresponds to the bridgeless island I_j.

Induction Hypothesis. Let I. hold for all $0, \dots, k - 1$, where k is a natural number.

Induction Step. Consider statement I. for k. Let I be a k-bridge island of G and let $b \in \mathcal{B}(G)$ be one of its k bridges. By removing b from the set of edges in I, we decompose it into two disjoint connected subgraphs J and K such that I is the union of J, K, and b.

As they are formed from a k-bridge island by removing one bridge, J and K contain m and l bridges of G, respectively, where m, l are integers satisfying $m + l + 1 = k$. It can be shown that J and K are an m-bridge island and an l-bridge island, respectively

– they are both connected and contain exactly the right number of bridges. It remains to show that neither is properly contained in another such subgraph of G.

Let us assume that there is an m-bridge island J' in G properly containing J. It follows that J' is necessarily connected and contains no bridges out of J, which also means that b is not in J'. Furthermore, J' and K are necessarily disjoint, because as they are both connected, J' and K having a non-empty intersection would imply the existence of a path connecting J and K without going through b, which would contradict the fact that b is a bridge in G. Therefore, the union of J', K and b is a connected subgraph of G containing exactly k bridges and properly containing I as a subgraph, which contradicts the presupposition that I is a k-bridge island. Therefore, there can be no such J', J is an m-bridge island, and the same argument can be used to prove that K is an l-bridge island.

By the induction hypothesis, J and K are composed of $m + 1$ and $l + 1$ bridgeless islands, respectively, along with the m and l bridges connecting them. As I is exactly the union of K, J and b, we can express I as the union of $m+l+2 = k+1$ bridgeless islands and $m + l + 1 = k$ bridges between them. These k bridges are also all the bridges of G between the nodes of K and J (so precisely $\mathcal{B}(G) \cap V \times V$, as the theorem requires), as no more than k bridges may connect $k+1$ bridgeless islands – the presence of more bridges would introduce loops, contradicting the assumption that they are bridges. Thus, the induction step for I. is completed.

Proof of II. (a sketch). Statement II. can also be proven using complete induction on $k \geq 0$ along with the already proven fact that statement I. holds. Let k be a positive integer, assume that statement II. holds for all $0, \dots, k-1$, and let $S = \{j_0, \dots, j_k\}$ be a set of $k + 1$ distinct indices satisfying $1 \leq j_i \leq n$ for each $0 \leq i \leq k$. Let us construct $I = (V, E)$ with S as a basis as described in the theorem. If I is disconnected, we are done. Otherwise, I is connected, and we know that the number of bridges in I is k, as more bridges would lead to loops and fewer bridges would not suffice to connect all of the $k + 1$ disjoint bridgeless islands. It remains to show the maximality of I, because if it is not properly contained in a k-bridge island, it is itself a k-bridge island.

Let I' be a k-bridge island containing I. Then, as follows from statement I., I' must be constructed as a union of $k + 1$ bridgeless islands specified by some set S' of indices. However, as I' also contains I, it must contain all of the bridgeless islands specified by S, leading to the conclusion that $S \subseteq S'$, which along with the fact that $\operatorname{card}(S') = \operatorname{card}(S) = k + 1$ implies $S' = S$. Finally, because I and I' are fully specified by S and S', respectively, it must also be the case that $I' = I$. □

We have already seen that any graph can be described as a union of bridgeless islands and the bridges connecting them (in Lemmas 3.2 and 3.3), and we have seen the structure of such a decomposition (in Theorem 3.6). Similar decompositions of a graph G can be made into the more general k-bridge islands (for varying k), usually in more than one way. What this comes down to is deciding which bridgeless islands to merge into larger islands. This corresponds to dividing the bridges in $\mathcal{B}(G)$ into two groups: bridges which merge their adjacent bridgeless islands into larger islands, and bridges which remain as bridges between the larger islands in the final decomposition. Therefore, for a graph G

with n bridges, given k such that $0 \leq k \leq n$, we can decompose G into $k+1$ islands in $\binom{n}{k}$ different ways.

Using such a decomposition might appear like an attempt to diverge from structures offered by bridgeless islands alone, but in fact, it is just a convenient shorthand – for any k-bridge island I found in the transition graph of a particular finite automaton M, we can slightly modify M to get an equivalent automaton containing a bridgeless island corresponding to I. As an example, consider any bridge (u, v) along with a rule $uw \to v$ in M. If we simply add a new state q and rules $u \to q$, $qw \to v$, the language accepted by the automaton remains the same, but (u, v) is no longer a bridge and the former two bridgeless islands connected by (u, v) are merged into a new bridgeless island in the new automaton. Similar actions can be repeated as necessary to merge larger groups of islands.

n-island finite automata

Let us now finally apply all the notions we have introduced to the formal definition of an n-island GFA:

Definition 3.9. Let $[n] = \{1, 2, \ldots, n\}$ be the set of all positive integers up to and including n and let $M = (Q, \Sigma, R, s, f)$ be a GFA. We say that M is an *n-island general finite automaton* (or *n-IGFA* for short) if and only if $Q = \bigcup_{i=1}^{n} Q_i$ for some family of pairwise disjoint sets $Q_1, \ldots, Q_n$ called *islands* of M such that:

1. For each $i \in [n]$, the island Q_i contains:
    - An *entry state* $s_i \in Q_i$,
    - An *exit state* $f_i \in Q_i$;
2. For each $pw \to q \in R$, exactly one of the following holds:
    - Either $p, q \in Q_i$ for some $i \in [n]$, in which case the rule is an *internal rule* of island i,
    - Or $p = f_i \in Q_i, q = s_{i+1} \in Q_{i+1}$ for some $i \in [n]$, $i \neq n$, in which case the rule is a *bridge rule*;
3. $s = s_1 \in Q_1$;
4. $f = f_n \in Q_n$.

