Networks of Evolutionary Processors with Subregular Filters

Jürgen Dassow, Florin Manea[⋆], and Bianca Truthe

Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik
PSF 4120, D-39016 Magdeburg, Germany
{dassow,manea,truthe}@iws.cs.uni-magdeburg.de

Abstract. In this paper we propose a hierarchy of classes of languages, generated by networks of evolutionary processors with the filters in several special classes of regular sets. More precisely, we show that the use of filters from the class of ordered, non-counting, power-separating, circular, suffix-closed regular, union-free, definite and combinational languages is as powerful as the use of arbitrary regular languages and yields networks that can generate all the recursively enumerable languages. On the other hand, the use of filters that are only finite languages allows only the generation of regular languages, but not all regular languages can be generated. If we use filters that are monoids, nilpotent languages or commutative regular languages, we obtain the same family of languages which contains non-context-free languages but not all regular languages. These results seem to be of interest because they provide both upper and lower bounds on the classes of languages that one can use as filters in a network of evolutionary processor in order to obtain a complete computational model.

1 Introduction

An important part of theoretical computer science is the study of problems and processes connected with regular sets. In the last years a lot of papers appeared in which, for such problems and processes, the effect of going from arbitrary regular sets to special regular sets was studied. We here mention four such topics.

– It is a classical result that any nondeterministic finite automaton with n states can be transformed into a deterministic one with 2^n states, which accepts the same language, and that this exponential blow-up with respect to the number of states is necessary in the worst cases. In [2], this problem is studied if one restricts to the case that the automata accept special regular languages only. It is shown, that the situation does not change for suffix-closed and star-free regular languages; however, for some classes of definite languages, the size of the deterministic automaton is bounded by $2^{n-1} + 1$.

[⋆] Also at: *Faculty of Mathematics and Computer Science, University of Bucharest, Str. Academiei 14, RO-010014 Bucharest, Romania (*`flmanea@fmi.unibuc.ro`*).* The work of Florin Manea is supported by the *Alexander von Humboldt Foundation.*

- A number α, $n \leq \alpha \leq 2^n$, is called magic (w.r.t. n), if there is no nondeterministic finite automaton with n states such that the minimal deterministic finite automaton has α states. It is known that no magic numbers exist if $n \geq 3$. This situation changes if one considers subregular families of languages. For instance, only the values α with $n + 1 \leq \alpha \leq 2^{n-1} + 1$ are possible for prefix-free regular languages (see [16]).
- In the last 20 years the behaviour of the (nondeterministic) state complexity under operations is intensively studied, i.e., it is asked for the size of the minimal (non)deterministic finite automaton for the language obtained from languages with given sizes. For many operations, the worst case is exactly determined. It has been shown that one gets smaller sizes if one restricts to special regular languages (see [13], [14], [3], and [17]).
- In order to enlarge the generative power, some mechanisms connected with regular languages were introduced, which control the derivations in context-free grammars. For instance, the sequence of applied rules in a regularly controlled grammar, the current sentential form in a conditional grammar and the levels of the derivation tree in a tree controlled grammar have to belong to given regular languages. In the papers [7], [9], [8], and [11], the change in the generative power, if one restricts to special regular sets, is investigated.

In this paper we continue the research along this direction. We consider the effect of special regular filters for generating evolutionary networks.

Networks of language processors have been introduced in [6] by E. Csuhaj-Varjú and A. Salomaa. Such a network can be considered as a graph where the nodes are sets of productions and at any moment of time a language is associated with a node. In a derivation step any node derives from its language all possible words as its new language. In a communication step any node sends those words to other nodes where the outgoing words have to satisfy an output condition given as a regular language (called output filter), and any node takes words sent by the other nodes if the words satisfy an input condition also given by a regular language (called input filter). The language generated by a network of language processors consists of all (terminal) words which occur in the languages associated with a given node.

Inspired by biological processes, in [4] a special type of networks of language processors was introduced which are called networks with evolutionary processors because the allowed productions model the point mutation known from biology. The sets of productions have to be substitutions of one letter by another letter or insertions of letters or deletion of letters; the nodes are then called substitution node or insertion node or deletion node, respectively. Results on networks of evolutionary processors can be found, e. g., in [4], [5], [18]. For instance. in [5], it was shown that networks of evolutionary processors are complete in that sense that they can generate any recursively enumerable language.

Modifications of evolutionary networks with evolutionary processors concern restrictions in the type of the nodes and the mode of applying a rule. In [1], it is investigated how the generative power behaves if one restricts to networks with at most two types of nodes only. Moreover, in the case that one allows that some insertions and deletions can only be performed at the begin or end of the word one has also restricted to special regular filters given by random context conditions.

