RAIRO-Theor. Inf. Appl. 55 (2021) 8
https://doi.org/ 10.1051/ita/2021004

RAIRO - Theoretical Informatics and Applications
www.rairo-ita.org

ACCEPTING NETWORKS OF EVOLUTIONARY PROCESSORS WITH RESOURCES RESTRICTED AND STRUCTURE LIMITED FILTERS*

Jürgen Dassow[1] and Bianca Truthe[2,**]

Abstract. In this paper, we continue the research on accepting networks of evolutionary processors where the filters belong to several special families of regular languages. We consider families of codes or ideals and subregular families which are defined by restricting the resources needed for generating or accepting them (the number of states of the minimal deterministic finite automaton accepting a language of the family as well as the number of non-terminal symbols or the number of production rules of a right-linear grammar generating such a language). We insert the newly defined language families into the hierachy of language families obtained by using as filters languages of other subregular families (such as ordered, non-counting, power-separating, circular, suffix-closed regular, union-free, definite, combinational, finite, monoidal, nilpotent, or commutative languages).

Mathematics Subject Classification. 68Q42, 68Q45.

Received December 17, 2019. Accepted March 25, 2021.

1. Introduction

Networks of language processors have been introduced in [3] by Csuhaj-Varjú and Salomaa. Such a network can be considered as a graph where the nodes represent processors which apply production rules to the words they contain. In a derivation step (an evolutionary step), any node derives from its language all possible words as its new language. In a communication step, any node sends copies of those words to other nodes which satisfy an output condition given as a regular language (called the output filter) and any node adopts (copies of) words sent by the other nodes if the words satisfy an input condition also given by a regular language (called the input filter).

Inspired by biological processes, in [1] a special type of networks of language processors was introduced. The nodes of such networks are called evolutionary processors because the allowed productions model the point mutation known from biology. The productions of a node allow that one letter is substituted by a letter, letters are inserted, or letters are deleted; the nodes are then called substitution nodes, insertion nodes, or deletion nodes, respectively.

* This paper is a revised and extended version of the paper presented at NCMA 2019 [21].

Keywords and phrases: Network, evolutionary processor, computational power, regular filter, descriptional complexity, codes, ideals.

[1] Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, Postfach 4120, 39106 Magdeburg, Germany.
[2] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany.

** Corresponding author: bianca.truthe@informatik.uni-giessen.de

Networks of evolutionary processors can be defined as language generating and language accepting devices. In case of a generating device, the processors start working with finite sets of axioms and all words which are in a designated processor at some time form the generated language. In case of an accepting device, input words are accepted if there is a computation which leads to a word in a designated processor.

Results on generating networks of evolutionary processors can be found, *e.g.*, in [1, 2, 14]. Accepting networks of evolutionary processors were introduced in [13]. Further results, especially on accepting networks where the filters belong to certain subclasses of the family of the regular languages, were published in [6, 12].

In [5], the generative capacity of networks of evolutionary processors was investigated for cases that all filters belong to a certain subfamily of the set of all regular languages. In [20], networks of evolutionary processors were investigated where the filters are restricted by bounded resources, namely the number of non-terminal symbols or the number of production rules which are necessary for generating the languages or the number of states of a minimal deterministic finite automaton over an arbitrary alphabet which are necessary for accepting the filters.

In [12], the computational power of accepting networks was studied in which the filters are languages from certain subfamilies of the set of all regular languages. It was shown that the use of ordered, non-counting, power-separating, suffix-closed regular, union-free, definite, and combinational languages as filters is as powerful as the use of arbitrary regular languages and yields networks which can accept all the recursively enumerable languages. On the other hand, when the filters are restricted to finite languages, nilpotent languages, monoids, commutative or circular regular languages, then not all recursively enumerable languages can be accepted. The results obtained and methods used are different to the case of generating networks.

In the present paper, the research on accepting networks of evolutionary processors is continued as in [20] for generating networks, namely the computational power is investigated when the resources for generating or accepting the filters are bounded. Further, we study accepting networks of evolutionary processors where th filters are ideals or codes. The language classes obtained by these types of filters are compared to those defined by structural properties (obtained in [12]).

2. Definitions

We assume that the reader is familiar with the basic concepts of formal language theory (see, *e.g.*, [15]). and recall here only some notations used in the paper.

2.1. Languages, grammars, and automata

Let V be an alphabet. By V^* we denote the set of all words (strings) over the alphabet V (including the empty word λ). For a natural number k, we denote by V^k the set of all words over the alphabet V with length k. The cardinality of a set A is denoted by $|A|$.

A phrase structure grammar is a quadruple $G = (N, T, P, S)$ where N is a finite set of non-terminal symbols, T is a finite set of terminal symbols, P is a finite set of production rules which are written as $\alpha \to \beta$ with $\alpha \in (N \cup T)^* \setminus T^*$ and $\beta \in (N \cup T)^*$, and $S \in N$ is the axiom. A grammar is right-linear if, for any rule $\alpha \to \beta$, the left-hand side α consists of a non-terminal symbol only and the right-hand side β contains at most one non-terminal symbol and this is at the right end of the word: $\alpha \in N$ and $\beta \in T^* \cup T^*N$. A special case of right-linearity is regularity where each rule contains exactly one terminal symbol (with the only possible exception $S \to \lambda$). Let $G = (N, T, P, S)$ be a grammar. A word $v \in (N \cup T)^*$ is derived in one step from a word $u \in (N \cup T)^*$ by the grammar G, written as $u \Longrightarrow_G v$, if there are a rule $\alpha \to \beta \in P$ and two subwords x and y of u such that $u = x\alpha y$ and $v = x\beta y$. By $\Longrightarrow_G^*$, we denote the reflexive and transitive closure of the derivation relation $\Longrightarrow_G$. The language $L(G)$ generated by the grammar G is the set of all words which consist of terminal symbols and which are derivable from the axiom S:

$$L(G) = \{\, w \mid w \in T^* \text{ and } S \Longrightarrow_G^* w \,\}.$$

If G is clear from the context, we write $\Longrightarrow$ and $\Longrightarrow^*$ instead of $\Longrightarrow_G$ and $\Longrightarrow_G^*$, respectively.

Regular and right-linear grammars generate the same family of languages (the regular languages). Therefore, also right-linear grammars are often called regular. In the context of descriptional complexity, when the number of non-terminal symbols or the number of production rules which are necessary for generating a language are considered then there is a difference whether a language is generated by means of regular or right-linear rules. We use in this paper right-linear grammars.

By REG and RE, we denote the families of languages generated by regular and phrase structure grammars, respectively.

A finite automaton is a quintuple $\mathcal{A} = (V, Z, z_0, F, \delta)$ where V is an alphabet called the input alphabet, Z is a non-empty finite set of elements which are called states, $z_0 \in Z$ is the so-called start state, $F \subseteq Z$ is the set of accepting states, and $\delta : Z \times V \to \mathcal{P}(Z)$ is a mapping which is also called the transition function where $\mathcal{P}(Z)$ denotes the power set of Z (the set of all subsets of Z). A finite automaton is called deterministic if every set $\delta(z, a)$ for $z \in Z$ and $a \in V$ is a singleton set.

The transition function δ can be extended to a function $\delta^* : Z \times V^* \to \mathcal{P}(Z)$ where $\delta^*(z, \lambda) = \{z\}$ and

$$\delta^*(z, va) = \bigcup_{z' \in \delta^*(z,v)} \delta(z', a).$$

We will use the same symbol δ in both the original and extended version of the transition function.

Let $\mathcal{A} = (V, Z, z_0, F, \delta)$ be a finite automaton. A word w is accepted by the finite automaton $\mathcal{A}$ if and only if the automaton has reached an accepting state after reading the input word w, *i.e.*, $\delta(z_0, w) \cap F \neq \emptyset$. The language accepted by $\mathcal{A}$ is defined as

$$L(A) = \{\, w \mid \delta(z_0, w) \cap F \neq \emptyset \,\}.$$

The family of the languages accepted by finite automata is equal to the family of the regular languages.

2.2. Ideals and codes

In the sequel, let V be an alphabet. We first introduce the notion of an ideal in V^* from the theory of rings and semigroups.

A non-empty language $L \subseteq V^*$ is called a right (left) ideal if and only if, for any word $v \in V^*$ and any word $u \in L$, we have $uv \in L$ ($vu \in L$, respectively). It is easy to see that the language L is a right (left) ideal if and only if there is a language L' such that $L = L'V^*$ ($L = V^*L'$, respectively).

We now present some notions from coding theory, especially some special codes. For details, we refer to [11, 16].

For a word $x \in V^*$, let

$$E(x) = \{\, y \mid y \in V^+,\ vyv' = x \text{ for some } v, v' \in V^* \,\},$$

(*i.e.*, $E(x)$ is the set of all non-empty subwords of x).

A language $L \subseteq V^*$ is called

– a code if and only if, for any numbers $n \geq 1$, $m \geq 1$, and words

$$x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_m \in L$$

such that

$$x_1 x_2 \ldots x_n = y_1 y_2 \ldots y_m,$$

we have the equalities $n = m$ and $x_i = y_i$ for $1 \leq i \leq n$ (*i.e.*, a word of L^* has a unique decomposition into code words.

– a solid code if and only if, for any numbers $n \geq 1$, $m \geq 1$, words

$$x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_m \in L,$$

and words

$$v_1, v_2, \ldots, v_{n+1}, w_1, w_2, \ldots, w_{m+1}$$

with $E(v_i) \cap L = \emptyset$ for $1 \leq i \leq n+1$, and $E(w_j) \cap L = \emptyset$ for $1 \leq j \leq m+1$ such that

$$v_1 x_1 v_2 x_2 \ldots v_n x_n v_{n+1} = w_1 y_1 w_2 y_2 \ldots w_m y_m w_{m+1},$$

we have $n = m$, $x_i = y_i$ for $1 \leq i \leq n$, and $v_j = w_j$ for $1 \leq j \leq n+1$;

– uniform if and only if $L \subseteq V^n$ for some $n \geq 1$ (all words have the same length);

– prefix if and only if, for any words $u \in L$ and $v \in V^*$ such that $uv \in L$, we have $v = \lambda$ (*i.e.*, any proper prefix of a word in L is not in L);

– suffix if and only if, for any words $u \in L$ and $v \in V^*$ such that $vu \in L$, we have $v = \lambda$ (*i.e.*, any proper suffix of a word in L is not in L);

– bifix if and only if it is prefix as well as suffix;

– infix if and only if, for any $u \in L$, and $v, v' \in V^*$ such that $vuv' \in L$, we have $v = v' = \lambda$ (*i.e.*, any proper subword of a word in L is not in L).

Note that uniform, prefix, suffix, bifix, and infix languages are codes.

A code $L \subseteq V^*$ is called

– outfix if and only if, for any words $u \in V^*$ and $v, v' \in V^*$ such that $vv' \in L$ and $vuv' \in L$, we have $u = \lambda$;

– reflective if and only if, for any words $u, v \in V^*$ such that $uv \in L$, we have $vu \in L$.

By *rId*, *lId*, *C*, *SC*, *PfC*, *SfC*, *BfC*, *IfC*, *OfC*, *RC*, and *UC*, we denote the families of regular right ideals, regular left ideals, regular codes, regular solid codes, regular prefix codes, regular suffix codes, regular bifix codes, regular infix codes, regular outfix codes, regular reflective codes and uniform codes, respectively.

In [4], it is proved that any uniform code, any regular outfix code, and any regular reflective code is finite. Further relations, expecially those depicted in Figure 1, are proved in [4, 11], and [16].

Lemma 2.1. *The hierarchy of the classes REG, rId, lId, C, SC, PfC, SfC, BfC, IfC, OfC, RC, and UC is presented in Figure 1.*

2.3. Other subregular language families

For a language L over V, we set

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

In [12], the following restrictions for regular languages are considered. In order to relate our results of this paper to the results there, we explain here those special regular languages. Let L be a language and $V = alph(L)$ the minimal alphabet of L. We say that the language L, with respect to the alphabet V, is

– *monoidal* if $L = V^*$,

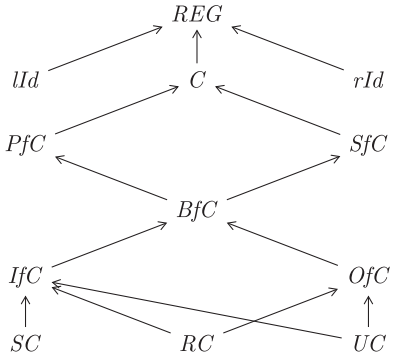


FIGURE 1. Hierarchy of regular ideals and codes (an arrow from X to Y denotes $X \subset Y$; if two families are not connected by a directed path, then they are incomparable).

- *combinational* if it has the form $L = V^*A$ for some subset $A \subseteq V$,
- *definite* if it can be represented in the form $L = A \cup V^*B$ where A and B are finite subsets of V^*,
- *nilpotent* if L is finite or $V^* \setminus L$ is finite,
- *commutative* if $L = Comm(L)$,
- *circular* if $L = Circ(L)$,
- *suffix-closed* if the relation $xy \in L$ for some words $x, y \in V^*$ implies that also the suffix y belongs to L or equivalently, $L = Suf(L)$,
- *non-counting* (or star-free) if there is an integer $k \geq 1$ such that, for any $x, y, z \in V^*$, the relation $xy^k z \in L$ holds if and only if also $xy^{k+1}z \in L$ holds,
- *power-separating* if for any word $x \in V^*$ there is a natural number $m \geq 1$ such that either the equality $J_x^m \cap L = \emptyset$ or the inclusion $J_x^m \subseteq L$ holds where $J_x^m = \{\, x^n \mid n \geq m \,\}$,
- *ordered* if 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 $a \in V$, $z \preceq z'$ implies $\delta(z, a) \preceq \delta(z', a)$,
- *union-free* if L can be described by a regular expression which is only built by product and star.

Among the commutative, circular, suffix-closed, non-counting, and power-separating languages, we consider only those which are also regular.