For each $i \in [n]$ set $R_i = R \cap (Q_i \times \Sigma^* \times Q_i)$, which is the set of internal rules of island i, and set $R_b = R \cap \bigcup_{i=1}^{n-1}(\{s_i\} \times \Sigma^* \times \{f_{i+1}\})$, which is the set of bridge rules of M. Clearly $R = (\bigcup_{i=1}^{n} R_i) \cup R_b$. Also set $\Gamma = \{(f_i, s_{i+1}) \mid 1 \leq i < n\}$, which is the *set of bridges of M*.

Furthermore, if M is a finite automaton, we say that M is an *n-island finite automaton* (or *n-IFA* for short).

In the transition graph G_M of the automaton, it is the case that $\Gamma \subseteq \mathcal{B}(G_M)$, meaning that each pair of states in Γ is a bridge in G_M, Therefore, $Q_1, \ldots, Q_n$ are islands in G_M, as they are necessarily composed of the remaining bridgeless islands and the bridges connecting them. The edges (f_i, s_{i+1}) connecting neighboring islands are called bridges, as they are bridges in the transition graph of the GFA. However, this graph may also contain bridges within the individual islands.

Note that while M is required to contain the island structure as described by the sets $Q_1, \ldots, Q_n$, the structure of M itself does not specify its islands or bridges in any way. These must therefore be defined by some additional structure, such as the set $\Gamma \subseteq \mathcal{B}(G_M)$ of selected bridges. The sets $Q_1, \ldots, Q_n$ along with their entry and exit states are fully defined by specifying the automaton M along with the set of bridges Γ.

Furthermore, note that for each $i \in [n]$, $M_i = (Q_i, \Sigma, R_i, s_i, f_i)$ is a GFA, which we can also refer to as the ith island of M.

An n-IGFA is just a special case of a GFA, so the previously defined notions of a configuration, move, and accepted language still apply. However, we can introduce some extensions of these notions:

Definition 3.10. Let n be a positive integer, $M = (Q, \Sigma, R, s, f)$ an n-IGFA, $\Gamma \subseteq \mathcal{B}(G_M)$ be its set of bridges, and let $p, q \in Q$ be any two states of M, $x, y \in \Sigma^*$ any two strings over Σ, and $r \in R$ be a rule of M such that $px \vdash qy\ [r]$.

If $r \in R_i$ for some $i \in [n]$, we can say that M makes a *move in island i with regard to* Γ, denoted as $px\ {}^{\Gamma}_{i}\!\vdash qy\ [r]$, and if $r \in R_b$, we can say that M makes a *bridge move with regard to* Γ, denoted as $px\ {}^{\Gamma}_{b}\!\vdash qy\ [r]$. We can omit the left superscript Γ when the set of bridges used is clear from the context.

For both of these relations, the upper index ^{k} for some non-negative integer k denotes their kth power, and the upper index * denotes their transitive and reflexive closure.

Using these notions, we now introduce a new kind of computation based on a restriction not possible in an ordinary GFA.

Definition 3.11. Let n be a positive integer, $M = (Q, \Sigma, R, s, f)$ an n-IGFA with its islands specified by the set Γ of selected bridges, and let$w \in \Sigma^*$ be any string over Σ. A computation $sw \vdash^* f$ of M is *even with regard to* Γ, denoted as $sw\ {}^{\Gamma}_{e}\!\vdash^* f$, if and only if there is a non-negative integer k and for each $i \in [n]$ configurations $\varsigma_i, \varphi_i \in Q_i\Sigma^*$ such that:

(i) $\varsigma_i\ {}^{\Gamma}_{i}\!\vdash^k \varphi_i$ for each $i \in [n]$,

(ii) $\varphi_i\ {}^{\Gamma}_{b}\!\vdash \varsigma_{i+1}$ for each $i \in [n]$, $i \neq n$,

(iii) $\varsigma_1 = sw$,

(iv) $\varphi_n = f$.

The *language accepted by M by even computations with regard to* Γ is denoted by $L_e(M, \Gamma)$ and defined as

$$L_e(M, \Gamma) = \{w \in \Sigma^* \mid sw\ {}^{\Gamma}_{e}\!\vdash^* f\}.$$

In other words, a computation is even if the same number of moves is performed by M in each island.

For any positive integer n, we denote the class of all n-IGFA and the class of all n-IFA by $\mathbf{GFA}_n$ and $\mathbf{FA}_n$, respectively. We also denote the classes of languages accepted by these automata as $\mathcal{L}(\mathbf{GFA}_n) = \{L(M) \mid M \in \mathbf{GFA}_n\}$ and $\mathcal{L}(\mathbf{FA}_n) = \{L(M) \mid M \in \mathbf{FA}_n\}$. Finally, we denote the classes of languages accepted by these automata by even computations by $\mathcal{L}_e(\mathbf{GFA}_n) = \{L_e(M, \Gamma) \mid M \in \mathbf{GFA}_n, \Gamma \subseteq \mathcal{B}(G_M)\}$ and $\mathcal{L}_e(\mathbf{FA}_n) = \{L_e(M, \Gamma) \mid M \in \mathbf{FA}_n, \Gamma \subseteq \mathcal{B}(G_M)\}$.