In this paper, we modify the filters. We require that the filters have to belong to a special subset of the set of all regular languages. We show that the use of filters from the class of ordered, non-counting, power-separating, circular, suffix-closed regular, union-free, definite and combinational languages is as powerful as the use of arbitrary regular languages and yields networks that can generate all the recursively enumerable languages. On the other hand, the use of filters that are only finite languages allows only the generation of regular languages, but not all regular languages can be generated. If we use filters that are monoids, nilpotent languages or commutative regular languages, we obtain the same family of languages which contains non-context-free languages but not all regular languages. These results seem to be of interest because they provide both upper and lower bounds on the classes of languages that one can use as filters in a network of evolutionary processor in order to obtain a complete computational model.

By reasons of space we omit some proofs. The omitted proofs can be found in [10] (see http://theo.cs.uni-magdeburg.de/pubs/preprints/pp-afl-2011-01.pdf).

2 Definitions

We assume that the reader is familiar with the basic concepts of formal language theory (see e. g. [19]). We here only recall some notations used in the paper.

By V^* we denote the set of all words (strings) over V (including the empty word λ). The length of a word w is denoted by $|w|$. By V^+ and V^k for some natural number k we denote the set of all non-empty words and the set of all words with length k, respectively. Let V_k be the set of all words over V with a length of at most k, i. e. $V_k = \bigcup_{i=0}^{k} V^i$.

By *REG*, *CF*, and *RE* we denote the families of regular, context-free, and recursively enumerable languages, respectively.

For a language L over V, we set

$$\begin{aligned} \mathit{Comm}(L) &= \{a_{i_1}\dots a_{i_n} \mid a_1\dots a_n \in L,\ n \geq 1,\ \{i_1, i_2, \dots, i_n\} = \{1, 2, \dots, n\}\},\\ \mathit{Circ}(L) &= \{vu \mid uv \in L,\ u, v \in V^*\},\\ \mathit{Suf}(L) &= \{v \mid uv \in L,\ u, v \in V^*\} \end{aligned}$$

We consider the following restrictions for regular languages. Let L be a language and $V = \mathit{alph}(L)$ the minimal alphabet of L. We say that L is

- *combinational* iff it can be represented in the form $L = V^*A$ for some subset $A \subseteq V$,
- *definite* iff it can be represented in the form $L = A \cup V^*B$ where A and B are finite subsets of V^*,
- *nilpotent* iff L is finite or $V^* \setminus L$ is finite,

– *commutative* iff $L = Comm(L)$,
– *circular* iff $L = Circ(L)$,
– *suffix-closed* (or *fully initial* or *multiple-entry* language) iff $xy \in L$ for some $x, y \in V^*$ implies $y \in L$ (or equivalently, $Suf(L) = L$),
– *non-counting* (or star-free) iff there is an integer $k \geq 1$ such that, for any $x, y, z \in V^*$, $xy^kz \in L$ if and only if $xy^{k+1}z \in L$,
– *power-separating* iff for any $x \in V^*$ there is a natural number $m \geq 1$ such that either $J_x^m \cap L = \emptyset$ or $J_x^m \subseteq L$ where $J_x^m = \{x^n \mid n \geq m\}$,
– *ordered* iff L is accepted by some finite automaton $\mathcal{A} = (Z, V, \delta, z_0, F)$ where $(Z, \preceq)$ is a totally ordered set and, for any $a \in V$, the relation $z \preceq z'$ implies the relation $\delta(z, a) \preceq \delta(z', a)$,
– *union-free* iff L can be described by a regular expression which is only built by product and star.

It is obvious that combinational, definite, nilpotent, ordered and union-free languages are regular, whereas non-regular languages of the other types mentioned above exist.

By *COMB*, *DEF*, *NIL*, *COMM*, *CIRC*, *SUF*, *NC*, *PS*, *ORD*, and *UF* we denote the families of all combinational, definite, nilpotent, regular commutative, regular circular, regular suffix-closed, regular non-counting, regular power-separating, ordered, and union-free languages, respectively. Moreover, we add the family *MON* of all languages of the form V^*, where V is an alphabet (languages of *MON* are target sets of monoids; we call them monoidal languages). We set

$$\mathcal{G} = \{FIN, MON, COMB, DEF, NIL, COMM, CIRC, SUF, NC, PS, ORD, UF\}.$$

The relations between families of $\mathcal{G}$ are investigated e. g. in [15] and [20]. and their set-theoretic relations are given in Figure 1.

We call a production $\alpha \to \beta$ a
– substitution if $|\alpha| = |\beta| = 1$,
– deletion if $|\alpha| = 1$ and $\beta = \lambda$.

