Fundamenta Informaticae XX (2013) 1–26
DOI 10.3233/FI-2012-0000
IOS Press

Power of S-kR-RRWW-automata

Petr Hoffmann[*]
Faculty of Mathematics and Physics
Charles University in Prague,
Malostranské nám. 25, 118 00, Praha 1, Czech Republic
krysar78@gmail.com

Abstract. Single k-reversible restarting automata are a special version of restarting automata which can be effectively learned from samples. We show that their power lies between GCSL and CSL. We show that their subclasses form an infinite hierarchy of classes of languages with respect to the reversibility level k and we also show that limiting types of allowed rewrites lowers the power of the model. Finally, we study their relation to strictly locally testable restarting automata.

1. Introduction

The so-called *analysis by reduction* (ABR for short) [8, 12, 18] forms an important tool for parsing natural language sentences. The ABR consists in a stepwise simplification of a given extended sentence until a simple sentence is obtained or an error is found, as illustrated by this example [8]: <u>Peter and Jane work</u> slowly. ⇒ Jane works <u>slowly</u>. ⇒ Jane works.

As you can see, the input sentence is iteratively simplified (changed parts are underlined) by either just removing a part or by replacing a part by another shorter part. Finally, we obtain a simple sentence. As we ended with a correct sentence and individual simplifications are not allowed to remove errors, we conclude that the original input sentence is correct as well.

The *restarting automaton* [7, 16, 17], defining a set of simplification rules and a set of simple sentences, can be used to model the ABR. Such a model defines a language as a set of sentences, which can be simplified to a simple sentence of the model by the simplification rules of the model.

In [5] we propose the Omega[*] method for learning a subclass of restarting automata, the single k-reversible restarting automata, from positive samples. A similar model, called strictly locally testable

[*]This work was supported by the Grant Agency of the Czech Republic under the projects P103/10/0783 and P202/10/1333.

automata, was considered in [14]. It is important to know the power of our model as well as the relation to the model of [14] to better understand the capabilities of the Omega* . We analyze those questions in this paper.

We will use common notions from formal languages theory [6, 11, 4, 1]. By FSA, GCSL, CSL, and EL we denote a (non-deterministic) finite state automaton, growing context-sensitive, context-sensitive, and even linear languages, respectively. By $\mathrm{RQ}(L, L')$ and $\mathrm{LQ}(L, L')$ we denote the right-quotient and the left-quotient of L by L'. The expression $\mathcal{L}(X)$ denotes the class of languages accepted by automata of type X. We will sometimes use regular expressions instead of languages to improve readability, e.g. instead of $\{0\} \cdot \Sigma$, we will write just 0Σ to represent the language of words consisting of the symbol 0 followed by a symbol from an alphabet Σ.

2. Restarting Automata

To model the ABR, we often use restarting automata defined below [7, 16, 17].

Definition 2.1. ([7, 16, 17])
A *restarting automaton* (RRWW-automaton) is a triplet $M = (\Sigma, \Gamma, I)$, where Σ is an input alphabet, Γ is a working alphabet, $\Sigma \subseteq \Gamma$ (the symbols from $\Gamma \setminus \Sigma$ are called auxiliary symbols), and I is a finite set of meta-instructions of the following two forms:

- a rewriting meta-instruction $(E_\ell, x \to y, E_\mathrm{r})$, where $x, y \in \Gamma^*$ such that $|x| > |y|$ and $E_\ell, E_\mathrm{r} \subseteq \Gamma^*$ are regular languages (we call the expression $x \to y$ a *rewrite*, the words x and y are called a *rewritten* and a *replacement* word, respectively, and $|x|$ is called the *length of the rewrite*; the maximum length of a rewrite taken over all rewriting meta-instructions is called the *size of the window of* M), or

- an accepting meta-instruction (E, Accept), where $E \subseteq \Gamma^*$ is a regular language.

We say that M can *directly reduce* a word u into a word v ($u \vdash_M v$) if $u = u_1 x u_2, v = u_1 y u_2$ for some $u_1, u_2, x, y \in \Gamma^*$ and there is a rewriting meta-instruction $(E_\ell, x \to y, E_\mathrm{r}) \in I$, where $u_1 \in E_\ell$ and $u_2 \in E_\mathrm{r}$. Let $\vdash_M^*$ denote the reflexive and transitive closure of $\vdash_M$. If $u \vdash_M^* v$ for some words u, v, we say that M can *reduce* u to v.

A word $w \in \Gamma^*$ is *accepted by* M *directly* if $w \in E$ for at least one accepting meta-instruction $(E, \mathrm{Accept}) \in I$. A word $w \in \Gamma^*$ is *accepted by* M if there exists a word $w' \in \Gamma^*$ such that $w \vdash_M^* w'$ and w' is accepted by M directly. The *characteristic language accepted by* M is the set of all words accepted by M, thus $\mathrm{L_C}(M) = \{w \in \Gamma^*; w \text{ is accepted by } M\}$. The *language accepted by* M is the set of all words over the input alphabet Σ accepted by M, thus $\mathrm{L}(M) = \mathrm{L_C}(M) \cap \Sigma^*$.

We define two subtypes of RRWW-automata. An RRWW-automaton having $\Gamma = \Sigma$ is called an RRW-automaton. An RR-automaton is an RRW-automaton such that for each meta-instruction $(E_\ell, u \to v, E_\mathrm{r})$, it holds that v can be obtained from u by deleting some symbols from u.

One of the fundamental properties of all models of restarting automata is the following so-called error preserving property. This property means that whatever reduction performs given restarting automaton on an input word not belonging to the language accepted by the automaton ("the word contains an error"), the resulting word is not accepted by the automaton (it contains still some error), too.

Proposition 2.2. (Error Preserving Property; [8])
Let $M = (\Sigma, \Gamma, I)$ be an RRWW-automaton, and let u, v be words over its input alphabet Σ. If $u \vdash_M^* v$ holds and $u \notin \mathrm{L}(M)$, then $v \notin \mathrm{L}(M)$, either.

This proposition is extensively used for proving that some language can not be accepted by any restarting automaton of a certain type (see [17]).