Example 3.12. Consider $M_1 = (\{s, q, f\}, \{a, b, c\}, R, s, f)$, where R consists of the rules

$$\begin{aligned} sa &\rightarrow s, \\ s &\rightarrow q, \\ qb &\rightarrow q, \\ q &\rightarrow f, \\ fc &\rightarrow f, \end{aligned}$$

and set $\Gamma = \{(s, q), (q, f)\}$, which as a set of selected bridges corresponds to the islands $Q_1 = \{s\}, Q_2 = \{q\}, Q_3 = \{f\}$. See Figure 3 for a diagram of this automaton, with the individual bridges enclosed in dashed rectangles.

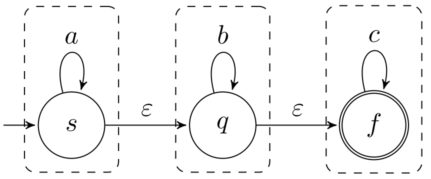

Fig. 3. The diagram of M_1.

Clearly, $L(M_1) = \{a\}^*\{b\}^*\{c\}^* \in \mathbf{REG}$. Considering the set Γ of selected bridges, M_1 is a 3-IFA, with each of the states $s = s_1 = f_1$, $q = s_2 = f_2$, $f = s_3 = f_3$ being both the entry and exit state of their respective islands. Consider the following even computation with input $aabbcc$, with the current state underlined in each configuration for clarity:

$$\begin{aligned} \underline{s}aabbcc \ {}_1{\vdash}\ & \underline{s}abbcc \\ {}_1{\vdash}\ & \underline{s}bbcc \\ {}_b{\vdash}\ & \underline{q}bbcc \\ {}_2{\vdash}\ & \underline{q}bcc \\ {}_2{\vdash}\ & \underline{q}cc \\ {}_b{\vdash}\ & \underline{f}cc \\ {}_3{\vdash}\ & \underline{f}c \\ {}_3{\vdash}\ & \underline{f} \end{aligned}$$

As the computation performs exactly 2 steps in each island, we can say that

$$saabbcc \, {}_e^{\Gamma}{\vdash} f,$$

meaning $aabbcc \in L_e(M_1, \Gamma)$. Observe that

$$L_e(M_1, \Gamma) = \{a^n b^n c^n \mid n \geq 0\},$$

which is a context-sensitive language, but not a context-free language.

4. RESULTS

In this section, we establish the main results of this paper. First, we prove $\mathbf{REG} = \mathcal{L}(\mathbf{GFA}_n) = \mathcal{L}(\mathbf{FA}_n)$ for all $n \geq 1$. Then, we demonstrate $\mathcal{L}_e(\mathbf{GFA}_n) \subset \mathcal{L}_e(\mathbf{GFA}_{n+1})$ for all $n \geq 1$.

Theorem 4.1. $\mathbf{REG} = \mathcal{L}(\mathbf{GFA}_n)$ for any $n \in \mathbb{N}$, $n \geq 1$.

Proof. As an n-IGFA is just a special case of a GFA, clearly $\mathcal{L}(\mathbf{GFA}_n) \subseteq \mathbf{REG}$ for any $n \in \mathbb{N}$, $n \geq 1$.

Let $L \in \mathbf{REG}$ be a regular language over an alphabet Σ. If $L = \emptyset$, it is trivial to construct an n-IGFA accepting L. Let $L \neq \emptyset$ and let $M = (Q, \Sigma, R, s, f)$ be a GFA with no useless states such that $L(M) = L$. As M is definitely a 1-IGFA, clearly $\mathbf{REG} \subseteq \mathcal{L}(\mathbf{GFA}_1)$. Let $n \geq 2$ and $Q' = \{q_2, \dots, q_n\}$ be a set of $n-1$ new states such that $Q' \cap Q = Q' \cap \Sigma = \emptyset$. We can introduce new rules $R' = \{f \to q_2\} \cup \{q_i \to q_{i+1} \mid 2 \leq i < n\}$. Clearly, $M' = (Q \cup Q', \Sigma, R \cup R', s, q_n)$ is an n-IGFA with regard to $\{(f, q_2)\} \cup \{(q_i, q_{i+1}) \mid 2 \leq i < n\}$ as the set of selected bridges and $L(M') = L(M) = L$. Therefore, $\mathbf{REG} \subseteq \mathcal{L}(\mathbf{GFA}_n)$, so $\mathbf{REG} = \mathcal{L}(\mathbf{GFA}_n)$ for any $n \in \mathbb{N}$, $n \geq 1$. $\square$

Rather than proving the analogous result for ordinary finite automata separately, we now show their relation to general finite automata, from which this analogous result follows.

Theorem 4.2. $\mathcal{L}(\mathbf{FA}_n) = \mathcal{L}(\mathbf{GFA}_n)$ and $\mathcal{L}_e(\mathbf{FA}_n) = \mathcal{L}_e(\mathbf{GFA}_n)$ for any $n \in \mathbb{N}$, $n \geq 1$.

Proof. As an n-FA is just a special case of an n-GFA, clearly it holds that $\mathcal{L}(\mathbf{FA}_n) \subseteq \mathcal{L}(\mathbf{GFA}_n)$ and $\mathcal{L}_e(\mathbf{FA}_n) \subseteq \mathcal{L}_e(\mathbf{GFA}_n)$.

To prove the converse inclusion, let n be any positive integer, let $M = (Q, \Sigma, R, s, f)$ be any n-IGFA, and let $\Gamma \subseteq \mathcal{B}(G_M)$ be the set of bridges defining the n islands of M. We construct an n-IFA $N = (O, \Sigma, P, s, f')$ along with a set $\Delta \subseteq \mathcal{B}(G_N)$ of its defining bridges such that $L(N) = L(M)$ and $L_e(N, \Delta) = L_e(M, \Gamma)$.