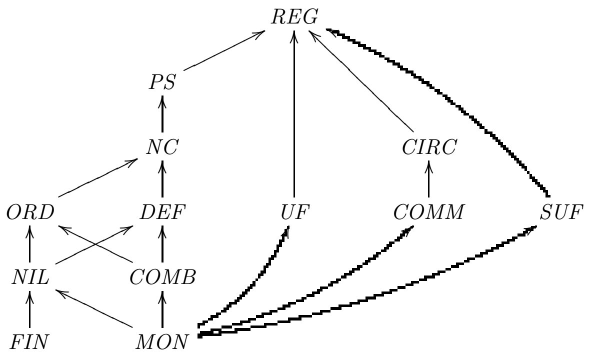

$$\begin{tikzcd}
 & & REG & & \\
 PS \arrow[urr] & & & & \\
 NC \arrow[u] & & & CIRC \arrow[uul] & \\
 ORD \arrow[ur] & DEF \arrow[ul] & UF \arrow[uuu] & COMM \arrow[u] & SUF \arrow[uuull] \\
 NIL \arrow[u] \arrow[ur] & COMB \arrow[u] \arrow[ul] & & & \\
 FIN \arrow[u] & MON \arrow[u] \arrow[ul] \arrow[uur] \arrow[uurr] \arrow[uurrr] & & &
\end{tikzcd}$$

Fig. 1. Hierarchy of subregular languages (an arrow from X to Y denotes $X \subset Y$, and if two families are not connected by a directed path then they are incomparable)

The productions are applied like context-free rewriting rules. We say that a word v derives a word w, written as $v \Longrightarrow w$, if there are words x, y and a production $\alpha \to \beta$ such that $v = x\alpha y$ and $w = x\beta y$. If the rule p applied is important, we write $v \Longrightarrow_p w$.

We introduce insertion as a counterpart of deletion. We write $\lambda \to a$, where a is a letter. The application of an insertion $\lambda \to a$ derives from a word w any word $w_1 a w_2$ with $w = w_1 w_2$ for some (possibly empty) words w_1 and w_2.

We now introduce the basic concept of this paper, the networks of evolutionary processors (NEPs for short).

Definition 1. *Let X be a family of regular languages.*

(i) *A network of evolutionary processors (of size n) with filters of the set X is a tuple*

$$\mathcal{N} = (V, N_1, N_2, \ldots, N_n, E, j)$$

where

- *V is a finite alphabet,*
- *for $1 \leq i \leq n$, $N_i = (M_i, A_i, I_i, O_i)$ where*
  - *M_i is a set of rules of a certain type: $M_i \subseteq \{a \to b \mid a, b \in V\}$ or $M_i \subseteq \{a \to \lambda \mid a \in V\}$ or $M_i \subseteq \{\lambda \to b \mid b \in V\}$,*
  - *A_i is a finite subset of V^*,*
  - *I_i and O_i are languages from X over V,*
- *E is a subset of $\{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$, and*
- *j is a natural number such that $1 \leq j \leq n$.*

(ii) *A configuration C of $\mathcal{N}$ is an n-tuple $C = (C(1), C(2), \ldots, C(n))$ where $C(i)$ is a subset of V^* for $1 \leq i \leq n$.*

(iii) *Let $C = (C(1), C(2), \ldots, C(n))$ and $C' = (C'(1), C'(2), \ldots, C'(n))$ be two configurations of $\mathcal{N}$. We say that C derives C' in one*

- *evolutionary step (written as $C \Longrightarrow C'$) if, for $1 \leq i \leq n$, $C'(i)$ consists of all words $w \in C(i)$ to which no rule of M_i is applicable and of all words w for which there are a word $v \in C(i)$ and a rule $p \in M_i$ such that $v \Longrightarrow_p w$ holds,*
- *communication step (written as $C \vdash C'$) if, for $1 \leq i \leq n$,*

$$C'(i) = (C(i) \setminus O_i) \cup \bigcup_{(k,i) \in E} (C(k) \cap O_k \cap I_i).$$

The computation of an evolutionary network $\mathcal{N}$ is a sequence of configurations $C_t = (C_t(1), C_t(2), \ldots, C_t(n))$, $t \geq 0$, such that

- *$C_0 = (A_1, A_2, \ldots, A_n)$,*
- *for any $t \geq 0$, C_{2t} derives C_{2t+1} in one evolutionary step,*
- *for any $t \geq 0$, C_{2t+1} derives C_{2t+2} in one communication step.*

(iv) *The language $L(\mathcal{N})$ generated by $\mathcal{N}$ is defined as*

$$L(\mathcal{N}) = \bigcup_{t \geq 0} C_t(j)$$

where $C_t = (C_t(1), C_t(2), \ldots, C_t(n))$, $t \geq 0$ is the computation of $\mathcal{N}$.