The general RRWW-automata are not suitable for learning from positive samples only. Therefore in [14] a subclass of restarting automata, called strictly locally testable automata, was introduced, that is based on strictly locally testable languages [19, 9].

Definition 2.3. ([19, 9])
Let $k > 0$ and let Σ be an alphabet. A language $L \subseteq \Sigma^*$ is *strictly k-testable* if there exist finite sets $A, B, C \subseteq \Sigma^k$ such that for each $w \in \Sigma^{\geq k}$, it holds that $w \in L$ if and only if $\mathrm{P}_k(w) \in A$, $\mathrm{S}_k(w) \in B$, and $\mathrm{I}_k(w) \subseteq C$, where $\mathrm{P}_k(w), \mathrm{S}_k(w)$, and $\mathrm{I}_k(w)$ denote k-length prefix of w, k-length suffix of w, and the set of all interior subwords of the length k of w, respectively. The (A, B, C) is called a *triple* for L. A language L is called *strictly locally testable* if L is strictly k-testable for some k.

The considered restriction proposed in the following definition allowed to learn the obtained model from positive samples.

Definition 2.4. ([14])
Let $k > 0$. A rewriting meta-instruction $(E_\ell, x \to y, E_\mathrm{r})$ is *strictly k-testable*, if both E_ℓ and E_r are strictly k-testable and $|x| \leq k$. An accepting meta-instruction (E, Accept) is *strictly k-testable* if E is strictly k-testable. A restarting automaton M is *strictly k-testable* (k-SLT-R-automaton), if all its meta-instructions are strictly k-testable. A restarting automaton is *strictly locally testable* (SLT-R-automaton), if it is strictly k-testable for some k.

The Omega* method is similar to the approach of [14]. The considered model utilizes the k-reversible languages [2, 13] instead of strictly locally testable languages.

Definition 2.5. ([2, 13])
Let $k \geq 0$. Let $A = (Q, \Sigma, \delta, I, F)$ be a FSA. Then $u \in \Sigma^k$ is a *k-follower of $q \in Q$ in A* if $\delta^*(q, u) \neq \emptyset$. The FSA A is *deterministic with lookahead k* if for any pair $p, q \in Q$ such that $p \neq q$, if $p, q \in I$ or $p, q \in \delta(r, a)$ for some $r \in Q, a \in \Sigma$, then no word is a k-follower of both p and q. A is *k-reversible* if A is deterministic and A^R (the reverse of A) is deterministic with lookahead k. A language L is *k-reversible* if there is a k-reversible automaton A accepting L. A language L is *reversible* if it is k-reversible for some k.

Moreover, the learning method Omega* has the following feature: for all $u \in \Gamma^+, v \in \Gamma^*$ (Γ being the considered working alphabet), the resulting automata have at most one rewriting meta-instruction performing a rewrite $u \to v$ (where u and v share neither a common non-empty prefix nor a common non-empty suffix) and also only one accepting meta-instruction. It is therefore natural to introduce these restrictions into the model as well. It will help us better understand the power of the models returned by Omega* . The formal definition follows.

Definition 2.6. A *rewriting meta-instruction* $(E_\ell, x \to y, E_\mathrm{r})$ is called *k-reversible*, if both E_ℓ and E_r are k-reversible languages. An *accepting meta-instruction* (E, Accept) is called *k-reversible*, if E is a

k-reversible language. A *restarting automaton* is called *k-reversible* (kR-RRWW-automaton), if all its meta-instructions are k-reversible.

A *restarting automaton M* is called *single* (a S-RRWW-automaton), if M has at most one accepting meta-instruction, for each pair $x \to y$ (where $x, y \in \Gamma^*$) M has at most one meta-instruction of the form $(E_\ell, x \to y, E_\mathrm{r})$, and for each rewriting meta-instruction $(E_\ell, x \to y, E_\mathrm{r})$, it holds that x and y share neither a common non-empty prefix nor a common non-empty suffix.

A *restarting automaton* is called *single k-reversible* (S-kR-RRWW-automaton), if it is both single and k-reversible. A S-kR-RRWW-automaton not using auxiliary symbols is called a S-kR-RRW-automaton. A S-kR-RR-automaton is a S-kR-RRW-automaton such that for each rewriting meta-instruction $(E_\ell, u \to v, E_\mathrm{r})$, it holds that v can be obtained from u by deleting some symbols from u.

There is an important result [9] stating that the class of strictly locally testable languages is *properly* included in the class of reversible languages. This is of great importance because it shows that considering reversible languages leads to a larger class of languages to be used in meta-instructions compared to the strictly locally testable languages used in [14]. Note that there are more differences between S-kR-RRWW-automata and SLT-R-automata than the classes of languages allowed in meta-instructions, particularly, we consider only *single* restarting automata.

3. Results

Here we characterize the power of the discussed model. Let us start by relating our model to Chomsky's hierarchy. We will need the following lemma:

Lemma 3.1. For each kR-RRWW-automaton M, there is a S-kR-RRWW-automaton M' such that $\mathrm{L}(M) = \mathrm{L}(M')$.

Proof:
Let $M = (\Sigma, \Gamma, I)$ be a kR-RRWW-automaton. We will transform M into a S-kR-RRWW-automaton M' such that $\mathrm{L}(M) = \mathrm{L}(M')$.

If $\mathrm{L}(M) = \emptyset$, then it is easy to create a S-kR-RRWW-automaton accepting the same language. Below we suppose that M accepts a non-empty language.

If a kR-RRWW-automaton is not a S-kR-RRWW-automaton, then at least one of the following conditions holds:

(i) it has at least two accepting meta-instructions,

(ii) it has at least two rewriting meta-instructions performing the same rewrite, or

(iii) it has at least one rewriting meta-instruction such that the rewritten and the replacement words share a common non-empty prefix or a common non-empty suffix.

We will deal with all those possible problems below.