By *FIN*, *MON*, *COMB*, *DEF*, *NIL*, *COMM*, *CIRC*, *SUF*, *NC*, *PS*, *ORD*, and *UF*, we denote the families of all finite, monoidal, combinational, definite, nilpotent, regular commutative, regular circular, regular suffix-closed, regular non-counting, regular power-separating, ordered, and union-free languages, respectively.

In this paper, families of languages are of special interest which are defined by bounding the resources which are necessary for accepting or generating these languages.

Let *RLG* be the set of all right-linear grammars and *DFA* the set of all deterministic finite automata. Further, let

$$G = (N, T, P, S) \in RLG \quad \text{and} \quad \mathcal{A} = (V, Z, z_0, F, \delta) \in DFA.$$

Then we define the following measures of descriptional complexity:

$$Var(G) = |N|, \quad Prod(G) = |P|, \quad State(\mathcal{A}) = |Z|.$$

For these complexity measures, we define the following families of languages (we abbreviate the measure *Var* by V, the measure *Prod* by P, and the measure *State* by Z):

$$RL_n^V = \{\, L \mid \exists G \in RLG : L = L(G) \text{ and } Var(G) \leq n \,\},$$

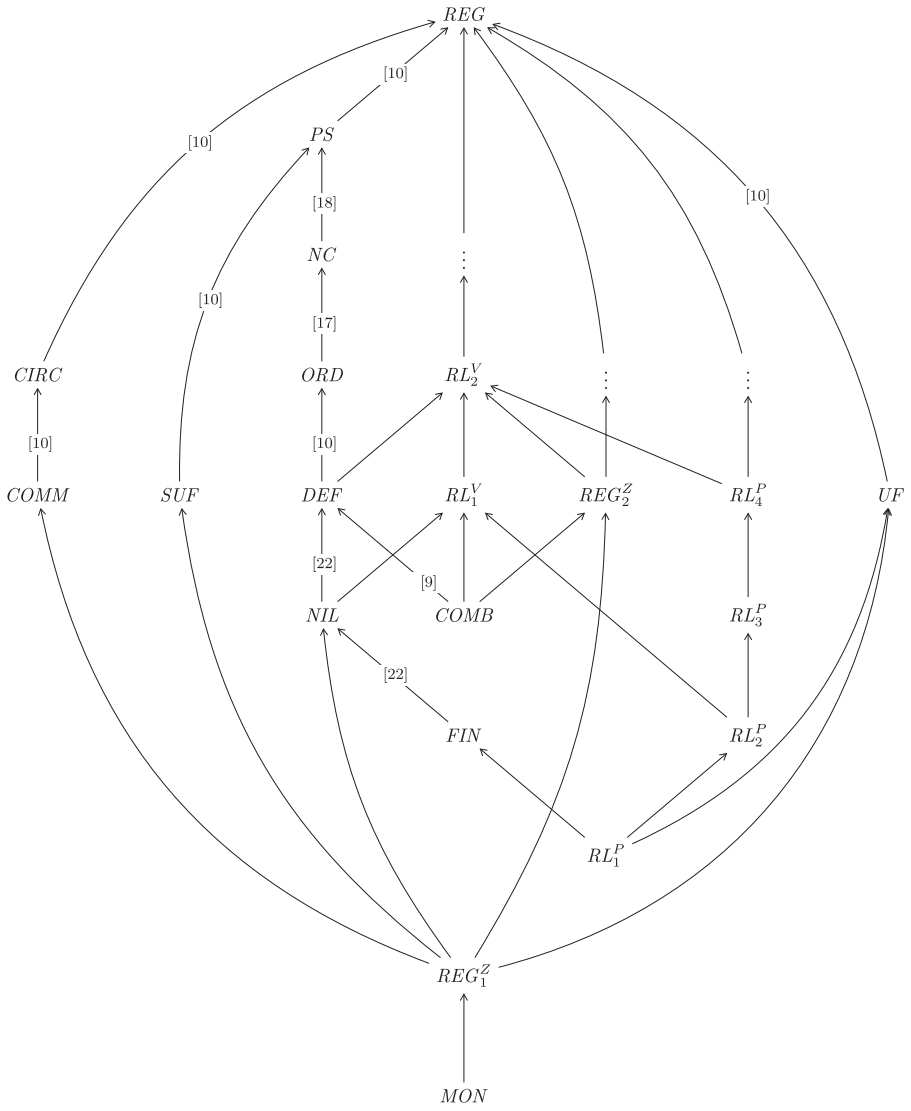


FIGURE 2. Hierarchy of further subregular language families. An edge label refers to the paper where the respective inclusion is proved. The proper inclusions where no reference is given as well as the incomparabilities are proved in [19].

$$RL_n^P = \{\, L \mid \exists G \in RLG : L = L(G) \text{ and } Prod(G) \leq n \,\},$$
$$REG_n^Z = \{\, L \mid \exists \mathcal{A} \in DFA : L = L(\mathcal{A}) \text{ and } State(\mathcal{A}) \leq n \,\}.$$

The relations between the considered families are investigated, *e.g.*, in [9, 10, 17–19, 22]. They are illustrated in Figure 2.

Regarding the families defined by bounded resources, we note the following relations: $K_i \subset K_{i+1}$ for $K \in \{RL^V, RL^P, REG^Z\}$ and $i \geq 1$ as well as $RL_{2i}^P \subset RL_i^V$ and $REG_i^Z \subset RL_i^V$.

Lemma 2.2. *The inclusion relations presented in Figure 2 hold. An arrow from an entry X to an entry Y depicts the proper inclusion $X \subset Y$; if two families are not connected by a directed path, then they are incomparable.*

2.4. Accepting networks of evolutionary processors

We call a production $\alpha \to \beta$ a substitution if $|\alpha| = |\beta| = 1$ and deletion if $|\alpha| = 1$ and $\beta = \lambda$. 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 $w = x\beta y$.

A production $\lambda \to a$, where a is a letter, is called an insertion. 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.

In order to indicate also the applied rule p in a derivation step, we write $v \Longrightarrow_p w$. For a set P of rules, we write $v \Longrightarrow_P w$ if and only if $v \Longrightarrow_p w$ for some rule $p \in P$. The reflexive and transitive closure of the relation is denoted by $\Longrightarrow_P^*$: we write $x \Longrightarrow_P^* y$ if there are rules $p_1, \ldots, p_n$ $(n \geq 0)$ in P and words $w_0, \ldots, w_n$ such that $x = w_0$, $w_i \Longrightarrow_{p_{i+1}} w_{i+1}$ for $i = 0, \ldots, n-1$, and $w_n = y$. If at least one rule is applied, we write $x \Longrightarrow_P^+ y$.

We now present the basic concept of this paper, namely accepting networks of evolutionary processors (ANEPs for short).

Definition 2.3. Let X be a family of regular languages and n a natural number.

1. An accepting network of evolutionary processors of size n and with filters from the family X is an $(n+5)$-tuple $\mathcal{N} = (V, U, N_1, N_2, \ldots, N_n, E, n_{\mathrm{i}}, n_{\mathrm{o}})$ where
   - V is a finite alphabet, called the input alphabet of the network,
   - U is a finite alphabet with $V \subseteq U$, called the working alphabet of the network,
   - $N_i = (M_i, I_i, O_i)$ for $1 \leq i \leq n$ are the processors where
     - M_i is a set of rules of a certain type: $M_i \subseteq \{\, a \to b \mid a, b \in U \,\}$ or $M_i \subseteq \{\, a \to \lambda \mid a \in U \,\}$ or $M_i \subseteq \{\, \lambda \to b \mid b \in U \,\}$,
     - I_i and O_i are languages from the class X over some subset of the alphabet U; the set I_i is called the input filter and O_i the output filter of the processor,
   - E is a subset of $\{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$, and
   - n_{i} and n_{o} are two natural numbers from the set $\{1, 2, \ldots, n\}$; the processor $N_{n_{\mathrm{i}}}$ is called the input node and $N_{n_{\mathrm{o}}}$ the output node of the network.
2. A configuration C of an ANEP $\mathcal{N}$ is an n-tuple $C = (C(1), C(2), \ldots, C(n))$ where $C(i)$ is a subset of U^* for $1 \leq i \leq n$.
3. 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 (no left-hand side of a rule is present in the word w; note that insertion rules can always be applied since the empty word is always a subword) 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 ANEP $\mathcal{N}$ on an input word $w \in V^*$ is a sequence of configurations $C_t^w = (C_t^w(1), C_t^w(2), \ldots, C_t^w(n))$, $t \geq 0$, such that
   - $C_0^w(n_{\mathrm{i}}) = \{w\}$ and $C_0^w(j) = \emptyset$ for $j \in \{1, \ldots, n\} \setminus \{n_{\mathrm{i}}\}$,
   - for any $t \geq 0$, C_{2t}^w derives C_{2t+1}^w in one evolutionary step,
   - for any $t \geq 0$, C_{2t+1}^w derives C_{2t+2}^w in one communication step.

   The computation of an ANEP $\mathcal{N}$ on an input word $w \in V^*$ is said to be accepting if there exists a step $t \geq 0$ in which the component $C_t^w(n_{\mathrm{o}})$ of the configuration representing the content of the output node is not empty.

4. The language $L(\mathcal{N})$ accepted by $\mathcal{N}$ is defined as

$$L(\mathcal{N}) = \{ w \mid w \in V^* \text{ and the computation of } \mathcal{N} \text{ on } w \text{ is accepting} \}.$$

In the sequel, we also use the terms evolutionary network or network or net instead of ANEP.

Intuitively, an accepting network with n evolutionary processors is a graph consisting of n nodes $N_1, \ldots, N_n$ (also called processors) and a set of edges given by E such that there is a directed edge from node N_k to node N_i if and only if $(k, i) \in E$. Among the processors, two are distinguished: an input processor N_{n_i} and an output processor N_{n_o}. Any processor N_i consists of a set M_i of evolutionary rules, an input filter I_i, and an output filter O_i. We say that N_i is

- a substitution node if $M_i \subseteq \{ a \to b \mid a, b \in U \}$ (by any rule, a letter is substituted by another one),
- a deletion node if $M_i \subseteq \{ a \to \lambda \mid a \in U \}$ (by any rule, a letter is deleted), or
- an insertion node if $M_i \subseteq \{ \lambda \to b \mid b \in U \}$ (by any rule, a letter is inserted).

Every node has only one type of rules. 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. We say that an accepting network has filters from a subclass X of regular languages if both filters of any processor belong to the class X. 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, the node N_{n_i} contains exactly one word – the input word; all other nodes do not contain any word. In an evolutionary step, we derive from $C_t(i)$ all words by applying rules from the set M_i to the present words (all words, for which there is no rule applicable, survive an evolutionary step; to the others, any applicable rule is applied). In a communication step, any processor N_i sends out copies of all words from the set $C_t(i) \cap O_i$ (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 whose copies are sent by N_k and pass the input filter I_i of N_i, *i.e.*, the processor N_i gets in addition all words of $C_t(k) \cap O_k \cap I_i$. The input filter of a node controls which words are allowed to enter the node and the output filter controls which words are allowed to leave the node. Those words which do not leave a node remain there. Those words which leave a node move to every adjacent node. There, a word enters the processor if it passes the input filter, otherwise it is lost (it vanishes from the network). We start with an evolutionary step and then communication steps and evolutionary steps are alternately performed. An input word is accepted by a network if and only if there exists a step in the computation in which the output processor contains at least one string. The accepted language consists of all accepted words. In what follows, we do not distinguish between nodes and processors. If two ANEPs accept the same language, we say that the networks are equivalent to each other.

For a family X, we denote the family of languages accepted by networks of evolutionary processors where all filters are of type X by $\mathcal{A}(X)$. In this paper, we assume that a filter language belongs to some family X if it belongs to it with respect to its smallest alphabet, not necessarily to the alphabet of all letters which might occur in the node or even in the entire network. A word passes a filter if it is an element of the language representing the filter otherwise it does not pass the filter.

The following theorem is known (see [12]).

Theorem 2.4 ([12]). *We have* $\mathcal{A}(REG) = RE$.

This result appeared already earlier in other publications, however, [12] is the first paper where the evolutionary step is defined as here (in the present paper, in an evolutionary step, a word remains only in a node if there is no rule which can be applied to it or, for some letter a in the word, $a \to a$ is a rule associated to the node; this is in contrast to earlier literature, where a word also remains, if there are a rule which cannot be applied and a another rule which can be applied).

Also the following two lemmas are from [12] and will be used here as well.

Lemma 2.5 ([12]). *Let X and Y be two families of languages such that $X \subseteq Y$. Then the inclusion $\mathcal{A}(X) \subseteq \mathcal{A}(Y)$ holds.*

Lemma 2.6 ([12]). *Let X be a subclass of REG. Then the inclusion $X \subseteq \mathcal{A}(X)$ holds.*

3. Results

We investigate the impact of filters defined by restricting resources or by properties as being an ideal or a code for the computational power. We compare the language families obtained in such a way to those investigated earlier in the literature (see [12]).

3.1. Resources restricted filters

We first consider the number of non-terminal symbols sufficient for generating a language. We obtain that any recursively enumerable language is accepted by a network of evolutionary processors were each filter is generated by a right-linear grammar with only one non-terminal symbol.

Theorem 3.1. *We have $\mathcal{A}(RL_i^V) = RE$ for all natural numbers $i \geq 1$.*

Proof. Every combinational language can be generated by a regular grammar with only one non-terminal symbol: For generating a language V^*A, the rules $S \to vS$ for every letter $v \in V$ and $S \to a$ for every letter $a \in A$ are sufficient ([19]). According to Lemma 2.5, the chain of inclusions

$$COMB \subseteq RL_1^V \subseteq RL_2^V \subseteq \cdots \subseteq REG$$

implies the chain of inclusions

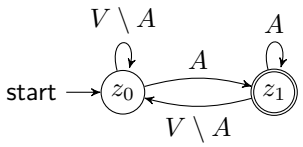
$$\mathcal{A}(COMB) \subseteq \mathcal{A}(RL_1^V) \subseteq \mathcal{A}(RL_2^V) \subseteq \cdots \subseteq \mathcal{A}(REG).$$

In [12], the relation $\mathcal{A}(COMB) = RE$ was proved. Hence, together with Theorem 2.4, we also have $\mathcal{A}(RL_i^V) = RE$ for all natural numbers $i \geq 1$. □

We now consider networks where the filters are accepted by deterministic finite automata with a bounded number of states. Analogously to Theorem 3.1, we obtain a similar result.