As an intermediate step to simplify the construction, let us define the n-GFA M' with bridges Γ' equivalent to M in every way except that nothing is read during bridge moves. This reading is done just before exiting the island instead. Let $s_1, \dots, s_n$ denote the entry states and $f_1, \dots, f_n$ the exit states of each island of M. Clearly, $s = s_1$, $\Gamma = \{(f_i, s_{i+1}) \mid 1 \leq i < n\}$, and $f = f_n$. We introduce new states $f'_1, \dots, f'_n \notin Q$ which serve as the new exit states of each island. We define M' as the pentuple (Q', Σ, R', s, f'), where:

- $Q' = Q \cup \{f'_1, \dots, f'_n\}$;

- $R' = R_n \cup R_f \cup R_b$, where:
  - $R_n = R \setminus \{px \to q \in R \mid (p,q) \in \Gamma\}$ is the set of original non-bridge rules in M,
  - $R_f = \{f_i x \to f'_i \mid (f_i x \to s_{i+1}) \in R \text{ for some } 1 \leq i < n\} \cup \{f_n \to f'_n\}$ is the set of new rules for the reading done in bridges in M,
  - $R_b = \{f'_i \to s_{i+1} \mid 1 \leq i < n\}$ is the set of new bridge rules;
- $f' = f'_n$ is the new exit state of the final island;
- Σ and s are the same as in M.

We also set $\Gamma' = \{(f'_i, s_{i+1}) \mid 1 \leq i < n\}$. Clearly, $L(M') = L(M)$ and $L_e(M,\Gamma) = L_e(M',\Gamma')$, as exactly one additional step must occur in each island, so for each even computation in M there is a corresponding even computation in M', and vice versa.

Next, let $m = \max\{\text{len}(x) \mid px \to q \in R', \text{ where } p,q \in Q', x \in \Sigma^*\}$ be the smallest integer greater than or equal to the length of any string that can be read by a single rule in M' (or, equivalently, in M). We simulate each internal island rule of M' in N by m new rules and $m-1$ new intermediate states as follows.

Let $r : px \to q \in (R_n \cup R_f)$ be a non-bridge rule of M', let $x = a_1 \cdots a_k$ with $k = \text{len}(x) \leq m$, and let a_i denote ε for $i > k$. We introduce a new state $\langle r, i\rangle$ for each $1 \leq i < m$. To simulate r, we add the rules $pa_1 \to \langle r, 1\rangle$, $\langle r, m-1\rangle a_m \to q$ and for each $1 \leq i \leq m-2$ a rule of the form $\langle r, i\rangle a_{i+1} \to \langle r, i+1\rangle$. The other states and bridge rules can be taken over unchanged.

Leaving a strictly rigorous version of the definition of N to the reader, we see that for each computation of M', a corresponding computation in N can be made such that for each move in an island of M', exactly m moves are performed in the corresponding island in N while reading exactly the same string from the input. This also means that an even computation of M' is always simulated by an even computation in N, because where M' performs exactly k moves in each island for some $k \geq 0$, N performs exactly mk moves in each island. Furthermore, no additional computations can be performed in N, as the new states can only be used in the simulation of rules from M'. Therefore, $L(N) = L(M') = L(M')$ and $L_e(N,\Delta) = L_e(M',\Gamma') = L_e(M,\Gamma)$. This construction can be performed for any n-IGFA, meaning that the inclusions $\mathcal{L}(\mathbf{GFA}_n) \subseteq \mathcal{L}(\mathbf{FA}_n)$ and $\mathcal{L}_e(\mathbf{GFA}_n) \subseteq \mathcal{L}_e(\mathbf{FA}_n)$ hold for any positive integer n, finalizing the proof of the theorem. □

Theorem 4.3. $\mathbf{REG} = \mathcal{L}(\mathbf{FA}_n)$ for any $n \in \mathbb{N}$, $n \geq 1$.

Proof. Follows immediately from Theorems 4.1 and 4.2. □

Accepting power

In this subsection we show that n-IGFA are equivalent to another computational model, *n-parallel right-linear grammars*, defined as follows:

Definition 4.4. Let n be a positive integer. An *n-parallel right-linear grammar* (or *n-PRLG* for short) is a quadruple $G = (N, \Sigma, P, S)$, where:

- N is a finite, non-empty set of *nonterminals*;
- Σ is a finite set of *terminals*, $N \cap \Sigma = \emptyset$;
- $S \in N$ is the *start symbol*;
- P is a finite set of *production rules*, each rule being of one of the following four forms:
  - (i) $S \to x$, where $x \in \Sigma^*$,
  - (ii) $S \to A_1 \cdots A_n$, where $A_i \in N \setminus \{S\}$ for $1 \leq i \leq n$,
  - (iii) $A \to xB$, where $A, B \in N \setminus \{S\}$ and $x \in \Sigma^*$,
  - (iv) $A \to x$, where $A \in N \setminus \{S\}$ and $x \in \Sigma^*$.

A derivation step in a given n-PRLG $G = (N, \Sigma, P, S)$ is defined as follows. For $\alpha, \beta \in (N \cup \Sigma)^*$, $\alpha \Rightarrow_G \beta$ holds if and only if one of the following three options holds:

a) $\alpha = S$ and $(S \to \beta) \in P$;

b) $\alpha = x_1 A_1 x_2 \cdots x_n A_n$ and $\beta = x_1 y_1 x_2 \cdots x_n y_n$, where $x_i \in \Sigma^*$, $A_i \in N \setminus \{S\}$, $y_i \in \Sigma^*(N \setminus \{S\})$, and $(A_i \to y_i) \in P$ for all $1 \leq i \leq n$;

c) $\alpha = x_1 A_1 x_2 \cdots x_n A_n$ and $\beta = x_1 y_1 x_2 \cdots x_n y_n$, where $x_i \in \Sigma^*$, $A_i \in N \setminus \{S\}$, $y_i \in \Sigma^*$, and $(A_i \to y_i) \in P$ for all $1 \leq i \leq n$.