Let us suppose that the condition (i) holds, i.e. let M have more than one accepting meta-instruction. Of course, it may not be possible to merge all accepting meta-instructions into one and to preserve k-reversibility at the same time. However, we can use the power of auxiliary symbols to process the words accepted directly as follows. Let L_{ad} be the set of all words accepted by M directly. It is surely a regular language. Thus, there is a deterministic FSA $D = (Q, \Gamma, \delta, \{q_0\}, F)$ such that $L_{\mathrm{ad}} = \mathrm{L}(D)$. We define the following meta-instructions and we put them into I_{ad} (Q_q for $q \in Q$ are new auxiliary symbols):

- $(\lambda, ab \to Q_q, \Gamma^*)$ for each $a, b \in \Gamma$ and $q = \delta^*(q_0, ab)$,
- $(\lambda, Q_q a \to Q_{q'}, \Gamma^*)$ for each $q, q' \in Q, a \in \Gamma$, and $q' = \delta(q, a)$,
- for $\lambda \notin L_{\mathrm{ad}}$, we add $(\{Q_q; q \in F\} \cup (L_{\mathrm{ad}} \cap \Gamma), \mathrm{Accept})$, and
- for $\lambda \in L_{\mathrm{ad}}$, we add
  - $(\lambda, Q_q \to \lambda, \lambda)$ for all $q \in F$,
  - $(\lambda, x \to \lambda, \lambda)$ for all $x \in L_{\mathrm{ad}} \cap \Gamma$, and
  - $(\{\lambda\}, \mathrm{Accept})$.

The meta-instructions from I_{ad} can be surely used to accept exactly the words from L_{ad} and they can be safely used in place of the original accepting meta-instructions. Obviously, all meta-instructions from I_{ad} are zero-reversible and there is exactly one accepting meta-instruction. In what follows, we suppose that M has one accepting meta-instruction. Let it be $(L_{\mathrm{accept}}, \mathrm{Accept})$.

To deal with the condition (ii), we will need non-empty replacement words in nearly all rewriting meta-instructions. Then we will use new auxiliary symbols to make individual rewrites unique. We will prepare this as follows. Let I_0 denote the set of all rewriting meta-instructions with a non-empty replacement word. Then, for each rewriting meta-instruction $(L_\ell, x \to \lambda, L_{\mathrm{r}}) \in I \setminus I_0$ (i.e. a meta-instruction with the empty replacement word) we distinguish the following cases:

(C1) If both L_ℓ and L_{r} contain only the empty word λ, we put this meta-instruction into I_λ.

(C2) If L_{r} contains some non-empty words, we collect the first symbols of all such words — let them form a set Δ. For each $a \in \Delta$, we insert the meta-instruction $(L_\ell, xa \to a, \mathrm{LQ}(L_{\mathrm{r}}, a))$ into I_0 (here again the right context obviously remains k-reversible). You may notice that this meta-instruction satisfies the requirement mentioned in the condition (iii), but we will fix that later. If $\lambda \in L_{\mathrm{r}}$ and $L_\ell \neq \{\lambda\}$, we process $(L_\ell, x \to \lambda, \lambda)$ according to the step $(C3)$. If $\lambda \in L_{\mathrm{r}}$ and $L_\ell = \{\lambda\}$, we add $(\lambda, x \to \lambda, \lambda)$ into I_λ.

(C3) If L_{r} contains only λ but L_ℓ contains a non-empty word, we proceed similarly. We collect the last symbols of all such words — let them form a set Δ. For each $a \in \Delta$, we insert the meta-instruction $((\mathrm{LQ}(L_\ell^{\mathrm{R}}, a))^{\mathrm{R}}, ax \to a, L_{\mathrm{r}})$ into I_0. Again, we obtained a k-reversible meta-instruction — here we additionally used the fact that the class of k-reversible languages is closed under reversal [2]. Again, the problem with the condition (iii) will be fixed later. If $\lambda \in L_\ell$, we add $(\lambda, x \to \lambda, \lambda)$ into I_λ.

Obviously, the rewriting meta-instructions from $I_0 \cup I_\lambda$ perform the same reductions as those from I. Note that no rewriting meta-instruction from I_0 uses the empty replacement word. Moreover, all those meta-instructions are k-reversible. We would like to use the meta-instructions from I_0 in M', but their presence could mean that M' still satisfies the condition (ii). Therefore, for each rewriting meta-instruction $r \in I_0$, we create three new alphabets $\Gamma_{r,0}$, $\Gamma_{r,1}$, and $\Gamma_{r,2}$, forming together an alphabet Γ_r (for $a \in \Gamma$ there will thus be three associated symbols $a_{r,i} \in \Gamma_r$ for $i \in \{0,1,2\}$ — we will further use a_r to denote one of the symbols associated with a), we rewrite the replacement word in r using the associated symbols from $\Gamma_{r,0}$ in the obvious way, and we put the resulting meta-instruction into I_1. The purpose of $\Gamma_{r,1}$ and $\Gamma_{r,2}$ will be obvious later. Note that I_1 contains no pair of rewriting meta-instructions performing the same rewrite as the replacement words are unique — each one is composed of symbols from an alphabet specific to the particular meta-instruction.

Let $\Gamma_* = \Gamma \cup \bigcup_{r \in I_0} \Gamma_r$. Note that for each symbol $a \in \Gamma$, there can be therefore many associated symbols in Γ_* (there are three for each rewriting meta-instruction from I_0). We will say that a symbol $a \in \Gamma$ is *compatible* with itself and also with $a_{r,0}, a_{r,1}, a_{r,2} \in \Gamma_r$ for all $r \in I_0$. For a word w, we define a set of compatible words as $\mathrm{c}(w) = \{w' \in \Gamma_*^{|w|};$ for each i such that $0 \leq i < |w|$, it holds that w_i is compatible with $w'_i\}$. For a language $L_{\mathrm{orig}} \subseteq \Gamma^*$, we define $\mathrm{c}(L_{\mathrm{orig}}) = \bigcup_{w \in L_{\mathrm{orig}}} \mathrm{c}(w)$. For each rewriting meta-instruction $(L_\ell, x \to y, L_{\mathrm{r}}) \in I_1$ and each $x' \in \mathrm{c}(x)$, we add the meta-instruction $(\mathrm{c}(L_\ell), x' \to y, \mathrm{c}(L_{\mathrm{r}}))$ to I_2. The resulting meta-instruction is k-reversible as can be easily seen from the Definition 2.5. We will call the set of all meta-instructions created from the same meta-instruction in this step a *group*.