Theorem 3.2. *We have $\mathcal{A}(REG_i^Z) = RE$ for all natural numbers $i \geq 2$.*

Proof. Every combinational language can be accepted by a deterministic finite automaton with two states only: A language V^*A with an alphabet V and a subset $A \subseteq V$ is accepted by an automaton whose transition graph is as follows ([19]):



According to Lemma 2.5, the chain of inclusions

$$COMB \subseteq REG_2^Z \subseteq REG_3^Z \subseteq \cdots \subseteq REG$$

implies the chain of inclusions

$$\mathcal{A}(COMB) \subseteq \mathcal{A}(REG_2^Z) \subseteq \mathcal{A}(REG_3^Z) \subseteq \cdots \subseteq \mathcal{A}(REG).$$

Since $\mathcal{A}(COMB) = RE = \mathcal{A}(REG)$ (by [12] and Thm. 2.4), also the relation $\mathcal{A}(REG_i^Z) = RE$ holds for every natural number $i \geq 2$. □

For networks with filters which are accepted by deterministic finite automata with one state only, the situation is different.

Theorem 3.3. *We have* $\mathcal{A}(REG_1^Z) = \mathcal{A}(MON)$.

Proof. The inclusion $MON \subset REG_1^Z$ ([19]) implies the inclusion

$$\mathcal{A}(MON) \subseteq \mathcal{A}(REG_1^Z)$$

(according to Lem. 2.5). We now prove that the inverse inclusion

$$\mathcal{A}(REG_1^Z) \subseteq \mathcal{A}(MON)$$

holds as well.

Every language of the family REG_1^Z is the empty set or a monoidal language. A network with filters in REG_1^Z which contains empty sets as filters can be transformed into an equivalent network which has only monoidal languages as filters, which can be seen as follows.

If the input filter of a node N is the empty set, then no word can enter this node. Thus, we can remove all edges that lead to the node N and set the input filter to an arbitrary monoidal language without changing the possible communications. If the output filter of a node is the empty set, then no word can leave this node. If this is the case for a node N which is not the output node, then this node N is useless in that sense that it does not contribute to the acceptance of a language. So, we can eliminate this node together with all incident edges without changing the language accepted. If the output node has an empty output filter, then we replace this filter by any monoidal language and cancel all its outgoing edges (once a word is present in this node, the input word is accepted, so there is no need that the computation continues; hence, the output filter has no effect on the acceptance of a language). This new network accepts the same language and has only monoidal filters. Hence, also the inclusion $\mathcal{A}(REG_1^Z) \subseteq \mathcal{A}(MON)$ holds, and, thus, also the equality $\mathcal{A}(REG_1^Z) = \mathcal{A}(MON)$. □

We finally investigate networks where the filters are restricted by the number of production rules necessary for generating the filters. Here, we obtain an infinite hierarchy. Further inclusion relations and incomparability results are proved in the sequel.

First, we give a lemma regarding the number of production rules $Prod(L)$ (for some regular language L) which are necessary for a right-linear grammar generating the language L. Note that

$$Prod(L) = \min\{\, Prod(G) \mid L = L(G) \,\}.$$

Lemma 3.4. *Let T and T' be two alphabets and $h : T^* \to (T')^*$ be a homomorphism. Then, for every language $L \subseteq T^*$, we have $Prod(h(L)) \leq Prod(L)$.*

Proof. Let $G = (N, T, P, S)$ be a right-linear grammar such that $L(G) = L$ and $Prod(G) = Prod(L)$. Then the grammar $G' = (N, T', P', S)$ with

$$P' = \{\, A \to h(w) \mid w \in T^* \text{ and } A \to w \in P \,\} \cup \{\, A \to h(w)B \mid B \in N \text{ and } A \to wB \in P \,\}$$

is right-linear and satisfies $L(G') = h(L)$. Thus,

$$Prod(h(L)) \leq |P'| \leq |P| = Prod(G) = Prod(L).$$

□

The next lemmas give lower bounds for the number of rules of right-linear grammars generating special languages (these will later be used to prove that some output filters cannot be generated with fewer rules).

A rule in a right-linear grammar is called a T-rule if and only if it has not the form $A \to B$ or $A \to \lambda$, where A and B are non-terminal symbols, (*i.e.*, by the application of a T-rule, at least one terminal symbol is produced).

Lemma 3.5. *Let $a_1, a_2, \ldots, a_i$ be i pairwise different letters and T an alphabet with $a_j \in T$ for $1 \leq j \leq i$. For every natural number j with $1 \leq j \leq i$, let $z_j \in \{a_j\}^* T \{a_j\}^*$ be a word over T with $|z_j| \geq 3$. Further, let $L \subseteq T$ be a regular language such that $\{z_1, z_2, \ldots, z_i\} \subseteq L$. Then it holds $Prod(L) \geq i$.*

Proof. Let everything be defined as stated in the lemma and let $G = (N, T, P, S)$ be a right-linear grammar generating the language L. Further, let I be the subset of all indices $j \in \{1, 2, \ldots, i\}$ such that there is a derivation D_j of the word z_j by rules of a subset $P_j \subseteq P$ such that all T-rules of this set P_j are not used in any derivation of any other word z_k with $1 \leq k \leq i$, $k \neq j$. For $j \in I$, we choose q_j as the first T-rule which is applied in the derivation D_j.

Let $j \notin I$. Then there is a derivation D'_j of z_j in which a T-rule $A \to wB$ or $A \to w$ with $w \in T^+$ is applied which is also applied in some derivation of another word z_k. But then w is a subword of z_j as well as of z_k. By the structure of the words z_p with $1 \leq p \leq i$, we obtain that $|w| \leq 2$ (a longer subword of z_j contains at least two letters a_j but no other word contains a_j twice). Hence, in the derivation D'_j, a T-rule is applied which produces a word of $\{a_j\}^+$ (the word w itself or a subword of the remaining part of z_j). We choose q_j as this rule.

By construction, the rules $q_1, q_2, \ldots, q_i$ are pairwise different. This implies for the number of rules $Prod(G) \geq i$. If the grammar G is minimal, we obtain also $Prod(L) = Prod(G) \geq i$. □

Lemma 3.6. *Let $a_1, a_2, \ldots, a_i$ be i pairwise different letters and T an alphabet with $a_j \in T$ for $1 \leq j \leq i$. Moreover, let $t > 0$ be a natural number and, for every natural number j with $1 \leq j \leq i$, let $z_j \in \{a_j\}^* T \{a_j\}^*$ be a word over T with $|z_j| \geq 3t$. Further, let $L \subseteq T$ be a regular language such that $\{z_1, z_2, \ldots, z_i\} \subseteq L$ and let $G = (N, T, P, S)$ be a right-linear grammar which generates the language L and where the longest right-hand side of a rule in P has a length of at most t. Then it holds $Prod(G) > i$.*

Proof. Let everything be defined as stated in the lemma. For every word z_j with $1 \leq j \leq i$, there is a derivation

$$S \Longrightarrow^* w_{j,1} \Longrightarrow^* w_{j,2} \Longrightarrow^* z_j$$

where $1 \leq |w_{j,1}| \leq t$ and $|w_{j,1}| + 1 \leq |w_{j,2}| \leq 2t$. If $w_{j,1}$ consists of the letter a_j only, we denote the first T-rule applied in the derivation by q_j. Otherwise, we take as q_j the first T-rule which is applied in the sub-derivation $w_{j,1} \Longrightarrow^* w_{j,2}$. In this way, the right-hand side of any rule q_j is from the set $\{a_j\}^+ N$ (any rule q_j is not terminating since $|w_{j,2}| \leq 2t$ whereas $|z_j| = 3t$). Additionally, we take as q_{i+1} the terminating rule applied in the derivation D_1.

By construction, all rules $q_1, q_2, \ldots, q_{i+1}$ are pairwise different. This implies for the number of rules $Prod(G) \geq i + 1$. If the grammar G is minimal, we obtain also $Prod(L) = Prod(G) \geq i + 1$. □

We now present some witness languages for separating language families.

Lemma 3.7. *Let $L = \{a, aaa\}$. Then $L \in (\mathcal{A}(FIN) \cap \mathcal{A}(RL_2^P)) \setminus \mathcal{A}(RL_1^P)$ holds.*

Proof. The language L is finite and can be generated by a right-linear grammar with two rules. According to Lemma 2.6, it holds $L \in \mathcal{A}(FIN) \cap \mathcal{A}(RL_2^P)$.

Assume that also $L \in \mathcal{A}(RL_1^P)$ holds. Then there is an ANEP

$$\mathcal{N} = (\{a\}, U, N_1, N_2, \ldots, N_n, E, n_{\mathrm{i}}, n_{\mathrm{o}})$$

with $L(\mathcal{N}) = L$ and where every filter contains at most one word. If the input and output node coincide ($n_\mathrm{i} = n_\mathrm{o}$), then the language $\{a\}^*$ is accepted which contradicts the assumption $L(\mathcal{N}) = L$. Hence, the input and output node are different. We distinguish now the possibilities for the input node $N_{n_\mathrm{i}} = (M_{n_\mathrm{i}}, I_{n_\mathrm{i}}, O_{n_\mathrm{i}})$:

1. The node has no rules. Then $L(\mathcal{N}) \subseteq O_{n_\mathrm{i}}$. Since the output filter O_{n_i} contains at most one word, the language L is not accepted.
2. The input node is a deletion node. If $O_{n_\mathrm{i}} \not\subseteq \{a\}^*$, then $L(\mathcal{N}) = \emptyset$, which is a contradiction. Hence, the filter O_{n_i} contains a word which consists of letters a only. Then $L(\mathcal{N}) \subseteq \{a\}^*\{w\}\{a\}^*$. Due to the assumption that $L(\mathcal{N}) = L$, we have $w = a$ (in case that a is not deleted in this node) or $w = \lambda$ (in case that the rule $a \to \lambda$ exists). In the first case, the word input word aaa cannot be changed. Since, it cannot leave the node, it is not accepted. In the second case, both the input words aa and aaa are derived to w by deletion, and hence, both are accepted, which is also a contradiction.
3. The input node is a substitution node. If $O_{n_\mathrm{i}} = \emptyset$, then $L(\mathcal{N}) = \emptyset$, which is a contradiction. Otherwise, $O_{n_\mathrm{i}} = \{w\}$ for a word $w \in U^*$. Since a^m with $m = |w|$ is the only input word which can be derived to the word w by substitution rules, this is the only word for which it is possible that it is accepted. This is again a contradiction.
4. The input node is an insertion node. If $O_{n_\mathrm{i}} = \emptyset$, then $L(\mathcal{N}) = \emptyset$, which is a contradiction. Otherwise, $O_{n_\mathrm{i}} = \{w\}$ for a word $w \in U^*$. The only input words which can be derived to the word w by insertion rules are a^m with $m < |w|$. Hence, $L(\mathcal{N}) \subseteq \{\, a^m \mid 0 \leq m < |w| \,\}$. Since, by assumption, $L(\mathcal{N}) = L$, it follows $|w| \geq 4$. Furthermore, aaa is a scattered subword of the word w (because $aaa \in L$ and, by insertion rules, only letters can be added to obtain the word w). Since also a must be derived to the word w, the node contains the rule $\lambda \to a$. But then, also the input word aa can be derived to the word w (by one insertion of a more than for the input word aaa). Thus, it holds $aa \in L(\mathcal{N})$ if and only if $aaa \in L(\mathcal{N})$, which is also a contradiction.

Since every possibility yields a contradiction, the assumption is wrong. Hence, we obtain the claim $L \notin \mathcal{A}(RL_1^P)$. □

The language from the previous lemma serves as a witness language for some inclusion relations.

Theorem 3.8. *The proper inclusions $\mathcal{A}(RL_1^P) \subset \mathcal{A}(FIN)$ and $\mathcal{A}(RL_1^P) \subset \mathcal{A}(RL_2^P)$ hold.*

Proof. The inclusions $\mathcal{A}(RL_1^P) \subseteq \mathcal{A}(FIN)$ and $\mathcal{A}(RL_1^P) \subseteq \mathcal{A}(RL_2^P)$ follow from Lemma 2.2 and Lemma 2.5. A witness language for the properness is in both cases the language $L = \{a, aaa\}$ as shown in Lemma 3.7. □

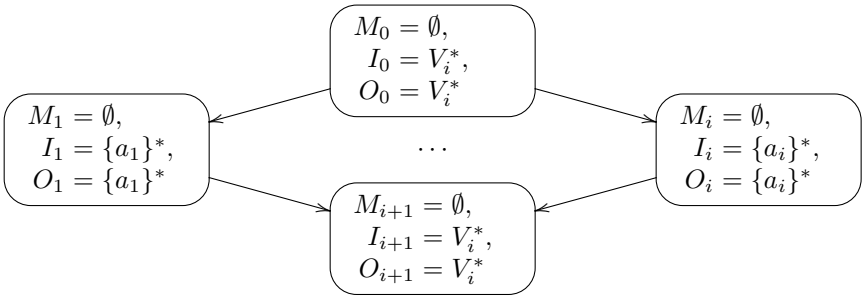
We now consider a sequence of languages which will also serve as witness languages.

Lemma 3.9. *For any natural number $i \geq 2$, let $V_i = \{a_1, a_2, \ldots, a_i\}$ be an alphabet with i letters. Further, let*

$$L_i = \bigcup_{k=1}^{i} \{a_k\}^*.$$

Then $L_i \in (\mathcal{A}(RL_{i+1}^P) \cap \mathcal{A}(MON)) \setminus \mathcal{A}(RL_i^P)$.