Intuitively, a network with evolutionary processors is a graph consisting of some, say n, nodes $N_1, N_2, \ldots, N_n$ (called processors) and the set of edges given by E such that there is a directed edge from N_k to N_i if and only if $(k,i) \in E$. Any processor N_i consists of a set of evolutionary rules M_i, a set of words A_i, an input filter I_i and an output filter O_i. We say that N_i is a substitution node or a deletion node or an insertion node if $M_i \subseteq \{a \to b \mid a,b \in V\}$ or $M_i \subseteq \{a \to \lambda \mid a \in V\}$ or $M_i \subseteq \{\lambda \to b \mid b \in V\}$, respectively. The input filter I_i and the output filter O_i control the words which are allowed to enter and to leave the node, respectively. With any node N_i and any time moment $t \geq 0$ we associate a set $C_t(i)$ of words (the words contained in the node at time t). Initially, N_i contains the words of A_i. In an evolutionary step, we derive from $C_t(i)$ all words applying rules from the set M_i. In a communication step, any processor N_i sends out all words $C_t(i) \cap O_i$ (which pass the output filter) to all processors to which a directed edge exists (only the words from $C_t(i) \setminus O_i$ remain in the set associated with N_i) and, moreover, it receives from any processor N_k such that there is an edge from N_k to N_i all words sent by N_k and passing the input filter I_i of N_i, i. e., the processor N_i gets in addition all words of $C_t(k) \cap O_k \cap I_i$. We start with an evolutionary step and then communication steps and evolutionary steps are alternately performed. The language consists of all words which are in the node N_j (also called the output node, j is chosen in advance) at some moment t, $t \geq 0$.

For a family $X \subseteq REG$, we denote the family of languages generated by networks of evolutionary processors where all filters are of type X by $\mathcal{E}(X)$.

The following fact is obvious.

Lemma 1. *Let X and Y be subfamilies of REG such that $X \subseteq Y$. Then the inclusion $\mathcal{E}(X) \subseteq \mathcal{E}(Y)$ holds.*

The following theorem is known (see, e. g., [5]).

Theorem 1. *$\mathcal{E}(REG) = RE$.*

3 Some General Results

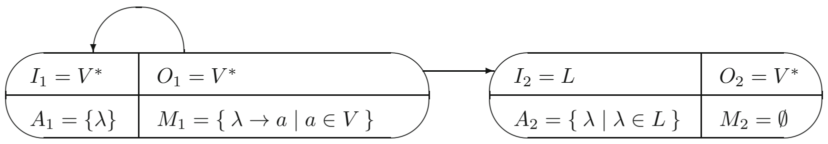
We start with some results which hold for every type of filters.

Lemma 2. *For every network $\mathcal{N}$ of evolutionary processors, there is a network $\mathcal{N}'$ of evolutionary processors that generates the same language as $\mathcal{N}$ and has the property that its output node N' has the form $N' = (\emptyset, \emptyset, I', O')$ for some regular languages I', O' over the network's working alphabet and no edge is leaving N'.*

Theorem 2. *Let $X \in \mathcal{G}$. Then each language $L \in X$ can be generated by a NEP $\mathcal{N}$ with at most two nodes and with filters from X.*

Proof. Let $X = FIN$. Let L be a finite set over V. Then the evolutionary network $(V, (\emptyset, L, \emptyset, \emptyset), \emptyset, 1)$ with all filters from FIN generates L.

If $X \neq FIN$, then $MON \subseteq X$ holds by Figure 1. Moreover, let $L \in X$ be a language over an alphabet V. We construct the NEP $\mathcal{N} = (V, N_1, N_2, E, 2)$ given as



$I_1 = V^*$
$O_1 = V^*$
$A_1 = \{\lambda\}$
$M_1 = \{ \lambda \to a \mid a \in V \}$
$I_2 = L$
$O_2 = V^*$
$A_2 = \{ \lambda \mid \lambda \in L \}$
$M_2 = \emptyset$

Every word $w \in V^+$ will be derived in node N_1 and be communicated to node N_2 which accepts all words that also belong to L. The language generated by $\mathcal{N}$ is $L(\mathcal{N}) = A_2 \cup (V^+ \cap L) = L$. All filters are of type X.