It is clear that groups can be distinguished by the replacement words because they remain the same as before and thus are group specific (remember that there were no duplicate replacement words in I_1). Also, no pair of meta-instructions from the same group share the same rewritten word.

Now no pair of rewriting meta-instructions from I_2 performs the same rewrite. However, the condition (iii) can still be a problem. We will inspect all rewriting meta-instructions as follows — let $r' = (L_\ell, x \to y, L_{\mathrm{r}}) \in I_2$. Note that $y \neq \lambda$ holds for all $r' \in I_2$. Let $x = awb$ where $a \in \Gamma_{r,i}, b \in \Gamma_{r,j}$, and $w \in \Gamma_r^*$ for some $r \in I_0$, then we rewrite $y \in \Gamma_{r,0}^*$ using symbols from $\Gamma_{r,\ell}$ where $\ell \in \{0,1,2\} \setminus \{i,j\}$ is selected arbitrarily — here we replace each symbol $a_r \in \Gamma_{r,0}$ with $a'_r \in \Gamma_{r,\ell}$ and then we insert this rewriting meta-instruction into I_3. Note that the meta-instructions from I_3 do not satisfy the requirements of the condition (iii). Also note that the new meta-instructions are still k-reversible. As the new replacement words remain unique for each group, no additional action is needed to avoid satisfying the condition (ii).

Recall the meta-instructions from I_λ. As we introduced new symbols in the previous steps, we need to adapt meta-instructions from I_λ to this new situation. Therefore, for each $(\lambda, w \to \lambda, \lambda) \in I_\lambda$, we put $(\lambda, w' \to \lambda, \lambda)$ into I'_λ for each $w' \in \mathrm{c}(w)$.

Finally, let $M' = (\Sigma, \Gamma_*, I_3 \cup I'_\lambda \cup \{(\mathrm{c}(L_{\mathrm{accept}}), \mathrm{Accept})\})$.

Note that the computation of M' resembles the computation of M. Only some compatible symbols occur in the processed words in place of the ones used by M.

Now, the following holds:

- M' contains exactly one accepting meta-instruction.
- Rewriting meta-instructions originating from the same group are distinguished by the rewritten word.
- Rewriting meta-instructions originating from different groups are distinguished by the alphabet used within the replacement word.

- Rewriting meta-instructions having the empty replacement word have contexts containing exactly the empty word.
- There is no rewriting meta-instruction performing a rewrite $x \to y$ such that x and y share the same non-empty prefix or the same non-empty suffix (note that the replacement word is either empty or it is composed of symbols from an alphabet that does not contain the first and the last symbol of the rewritten word).
- M' is k-reversible.

Thus, we obtained a $\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRWW}$-automaton M'.

It remains to prove that $\mathrm{L}(M) = \mathrm{L}(M')$. It is clear that words accepted by M directly are accepted by M' directly as well, and the same holds for all words compatible with words accepted by M directly. If M reduces u to v, then each word $u' \in \mathrm{c}(u)$ (i.e. compatible with u) can be reduced by M' to a word compatible with v. Thus, $\mathrm{L}(M) \subseteq \mathrm{L}(M')$. We will now prove that $\mathrm{L}(M') \subseteq \mathrm{L}(M)$. Let $\mathrm{c}^{-1}(w)$ for $w \in \Gamma_*^*$ denote the unique word $w' \in \Gamma^*$ such that $w \in \mathrm{c}(w')$. If $w \in \mathrm{L}_\mathrm{c}(M')$ is accepted by M' directly, then surely $\mathrm{c}^{-1}(w) \in \mathrm{L}(M)$. If M' reduces $u \in \mathrm{L}_\mathrm{c}(M')$ to λ using a meta-instruction from I'_λ, then M reduces $\mathrm{c}^{-1}(u)$ to λ. If M' reduces $u \in \mathrm{L}_\mathrm{c}(M')$ to v using a meta-instruction from I_3, then M reduces $\mathrm{c}^{-1}(u)$ to $\mathrm{c}^{-1}(v)$. Overall, we have $\mathrm{L}(M') \subseteq \mathrm{L}(M)$.

□

The following theorem is a modification of a similar one from [14].

Theorem 3.2. Let $k \geq 0$. It holds that $\mathsf{GCSL} \subset \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRWW}) \subseteq \mathsf{CSL}$.

Proof:
First, we prove that $\mathsf{GCSL} \subseteq \mathcal{L}(\mathsf{S}\text{-}0\mathsf{R}\text{-}\mathsf{RRWW})$. Let $G = (\Sigma, \Gamma, S, P)$ be a growing context-sensitive grammar. We will construct a $\mathsf{S}\text{-}0\mathsf{R}\text{-}\mathsf{RRWW}$-automaton M such that $\mathrm{L}(M) = \mathrm{L}(G)$. Without loss of generality, we suppose that P does not contain any rules $S \to X$ for $X \in \Gamma$ (otherwise, we could replace each such rule with several rules $S \to \alpha$ for all α such that $X \to \alpha \in P$). We will iteratively modify the original grammar, until we reach a form suitable for an easy transformation to a $\mathsf{S}\text{-}0\mathsf{R}\text{-}\mathsf{RRWW}$-automaton:

1. With each terminal symbol $a \in \Sigma$, we associate a new non-terminal symbol $\overline{a}$ (let the new symbols form the alphabet $\overline{\Sigma}$). This step will later help us to manipulate with "terminal" symbols during rewrites as needed.
2. For each rule $\alpha \to \beta \in P$, we create a set of new rules of the form $\alpha' \to \beta'$ (these new rules will be placed into $\overline{P}$) where
   - α' is obtained from α by replacing each occurrence of each terminal symbol a with the associated non-terminal symbol $\overline{a}$ (note that this means there is no terminal symbol on the left-hand side of the resulting rule), and
   - β' is obtained from β by replacing *some* occurrences of each terminal symbol a with the associated non-terminal symbol $\overline{a}$ in all possible ways (including replacing all occurrences of a and also none of them).