Proof. Let $i \geq 2$ be a natural number. The language L_i is generated by the following network with $i + 2$ nodes where the index of the input node is $n_\mathrm{i} = 0$ and that of the output node $n_\mathrm{o} = i + 1$:



The language accepted is L_i because any word a_k^m with $1 \leq k \leq i$ and $m \geq 0$ moves from the input node to node N_k and further to the output node and any other word is lost in the first communication step. Each filter in the presented network is a monoidal language and belongs to the family RL_{i+1}^P (the language V_i^* can be generated by the $i+1$ regular rules $S \to xS$ for $x \in V_i$ and $S \to \lambda$; any language $\{a_k\}^*$ for $1 \leq k \leq i$ can be generated by the two rules $S \to a_k S$ and $S \to \lambda$). Hence, $L_i \in \mathcal{A}(RL_{i+1}^P) \cap \mathcal{A}(MON)$.

Assume that also $L_i \in \mathcal{A}(RL_i^P)$ holds. Then there is an ANEP

$$\mathcal{N} = (V_i, U, N_1, \ldots, N_n, E, n_{\mathrm{i}}, n_{\mathrm{o}})$$

with $L(\mathcal{N}) = L_i$ and where every filter is generated by a right-linear grammar with at most i rules. If the input and output node coincide, then the language V_i^* is accepted which contradicts the assumption $L(\mathcal{N}) = L_i$. Hence, the input and output node are different. We consider the input node $N_{n_{\mathrm{i}}} = (M_{n_{\mathrm{i}}}, I_{n_{\mathrm{i}}}, O_{n_{\mathrm{i}}})$ and prove that already the output filter $O_{n_{\mathrm{i}}}$ of this node cannot be generated by a grammar with at most i production rules. Let $G_{n_{\mathrm{i}}}$ be a right-linear grammar which generates the filter $O_{n_{\mathrm{i}}}$.

We distinguish some cases with respect to the rule set $M_{n_{\mathrm{i}}}$:

1. No rule $x \to \lambda$ with $x \in V_i$, no rule $x \to y$ with $x \in V_i$ and $y \in U$, and no rule $\lambda \to y$ with $y \in U$ is contained in $M_{n_{\mathrm{i}}}$. Then input words cannot be changed in the node $N_{n_{\mathrm{i}}}$, but have to leave it in order to yield a word arriving in the output node after some steps. Thus, especially all words a_j^{3t} with $1 \leq j \leq i$ are contained in $O_{n_{\mathrm{i}}}$. Since the generation of these words by a right-linear grammar requires at least $i+1$ rules (one for each first letter and a terminating rule because these words cannot be derived in one step due to the choice of the length), we obtain a contradiction to our assumption.
2. The set $M_{n_{\mathrm{i}}}$ contains a rule for deleting a letter of the input alphabet. First, we discuss the case that $M_{n_{\mathrm{i}}}$ contains only one such rule $a_k \to \lambda$. The words a_j with $1 \leq j \leq i$ and $j \neq k$ cannot be changed in the node $N_{n_{\mathrm{i}}}$. Each such word must leave this node such that it can be accepted by yielding a word z_j which arrives in the output node. Thus, all these words a_j belong to $O_{n_{\mathrm{i}}}$. But then a mixed input word $a_k a_j$ derives a_j which leaves the node $N_{n_{\mathrm{i}}}$ and also yields the word z_j which arrives in the output node. Thus, the word $a_k a_j \notin L_i$ is accepted in contrast to our assumption.

   If there are two rules $a_k \to \lambda$ and $a_\ell \to \lambda$ in $M_{n_{\mathrm{i}}}$ with $1 \leq k < \ell \leq i$, then we derive in the first step in the node $N_{n_{\mathrm{i}}}$ the word a_k both from the input word a_k^2 as well as from the input word $a_\ell a_k$. Since a_k^2 is accepted, also $a_\ell a_k$ is accepted, but $a_\ell a_k \notin L_i$. Thus, we have a contradiction, again.
3. The set $M_{n_{\mathrm{i}}}$ contains insertion rules. Let t be the length of the maximal number of letters of V_i in the words z occurring on the right-hand sides of rules of $G_{n_{\mathrm{i}}}$. Since $M_{n_{\mathrm{i}}}$ contains only insertion rules and a word obtained from a_1^2 has to leave the input node, $t > 0$ holds. We consider the word a_j^{3t} for an index j with $1 \leq j \leq i$. Since this word is accepted, there is a sequence of words $u_{j,1}, u_{j,2}, \ldots, u_{j,r_j}$ such that

$$a_j^{3t} \Longrightarrow u_{j,1} \Longrightarrow u_{j,2} \Longrightarrow \cdots \Longrightarrow u_{j,r_j-1} \Longrightarrow u_{j,r_j} \tag{3.1}$$

   is a derivation in the node $N_{n_{\mathrm{i}}}$ where $u_{j,p} \notin O_{n_{\mathrm{i}}}$ for $1 \leq p \leq r_j - 1$ and $u_{j,r_j} \in O_{n_{\mathrm{i}}}$ and u_{j,r_j} is transformed in the sequel into a word arriving in the output node.

If all letters inserted into the word a_j^{3t} in the derivation (3.1) to get u_{j,r_j} are not in $V_i \setminus \{a_j\}$, we set $z_j = u_{j,r_j}$. Obviously, z_j contains a_j^q for some natural number $q \geq 3t$ as a scattered subword ($q > 3t$ occurs if a_j is inserted in some steps) and the other letters do not belong to V_i.

If a letter of $V_i \setminus \{a_j\}$ is inserted, then let $u_{j,s_j} \Longrightarrow u_{j,s_j+1}$ be the first step where such a letter is inserted and let a_k be this letter. Then the word $a_j^{q_1} a_k a_j^{q_2}$ for some numbers q_1 and q_2 with $q_1 + q_2 \geq 3t$ is a scattered subword of u_{j,s_j+1} (the case $q_1 + q_2 > 3t$ occurs if a_j is inserted in some earlier steps) and the other letters of u_{j,s_j+1} do not belong to V_i. We now consider an input word $u = a_j^{q_1'} a_k a_j^{q_2'}$ such that $q_1' \leq q_1$, $q_2' \leq q_2$, and $q_1' + q_2' = 3t$, and then we start to insert the letters inserted in the first s_j steps of the derivation (3.1) at the corresponding positions in u such that u_{j,s_j+1} is obtained if all insertions can be performed (*i.e.*, the intermediate words do not leave the input node). If all steps can be performed, we obtain the word u_{j,s_j+1} where we insert further letters as above to produce the word u_{j,r_j} in the input node. This word u_{j,r_j} leaves the node and is transformed into a word arriving in the output node. But then $u \notin L_i$ is accepted which gives a contradiction. Therefore, a word u' produced in less than s_j steps has to belong to O_{n_i} (the word u' leaves the node N_{n_i} and does not yield a word arriving in the output node after transformations). Then, we set $z_j = u'$. Note that z_j contains a scattered subword $a_j^{q_1''} a_k a_j^{q_2''}$ with $q_1'' + q_2'' \geq 3t$ such that all other letters of z_j do not belong to V_i.

By our construction, we have $\{z_1, z_2, \ldots, z_i\} \subseteq O_{n_i}$. We consider the homomorphism h which maps every letter of V_i onto itself and every letter not in V_i onto the empty word. Then $\{h(z_1), h(z_2), \ldots, h(z_i)\} \subseteq h(O_{n_i})$. Let H_{n_i} be a right-linear grammar constructed from the grammar G_{n_i} as in the proof of Lemma 3.4. Then $L(H_{n_i}) = h(O_{n_i})$ and the maximal length of the words z' occurring on the right-hand sides of rules of H_{n_i} is t. By Lemma 3.6, we obtain that $Prod(H_{n_i}) > i$. By the construction of H_{n_i}, we get $Prod(G_{n_i}) > i$, too. Since the above argumentation is valid for any right-linear grammar generating the output filter O_{n_i}, we obtain that also $Prod(O_{n_i}) > i$ holds. Thus, $O_{n_i} \notin RL_i^P$ which is a contradiction to our assumption.

4. The set M_{n_i} contains a rule for substituting a letter of the input alphabet.

For every natural number j with $1 \leq j \leq i$, we set Y_j as the set of all letters which can be obtained in the input node from the letter a_j by substitutions and only from this letter and which do not belong to the alphabet V_i:

$$Y_j = \{\, x \mid x \in U \setminus V_i \text{ and } a_j \Longrightarrow^* x \text{ and } \forall k \neq j : a_k \not\Longrightarrow^* x \,\}.$$

We further set Y as the set of all letters which can be obtained in the input node from two different letters of the input alphabet by substitutions and which do not belong to the alphabet V_i:

$$Y = \{\, x \mid x \in U \setminus V_i \text{ and } \exists j,k : 1 \leq j < k \leq i \text{ and } a_j \Longrightarrow^* x \text{ and } a_k \Longrightarrow^* x \,\}.$$

Let t be the maximal length of the words z occurring on the right-hand side of rules of G_{n_i}. Since the length is not changed by the application of substitution rules and an infinite set is accepted, it is clear that $t > 0$. We consider the word a_j^{3t} for an index j with $1 \leq j \leq i$. Since this word is accepted, there is a sequence of words $u_{j,1}, u_{j,2}, \ldots, u_{j,r_j}$ such that

$$a_j^{3t} \Longrightarrow u_{j,1} \Longrightarrow u_{j,2} \Longrightarrow \cdots \Longrightarrow u_{j,r_j-1} \Longrightarrow u_{j,r_j} \tag{3.2}$$

is a derivation in the node N_{n_i} where $u_{j,p} \notin O_{n_i}$ for $1 \leq p \leq r_j - 1$ and $u_{j,r_j} \in O_{n_i}$ and u_{j,r_j} is transformed in the sequel into a word arriving in the output node.

If the word u_{j,r_j} consists of letters from the set $\{a_j\} \cup Y_j$ only (the letters different from a_j do not belong to the set V_i and cannot be obtained from another letter a_k), we set $z_j = u_{j,r_j}$.

If a letter of the set $(V_i \cup Y) \setminus \{a_j\}$ appears after a substitution, then let $u_{j,s_j} \Longrightarrow u_{j,s_j+1}$ be the first step where such a letter appears and let x be this letter. Then we have

$$u_{j,s_j+1} \in (\{a_j\} \cup Y_j)^{q_1}\{x\}(\{a_j\} \cup Y_j)^{q_2}$$

for some numbers q_1 and q_2 with $q_1 + q_2 = 3t - 1$. If $x \in V_i$, then $x = a_k$ for some index k with $1 \leq k \leq i$ and $k \neq j$. If $x \in Y$, then there is a letter a_k with $1 \leq k \leq i$ and $k \neq j$ and $a_k \Longrightarrow^* x$.

We now consider the input word $u = a_j^{q_1} a_k a_j^{q_2}$. We first substitute the letters as in the first s_j steps of the derivation (3.2) at the corresponding positions in u ignoring those derivation steps which substitute the letter at position $q_1 + 1$ and then we execute the substitutions at position $q_1 + 1$ according to the derivation $a_k \Longrightarrow^* x$ such that u_{j,s_j+1} is obtained if all substitutions can be performed (*i.e.*, the intermediate words do not leave the input node). If all steps can be performed, we obtain the word u_{j,s_j+1} where we substitute further letters as above in the derivation (3.2) to produce the word u_{j,r_j} in the input node. This word u_{j,r_j} leaves the node and is transformed into a word arriving in the output node. But then $u \notin L_i$ is accepted which gives a contradiction. Therefore, a word u' produced earlier than u_{j,s_j+1} belongs to O_{n_i} (such that the word u' leaves the node N_{n_i} but does not yield a word arriving in the output node after transformations). Then, we set $z_j = u'$. Note that z_j belongs to the set $(\{a_j\} \cup Y_j)^{q_1} U (\{a_j\} \cup Y_j)^{q_2}$.

By our construction, we have $\{z_1, z_2, \ldots, z_i\} \subseteq O_{n_i}$. We consider the homomorphism h which maps every letter of V_i onto itself, every letter of a set Y_j with $1 \leq j \leq i$ onto a_j, and all letters from Y onto a new letter b. Then $\{h(z_1), h(z_2), \ldots, h(z_i)\} \subseteq h(O_{n_i})$. Let H_{n_i} be a right-linear grammar constructed from the grammar G_{n_i} as in the proof of Lemma 3.4. Then $L(H_{n_i}) = h(O_{n_i})$ and the maximal length of words z' occurring on the right-hand sides of rules of G_{n_i} is t. By Lemma 3.6, we obtain that $Prod(H_{n_i}) > i$. By the construction of H_{n_i}, we get $Prod(G_{n_i}) > i$, too. Since the above argumentation is valid for any right-linear grammar generating the output filter O_{n_i}, we obtain that also $Prod(O_{n_i}) > i$ holds. Thus, $O_{n_i} \notin RL_i^P$ which is a contradiction to our assumption.

Our assumption is false in any case and, thus, $L_i \notin \mathcal{A}(RL_i^P)$ holds. □

The previous lemma is now used to prove an infinite hierarchy.

Theorem 3.10. *For all natural numbers $i \geq 1$, the proper inclusion*

$$\mathcal{A}(RL_i^P) \subset \mathcal{A}(RL_{i+1}^P)$$

holds.

Proof. The proper inlusion $\mathcal{A}(RL_1^P) \subset \mathcal{A}(RL_2^P)$ was already proved in Theorem 3.8.

Let $i \geq 2$ be a natural number. The inclusion $\mathcal{A}(RL_i^P) \subseteq \mathcal{A}(RL_{i+1}^P)$ follows from Lemma 2.2 and Lemma 2.5. A witness language for the properness is the language L_i as was shown in Lemma 3.9. □

The inclusion results obtained here and in [12] are illustrated in Figure 3. Whenever there is no directed path from an entry X to an entry Y, the families X and Y are incomparable. Before we prove the incomparabilities, we give some helpful lemmas.