Corollary 1. *For each class $X \in \mathcal{G}$, we have $X \subseteq \mathcal{E}(X)$.*

Corollary 2. *For each class $X \in \mathcal{G}$, we have $MON \subseteq \mathcal{E}(X)$.*

Proof. By the relations given in Figure 1 and Corollary 1, it is sufficient to show that $MON \subseteq \mathcal{E}(FIN)$. Let V be an alphabet and $L = V^*$. Then the evolutionary network $(V, (\{ \lambda \to a \mid a \in V \}, \{\lambda\}, \emptyset, \emptyset), \emptyset, 1)$ with all filters from FIN generates L. Thus, any monoidal language $L = V^*$ belongs to $\mathcal{E}(FIN)$.

4 Computationally Complete Cases

In this section we present the computational completeness of some families $\mathcal{E}(X)$.

Theorem 3. *$\mathcal{E}(SUF) = RE$ and $\mathcal{E}(CIRC) = RE$.*

Proof. First we show that $\mathcal{E}(SUF) = RE$.

Let L be a recursively enumerable set. Let $\mathcal{N} = (V, N_1, N_2, \ldots, N_n, E, j)$ be a network with evolutionary processors and filters from REG such that $L(\mathcal{N}) = L$. For any node $N_i = (M_i, A_i, I_i, O_i)$, we construct the sets

$$I'_i = \{X\}I_i\{Y\} \cup Suf(I_i)\{Y\} \cup \{\lambda\},$$
$$O'_i = \{X\}O_i\{Y\} \cup Suf(O_i)\{Y\} \cup \{\lambda\},$$

where X and Y are two new symbols. By definition, I'_i and O'_i are suffix-closed. We assume that the network $\mathcal{N}$ has the property $N_j = (\emptyset, \emptyset, I_j, O_j)$ and no edge leaves the output node (according to the previous Lemma).

We consider the network

$$\mathcal{N}' = (V \cup \{X, Y\}, N'_1, N'_2, \ldots, N'_n, N'_{n+1}, N'_{n+2}, E', n+2)$$

with

$$\begin{aligned} N'_i &= (M_i, \{X\}A_i\{Y\}, I'_i, O'_i) \text{ for } 1 \le i \le n, \\ N'_{n+1} &= (\{X \to \lambda,\ Y \to \lambda\}, \emptyset, I'_j, V^*), \\ N'_{n+2} &= (\emptyset, \emptyset, V^*, \emptyset), \\ E' &= E \cup \{ (i, n+1) \mid (i, j) \in E \} \cup \{ (n+1, n+2) \}. \end{aligned}$$

It is obvious that the filters of N'_{n+1} and N'_{n+2} are suffix-closed, too. Thus $\mathcal{N}'$ is a network of type SUF.

We now prove that $L(\mathcal{N}) = L(\mathcal{N}')$. We start with words of the form XwY and as long as these words are changed according to rules of M_i, $1 \leq i \leq n$, they can only be sent to nodes N'_s, $1 \leq s \leq n$, and N'_{n+1}. Thus we simulate a derivation in $\mathcal{N}$ (in $\mathcal{N}'$ we have an X in front of and a Y behind the word w occurring in $\mathcal{N}$) and get into N'_{n+1} exactly those words XwY whose subword w comes into N_j. Now X and Y are removed and the resulting word w is sent to N'_{n+2}. Other words cannot arrive in N'_{n+2} and other words do not appear in N_j. Hence, $L(\mathcal{N}') = L(\mathcal{N})$.

To show that $\mathcal{E}(CIRC) = RE$, we repeat the previous proof with the following modifications. We set

$$I'_i = Circ(\{X\}I_i\{Y\}) \text{ and } O'_i = Circ(\{X\}O_i\{Y\}) \text{ for } 1 \leq i \leq n.$$

This ensures that $Circ(F) = F$ for all filters F of the new network $\mathcal{N}'$. Then the proof proceeds as in the case of suffix-closed filters.

Theorem 4. *$\mathcal{E}(COMB) = \mathcal{E}(DEF) = \mathcal{E}(UF) = RE$.*

By the relations shown in Figure 1, Lemma 1, and Theorem 1, we obtain the following theorem.

Theorem 5. *$\mathcal{E}(ORD) = \mathcal{E}(NC) = \mathcal{E}(PS) = RE$.*

5 Computationally Non-complete Cases

We first discuss the case of finite filters. We start with a certain normal form for networks with finite filters.