Let $\Gamma_1 = \Gamma \cup \overline{\Sigma}$ and let the resulting grammar be $G_1 = (\Sigma, \Gamma_1, S, \overline{P})$.

3. With each non-terminal symbol $X \in \Gamma_1$, we associate a new non-terminal symbol $\overline{X}$ (let those new non-terminal symbols form the alphabet $\overline{\Gamma_1}$).

4. For each rule $\alpha \to \beta \in \overline{P}$, we create the rules $\alpha' \to \beta'$ (these new rules will form P') where

   - α' is obtained from α by replacing *some* occurrences of each non-terminal symbol X with the associated non-terminal symbol $\overline{X}$ in all possible ways (including replacing all occurrences of X and also none of them), and
   - β' is obtained from β by changing its first symbol and its last symbol to be different from the first occurrence of a non-terminal symbol in α' and from the last occurrence of a non-terminal symbol in α', respectively, by replacing the particular occurrences of non-terminal symbols X in β with the associated non-terminal symbols $\overline{X}$ (when necessary, i.e. if the first or last symbol of β is a terminal symbol or it is different from the first or the last symbol of α', then no change is performed).

   Let $\Gamma' = \Gamma_1 \cup \overline{\Gamma_1}$ and let the obtained grammar be $G' = (\Sigma, \Gamma', S, P')$.

Let us show that $\mathrm{L}(G) = \mathrm{L}(G')$. The first step just added some new non-terminal symbols. According to the second step, they are used only in places where original terminal symbols can occur. Thus, if we ignore the difference between the original terminal symbols and the associated non-terminal symbols, the grammar G_1 rewrites in the same way as G. Moreover, once G_1 yields a word over Σ, this word thus surely belongs to $\mathrm{L}(G)$. On the other hand, consider a derivation of $w \in \mathrm{L}(G)$. We will modify each applied rule $\alpha \to \beta$ as follows. We replace terminal symbols in α with associated non-terminal symbols, and all occurrences of terminal symbols from β such that they are rewritten in a later step, are replaced with associated non-terminal symbols as well. This obviously gives a derivation for w using the rules of G_1.

The third step just adds new non-terminal symbols that will be used in the fourth one. The idea of the transformation of G into a S-0R-RRWW-automaton is to associate each rule $\alpha \to \beta$ to the rewriting meta-instruction $(Z^*, \beta \to \alpha, Z^*)$ (where Z denotes the working alphabet of the restarting automaton). However, e.g. in the case of the rule $0X0 \to 0110$ this idea would lead to the rewriting meta-instruction $(Z^*, 0110 \to 0X0, Z^*)$ that violates the condition for an automaton to be single. Unfortunately, in the case of zero-reversible restarting automata, the contexts of rewriting meta-instructions can be too weak for us to be able to convert this meta-instruction into $(Z^* \cdot \{0\}, 11 \to X, \{0\} \cdot Z^*)$ to avoid the mentioned violation. Therefore, we change the rewriting rules so that the left and right sides will both start and end with different symbols. This is exactly what is performed in the fourth step.

Again, does this preserve the generated language? As before, we see that the rewriting performed by both grammars G_1 and G' is the same if we ignore the difference between the original symbols and the associated non-terminal symbols. Particularly, if G' yields a word over Σ, it surely belongs to $\mathrm{L}(G_1)$. On the other hand, let us have a derivation for $w \in \mathrm{L}(G_1)$. Because of the changes made in the first two steps, we know that the terminal symbols occur only on the right-hand sides of rules used in the derivation. Note that for each rule used in the derivation, we have at least a similar rule that differs only in using some associated non-terminal symbols in place of the original ones. We start at the beginning of the derivation. We proceed iteratively, using rules of G' corresponding to the rules used originally (i.e. the only difference is the presence of associated non-terminal symbols in place of some original non-terminal symbols). We know the corresponding rule is always available in P' as our left sides cover all possible ways of using those associated non-terminal symbols. Eventually, there will remain no non-terminal symbols. So, w is derived using G'.

Finally, we define $\Gamma'' = \Sigma \cup \Gamma'$ and $M = (\Sigma, \Gamma'', I)$, where I is as follows. For each rule $\alpha \to \beta \in P'$ such that $|\alpha| < |\beta|$, there is $(\Gamma''^*, \beta \to \alpha, \Gamma''^*) \in I$. If $\lambda \notin \mathrm{L}(G)$, then $(\{S\} \cup \{a \in \Sigma; S \to a \in P'\}, \mathrm{Accept}) \in I$. Otherwise, if $\lambda \in \mathrm{L}(G)$, then $(\{\lambda\}, \mathrm{Accept}) \in I$, $(\{\lambda\}, S \to \lambda, \{\lambda\}) \in I$, and $(\{\lambda\}, a \to \lambda, \{\lambda\}) \in I$ for all $a \in \Sigma$ such that $S \to a \in P'$.

It can be easily seen that M is a $\mathsf{S\text{-}0R\text{-}RRWW}$-automaton. The automaton M tries to find derivations of given words (with a few exceptions handled by the accepting meta-instruction) and accepts or rejects the input words accordingly. Thus, $\mathrm{L}(G) = \mathrm{L}(M)$.