Lemma 3.11. *Let $\mathcal{L}$ be the family of the regular commutative languages or the family of the regular circular languages, then $\mathcal{A}(\mathcal{L}) = \mathcal{L}$ holds.*

Proof. Let $\mathcal{L} \in \{COMM, CIRC\}$ and

$$f_{\mathcal{L}} = \begin{cases} Comm, & \text{if } \mathcal{L} = COMM, \\ Circ, & \text{otherwise} \end{cases}$$

be the function which maps a language to its commutative or circular closure. By Lemma 2.6, we have the inclusion $\mathcal{L} \subseteq \mathcal{A}(\mathcal{L})$. Hence, it suffices to show that also the inclusion $\mathcal{A}(\mathcal{L}) \subseteq \mathcal{L}$ holds. Let $\mathcal{N}$ be a network where all filters belong to the class $\mathcal{L}$ and let $L = L(\mathcal{N})$ be the accepted language. If the input and output node of the network $\mathcal{N}$ coincide, then $L = V^*$ where V is the input alphabet and $L \in \mathcal{L}$. We now consider the case, that the input and output node are different. Assume that $L \notin \mathcal{L}$. Then there are a word $u \in L$ and a word $v \in f_{\mathcal{L}}(\{u\}) \setminus L$. Every operation on u in an evolutionary step leading to a word u' can be executed also on v at an appropriate position such that the resulting word v' belongs to the set $f_{\mathcal{L}}(u')$. Since all the filters belong to the family $\mathcal{L}$, both words u' and v' leave the node and enter the same nodes or both remain in the node. This argumentation can be repeated for all further steps in a computation which leads to the acceptance of the input word u. This implies that there is also a computation which leads to the acceptance of the input word v but this is a contradiction to $v \notin L$. Thus, $f_{\mathcal{L}}(L) = L(\mathcal{N})$ and, therefore, $L \in \mathcal{L}$. □

Lemma 3.12. *Let $L = \{ab\}$. Then $L \in \mathcal{A}(RL_1^P) \setminus \mathcal{A}(CIRC)$ holds.*

Proof. Since $L \in RL_1^P$, it follows $L \in \mathcal{A}(RL_1^P)$ by Lemma 2.6. According to Lemma 3.11, it holds $\mathcal{A}(CIRC) = CIRC$. The language L is not circular and, therefore, $L \notin \mathcal{A}(CIRC)$. □

Lemma 3.13. *Let $L = \{a\}^*\{b\}$. Then $L \in \mathcal{A}(RL_2^P) \setminus (\mathcal{A}(NIL) \cup \mathcal{A}(CIRC))$ holds.*

Proof. The language L can be generated by a regular grammar with two rules (for instance, $S \to aS$ and $S \to b$). By Lemma 2.6, it follows $L \in \mathcal{A}(RL_2^P)$. By Lemma 3.11, it holds $\mathcal{A}(CIRC) = CIRC$. The language L is not circular and, therefore, $L \notin \mathcal{A}(CIRC)$. We now prove that $L \notin \mathcal{A}(NIL)$.

Assume that $L \in \mathcal{A}(NIL)$. Then there is a network

$$\mathcal{N} = (\{a, b\}, U, N_1, N_2, \ldots, N_n, n_{\mathrm{i}}, n_{\mathrm{o}})$$

with $L(\mathcal{N}) = L$ where each filter is finite or its complement is finite. Let m_1 be the length of a longest word occurring in a finite filter of the network $\mathcal{N}$ and let m_2 be the length of a longest word occurring in the complement of an infinite filter of the network $\mathcal{N}$. Further, let $m = \max\{m_1, m_2\}$ be the maximum of these two lengths.

Any word of length at least $m+1$ can be communicated to another node only by passing infinite filters. For any such word, also every permutation goes the same way (it is too long for passing the finite filters and too long for not passing the infinite filters). A substitution or insertion of a letter in a permutation yields the same words as first substituting or inserting a letter and then permuting the words. Thus, as long as the network $\mathcal{N}$ does not apply deleting rules, it works on a word with at least $m+1$ letters in the same way as on every permutation of it. Hence, especially the word $a^m b$ is accepted if and only if also the word $a^{m-1}ba$ is accepted. Since this contradicts $L(\mathcal{N}) = L$, at some moment during the computation, a deleting rule is applied which reduces the original length. Hence, this deletion or a deletion in an earlier step effects a letter of the original input word – possibly after some substitutions. Let us consider the two words $w_1 = a^m b$ and $w_2 = a^{m-1}ba$ as input words. If that letter will be deleted which was originally the b and all other operations regarding the letters a before are applied in both words w_1 and w_2 at the same positions, then the computations on the words w_1 and w_2 lead to the same word. If, in continuation, the word w_1 is accepted then also w_2 is accepted which is a contradiction. If the considered deletion effects a letter which was originally an a, then there is a derivation of the word $a^m b$ in the network before the mentioned deletion of the i-th a for some i with $1 \leq i \leq m$ such that these letters are only substituted or not changed ($x_i \in U$ for $1 \leq m$, $y \in U$) and letters are inserted, which are later maybe also substituted and deleted (leading to words $w_i \in U^*$ with $1 \leq i \leq m+2$):

$$\begin{array}{ccccccccccccccccccccc} & a & & a & & \ldots & & a & & a & & a & & \ldots & & a & & a & & b & \\ & | & & | & & & & | & & | & & | & & & & | & & | & & | & \\ w_1 & x_1 & w_2 & x_2 & w_3 & \ldots & w_{i-1} & x_{i-1} & w_i & x_i & w_{i+1} & x_{i+1} & w_{i+2} & \ldots & w_{m-1} & x_{m-1} & w_m & x_m & w_{m+1} & y & w_{m+2} \end{array}$$

All intermediate words in this derivation have a length of at least $m+1$. Thus, the same evolutionary and communication steps are also possible, if we start with the word $a^{m-1}ba$ and operate on the last a as on the i-th a of the word $a^m b$:

$$\begin{array}{ccccccccccccccccccc} & a & & a & & \dots & & a & & a & & \dots & & a & & a & & b & & a & \\ & | & & | & & & & | & & | & & & & | & & | & & | & & | & \\ w_1 & x_1 & w_2 & x_2 & w_3 & \dots & w_{i-1} & x_{i-1} & w_i & w_{i+1} & x_{i+1} & w_{i+2} & \dots & w_{m-1} & x_{m-1} & w_m & x_m & w_{m+1} & y & w_{m+2} & x_i \end{array}$$

Then we delete the letter x_i and obtain in both cases the same word. Hence, also in the case that a letter is deleted which was originally an a, the computations on the words w_1 and w_2 lead to the same word. If, in continuation, the word w_1 is accepted then also w_2 is accepted which is a contradiction as before.

By these contradictions, we obtain that a network with nilpotent filters cannot accept the language L. Thus, we also have $L \notin \mathcal{A}(NIL)$. □

Lemma 3.14. *For any natural number $i \geq 1$, let $V_i = \{a_1, a_2, \dots, a_i\}$ be an alphabet with i letters. Further, let F_{i+1} be the following language over the alphabet V_{i+1}:*

$$F_{i+1} = \bigcup_{k=1}^{i+1} \{a_k^3\}.$$

Then $F_{i+1} \in (\mathcal{A}(RL_{i+1}^P) \cup \mathcal{A}(FIN)) \setminus \mathcal{A}(RL_i^P)$.

Proof. Let $i \geq 1$ be a natural number. Since the language F_{i+1} is finite and can be generated by a right-linear grammar with $i+1$ rules, we know from Lemma 2.6 that

$$F_{i+1} \in \mathcal{A}(RL_{i+1}^P) \cup \mathcal{A}(FIN).$$

Assume that $F_{i+1} \in \mathcal{A}(RL_i^P)$ holds. Then there is an accepting network

$$\mathcal{N} = (V_{i+1}, U, N_1, \dots, N_n, E, n_{\mathrm{i}}, n_{\mathrm{o}})$$

with $L(\mathcal{N}) = F_{i+1}$ and where every filter is generated by a right-linear grammar with at most i rules. If the input and output node coincide ($n_{\mathrm{i}} = n_{\mathrm{o}}$), then the language V_{i+1}^* is accepted which contradicts the assumption $L(\mathcal{N}) = F_{i+1}$. Hence, the input and output node are different.

We consider the input node $N_{n_{\mathrm{i}}} = (M_{n_{\mathrm{i}}}, I_{n_{\mathrm{i}}}, O_{n_{\mathrm{i}}})$ and prove that already the output filter $O_{n_{\mathrm{i}}}$ of this node cannot be generated by a grammar with at most i production rules. We distinguish some cases with respect to the rule set $M_{n_{\mathrm{i}}}$:

1. No rule $x \to \lambda$ with $x \in V_{i+1}$, no rule $x \to y$ with $x \in V_{i+1}$ and $y \in U$, and no rule $\lambda \to y$ with $y \in U$ is contained in $M_{n_{\mathrm{i}}}$. Then input words cannot be changed in the node $N_{n_{\mathrm{i}}}$, but have to leave it in order to yield a word arriving in the output node after some steps. Thus, especially all words a_j^3 with $1 \leq j \leq i+1$ are contained in $O_{n_{\mathrm{i}}}$. Since the generation of these words by a right-linear grammar requires $i+1$ rules (one for each first letter), we obtain a contradiction to our assumption.
2. The set $M_{n_{\mathrm{i}}}$ contains a rule for deleting a letter of the input alphabet. First, we discuss the case that $M_{n_{\mathrm{i}}}$ contains only one such rule $a_k \to \lambda$. The words a_j^3 with $1 \leq j \leq i+1$ and $j \neq k$ cannot be changed in the node $N_{n_{\mathrm{i}}}$. Each such word must leave this node such that it can be accepted by yielding a word z_j which arrives in the output node. Thus, all these words a_j^3 belong to $O_{n_{\mathrm{i}}}$. But then a mixed input word $a_k a_j^3$ derives a_j^3 which leaves the node $N_{n_{\mathrm{i}}}$ and also yields the word z_j which arrives in the output node. Thus, the word $a_k a_j^3 \notin F_{i+1}$ is accepted in contrast to our assumption.

   If there are two rules $a_k \to \lambda$ and $a_\ell \to \lambda$ in $M_{n_{\mathrm{i}}}$ with $1 \leq k < \ell \leq i+1$, then we derive in the first step in the node $N_{n_{\mathrm{i}}}$ the word a_k^2 both from the input word a_k^3 as well as from the input word $a_\ell a_k^2$. Since a_k^3 is accepted, also $a_\ell a_k^2$ is accepted, but $a_\ell a_k^2 \notin F_{i+1}$. Thus, we have a contradiction, again.

3. The set M_{n_i} contains insertion rules. We consider the word a_j^3 for an index j with $1 \leq j \leq i+1$. Since this word is accepted, there is a sequence of words $u_{j,1}, u_{j,2}, \ldots, u_{j,r_j}$ such that

$$a_j^3 \Longrightarrow u_{j,1} \Longrightarrow u_{j,2} \Longrightarrow \cdots \Longrightarrow u_{j,r_j-1} \Longrightarrow u_{j,r_j} \tag{3.3}$$

is a derivation in the node N_{n_i} where $u_{j,p} \notin O_{n_i}$ for $1 \leq p \leq r_j - 1$ and $u_{j,r_j} \in O_{n_i}$ and u_{j,r_j} is transformed in the sequel into a word arriving in the output node.

If all letters inserted into the word a_j^3 in the derivation (3.3) to get u_{j,r_j} are not in $V_{i+1} \setminus \{a_j\}$, we set $z_j = u_{j,r_j}$. Obviously, z_j contains a_j^q for some natural number $q \geq 3$ as a scattered subword ($q > 3$ occurs if a_j is inserted in some steps) and the other letters do not belong to V_{i+1}. If a letter of the set $V_{i+1} \setminus \{a_j\}$ is inserted, then let $u_{j,s_j} \Longrightarrow u_{j,s_j+1}$ be the first step where such a letter is inserted and let a_k be this letter. Then the word $a_j^{q_1} a_k a_j^{q_2}$ for some numbers q_1 and q_2 with $q_1 + q_2 \geq 3$ is a scattered subword of u_{j,s_j+1} (the case $q_1 + q_2 > 3$ occurs if a_j is inserted in some earlier steps) and the other letters of u_{j,s_j+1} do not belong to V_{i+1}.

We now consider an input word $u = a_j^{q_1'} a_k a_j^{q_2'}$ such that $q_1' \leq q_1$, $q_2' \leq q_2$, and $q_1' + q_2' = 3$, and then we start to insert the letters inserted in the first s_j steps of the derivation (3.3) at the corresponding positions in u such that u_{j,s_j+1} is obtained if all insertions can be performed (*i.e.*, the intermediate words do not leave the input node). If all steps can be performed, we obtain the word u_{j,s_j+1} where we insert further letters as above to produce the word u_{j,r_j} in the input node. This word u_{j,r_j} leaves the node and is transformed into a word arriving in the output node. But then the word $u \notin F_{i+1}$ is accepted which gives a contradiction. Therefore, a word u' produced in less than s_j steps has to belong to O_{n_i} (the word u' leaves the node N_{n_i} and does not yield a word arriving in the output node after transformations). Then, we set $z_j = u'$. Note that z_j contains a scattered subword $a_j^{q_1''} a_k a_j^{q_2''}$ with $q_1' \leq q_1'' \leq q_1$ and $q_2' \leq q_2'' \leq q_2$ such that all other letters of z_j do not belong to V_{i+1}.

By our construction, we have $\{z_1, z_2, \ldots, z_{i+1}\} \subseteq O_{n_i}$. We consider the homomorphism h which maps every letter of V_{i+1} onto itself and every letter not in V_{i+1} onto the empty word. Then

$$\{h(z_1), h(z_2), \ldots, h(z_{i+1})\} \subseteq h(O_{n_i}).$$

By Lemma 3.5, we obtain that $Prod(h(O_{n_i})) \geq i+1$ holds. From Lemma 3.4, we also know that $Prod(O_{n_i}) \geq Prod(h(O_{n_i}))$ holds. Thus, we obtain also in this case $O_{n_i} \notin RL_i^P$ which is a contradiction to our assumption.