Lemma 3. *For each NEP $\mathcal{N}$ with only finite filters, we can construct a NEP $\mathcal{N}'$ with only one processor and finite filters that generates the same language as $\mathcal{N}$.*

Theorem 6. *$\mathcal{E}(FIN) \subset REG$.*

Proof. Let $\mathcal{N} = (V, N_1, N_2, \ldots, N_n, E, j)$ be a network with finite filters. Obviously, a word w is in N_j if and only if it is in A_j or satisfies I_j or is obtained from a word in N_j by application of a rule in M_j. We set

$$U = \{\, a \mid \lambda \to a \in M_j \,\},\ V' = \{\, a' \mid a \in V \,\},\ \text{and } U' = \{\, a' \mid a \in U \,\}.$$

Let $h : (V \cup V')^* \to V^*$ be the homomorphism defined by

$$h(a) = a \text{ for } a \in V \quad \text{and} \quad h(a') = \begin{cases} \lambda, & \text{for } a' \in U', \\ a, & \text{for } a' \in V' \setminus U', \end{cases}$$

and $\tau : (V \cup V')^* \to V^*$ be the finite substitution where $\tau(a) = \tau(a')$ for $a \in V$ and $\tau(a)$ consists of all $b \in V \cup \{\lambda\}$ such that there are an integers

$s \geq 0$ and $b_0, b_1, \ldots, b_{s-1} \in V$ and $b_s \in V \cup \{\lambda\}$ such that $a = b_0$, $b = b_s$, and $b_i \to b_{i+1} \in M_j$ for $0 \leq i \leq s-1$ (note that $s = 0$ implies $a = b$). Furthermore, let

$$k = \max\{\, |w| \mid w \in O_j \cup I_j \cup A_j \,\} + 1.$$

We note the following facts:

- Assume that there is a word w of length at least k in $L(\mathcal{N})$. Then w is in $C_t(j)$ for some t. By its length, it cannot leave the node, and thus all words which have a length at least k and can be obtained by application of rules of M_j to w belong to $L(\mathcal{N})$, too.
- If w with $|w| \geq k+1$ is in $L(\mathcal{N})$, then w is obtained from a word $v \in L(\mathcal{N})$ of length k by application of rules in M_j (since substitutions and deletions do not increase the length, the shortest words in $L(\mathcal{N})$ with length at least k are obtained by an insertion from a word of length less than k and thus they have length k).

Now it is easy to see that

$$L(\mathcal{N}) = (L(\mathcal{N}) \cap \bigcup_{i=0}^{k-1} V^i) \cup (\tau(h^{-1}(L(\mathcal{N}) \cap V^k)) \cap \bigcup_{i \geq k} V^i)$$

holds. Since finite languages are regular and regular languages are closed under inverse homomorphisms, finite substitutions, intersection, and union, $L(\mathcal{N})$ is regular. Hence $\mathcal{E}(FIN) \subseteq REG$ holds.

Let $V = \{a\}$ and $L = \{a\} \cup \{\, a^n \mid n \geq 3 \,\}$. Obviously, L is regular.

Suppose the language L is generated by a network with only finite filters. Then, by Lemma 3, there is a network $\mathcal{N}$ with only one node $N = (M, A, \emptyset, O)$ that generates L. Since L is infinite, this node must be inserting. Hence, the rule set is $M = \{\, \lambda \to a \,\}$. If the initial set A contains λ then $\lambda \in L(\mathcal{N})$ which is in contrast to $\lambda \notin L$. If the initial set A contains a or aa then the word aa belongs to the generated language $L(\mathcal{N})$ which is in contrast to $aa \notin L$. If the initial set only contains words a^n with $n \geq 3$ then the word a cannot be generated but $a \in L$ which is a contradiction, too. Hence, there is no network with only finite filters that generates L. Thus, $L \in REG \setminus \mathcal{E}(FIN)$.

The following result shows that the use of filters from the remaining language families, i. e., from MON or NIL or $COMM$ leads to the same class of languages.

Theorem 7. *$\mathcal{E}(MON) = \mathcal{E}(COMM) = \mathcal{E}(NIL)$.*

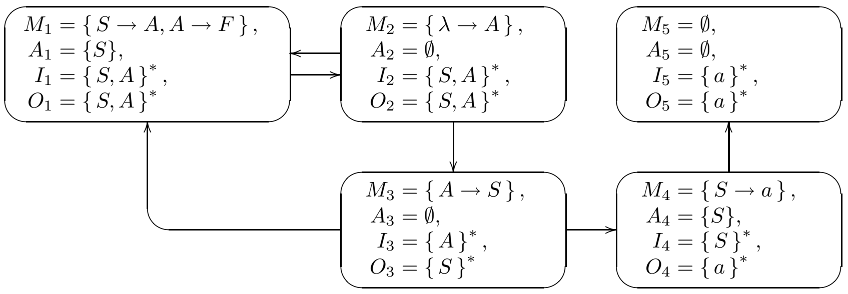
We now present some relations of $\mathcal{E}(MON)$ to other language families.