We can also simply write $x \Rightarrow y$ where G is clear from the context. We define $\Rightarrow^*$, $\Rightarrow^+$ and $\Rightarrow^k$ for non-negative integers k as usual.

The language generated by an n-PRLG G is denoted by $L(G)$ and defined as

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}.$$

Note that unlike other definitions of this model, we allow erasing rules. However, this has no effect on the generative power of this model in comparison with the variant without erasing rules (see [8] for a proof). In what follows, we denote the class of languages generated by n-PRLGs by $\mathbf{PRL}_n$.

Theorem 4.5. Let n be a positive integer, $M = (Q, \Sigma, R, s, f)$ be an n-IGFA and $\Gamma \subseteq \mathcal{B}(G_M)$ be a set of bridges of M. Then there is an n-PRLG $G = (N, \Sigma, P, S)$ such that $L(G) = L_e(M, \Gamma)$.

Proof. Given the n-IGFA $M = (Q, \Sigma, R, s, f)$ and the set of bridges Γ, we can infer the individual islands $Q_1, \dots, Q_n$ and their entry and exit states s_i, f_i for each $1 \leq i \leq n$. We can simply take the states of the automaton as the nonterminals for the grammar, add a start symbol S and an initial rule of the form $S \to s_1 \cdots s_n$, and add a rewriting rule of the form $p \to xq$ for each internal rule $px \to q$ within an island of M. Finally, we add rules to allow the rewriting of exit states to whatever strings can be read on the following bridge, or to the empty string in the case of the final state.

Formally, let $G = (Q \cup \{S\}, \Sigma, P, S)$, where $S \notin Q \cup \Sigma$ and $P = P_s \cup P_i \cup P_f$, with the individual components of the rule set defined as follows:

- $P_s = \{S \to s_1 \cdots s_n\}$ is the set containing the single initial rule;
- $P_i = \{p \to xq \mid (px \to q) \in R \wedge p, q \in Q_j$ for some island $j\}$ is the set containing rules corresponding to the internal rules of each island;
- $P_f = \{f_i \to x \mid 1 \leq i < n \wedge (f_i x \to s_{i+1}) \in R\} \cup \{f \to \varepsilon\}$ is the set containing the terminating rules for each island.

After applying the initial rule, the ith nonterminal represents the ith island of M. In any derivation, exactly the same number of steps is taken in each simulated island, fulfilling the requirement that only even computations get simulated. This is because an n-PRLG requires that in a single step, either all n nonterminals in its sentential form get rewritten to terminal strings (which corresponds to applying the terminating rules), or they all get rewritten to strings each containing exactly one nonterminal (which corresponds to applying the internal rules). Furthermore, because of the way the rewriting rules of the grammar are based on M, G generates exactly those strings which M accepts. Therefore, $L(G) = L_e(M, \Gamma)$, concluding the proof. □

It follows from this theorem that $\mathcal{L}_e(\mathbf{GFA}_i) \subseteq \mathbf{PRL}_i$. The converse relation is not as straightforward to prove. It is easy to imagine that we can simply construct an n-IGFA based on the rules of a given n-PRLG and let each island simulate the derivations starting from a particular nonterminal, but an n-PRLG may have multiple rules of the form $S \to A_1 \cdots A_n$, which requires each island to simulate derivations from different nonterminals depending on which initial rule was used, with only the number of steps taken in each island as a means of communication between the islands.

Consider the relatively simple language $L_2 = \{a^n b^n, b^n a^n \mid n \geq 0\}$, which is clearly a $\mathbf{PRL}_2$ language, as it is generated by the grammar $G_2 = (\{S, A, B\}, \{a, b\}, P, S)$, where P contains the following production rules:

$$\begin{array}{rcl} S & \to & AB, \\ S & \to & BA, \\ A & \to & aA, \\ A & \to & \varepsilon, \\ B & \to & bB, \\ B & \to & \varepsilon. \end{array}$$

Let us now think about how to construct a 2-IFGA which would accept L_2 by even computations. While it is easy to imagine how the concept of even computations can be employed to ensure the same number of a's and b's in a string, it is not immediately obvious how the second island can be forced to only accept b's if the first island only accepted a's, and vice versa. Regardless of what computation path is performed in the first island, the computation must go through the single available bridge, so the only information about the computation in the first island that is available to the automaton in the second island is how many steps were performed. However, this is enough to communicate which kind of string is being accepted – by adding appropriate ε-rules, we can ensure that for strings of the form $a^n b^n$, an odd number of steps is performed in each island, whereas for strings of the form $b^n a^n$ this number would be even. This idea is demonstrated in the following example.

Example 4.6. Consider the automaton $M_2 = (Q, \{a, b\}, R, s_1, f_2)$ where

- $Q = \{s_1, s_2, p_1, p_2, p_3, p_4, q_1, q_2, q_3, q_4, f_1, f_2\}$,
- $R = \{s_1 \to p_1, s_1 \to q_1, p_1a \to p_2, p_2 \to p_1, p_2 \to f_1, q_1 \to q_2, q_1 \to f_1, q_2b \to q_1, f_1 \to s_2, s_2 \to p_3, s_2 \to q_3, p_3b \to p_4, p_4 \to p_3, p_4 \to f_2, q_3 \to q_4, q_3 \to f_2, q_4a \to q_3\}$,

along with the set of bridges $\Gamma = \{(f_1, s_2)\}$. A diagram of this automaton can be seen in Figure 4.6 with dashed rectangles delimiting the individual islands.