4. The set M_{n_i} contains a rule for substituting a letter of the input alphabet.

For every natural number j with $1 \leq j \leq i+1$, we set Y_j as the set of all letters which can be obtained in the input node from the letter a_j by substitutions and only from this letter and which do not belong to the alphabet V_{i+1}:

$$Y_j = \{\, x \mid x \in U \setminus V_{i+1} \text{ and } a_j \Longrightarrow^* x \text{ and } \forall k \neq j : a_k \not\Longrightarrow^* x \,\}.$$

We further set Y as the set of all letters which can be obtained in the input node from two different letters of the input alphabet by substitutions and which do not belong to the alphabet V_{i+1}:

$$Y = \{\, x \mid x \in U \setminus V_{i+1} \text{ and } \exists j,k : 1 \leq j < k \leq i+1 \text{ and } a_j \Longrightarrow^* x \text{ and } a_k \Longrightarrow^* x \,\}.$$

We consider the word a_j^3 for an index j with $1 \leq j \leq i+1$. Since this word is accepted, there is a sequence of words $u_{j,1}, u_{j,2}, \ldots, u_{j,r_j}$ such that

$$a_j^3 \Longrightarrow u_{j,1} \Longrightarrow u_{j,2} \Longrightarrow \cdots \Longrightarrow u_{j,r_j-1} \Longrightarrow u_{j,r_j} \tag{3.4}$$

is a derivation in the node N_{n_i} where $u_{j,p} \notin O_{n_i}$ for $1 \leq p \leq r_j - 1$ and $u_{j,r_j} \in O_{n_i}$ and u_{j,r_j} is transformed in the sequel into a word arriving in the output node.

If the word u_{j,r_j} consists of letters from the set $\{a_j\} \cup Y_j$ only (the letters different from a_j do not belong to the set V_{i+1} and cannot be obtained from another letter a_k), we set $z_j = u_{j,r_j}$.

If a letter of the set $(V_{i+1} \cup Y) \setminus \{a_j\}$ appears after a substitution, then let $u_{j,s_j} \Longrightarrow u_{j,s_j+1}$ be the first step where such a letter appears and let x be this letter. Then we have

$$u_{j,s_j+1} \in (\{a_j\} \cup Y_j)^{q_1}\{x\}(\{a_j\} \cup Y_j)^{q_2}$$

for some numbers q_1 and q_2 with $q_1 + q_2 = 2$. If $x \in V_{i+1}$, then $x = a_k$ for some index k with $1 \leq k \leq i+1$ and $k \neq j$. If $x \in Y$, then there is a letter a_k with $1 \leq k \leq i+1$ and $k \neq j$ and $a_k \Longrightarrow^* x$.

We now consider the input word $u = a_j^{q_1} a_k a_j^{q_2}$. We first substitute the letters as in the first s_j steps of the derivation (3.4) at the corresponding positions in u ignoring those derivation steps which substitute the letter at position $q_1 + 1$ and then we execute the substitutions at position $q_1 + 1$ according to the derivation $a_k \Longrightarrow^* x$ such that u_{j,s_j+1} is obtained if all substitutions can be performed (*i.e.*, the intermediate words do not leave the input node). If all steps can be performed, we obtain the word u_{j,s_j+1} where we substitute further letters as above in the derivation (3.4) to produce the word u_{j,r_j} in the input node. This word u_{j,r_j} leaves the node and is transformed into a word arriving in the output node. But then the word $u \notin F_{i+1}$ is accepted which gives a contradiction. Therefore, a word u' produced earlier than u_{j,s_j+1} belongs to O_{n_i} (such that the word u' leaves the node N_{n_i} but does not yield a word arriving in the output node after transformations). Then, we set $z_j = u'$. Note that z_j belongs to the set $(\{a_j\} \cup Y_j)^{q_1} U (\{a_j\} \cup Y_j)^{q_2}$.

By our construction, we have $\{z_1, z_2, \ldots, z_{i+1}\} \subseteq O_{n_i}$. We consider the homomorphism h which maps every letter of V_{i+1} onto itself, every letter of a set Y_j with $1 \leq j \leq i+1$ onto a_j, and all letters from Y onto a new letter b. Then $\{h(z_1), h(z_2), \ldots, h(z_{i+1})\} \subseteq h(O_{n_i})$. By Lemma 3.5, we obtain that $Prod(h(O_{n_i})) \geq i+1$ holds. From Lemma 3.4, we also know that $Prod(O_{n_i}) \geq Prod(h(O_{n_i}))$ holds. Thus, $O_{n_i} \notin RL_i^P$ which is a contradiction to our assumption.

Our assumption is false in any case and, thus, $F_{i+1} \notin \mathcal{A}(RL_i^P)$ holds. □

We now prove the incomparability results.

Theorem 3.15. *Every family $\mathcal{A}(MON)$, $\mathcal{A}(COMM)$, $\mathcal{A}(REG_1^Z)$, and $\mathcal{A}(CIRC)$ is incomparable to $\mathcal{A}(RL_1^P)$*

Proof. Due to the relations

$$\mathcal{A}(MON) = \mathcal{A}(COMM) = \mathcal{A}(REG_1^Z) \quad \text{and} \quad \mathcal{A}(COMM) \subset \mathcal{A}(CIRC),$$

it suffices to show that there is a language in the set $\mathcal{A}(MON) \setminus \mathcal{A}(RL_1^P)$ and that there is a language which belongs to the set $\mathcal{A}(RL_1^P) \setminus \mathcal{A}(CIRC)$.

From Lemma 3.9 with $L_2 = \{a_1\}^* \cup \{a_2\}^*$, we know that $L_2 \in \mathcal{A}(MON)$ but $L_2 \notin \mathcal{A}(RL_2^P)$ and, by Theorem 3.10, we also have $L_2 \notin \mathcal{A}(RL_1^P)$.

For the language $L = \{ab\}$, the relation $L \in \mathcal{A}(RL_1^P) \setminus \mathcal{A}(CIRC)$ was proved in Lemma 3.12. Thus, this is a witness language for the other case. □

Theorem 3.16. *Each of the families $\mathcal{A}(FIN)$, $\mathcal{A}(MON)$, $\mathcal{A}(COMM)$, $\mathcal{A}(REG_1^Z)$, $\mathcal{A}(NIL)$, and $\mathcal{A}(CIRC)$ is incomparable to every family $\mathcal{A}(RL_i^P)$ for $i \geq 2$.*

Proof. Due to the equality $\mathcal{A}(MON) = \mathcal{A}(COMM) = \mathcal{A}(REG_1^Z)$ and the inclusion relations $\mathcal{A}(FIN) \subset \mathcal{A}(NIL)$, $\mathcal{A}(COMM) \subset \mathcal{A}(CIRC)$, and $\mathcal{A}(RL_i^P) \subset \mathcal{A}(RL_{i+1}^P)$ for all $i \geq 1$, it suffices to show that the following sets are not empty:

$$\mathcal{A}(MON) \setminus \mathcal{A}(RL_i^P) \quad \text{and} \quad \mathcal{A}(FIN) \setminus \mathcal{A}(RL_i^P) \quad \text{for} \quad i \geq 2$$

as well as

$$\mathcal{A}(RL_2^P) \setminus \mathcal{A}(NIL) \quad \text{and} \quad \mathcal{A}(RL_2^P) \setminus \mathcal{A}(CIRC).$$

According to Lemma 3.9, we have $L_i \in \mathcal{A}(MON) \setminus \mathcal{A}(RL_i^P)$ for all $i \geq 2$ which proves the first case. From Lemma 3.14 follows that $F_i \in \mathcal{A}(FIN) \setminus \mathcal{A}(RL_i^P)$ for all $i \geq 2$ which proves the second case. The last two cases follow from Lemma 3.13 where $L \in \mathcal{A}(RL_2^P) \setminus (\mathcal{A}(NIL) \cup \mathcal{A}(CIRC))$ was shown for the witness language $L = \{a\}^*\{b\}$. □

3.2. Ideals and codes as filters

We start with accepting networks of evolutionary processors where the filters are either all left ideals or all right ideals and prove that this restriction does not decrease the computational power compared to the use of arbitrary regular languages as filters.

Theorem 3.17. *We have* $\mathcal{A}(lId) = RE$.

Proof. In [12], it was shown that, for every recursively enumerable language Z, there is an ANEP with definite filters only which accepts the language Z. Each filter in the constructed network has the form V^*L for some alphabet V and a finite language $L \subseteq V^*$. Hence, all these filters are also left ideals which proves that also the equivalence $\mathcal{A}(lId) = RE$ holds. □

The symmetry between left and right ideals can be used for showing that also right ideals as filters do not restrict the computational power.

Theorem 3.18. *We have* $\mathcal{A}(rId) = RE$.

Proof. Let $L \in RE$ be a recursively enumerable language. Then also the language L^R of all reversed words,

$$L^R = \{\, w^R \mid w \in L \,\},$$

is recursively enumerable. According to Theorem 3.17, this language is accepted by a network with left ideals as filters. If we change, in an accepting network for L^R, every occurring filter F to F^R, this new network accepts the language $(L^R)^R$ which is L and has right ideals as filters. Hence, $\mathcal{A}(rId) = RE$ holds. □

We now consider codes as filters. In order to have a better readability, we recall that by C, SC, PfC, SfC, BfC, IfC, OfC, RC, and UC, we denote the families of regular codes, regular solid codes, regular prefix codes, regular suffix codes, regular bifix codes, regular infix codes, regular outfix codes, regular reflective codes and uniform codes, respectively.

Theorem 3.19. *For any* $\mathcal{L} \in \{C, PfC, SfC, BfC, IfC, SC\}$, *we have* $\mathcal{A}(\mathcal{L}) = RE$.

Proof. Let $\mathcal{L} \in \{C, PfC, SfC, BfC, IfC, SC\}$. We mention that, for any regular language R and two additional letters A and B not occurring in the alphabet of R, the language $\{A\}R\{B\}$ belongs to the class $\mathcal{L}$.

Let L be a recursively enumerable language. By [12], Theorem 3.1, there is an accepting NEP

$$\mathcal{N} = (V, V', N_1, N_2, \ldots, N_n, E, n_\mathrm{i}, n_\mathrm{o})$$

with regular filters and $N_{n_\mathrm{i}} = (\emptyset, V^*, V^*)$ such that $L(\mathcal{N}) = L$. Let A and B be two new symbols not occurring in V'. We modify the net to

$$\mathcal{N}' = (V, V' \cup \{A, B\}, N_0', N_1', N_2', \ldots, N_n', N_{n+1}', E', 0, n_\mathrm{o})$$

where

$$\begin{aligned} N'_0 &= (\{\lambda \to A, \lambda \to B\}, \{B\}V^*\{A\}, \{A\}V^*\{B\}), \\ N'_{n+1} &= (M_{n_i}, \{A\}V^*\{B\}, \{A\}O_{n_i}\{B\}), \\ N'_k &= (M_k, \{A\}I_k\{B\}, \{A\}O_k\{B\}) \quad \text{for } N_k = (M_k, I_k, O_k),\ 1 \le k \le n, \\ E' &= \{(N'_0, N'_{n+1})\} \cup \{\, (N'_{n+1}, N'_k) \mid (N_{n_i}, N_k) \in E \,\} \\ &\qquad \cup \{\, (N'_k, N'_l) \mid (N_k, N_l \in E \,\}. \end{aligned}$$

Let $w \in L$. By definition, using the insertion rules $\lambda \to A$ and $\lambda \to B$ of the initial node, we produce words which cannot leave the node or AwB which is send to the node N'_{n+1}. Note that this node N'_{n+1} behaves on a word AxB as N_i on the word x. Now let AxB with $x \in (V')^*$ be a word in some node N'_k. Then the word can be changed to $aAxB$ or $AxBa$ for some letter $a \in V'$ if M_k contains the insertion rule $\lambda \to a$ or to $Ax'B$. In the former two cases, the words and all words which can be obtained by further applications of M_k cannot leave the node. In the latter case, the word $Ax'B$ remains in the node N'_k if and only if x' remains in N_k or the word $Ax'B$ arrives in all nodes N'_l if x' moves from N_k to N_l. Essentially, this situation also holds for N'_{n+1} which behaves as N_{n_i}. The difference is that in the sequel no word can be send to N'_{n+1} by the structure of E'. Thus, it is easy to see that a word AyB arrives in the new output node N'_{n_o} if and only if y arrives in node N_{n_o} (starting with w). This proves $L(\mathcal{N}') = L$. Hence, $RE \subseteq \mathcal{A}(\mathcal{L})$ and, since $\mathcal{L} \subseteq REG$ and $\mathcal{A}(REG) = RE$ (Thm. 2.4), we also have $\mathcal{A}(\mathcal{L}) = RE$. □

The next lemmas present witness languages for proving incomparabilities of language families or the properness of inclusion relations.

Lemma 3.20. *Let $x \in V^*$ be a word over some alphabet V such that there is a circular shift of x which is different from x. Then $\{x\} \in (\mathcal{A}(UC) \cup \mathcal{A}(RL_1^P)) \setminus \mathcal{A}(CIRC)$.*

Proof. We consider the network

$$(V, V, (\emptyset, \{x\}, \{x\}), (\emptyset, \{x\}, \{x\}), (1, 2), 1, 2),$$

which accepts $\{x\}$. Since $\{x\}$ is a uniform code and can be generated by one rule, we get $\{x\} \in \mathcal{A}(UC) \cup \mathcal{A}(RL_1^P)$.