As each 0-reversible language is also k-reversible for all $k > 0$ [2], it immediately follows that $\mathsf{GCSL} \subseteq \mathcal{L}(\mathsf{S\text{-}}k\mathsf{R\text{-}RRWW})$. To prove that this inclusion is proper, we will show that our model can accept the language $L_{\neg\mathsf{GCSL}} = \{w\overline{w}; w \in \{0,1\}^+\}$ that is known not to belong to GCSL [10, 3]. Let $M_{\neg\mathsf{GCSL}} = (\{0,1,\overline{0},\overline{1}\} \cup \{X_{x,y}, X'_{x,y}; x,y \in \{0,1\}\} \cup \{Y_x, Y'_x; x \in \{0,1\}\}, \{0,1,\overline{0},\overline{1}\}, I)$, where I contains the following meta-instructions (where $a,b,c,d,e \in \{0,1\}$):

(I1) $(\{\lambda\}, ab \to X_{a,b}, \Sigma^*)$,

(I2) $(\{X_{a,b}\} \cdot \Sigma^*, \overline{ac} \to Y_c, \overline{\Sigma}^*)$,

(I3) $(\{\lambda\}, X_{a,b}c \to X'_{b,c}, \Sigma^* \cdot \{Y_d\} \cdot \overline{\Sigma}^*)$,

(I4) $(\{X'_{b,c}\} \cdot \Sigma^*, Y_b\overline{d} \to Y'_d, \overline{\Sigma}^*)$,

(I5) $(\{\lambda\}, X'_{b,c}d \to X_{c,d}, \Sigma^* \cdot \{Y'_e\} \cdot \overline{\Sigma}^*)$,

(I6) $(\{X_{b,a}\} \cdot \Sigma^*, Y'_b\overline{d} \to Y_d, \overline{\Sigma}^*)$,

(I7) $(\{X_{a,b}Y_b; a,b \in \{0,1\}\} \cup \{X'_{a,b}Y'_b, a,b \in \{0,1\}\} \cup \{0\overline{0}, 1\overline{1}\}, \mathrm{Accept})$.

Words from $L_{\neg\mathsf{GCSL}}$ shorter than four symbols are accepted directly. For longer words, we iteratively check that corresponding symbols of w and $\overline{w}$ match. The left half of the current word will contain $X_{x,y}$ or $X'_{x,y}$ meaning that initially, this place contained symbols xy and we are checking that x matches the symbol on the right hand side. The right half contains Y_x or Y'_x meaning that this place initially contained symbol x. Moreover, if the word contains either $X_{x,y}$ and Y_z or $X'_{x,y}$ and Y'_z (for some symbols $x,y,z \in 0,1$), then we already know that the symbol x had a matching $\overline{x}$ in the initial word. On the other hand, if the word contains either $X_{x,y}$ and Y'_z or $X'_{x,y}$ and Y_z (for some symbols $x,y,z \in 0,1$), then we still have to check that there is a matching symbol for the symbol x (and that should be the symbol $\overline{z}$). At the beginning of the computation, there will be just $X_{x,y}$ and no Y_z or Y'_z meaning that the initial word contained xy in place of $X_{x,y}$ and that we are about to check that there is a matching symbol for the symbol x.

Now, it is easy to follow the meaning of individual meta-instructions. Let us show that $L_{\neg\mathsf{GCSL}} \subseteq \mathrm{L}(M_{\neg\mathsf{GCSL}})$. Let us have a word from $L_{\neg\mathsf{GCSL}}$. At the beginning, we apply (I1) to start the check of the first symbol of w and the first symbol of $\overline{w}$. This will be finished by applying (I2). Since now, there will always be one of the symbols $X_{x,y}$ and $X'_{x,y}$ and one of the symbols Y_z and Y'_z (for some $x,y,z \in \{0,1\}$)

denoting both the symbol present on the specified place in the initial input word and the current status of checking the symbol x. Thus, we iteratively apply (I3)–(I6) until we reach either a word $X_{x,y}Y_y$ or a word $X'_{x,y}Y'_y$ for some $x, y \in \{0, 1\}$. This word will then be directly accepted.

On the other hand, it is easy to see that $\mathrm{L}(M_{\neg\mathsf{GCSL}}) \subseteq L_{\neg\mathsf{GCSL}}$. Directly accepted words $0\overline{0}$ and $1\overline{1}$ obviously belong to $L_{\neg\mathsf{GCSL}}$. For longer words accepted by $M_{\neg\mathsf{GCSL}}$, there is always exactly one possible accepting computation and the meta-instructions obviously ensure that only words from $L_{\neg\mathsf{GCSL}}$ will be accepted.

The automaton $M_{\neg\mathsf{GCSL}}$ is obviously 0-reversible. It is not a single automaton, but according to Lemma 3.1, we know that it can be converted into such an automaton. Thus, the inclusion is proper and we have $\mathsf{GCSL} \subset \mathcal{L}(\mathsf{S}\text{-}0\mathsf{R}\text{-}\mathsf{RRWW})$.

The inclusion $\mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRWW}) \subseteq \mathsf{CSL}$ follows from the fact that each restarting automaton can be simulated by a linear bounded automaton [14]. □

The kind of allowed rewriting meta-instructions significantly impacts the power of restarting automata as proven below.

Theorem 3.3. It holds that $\mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RR}) \subset \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW}) \subset \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRWW})$ for all $k \geq 0$.

Proof:
It is obvious that $\mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RR}) \subseteq \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW}) \subseteq \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRWW})$ holds as allowing more general rewriting meta-instructions can not decrease the power of the models. It remains to prove that the inclusions are proper. Let us prove $\mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW}) \setminus \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RR}) \neq \emptyset$. We will use the language $L = \{\oplus, \circ\circ\} \cdot \{0^n1^n; n \geq 0\} \cup \{\otimes, \circ\circ\} \cdot \{0^n1^m; m > 2n \geq 0\}$ that was shown not to be accepted by any RR-automaton in [8]. Below we prove that it can be accepted by a $\mathsf{S}\text{-}0\mathsf{R}\text{-}\mathsf{RRW}$-automaton $M = (\{\oplus, \otimes, \circ, 0, 1\}, \{\oplus, \otimes, \circ, 0, 1\}, I)$, where I contains the following meta-instructions:

(I1) $(\{\lambda\}, \circ\circ \to \oplus, \{0,1\}^*)$	(I3) $(\{\oplus\} \cdot \{0\}^*, 01 \to \lambda, \{1\}^*)$	(I5) $(\{\otimes 1\}, 1 \to \lambda, \{1\}^*)$
(I2) $(\{\lambda\}, \circ\circ \to \otimes, \{0,1\}^*)$	(I4) $(\{\otimes\} \cdot \{0\}^*, 011 \to \lambda, \{1\}^*)$	(I6) $(\{\oplus, \otimes 1\}, \mathrm{Accept})$

Consider $w \in \{\oplus, \circ\circ\} \cdot \{0^n1^n; n \geq 0\}$. If w starts with $\circ\circ$, this prefix is replaced with $\oplus$ (using (I1)) at first. Then subwords 01 are iteratively removed using (I3), until a word consisting of single $\oplus$ is obtained and this word is then directly accepted by (I6). The case of $w \in \{\otimes, \circ\circ\} \cdot \{0^n1^m; m > 2n \geq 0\}$ is similar. If there is a prefix $\circ\circ$, it is replaced with $\otimes$ using (I2). Then subwords 011 are removed by (I4) while preserving the required form of the word. As the number of 1's is more than twice the number of 0's, there must be some 1's left after all 0's have been removed. They will be removed by (I5) except the last one to obtain the word $\otimes 1$ that will be directly accepted by (I6).