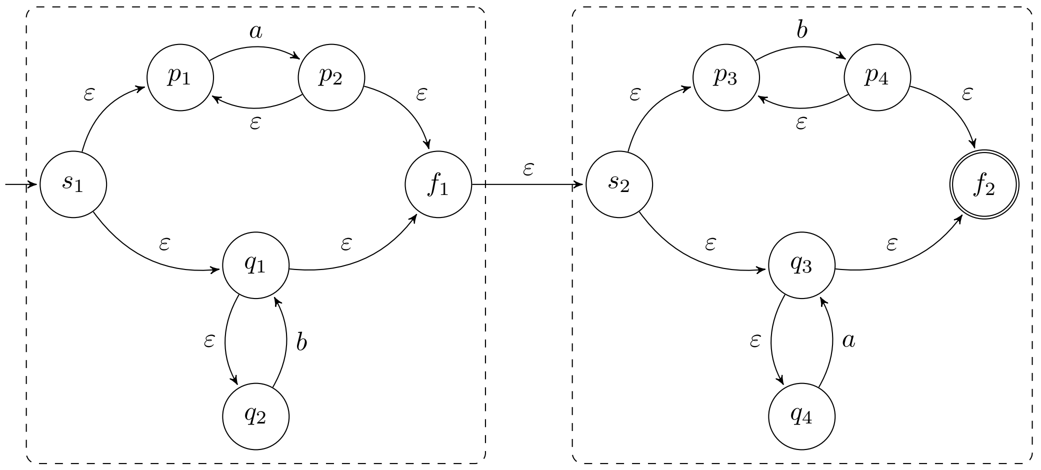

Fig. 4. The diagram of M_2.

Examine R to see that any computation within the first island ending in its exit state is necessarily of one of the following two forms, where $w \in \Sigma^*$:

(1) $s_1a^kw \vdash p_1a^kw \vdash^{2k-1} p_2w \vdash f_1w$, where $k \geq 1$,

(2) $s_1b^kw \vdash q_1b^kw \vdash^{2k} q_1w \vdash f_1w$, where $k \geq 0$.

Analogously, any computation within the second island that processes the entire remainder of the input string and ends up in the final state is necessarily of one of the following two forms:

(1') $s_2b^k \vdash p_3b^k \vdash^{2k-1} p_4 \vdash f_2$, where $k \geq 1$,

(2') $s_2a^k \vdash q_3a^k \vdash^{2k} q_3 \vdash f_2$, where $k \geq 0$.

Notice that computations of forms (1), (1') consist of $2k + 1$ steps for a given $k \geq 1$, so the number of steps is always odd, whereas computations of forms (2), (2') consist of $2k + 2$ steps for a given $k \geq 0$, so the number of steps is always even. Recall that in an even computation, exactly the same number of steps must be taken in each island. From this and from what we have established about the possible computations in each island, there are only two possible forms of even computations in automaton M_2:

(a) $s_1 a^k b^k \; {}_1{\vdash}^{2k+1} f_1 b^k \; {}_b{\vdash} s_2 b^k \; {}_2{\vdash}^{2k+1} f_2$, where $k \geq 1$,

(b) $s_1 b^k a^k \; {}_1{\vdash}^{2k+2} f_1 a^k \; {}_b{\vdash} s_2 a^k \; {}_2{\vdash}^{2k+2} f_2$, where $k \geq 0$.

Notice that even computations of form (a) go through states p_i and do not go through states q_i for $i \in \{1, 2, 3, 4\}$, whereas it is the other way around for computations of form (b)[2].

In summary,

$$L_e(M_2, \Gamma) = \{a^n b^n, b^n a^n \mid n \geq 0\} = L_2.$$

In an n-PRLG, we can achieve the generation of the two different string forms just by having two different starting rules ($S \to AB$ and $S \to BA$). In an n-IGFA, no such simple approach is possible in the general case, but as seen in the previous example, we can associate each string form with a different remainder when dividing by a preselected number (in this case 2) and require the number of steps taken in each island when accepting a string of a given form to give the associated remainder; 0 for strings of the form $b^n a^n$, 1 for strings of the form $a^n b^n$.

To allow us to simulate any n-PRLG, we can generalize this approach. To simulate a grammar with k starting rules, we proceed as follows. First, we associate a unique remainder from $\{0, \ldots, k-1\}$ with each starting rule. Then, we construct the automaton in such a way, that to simulate a derivation starting with a given starting rule, the number of steps taken in each island must give the corresponding remainder when divided by k.

Because we do not need to differentiate between different terminal-generating starting rules in this way (that is, rules of the form $S \to x$ where $x \in \Sigma^*$), we instead use the remainder when dividing by $m + 1$, where m is the number of different nonterminal-generating starting rules (that is, rules of the form $S \to \alpha$ where $\alpha \in N^n$). We reserve remainder 0 for terminal-generating starting rules and associate a remainder value from $\{1, \ldots, m\}$ with each of the m nonterminal-generating starting rules. This approach is formalized in the proof of the following theorem.

Theorem 4.7. Let n be a positive integer, and let $G = (N, \Sigma, P, S)$ be an n-PRLG. Then there is an n-IGFA $M = (Q, \Sigma, R, s, f)$ and a set of bridges $\Gamma \subseteq \mathcal{B}(G_M)$ such that $L_e(M, \Gamma) = L(G)$.