By Lemma 3.11, we know that $\mathcal{A}(CIRC) = CIRC$. By our assumption, we have $\{x\} \notin CIRC$, and thus, we obtain $\{x\} \notin \mathcal{A}(CIRC)$. □

Lemma 3.21. *Let $L = Circ(\{abc\})$. Then $L \in (\mathcal{A}(UC) \cup \mathcal{A}(RC)) \setminus \mathcal{A}(COMM)$.*

Proof. The language L is uniform and circular. Therefore, it is also a reflective code. By Lemma 2.6, it follows $L \in \mathcal{A}(UC) \cup \mathcal{A}(RC)$. According to Lemma 3.11, it holds $\mathcal{A}(COMM) = COMM$. The language L is not commutative and, therefore, it holds $L \notin \mathcal{A}(COMM)$. □

Lemma 3.22. *For any natural number $i \ge 1$, let $V_i = \{a_1, a_2, \ldots, a_i\}$ be an alphabet with i letters. Further, let F_{i+1} be the following language over the alphabet V_{i+1}:*

$$F_{i+1} = \bigcup_{k=1}^{i+1} \{a_k^3\}.$$

Then $F_{i+1} \in (\mathcal{A}(UC) \cup \mathcal{A}(RC)) \setminus \mathcal{A}(RL_i^P)$.

Proof. Let $i \ge 1$ be a natural number. The language F_i is uniform and circular. Therefore, it is a uniform and reflective code. Hence, $F_i \in \mathcal{A}(UC) \cup \mathcal{A}(RC)$ according to Lemma 2.6. As proved in Lemma 3.14, we also know that $F_{i+1} \notin \mathcal{A}(RL_i^P)$. □

Lemma 3.23. *Let i be an integer with $i \geq 5$ and*

$$C_i = \{\, a^{n_1}ba^{n_2} \mid n_1 \geq 0, n_2 \geq 0, n_1 + n_2 = i \,\} \cup \{\, b^{m_1}ab^{m_2} \mid m_1 \geq 0, m_2 \geq 0, m_1 + m_2 = i+1 \,\}$$

and

$$L_i = \{\, x_0y_1x_1y_2 \dots y_rx_r \mid x_s \in \{c\}^* \text{ for } 0 \leq s \leq r,\ y_1y_2 \dots y_r \in C_i \,\}.$$

Then $L_i \in \mathcal{A}(RC)$ and $L_i \notin \mathcal{A}(UC)$.

Proof. Let $i \geq 5$. The network

$$(\{a,b,c\}, \{a,b,c\}, (\{c \to \lambda\}, C_i, C_i), (\emptyset, C_i, C_i), (1,2), 1, 2)$$

accepts the language L_i. Since C_i is a reflective code, we get $L_i \in \mathcal{A}(RC)$.

Assume that $L_i \in \mathcal{A}(UC)$. Then there is an accepting network

$$\mathcal{N} = (\{a,b,c\}, U, N_1, \dots, N_n, E, n_\mathrm{i}, n_\mathrm{o})$$

with filters in UC such that $L(\mathcal{N}) = L_i$. We get $n_\mathrm{i} \neq n_\mathrm{o}$ since otherwise all words over $\{a,b,c\}$ are accepted. Therefore, any word of the language L_i is possibly changed by some applications of rules of M_{n_i} and then it leaves the input node $N_{n_\mathrm{i}} = (M_{n_\mathrm{i}}, I_{n_\mathrm{i}}, O_{n_\mathrm{i}})$. Since especially the output filter O_{n_i} is a uniform code, we get $O_{n_\mathrm{i}} \subseteq U^k$ for some natural number k. We distinguish two cases:

1. Let $k \leq i+1$. Since $b^{m_1}ab^{m_2}$ of length $i+2$ is in L_n, it has to leave the input node after some transformations. Consequently, M_{n_i} contains deletion rules, *i.e.*, $a \to \lambda \in M_{n_\mathrm{i}}$ or $b \to \lambda \in M_{n_\mathrm{i}}$ or both these rules are in the set M_{n_i}. We discuss the situation where $a \to \lambda \in M_{n_\mathrm{i}}$ (the other cases can be handled analogously). Then we consider the input word $w = ab^{i+1}a$. By deleting the first a, we obtain the word $w' = b^{i+1}a$. The word w' cannot leave the input node since its length is $i+2 > k$. Since w' as an input word is accepted (because $w' \in L_i$), we get that w is accepted, too, which contradicts $L(\mathcal{N}) = L_i$.
2. Let $k \geq i+2$. Since $b^{m_1}ab^{m_2}$ of length $i+1$ is in L_n, it has to leave the input node after some transformations. Thus, M_{n_i} contains insertion rules. However, since the word $c^{k+1}b^{i+1}a$ has to be transformed to a word of length k, too, the set M_{n_i} has to contain also deletion rules. This is a contradiction to the definition of a network.

Since there is a contradiction in any case, we have $L_i \notin \mathcal{A}(UC)$. □

Lemma 3.24. *Let $Z_1 = \{\, w \mid w \in \{a,b\}^*, |w|_a \in \{1,3\} \,\}$. Then*

$$Z_1 \in (\mathcal{A}(FIN) \cap \mathcal{A}(COMM)) \setminus \mathcal{A}(OfC).$$

Proof. The network

$$(\{a,b\}, \{a,b\}, (\{b \to \lambda\}, \{a, aaa\}, \{a, aaa\}), (\emptyset, \{a, aaa\}, \emptyset), \{(1,2)\}, 1, 2)$$

accepts the language Z_1 (by deletion of all the letters b in the input node, we get a or aaa and this word is sent to the output node). Since the filters are finite and commutative, we have $Z_1 \in \mathcal{A}(FIN) \cap \mathcal{A}(COMM)$.

Let us assume that there is an accepting network

$$\mathcal{N} = (\{a,b\}, U, N_1, \dots, N_n, E, n_\mathrm{i}, n_\mathrm{o})$$

with filters in OfC such that $L(\mathcal{N}) = Z_1$. Since Z_1 is a proper subset of $\{a,b\}^*$, the input node and output node are different. We distinguish the cases for the input node $N_{n_\mathrm{i}} = (M_{n_\mathrm{i}}, I_{n_\mathrm{i}}, O_{n_\mathrm{i}})$:

1. The set M_{n_i} is empty. Then a and aaa are in the output filter O_{n_i} since these words are accepted and thus they have to leave the input node. But this contradicts the code property of O_{n_i}.
2. The set M_{n_i} contains deletion rules. If $a \to \lambda$ is a rule of M_{n_i}, then we obtain $a \Longrightarrow \lambda$ in the input node N_{n_i}. Moreover, λ cannot be changed by applications of rules of M_{n_i}. Hence, $\lambda \in O_{n_\mathrm{i}}$ since a is accepted. But the empty word is contained in no code. Thus, $a \to \lambda$ is not in M_{n_i}. Therefore, the input words a and aaa cannot be changed by M_{n_i}. Since these words are accepted, they both belong to the output filter O_{n_i}. However, this is a contradiction to the code property of O_{n_i}, again.
3. The set M_{n_i} contains substitution or insertion rules. Since O_{n_i} is finite, only a finite set of words can leave the input node and can finally be accepted, because the application of the rules of M_{n_i} cannot decrease the length of the words. This is a contradiction because Z_1 is infinite.

Since every case yields a contradiction, the assumption is wrong and $Z_1 \notin \mathcal{A}(OfC)$. □

From the previous results, we obtain the following proper inclusions.

Theorem 3.25. *The proper inclusions*

1. $\mathcal{A}(UC) \subset \mathcal{A}(OfC) \subset \mathcal{A}(FIN)$,
2. $\mathcal{A}(RC) \subset \mathcal{A}(OfC)$, *and*
3. $\mathcal{A}(RC) \subset \mathcal{A}(CIRC)$

hold.

Proof. The inclusions follow from the inclusion relations $UC \subseteq OfC \subseteq FIN$, $RC \subseteq OfC$, and $RC \subseteq CIRC$ together with Lemma 2.6. Their properness follows from the existence of witness languages:

1. According to Lemma 3.24, we have $Z_1 \in \mathcal{A}(FIN) \setminus \mathcal{A}(OfC)$ for the language $Z_1 = \{\, w \mid w \in \{a,b\}^*, |w|_a \in \{1,3\} \,\}$. From Lemma 3.23, we have the relation $L_5 \in \mathcal{A}(RC) \setminus \mathcal{A}(UC)$. Since $\mathcal{A}(RC) \subseteq \mathcal{A}(OfC)$, the language L_5 is also a witness language for the properness of the inclusion $\mathcal{A}(UC) \subset \mathcal{A}(OfC)$.
2. From Lemma 3.20, we know $\{ab\} \in \mathcal{A}(UC) \setminus \mathcal{A}(CIRC)$. We also have the inclusions $\mathcal{A}(UC) \subset \mathcal{A}(OfC)$ and $\mathcal{A}(RC) \subseteq \mathcal{A}(CIRC)$ (since $RC \subseteq CIRC$); hence, the language $\{ab\}$ is also a witness language for the properness of the inclusion $\mathcal{A}(RC) \subset \mathcal{A}(OfC)$.
3. By Lemma 3.24, it is $Z_1 \in \mathcal{A}(COMM) \setminus \mathcal{A}(OfC)$. Since $\mathcal{A}(RC) \subseteq \mathcal{A}(OfC)$ and $\mathcal{A}(COMM) \subseteq \mathcal{A}(CIRC)$, the language Z_1 is also a witness language for the properness of the inclusion $\mathcal{A}(RC) \subset \mathcal{A}(CIRC)$.

□

Before we prove the proper inclusion $\mathcal{A}(RL_1^P) \subset \mathcal{A}(UC)$, we give some characterization of the languages in the set $\mathcal{A}(RL_1^P)$.

Lemma 3.26. *A language $L \subseteq V^*$ is in $\mathcal{A}(RL_1^P)$ if and only if one of the following cases holds:*

1. *There are an alphabet U with $V \subseteq U$, a word $w \in U^+$, and a set M of deletion rules, insertion rules, or substitution rules over the alphabet U such that $L = \{\, v \mid v \in V^*, v \Longrightarrow_M^+ w \,\}$.*
2. *There is a subset V' of the alphabet V such that $L = (V')^*$.*
3. *It holds $L = \emptyset$.*
4. *It holds $L = \{\lambda\}$.*

Proof. Let $L = L(\mathcal{N})$ for some network $\mathcal{N} = (V, U, N_1, N_2, \ldots, N_n, E, n_\mathrm{i}, n_\mathrm{o})$ where all filters are in RL_1^P. If the input node and output node coincide, then we obtain $L(\mathcal{N}) = V^*$ which belongs to case (2).

We now assume that input node and output node differ. Let $(M_{n_\mathrm{i}}, I_{n_\mathrm{i}}, O_{n_\mathrm{i}})$ be the input node. Because any language in RL_1^P contains at most one word, we have one of the following cases especially for the output filter

of the input node: O_{n_i} is empty, $O_{n_i} = \{w\}$ for some non-empty word $w \in U$, or $O_{n_i} = \{\lambda\}$. We discuss these cases.

If O_{n_i} is the empty set, then no word can leave the input node, and consequently, $L(\mathcal{N}) = \emptyset$ which corresponds to case (3).

If $O_{n_i} = \{w\}$ for a non-empty word $w \in U^+$, then all and only the words of the set $K = \{ v \mid v \in V^*,\ v \Longrightarrow^+_{M_{n_i}} w \}$ lead to w and, hence, to a word which can leave the input node. If the outgoing word w leads to a word arriving in the output node (after some transformations), then $L(\mathcal{N}) = K$ and case (1) holds. If the outgoing word does not lead to a word arriving in the output node, then $L(\mathcal{N}) = \emptyset$ and case (3) holds.

If $O_{n_i} = \{\lambda\}$ and the input node is a substitution or insertion node, then no other input word than λ leads to a word which can leave the input node and we get $L(\mathcal{N}) = \{\lambda\}$ or $L(\mathcal{N}) = \emptyset$, *i.e.*, case (4) or case (3) holds, respectively. If $O_{n_i} = \{\lambda\}$ and the input node is a deletion node, we set V' as the set of all letters of the input alphabet V which can be deleted in the input node, $V' = \{ a \mid a \in V,\ a \to \lambda \in M_{n_i} \}$, and obtain that all words of $(V')^*$ and only these words of V^* yield a word which can leave the input node. Thus, we obtain $L(\mathcal{N}) = (V')^*$ or $L(\mathcal{N}) = \emptyset$ (depending on whether the outgoing empty word leads to a word arriving in the output node or not). Thus, case (2) or case (3) holds.

On the other hand, for each of the cases (1) to (4), we see in the above consideration how such a language can be accepted by a network where all filters belong to the class RL_1^P. □

We now prove the mentioned proper inclusion.

Theorem 3.27. *We have* $\mathcal{A}(RL_1^P) \subset \mathcal{A}(UC)$.

Proof. Let $L \in \mathcal{A}(RL_1^P)$. Furthermore, let $\mathcal{N} = (V, U, N_1, N_2, \ldots, N_n, E, n_i, n_o)$ be a network such that $L = L(\mathcal{N})$ and all filters belong to the class RL_1^P. By Lemma 3.26, there are four cases for L.

If $L = \{ v \mid v \in V^*,\ v \Longrightarrow^+_M w \}$, by the proof of Lemma 3.26, $M = M_{n_i}$ and $\{w\}$ is the output filter of the input node of $\mathcal{N}$. Then we construct the network

$$\mathcal{N}_1 = (V, U, (M_{n_i}, \{w\}, \{w\}), (\emptyset, \{w\}, \{w\}), \{(1,2)\}, 1, 2).$$

It is easy to see that $L(\mathcal{N}_1) = L$. Since all filters are uniform, we get $L \in \mathcal{A}(UC)$.

If $L = (V')^*$, we consider the network

$$\mathcal{N}_2 = (V', V' \cup \{\S\}, (\emptyset, \{\S\}, \{\S\}), \emptyset, 1),$$

where $\S \notin V'$ which accepts $(V')^*$. Since all filters of $\mathcal{N}_2$ are uniform codes, we have $L \in \mathcal{A}(UC)$ also in this case.