Theorem 8. *$\mathcal{E}(FIN) \subset \mathcal{E}(MON)$.*

Proof. Since $FIN \subset NIL$, we obtain $\mathcal{E}(FIN) \subseteq \mathcal{E}(NIL)$ by Lemma 1. By Theorem 2, the nilpotent language $L = \{a\} \cup \{\, a^n \mid n \geq 3 \,\}$ is contained in $\mathcal{E}(NIL)$. However, by the second part of the proof of Theorem 6, L is not contained in $\mathcal{E}(FIN)$. Thus $\mathcal{E}(FIN) \subset \mathcal{E}(NIL)$. The statement now follows from Theorem 7.

Lemma 4. *The family $\mathcal{E}(MON)$ contains a non-semi-linear (hence non-regular and non-context-free) language.*

Proof. Let $V = \{S, A, F, a\}$ and $\mathcal{N} = (V, N_1, N_2, N_3, N_4, N_5, E, 5)$ be the following network:


$M_1 = \{S \to A, A \to F\}$, $A_1 = \{S\}$, $I_1 = \{S, A\}^*$, $O_1 = \{S, A\}^*$
$M_2 = \{\lambda \to A\}$, $A_2 = \emptyset$, $I_2 = \{S, A\}^*$, $O_2 = \{S, A\}^*$
$M_5 = \emptyset$, $A_5 = \emptyset$, $I_5 = \{a\}^*$, $O_5 = \{a\}^*$
$M_3 = \{A \to S\}$, $A_3 = \emptyset$, $I_3 = \{A\}^*$, $O_3 = \{S\}^*$
$M_4 = \{S \to a\}$, $A_4 = \{S\}$, $I_4 = \{S\}^*$, $O_4 = \{a\}^*$

In the beginning, we have the word S in node N_1. We consider a word S^n for $n \geq 1$ in node N_1 in an even moment (in the beginning or after a communication step). One occurrence of S is replaced by A, then the word is sent to node N_2 where another copy of A is inserted. This word w goes back to node N_1 and it goes on to node N_3 which takes it if no S appears in the word. If in N_1 the rule $A \to F$ is applied then the symbol F is introduced which cannot be replaced. Due to the output filter O_1, the word will be trapped in N_1 for ever. If, in the word w, no S is present then the only rule which can be applied is $A \to F$ and the cycle is stopped. If w still contains an S then it is replaced by A and N_2 inserts another A. So, the words move between N_1 and N_2 where alternatingly an S is replaced by A and an A is inserted until the word only contains As. The word is then A^{n+1}. Hence, the number of letters has been doubled.

In N_3, each A is replaced by S. The word is S^{n+1} when it leaves N_3. It moves to N_1 and to N_4. In N_1, the cycle starts again with a word S^m for $m \geq 1$. All arriving words in N_4 have the form S^n with $n \geq 2$. In order to cover also the case $n = 1$, the initial language of this node consists of S. In N_4, every letter S is replaced by the symbol a before the word leaves to node and moves to the output node N_5.

Hence, $L(\mathcal{N}) = \{ a^{2^n} \mid n \geq 0 \}$.

Corollary 3. *$NIL \subset \mathcal{E}(MON)$ and $COMM \subset \mathcal{E}(MON)$.*

Proof. The inclusions follow from Corollary 1 and Theorem 7. The strictness follows from Lemma 4.

Finally, we give a result which can be understood as a lower bound for the generative power of monoidal filters.

Theorem 9. *Let L be a semi-linear language. Then $Comm(L) \in \mathcal{E}(MON)$.*

6 Conclusion

If we combine all the results of the preceding sections, we get the following diagram which we state as a theorem.

Theorem 10. *The following diagram holds.*

$$\begin{tikzcd}
 & \begin{array}{l} RE = \mathcal{E}(REG) = \mathcal{E}(PS) = \mathcal{E}(NC) \\ = \mathcal{E}(ORD) = \mathcal{E}(SUF) = \mathcal{E}(CIRC) \\ = \mathcal{E}(DEF) = \mathcal{E}(COMB) = \mathcal{E}(UF) \end{array} & \\
CF \arrow[ur] & & \\
REG \arrow[u] & & \begin{array}{l} \mathcal{E}(MON) = \mathcal{E}(NIL) \\ \quad = \mathcal{E}(COMM) \end{array} \arrow[uul] \\
\mathcal{E}(FIN) \arrow[u] \arrow[urr] & NIL \arrow[ul] \arrow[ur] & COMM \arrow[ull] \arrow[u] \\
FIN \arrow[u] \arrow[ur] & & MON \arrow[ul] \arrow[ull] \arrow[u]
\end{tikzcd}$$