Proof. Let $G = (N, \Sigma, P, S)$ be an n-PRLG. First, for convenience, let us divide the set of production rules P of G into four pairwise disjoint subsets:

- $P_s = \{p \in P \mid p \text{ is of the form } S \to A_1 \cdots A_n, \text{ where } A_1, \ldots, A_n \in N \setminus \{S\}\}$,
- $P_c = \{p \in P \mid p \text{ is of the form } S \to x, \text{ where } x \in \Sigma^*\}$,
- $P_n = \{p \in P \mid p \text{ is of the form } A \to xB, \text{ where } A, B \in N \setminus \{S\}, x \in \Sigma^*\}$,
- $P_t = \{p \in P \mid p \text{ is of the form } A \to x, \text{ where } A \in N \setminus \{S\}, x \in \Sigma^*\}$.

[2]Although of course q_2 and q_4 are not visited when accepting the empty string.

Note that $P_s \cup P_c \cup P_n \cup P_t = P$.

Let $m = \text{card}(P_s)$ be the number of different initial rules generating nonterminals, and let these rules be indexed, meaning we can write $P_s = \{p_1, \dots, p_m\}$ where for any given integer j satisfying $1 \leq j \leq m$, p_j denotes a specific rule, and $p_i = p_j$ implies $i = j$. Also, let A_{ij} denote the ith nonterminal on the right-hand side of the jth rule of P_s, meaning that p_j is of the form $S \to A_{1j} \cdots A_{nj}$. We construct M in such a way that for a derivation starting with rule p_j, the corresponding computation performs in each island a number of steps giving the remainder j when divided by $m+1$. This ensures that even computations can only be created by combining paths within individual islands intended for simulating the same initial rule of G. The remainder 0 is used for derivations starting (and thus necessarily also ending) with rules from P_c.

We can now start constructing M. The set Q consists of the following states:

- s_i and f_i for each $1 \leq i \leq n$, which are the entry and exit states for each island;
- States of the form $\langle i, j \rangle$ for each $1 \leq i \leq n$, $1 \leq j \leq m$, which are used for the initial generation of the remainder in each island;
- States of the form $\langle A, i, j \rangle$ for each $1 \leq i \leq n$, $1 \leq j \leq m$, $A \in N \setminus \{S\}$, which represent the nonterminal A simulated in the ith island in a computation with predetermined remainder j (meaning that this state is intended to be used for simulations of derivations starting with the rule $p_j \in P_s$);
- States of the form $\langle i, j, k, B \rangle$ for each $1 \leq i \leq n$, $1 \leq j \leq m$, $1 \leq k \leq m$, $B \in (N \setminus \{S\}) \cup \{\varepsilon\}$, which allow the rules from P_n and P_t with the right-hand side xB for some $x \in \Sigma^*$ to be simulated using $m+1$ moves in M. In each of these states, i identifies the island, j represents the remainder for this branch of computation (or, equivalently, it represents that the derivation being simulated starts with the rule $p_j \in P_s$), and k is the counter used to ensure that exactly $m+1$ moves are performed for the simulation of any given rule from $P_n \cup P_t$.

We now have the necessary groundwork to describe the rules of the automaton. Specifically, R must contain the following rules:

- Rules to generate the remainder:
  - A rule of the form $s_i \to \langle i, 1 \rangle$ for each $1 \leq i \leq n$,
  - A rule of the form $\langle i, j \rangle \to \langle i, j+1 \rangle$ for each $1 \leq i \leq n$, $1 \leq j < m$;
- Rules to pair the remainder 0 with simulating derivations starting with rules from P_c:
  - A rule of the form $\langle 1, m \rangle x \to f_1$ for each production rule $S \to x \in P_c$;
  - A rule of the form $\langle i, m \rangle \to f_i$ for each $1 < i \leq n$;
- Rules to pair the remainder j, $1 \leq j \leq m$, in island i with the nonterminal A_{ij} derived at position i by the initial rule $p_j : S \to A_{1j} \cdots A_{nj} \in P_s$ corresponding to the remainder:

  - A rule of the form $s_i \to \langle A_{i1}, i, 1\rangle$ for each $1 \le i \le n$;
  - A rule of the form $\langle i, j-1\rangle \to \langle A_{ij}, i, j\rangle$ for each $1 \le i \le n$, $1 < j \le m$;

- Rules to simulate production rules from P_n and P_t, including extra ε-rules to ensure that the simulation of a given rule always takes $m+1$ steps:
  - A rule of the form $\langle A, i, j\rangle x \to \langle i, j, 1, B\rangle$ for each $1 \le i \le n$, $1 \le j \le m$ and each rule $A \to xB \in P_n \cup P_t$, where $x \in \Sigma^*$ and $B \in (N \setminus \{S\}) \cup \{\varepsilon\}$;
  - A rule of the form $\langle i, j, k, B\rangle \to \langle i, j, k+1, B\rangle$ for each $1 \le i \le n$, $1 \le j \le m$, $1 \le k < m$, $B \in (N \setminus \{S\}) \cup \{\varepsilon\}$;
  - A rule of the form $\langle i, j, m, B\rangle \to \langle B, i, j\rangle$ for each $1 \le i \le n$, $1 \le j \le m$, $B \in N \setminus \{S\}$;
  - A rule of the form $\langle i, j, m, \varepsilon\rangle \to f_i$ for each $1 \le i \le n$, $1 \le j \le m$;

- Bridge rules:
  - A rule of the form $f_i \to s_{i+1}$ for each $1 \le i < n$.

Given all this, we can define $M = (Q, \Sigma, R, s, f)$, where

- Q and R are as described above,
- Σ is the same as in G,
- $s = s_1$ and $f = f_n$.