On the other hand, let $w \notin L$ be the shortest word outside L accepted by M. It is surely not accepted directly. So it can be directly reduced to a word from L by a rewriting meta-instruction of M. Using (I1) or (I2) to replace $\circ\circ$ with either $\oplus$ or $\otimes$ obviously can not reduce w to any $w' \in L$. If (I3) was used to reduce w starting with $\oplus$, the removal of 01 in the middle of w preserves the difference of the number of 0's and the number of 1's and thus $w \in L$ as well. If (I4) was used to reduce w starting with $\otimes 0$, the removal of 011 in the middle of w again preserves the relation between the number of 0's and the number of 1's, so $w \in L$. The meta-instruction (I5) can only be applied onto a word from L, so we do not have to consider it now. So we conclude that $L = \mathrm{L}(M)$. Moreover, M is a $\mathsf{S}\text{-}0\mathsf{R}\text{-}\mathsf{RRW}$-automaton as can be checked directly according to the definition.

The relation $\mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW}) \neq \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRWW})$ follows easily from Theorem 3.2 and from the fact that to accept all context-free languages, auxiliary symbols are needed [8]. □

Auxiliary symbols are a very powerful tool. Even for the reversibility level equal to zero, i.e. in the case of S-0R-RRWW-automata, all GCSL languages can be accepted. However, we often have to work without auxiliary symbols, and not considering them would also help understand the raw power of the model. Therefore, we further consider this limited case. At first, we prove that S-kR-RRW-automata form an infinite hierarchy with respect to the degree of reversibility k. Thus, this parameter, k, can be used in the grammatical inference field to reach the required power of considered models, and also to lower the number of needed input samples.

In the following theorem, we consider a language that will help us separate classes of languages accepted by S-kR-RRW for different levels k. It is helpful to define it now. Later, we will utilize the fact that for each $k \geq 0$, the language $L_{\neg k} = \{0^\ell; \ell > k\}$ is $(k+1)$-reversible but not k-reversible [2]. The following language was designed having this fact in mind.

Definition 3.4. For all $k \geq 0$ we define the language $L_{\text{sep},k} = L_1 \cup L_2$, where $L_1 = \{0^r 1^s 0^t; r > k, s > 0, \text{ and } r = t\}$, and $L_2 = \{0^r 1^s 0^t; 0 \leq r \leq k, s > k, \text{ and } t > s\}$.

The following lemma will form one part of the theorem. It states that the language proposed above can be accepted by our model if we allow $(k+1)$-reversible languages in its internals.

Lemma 3.5. It holds that $L_{\text{sep},k} \in \mathcal{L}(\mathsf{S}\text{-}(k+1)\mathsf{R}\text{-}\mathsf{RR})$.

Proof:
We define $B = (\{0,1\}, \{0,1\}, I_B)$, where I_B contains the following meta-instructions:

(I1) $(\{0^i; i > k\}, 1 \to \lambda, \{1\} \cdot \{0,1\}^*)$ (note the language $L_{\neg k}$ used as the left context),

(I2) $(\{0^i; i > k\}, 010 \to 1, \{0\}^*)$ (note the language $L_{\neg k}$ again),

(I3) $(\{0^m 1^i; 0 \leq m \leq k, i > k\}, 10 \to \lambda, \{0\}^*)$,

(I4) $(\{0^m 1^{k+1} 0^{k+2}; 0 \leq m \leq k\}, 0 \to \lambda, \{0\}^*)$, and

(I5) $(\{0^{k+1} 1 0^{k+1}\} \cup \{0^m 1^{k+1} 0^{k+2}; 0 \leq m \leq k\}, \text{Accept})$.

Let us prove that $\mathrm{L}(B) = L$. Firstly, we will prove $L \subseteq \mathrm{L}(B)$. In Definition 3.4 the language $L_{\text{sep},k}$ consists of two sublanguages: L_1 and L_2. We will consider words from both of them individually.

Let $0^r 1^s 0^r \in L_1$ for some $r > k$ and $s > 0$. Then:

- If $s > 1$, then this number will be lowered using (I1) and a shorter word from L will be obtained.
- If $s = 1$ and $r > k+1$, then the number of 0's will be lowered on both the left and the right end of the word using (I2) and a shorter word from L is thus obtained.
- If $s = 1$ and $r = k+1$, then the current word will be accepted using (I5).

Let $0^r 1^s 0^t \in L_2$ for some $0 \leq r \leq k, s > k,$ and $s < t$. Then:

- If $s > k+1$, the word is shortened by (I3) and the result still belongs to L.

- If $s = k + 1$ and $t > k + 2$, the number of trailing 0's is lowered by (I4) and the obtained word is still in L.
- If $s = k + 1$ and $t = k + 2$, the current word is accepted by (I5).

Thus, each word from L is accepted by B.