The subregular classes considered in this paper are defined by combinatorial or algebraic properties of the languages. In [12], subclasses of REG defined by descriptional complexity have been considered. Let REG_n be the set of regular languages which can be accepted by deterministic finite automata. Then we have

$$REG_1 \subset REG_2 \subset REG_3 \subset \cdots \subset REG_n \subset \cdots \subset REG.$$

By Lemma 4 and [12], Lemma 4.1 and Theorems 4.3, 4.4., and 4.5, we get

$$\mathcal{E}(REG_1) \subset \mathcal{E}(MON) \subset \mathcal{E}(REG_2) = \mathcal{E}(REG_3) = \cdots = RE$$

and the incomparability of $\mathcal{E}(REG_1)$ with REG and CF.

References

1. Alhazov, A., Dassow, J., Martín-Vide, C., Rogozhin, Y., Truthe, B.: On networks of evolutionary processors with nodes of two types. Fundamenta Informaticae 91, 1–15 (2009)
2. Bordihn, H., Holzer, M., Kutrib, M.: Determinization of finite automata accepting subregular languages. Theoretical Computer Science 410, 3209–3222 (2009)
3. Brzozowski, J., Jirásková, G., Li, B.: Quotient complexity of ideal languages. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 208–221. Springer, Heidelberg (2010)

4. Castellanos, J., Martín-Vide, C., Mitrana, V., Sempere, J.M.: Solving NP-complete problems with networks of evolutionary processors. In: Mira, J., Prieto, A.G. (eds.) IWANN 2001. LNCS, vol. 2084, pp. 621–628. Springer, Heidelberg (2001)
5. Castellanos, J., Martín-Vide, C., Mitrana, V., Sempere, J.M.: Networks of evolutionary processors. Acta Informatica 39(6-7), 517–529 (2003)
6. Csuhaj-Varjú, E., Salomaa, A.: Networks of parallel language processors. In: Păun, G., Salomaa, A. (eds.) New Trends in Formal Languages. LNCS, vol. 1218, pp. 299–318. Springer, Heidelberg (1997)
7. Dassow, J.: Subregularly controlled derivations: the context-free case. Rostocker Mathematisches Kolloquium 34, 61–70 (1988)
8. Dassow, J.: Grammars with commutative, circular, and locally testable conditions. In: Automata, Formal Languages, and Related Topics – Dedicated to Ferenc Gécseg on the Occasion of his 70th Birthday, pp. 27–37. University of Szeged (2009)
9. Dassow, J., Hornig, H.: Conditional grammars with subregular conditions. In: Proc. Internat. Conf. Words, Languages and Combinatorics II, pp. 71–86. World Scientific, Singapore (1994)
10. Dassow, J., Manea, F., Truthe, B.: Networks of evolutionary processors with subregular filters. Technical report, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik (2011), `http://theo.cs.uni-magdeburg.de/pubs/preprints/pp-afl-2011-01.pdf`
11. Dassow, J., Stiebe, R., Truthe, B.: Generative capacity of subregularly tree controlled grammars. International Journal of Foundations of Computer Science 21, 723–740 (2010)
12. Dassow, J., Truthe, B.: On networks of evolutionary processors with filters accepted by two-state-automata. Fundamenta Informaticae (to appear)
13. Han, Y.S., Salomaa, K.: State complexity of basic operations on suffix-free regular languages. Theoretical Computer Science 410(27–29), 2537–2548 (2009)
14. Han, Y.S., Salomaa, K., Wood, D.: Nondeterministic state complexity of basic operations for prefix-suffix-free regular languages. Fundamenta Informaticae 90(1-2), 93–106 (2009)
15. Havel, I.M.: The theory of regular events II. Kybernetika 5(6), 520–544 (1969)
16. Holzer, M., Jakobi, S., Kutrib, M.: The magic number problem for subregular language families. In: Proceedings of 12th Internat. Workshop Descriptional Complexity of Formal Systems, pp. 135–146. University of Saskatchewan, Saskatoon (2010)
17. Jirásková, G., Masopust, T.: Complexity in union-free languages. In: Gao, Y., Lu, H., Seki, S., Yu, S. (eds.) DLT 2010. LNCS, vol. 6224, pp. 255–266. Springer, Heidelberg (2010)
18. Martín-Vide, C., Mitrana, V.: Networks of evolutionary processors: Results and perspectives. In: Molecular Computational Models: Unconventional Approaches, pp. 78–114 (2005)
19. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages. Springer, Berlin (1997)
20. Wiedemann, B.: Vergleich der Leistungsfähigkeit endlicher determinierter Automaten. Diplomarbeit, Universität Rostock (1978)