Finally, $\Gamma = \{(f_i, s_{i+1}) \mid 1 \le i < n\}$.

By examining the definition of R above, it can be seen that each non-initial rule of G is simulated by $m+1$ moves, and enforcing the remainder ensures that in even computations, the islands agree on the initial S-rule used. Even computations also make sure that in the simulated derivation, all nonterminals get rewritten to a terminal string in a single step, as an n-PRLG requires (this corresponds to moving into the exit state f_i of each island). Each island can on its own accept any string derivable from any single nonterminal different from S, but the structure helps simulate the synchronicity of the grammar. It therefore follows that $L_e(M, \Gamma) = L(G)$. □

Theorem 4.7 implies that $\mathbf{PRL}_n \subseteq \mathcal{L}_e(\mathbf{GFA}_n)$. From Theorems 4.5 and 4.7, we obtain the following main result of this paper.

Corollary 4.8. For any positive integer n, $\mathbf{PRL}_n = \mathcal{L}_e(\mathbf{GFA}_n)$.

Recall that

$$\mathbf{PRL}_n \subset \mathbf{PRL}_{n+1} \text{ for all } n \ge 1$$

(see [8]). From this hierarchy and Corollary 4.8 above, we get the next corollary.

Corollary 4.9. For any positive integer n, $\mathcal{L}_e(\mathbf{GFA}_n) \subset \mathcal{L}_e(\mathbf{GFA}_{n+1})$.

5. CONCLUSION

The present paper has introduced the notion of an island in transition graphs for finite automata. Based on this notion, it placed a restriction on the way finite automata work, referred to as even computation. As its main result, it demonstrated an infinite hierarchy of language families corresponding to the number of islands in evenly computing finite automata. This hierarchy coincides with a well-known infinite hierarchy of language families resulting from multi-parallel right-linear grammars in a very natural way. Consequently, it is obviously closely related to some well-known results about formal languages, on which it sheds light in an alternative way. Therefore, the authors suggest to continue with the study opened in the present paper. Specifically, this investigation should pay special attention to the following three open problem areas.

I. Introduce deterministic versions of the restricted finite automata defined in this paper. Study them by analogy with the study of classical deterministic finite automata.

II. We have based the present paper on one-way finite automata with useful states, out of which only one state is final. Observe that this concept fulfills an essential role in the proofs of the achieved results. In the classical automata theory, however, there exist many other equivalent versions of these automata. Do the achieved results hold in their terms as well? Specifically, do they hold in terms of two-way finite automata (see Section 2.6 in [3])?

III. There exist a broad variety of finite automata, ranging from quantum through probabilistic up to fuzzy finite automata. Restrict these automata by analogy with the bridge-based restriction discussed in the present paper. Study their power. From a more general viewpoint, investigate this topic in terms of automata that are stronger than finite automata, such as a broad variety of pushdown automata, including counters and one-turn pushdown automata.

ACKNOWLEDGEMENTS

We thank all three anonymous referees for their useful comments.

FUNDING

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II), from the project IT4Innovations excellence in science—LQ1602.

(Received January 21, 2021)

REFERENCES

[1] Z. Bavel: Structure and transition-preserving functions of finite automata. J. ACM *15* (1968), 1, 135–158. DOI:10.1145/321439.321448

[2] X. Han, Z. Chen, Z. Liu, and Q. Zhang: The detection and stabilisation of limit cycle for deterministic finite automata. Int. J. Control *91* (2018), 4, 874–886. DOI:10.1080/00207179.2017.1295319

[3] J. E. Hopcroft and J. D. Ullman: Introduction to Automata Theory, Languages and Computation. First edition. Addison-Wesley Publishing Company, 1979.

[4] A. Meduna: Formal Languages and Computation: Models and Their Applications. First edition. Auerbach Publications, 2014.

[5] A. Meduna and T. Masopust: Self-regulating finite automata. Acta Cybernetica *18* (2007), 1, 135–153.

[6] A. Meduna and P. Zemek: Jumping finite automata. Int. J. Found. Comput. Sci. *23* (2012), 1555–1578. DOI:10.1142/S0129054112500244

[7] A. Meduna and P. Zemek: Regulated Grammars and Automata. Springer, 2014.

[8] R. D. Rosebrugh and D. Wood: Restricted parallelism and right linear grammars. Util. Math. *7* (1975), 151–186.

[9] R. Schönecker: Automaty s několika čistými zásobníky. Master's Thesis, University of Technology, Faculty of Information Technology, Brno 2005.

[10] R. Sin'ya, K. Matsuzaki, and M. Sassa: Simultaneous finite automata: An efficient data-parallel model for regular expression matching. In: 42nd International Conference on Parallel Processing 2013, pp. 220–229.

[11] V. V. Skobelev and V. G. Skobelev: Finite automata over algebraic structures: models and some methods of analysis. Computer Sci. J. Moldova *23* (2015).

[12] A. Verma and A. Loura: A novel algorithm for the conversion of parallel regular expressions to non-deterministic finite automata. Appl. Math. Inform. Sci. *8* (2014), 95–105. DOI:10.12785/amis/080111

[13] A. Yli-Jyrä and K. Koskenniemi: Compiling contextual restrictions on strings into finite-state automata. In: Proc, Eindhoven FASTAR Days 2004.

Dušan Kolář, Božetěchova 2, 612 00 Brno. Czech Republic.
e-mail: kolar@fit.vutbr.cz

Alexander Meduna, Božetěchova 2, 612 00 Brno. Czech Republic.
e-mail: meduna@fit.vutbr.cz

Martin Tomko, Božetěchova 2, 612 00 Brno. Czech Republic.
e-mail: itomko@fit.vutbr.cz