If $L = \emptyset$, we consider the network

$$\mathcal{N}_3 = (V, V \cup \{\S\}, (\emptyset, \{\S\}, \{\S\}), (\emptyset, \{\S\}, \{\S\}), \{(1,2)\}, 1, 2),$$

which accepts the empty set because no word of V^* can be transformed into a word which can leave the input node. Since all filters of $\mathcal{N}_3$ are uniform codes, we have $L \in \mathcal{A}(UC)$ in this case, too.

If $L = \{\lambda\}$, we consider the network

$$\mathcal{N}_4 = (V, V \cup \{\S\}, (\{\lambda \to \S\}, \{\S\}, \{\S\}), (\emptyset, \{\S\}, \{\S\}), \{(1,2)\}, 1, 2).$$

If $w \in V^+$, then w cannot be transformed into $\S$ by the rules of the input node. Hence w is not in $L(\mathcal{N}_4)$. On the other hand, by one application of the insertion rule to the empty word, $\S$ is obtained and will be successfully sent to the output node, which gives $\lambda \in L(\mathcal{N}_4)$. Therefore $L(\mathcal{N}_4) = \{\lambda\}$. Since all filters of $\mathcal{N}_4$ are uniform codes, we have $L \in \mathcal{A}(UC)$.

Thus, we get the inclusion $\mathcal{A}(RL_1^P) \subseteq \mathcal{A}(UC)$. The properness follows with the language $F_2 = \{a_1^3, a_2^3\}$ which belongs to the class $\mathcal{A}(UC)$ but not to $\mathcal{A}(RL_i^P)$ according to Lemma 3.22. □

We now prove the incomparability results.

Theorem 3.28. *The classes $\mathcal{A}(UC)$ and $\mathcal{A}(RC)$ are incomparable.*

Proof. From Lemma 3.23, we have the relation $L_5 \in \mathcal{A}(RC) \setminus \mathcal{A}(UC)$. From Lemma 3.20, we know $\{ab\} \in \mathcal{A}(UC) \setminus \mathcal{A}(CIRC)$. Since $\mathcal{A}(RC) \subseteq \mathcal{A}(CIRC)$, we also have $\{ab\} \in \mathcal{A}(UC) \setminus \mathcal{A}(RC)$. □

Theorem 3.29. *The class $\mathcal{A}(RC)$ is incomparable to each of the classes $\mathcal{A}(MON)$, $\mathcal{A}(COMM)$, and $\mathcal{A}(REG_1^Z)$.*

Proof. Due to the equality $\mathcal{A}(MON) = \mathcal{A}(COMM) = \mathcal{A}(REG_1^Z)$, it suffices to show that there is a language in the set $\mathcal{A}(RC) \setminus \mathcal{A}(COMM)$ and a language in the set $\mathcal{A}(COMM) \setminus \mathcal{A}(RC)$.

In Lemma 3.21, we have shown for $L = Circ(\{abc\})$ that $L \in \mathcal{A}(RC) \setminus \mathcal{A}(COMM)$.

From Lemma 3.24, we know further that $Z_1 \in \mathcal{A}(COMM) \setminus \mathcal{A}(OfC)$. Due to the relation $\mathcal{A}(RC) \subseteq \mathcal{A}(OfC)$, we also have $Z_1 \in \mathcal{A}(COMM) \setminus \mathcal{A}(RC)$. □

Theorem 3.30. *The class $\mathcal{A}(RC)$ is incomparable to each of the classes $\mathcal{A}(RL_i^P)$ for $i \geq 1$.*

Proof. Due to the inclusions $\mathcal{A}(RL_i^P) \subset \mathcal{A}(RL_{i+1}^P)$, it suffices to show that there is a language in the set $\mathcal{A}(RL_1^P) \setminus \mathcal{A}(RC)$ and, for each $i \geq 1$, there is a language in the set $\mathcal{A}(RC) \setminus \mathcal{A}(RL_i^P)$.

We know that $\{ab\} \in \mathcal{A}(RL_1^P) \setminus \mathcal{A}(CIRC)$ by Lemma 3.20 and, together with the relation $\mathcal{A}(RC) \subset \mathcal{A}(CIRC)$, we also have $\{ab\} \in \mathcal{A}(RL_1^P) \setminus \mathcal{A}(RC)$.

In Lemma 3.22, we showed that $F_{i+1} \in \mathcal{A}(RC) \setminus \mathcal{A}(RL_i^P)$ for any natural number $i \geq 1$. □

Theorem 3.31. *The class $\mathcal{A}(UC)$ is incomparable to each of the classes $\mathcal{A}(MON)$, $\mathcal{A}(COMM)$, $\mathcal{A}(REG_1^Z)$, and $\mathcal{A}(CIRC)$.*

Proof. Due to the equality $\mathcal{A}(MON) = \mathcal{A}(COMM) = \mathcal{A}(REG_1^Z)$ and the inclusion $\mathcal{A}(COMM) \subset \mathcal{A}(CIRC)$, it suffices to show that there is a language in the set $\mathcal{A}(UC) \setminus \mathcal{A}(CIRC)$ and a language in the set $\mathcal{A}(COMM) \setminus \mathcal{A}(UC)$.

We know that $\{ab\} \in \mathcal{A}(UC) \setminus \mathcal{A}(CIRC)$ by Lemma 3.20.

From Lemma 3.24, we know further that $Z_1 \in \mathcal{A}(COMM) \setminus \mathcal{A}(OfC)$. Due to the relation $\mathcal{A}(UC) \subset \mathcal{A}(OfC)$, we also have $Z_1 \in \mathcal{A}(COMM) \setminus \mathcal{A}(UC)$. □

Theorem 3.32. *The class $\mathcal{A}(UC)$ is incomparable to each of the classes $\mathcal{A}(RL_i^P)$ for $i \geq 2$.*

Proof. Due to the inclusions $\mathcal{A}(RL_i^P) \subset \mathcal{A}(RL_{i+1}^P)$, it suffices to show that there is a language in the set $\mathcal{A}(RL_2^P) \setminus \mathcal{A}(UC)$ and, for each $i \geq 2$, there is a language in the set $\mathcal{A}(UC) \setminus \mathcal{A}(RL_i^P)$.

We know that $\{a\}^*\{b\} \in \mathcal{A}(RL_2^P) \setminus \mathcal{A}(NIL)$ by Lemma 3.13 and, together with the relation $\mathcal{A}(UC) \subset \mathcal{A}(NIL)$, we also have $\{a\}^*\{b\} \in \mathcal{A}(RL_2^P) \setminus \mathcal{A}(UC)$.

In Lemma 3.22, we showed that $F_{i+1} \in \mathcal{A}(UC) \setminus \mathcal{A}(RL_i^P)$ for any natural number $i \geq 1$. □

Summarizing, we have proved the hierarchy which is shown in Figure 3.

Theorem 3.33. *The relations shown in Figure 3 hold.*

4. Conclusions

We have investigated accepting networks of evolutionary processors where the filters belong to subregular language families which are defined by restricting the resources needed for generating or accepting them or which are ideals or codes. We have inserted the newly defined language families into the hierachy of language families obtained by using languages of other subregular families as filters which was published in [12]. The hierarchy with the new results is shown in Figure 3.

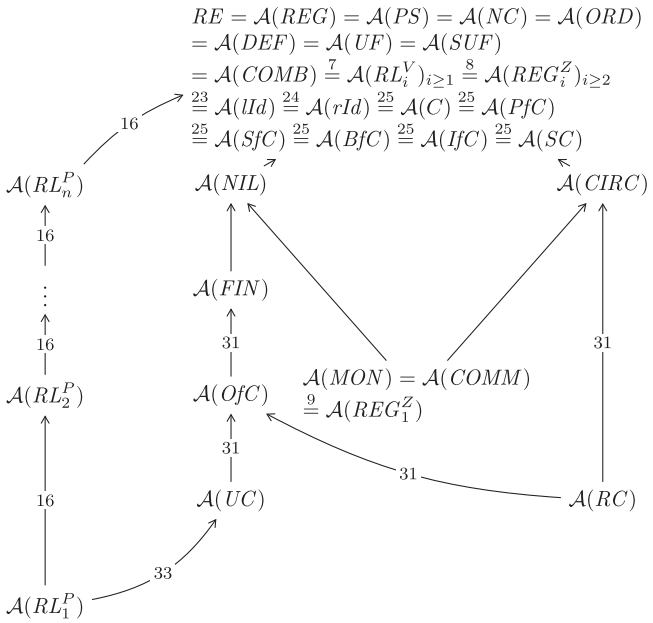


FIGURE 3. Hierarchy of language families by ANEPs with filters from subregular families. An arrow from a language family X to a language family Y stands for the proper inclusion $X \subset Y$. If two families X and Y are not connected by a directed path, then the families are incomparable. The labels at the arrows or equality signs refer to the theorems where the respective relation is shown. The other relations were proved in [12].

In [8], generating networks of evolutionary processors were investigated where the filters are ideals or special codes. In [20], generating networks of evolutionary processors were considered where the filters are determined by restricting resources. In [7], the two hierarchies obtained in the aforementioned papers were merged.

In the present paper, we have continued this research and considered the same language families for the filters in accepting networks of evolutionary processors. It is an extension of the paper [21].

REFERENCES

[1] J. Castellanos, C. Martín-Vide, V. Mitrana and J.M. Sempere, Solving NP-complete problems with networks of evolutionary processors, in *IWANN '01: Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks*. Vol. 2084 of *LNCS*. Springer-Verlag Berlin (2001) 621–628.

[2] J. Castellanos, C. Martín-Vide, V. Mitrana and J.M. Sempere, Networks of evolutionary processors. *Acta Inf.* **39** (2003) 517–529.

[3] E. Csuhaj-Varjú and A. Salomaa, Networks of parallel language processors, in *New Trends in Formal Languages – Control, Cooperation, and Combinatorics*. Vol. 1218 of *LNCS*. Springer-Verlag Berlin (1997) 299–318.

[4] J. Dassow, Grammars with control by ideals and codes. *J. Autom. Lang. Combinat.* **23** (2018) 143–164.

[5] J. Dassow, F. Manea and B. Truthe, Networks of evolutionary processors: the power of subregular filters. *Acta Inf.* **50** (2013) 41–75.

[6] J. Dassow and V. Mitrana, Accepting networks of non-inserting evolutionary processors, in *Proceedings of NCGT 2008 – Workshop on Natural Computing and Graph Transformations, Leicester, United Kingdom, September 8, 2008*, edited by I. Petre and G. Rozenberg. University of Leicester (2008) 29–41.

[7] J. Dassow and B. Truthe, Networks of evolutionary processors with resources restricted filters. To appear in *J. Autom. Lang. Combinat.* (2019).

[8] J. Dassow and B. Truthe, Networks with evolutionary processors and ideals and codes as filters. To appear *Int. J. Found. Comput. Sci.* (2019).

[9] I.M. Havel, The theory of regular events II. *Kybernetika* **5** (1969) 520–544.

[10] M. Holzer and B. Truthe, On relations between some subregular language families, in *Seventh Workshop on Non-Classical Models of Automata and Applications (NCMA), Porto, Portugal, August 31 – September 1, 2015, Proceedings*, edited by R. Freund, M. Holzer, N. Moreira and R. Reis. Vol. 318 of *books@ocg.at*. Österreichische Computer Gesellschaft (2015) 109–124.

[11] H. Jürgensen and S. Konstantinidis, Codes, in vol. 1 of *Handbook of Formal Languages*, edited by G. Rozenberg and A. Salomaa. Springer-Verlag, Berlin (1997) 511–607.

[12] F. Manea and B. Truthe, Accepting networks of evolutionary processors with subregular filters. *Theory Comput. Syst.* **55** (2014) 84–109.

[13] M. Margenstern, V. Mitrana and M.-J. Peréz-Jiménez, Accepting hybrid networks of evolutionary processors, in *DNA Computing – 10th International Workshop on DNA Computing*. Edited by C. Ferretti, G. Mauri and C. Zandron. In Vol. 3384 of *LNCS*. Springer-Verlag, Berlin (2004) 235–246.

[14] C. Martín-Vide and V. Mitrana, Networks of evolutionary processors: Results and perspectives, in *Molecular Computational Models: Unconventional Approaches*, edited by M. Gheorghe. Idea Group Publishing (2005) 78–114.

[15] G. Rozenberg and A. Salomaa, *Handbook of Formal Languages*. Springer-Verlag, Berlin (1997).

[16] H.-J. Shyr, Free Monoids and Languages. Hon Min Book Company, Taichung, Taiwan (1991).

[17] H.-J. Shyr and G. Thierrin, Ordered automata and associated languages. *Tankang J. Math.* **5** (1974) 9–20.

[18] H.-J. Shyr and G. Thierrin, Power-separating regular languages. *Math. Syst. Theory* **8** (1974) 90–95.

[19] B. Truthe, Hierarchy of subregular language families, Tech. Rep. 1801, Justus-Liebig-Universität Giessen, Institut für Informatik (2018).

[20] B. Truthe, Networks of evolutionary processors with resources restricted filters, in *Tenth Workshop on Non-Classical Models of Automata and Applications (NCMA), Košice, Slovakia, August 21–22, 2018, Proceedings*, edited by R. Freund, M. Hospodár, G. Jirásková and G. Pighizzini. Vol. 332 of *books@ocg.at*. Österreichische Computer Gesellschaft (2018) 165–180.

[21] B. Truthe, Accepting networks of evolutionary processors with resources restricted filters, in *Eleventh Workshop on Non-Classical Models of Automata and Applications (NCMA), Valencia, Spain, July 2–3, 2019, Proceedings*, edited by R. Freund, M. Holzer and J.M. Sempere. Vol. 336 of *books@ocg.at*. Österreichische Computer Gesellschaft (2019) 187–202.

[22] B. Wiedemann, Vergleich der Leistungsfähigkeit endlicher determinierter Automaten, diplomarbeit, Universität Rostock (1978).