Now we will show that $\mathrm{L}(B) \subseteq L$. Obviously, the words accepted by (I5) belong to L. It will suffice to prove that no word out of L can be directly reduced to a word from L. We will revert the way meta-instructions work, and rather than u can be reduced to v we will say v can be expanded to u. So our goal is to show that no word from L can be expanded to a word out of L. We will consider particular rewriting meta-instructions:

- In the case of (I1), we have the word 0^i1u for some $i > k$ and $u \in \{0, 1\}^*$. Here (I1) inserts the new symbol 1 right after the prefix 0^i and thus the new word is also from L.
- In the case of (I2), we have the word 0^i10^j for $i > k$ and $j \geq 0$. Because (I2) both prepends and appends the symbol 1 with the symbol 0, the resulting word belongs to L.
- In the case of (I3), we have the word $0^m1^i0^j$ for some $m \leq k, i > k$, and $j \geq 0$. Here (I3) inserts 10 right after the end of the subword 1^i. Therefore, the word belongs to L as well.
- In the case of (I4), we have the word $0^m1^{k+1}0^{k+2}0^j$ for some $m \leq k$ and $j \geq 0$. Because (I4) inserts another zero into the trailing sequence of 0's, the new word is from L.

In total, we start with a word from L (accepted using (I5)) and only a stepwise expansion to another words from L is possible. So we know $\mathrm{L}(B) \subseteq L$ and thus B accepts L.

It remains to prove that B is a S-$(k + 1)$R-RR-automaton. Obviously, B is a single RR-automaton. We need to prove that all languages used in meta-instructions are $(k + 1)$-reversible:

- $\{0^i; i > k\}$ is $(k + 1)$-reversible according to [2] (see Fig. 1 for a corresponding FSA).
- $\{1\} \cdot \{0, 1\}^*$ is 1-reversible (see Fig. 2 for a corresponding 1-reversible FSA) and thus also $(k+1)$-reversible [2].
- $\{0\}^*$ is 0-reversible (the one-state FSA accepting this language is obviously 0-reversible) and thus also $(k + 1)$-reversible.
- $\{0^m1^i; 0 \leq m \leq k, i > k\}$ is $(k + 1)$-reversible (see Fig. 3).
- $\{0^m1^{k+1}0^{k+2}; 0 \leq m \leq k\}$ is $(k + 1)$-reversible (see Fig. 4).
- $\{0^{k+1}10^{k+1}\} \cup \{0^m1^{k+1}0^{k+2}; 0 \leq m \leq k\}$ is just an extension of the previous language and it remains $(k + 1)$-reversible as shown in Fig. 5.

□

The above presented lemma will form one part of our separation theorem. Next, it is necessary to prove that no S-kR-RRW-automaton can accept the language $L_{\mathrm{sep},k}$. This proof will be significantly more complex. Therefore, we will start with several auxiliary lemmas.

Firstly, we will study how a S-kR-RRWW-automaton accepting $L_{\mathrm{sep},k}$ would work.

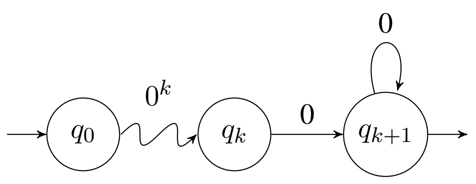

Figure 1. A $(k+1)$-reversible FSA accepting $\{0^i; i > k\}$.

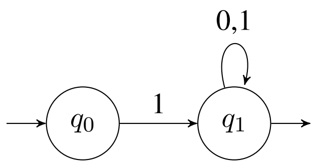

Figure 2. A 1-reversible FSA accepting $\{1\} \cdot \{0,1\}^*$.

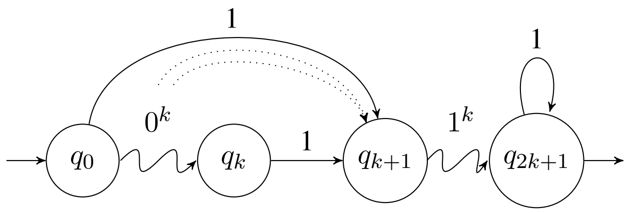

Figure 3. A $(k+1)$-reversible FSA accepting $\{0^m 1^i; 0 \leq m \leq k, i > k\}$. Note the dotted arrows denoting 1-transitions from all states present on the path from q_0 to q_k.

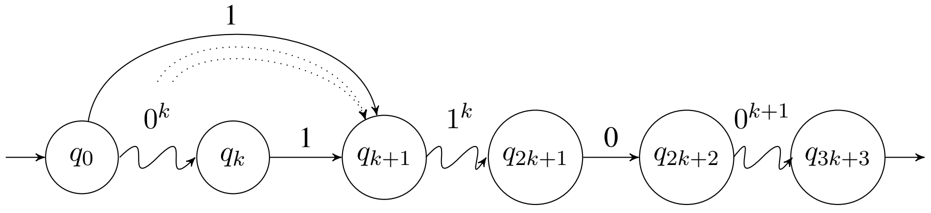

Figure 4. A $(k+1)$-reversible FSA accepting $\{0^m 1^{k+1} 0^{k+2}; 0 \leq m \leq k\}$. Note the dotted arrows denoting 1-transitions from all states present on the path from q_0 to q_k.

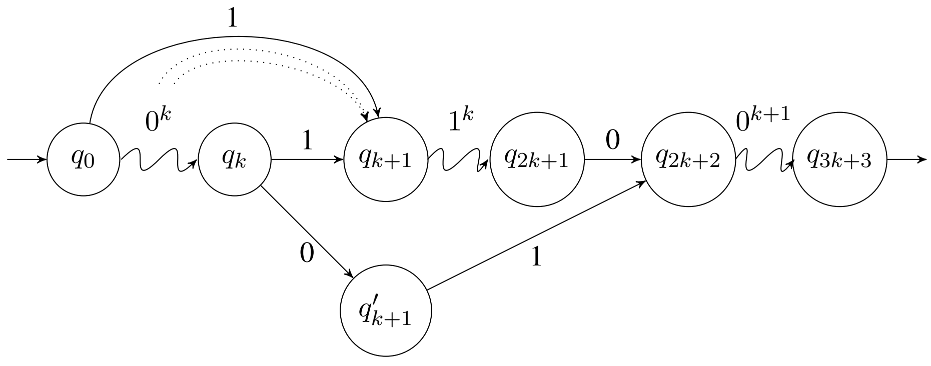

Figure 5. A $(k+1)$-reversible FSA accepting $\{0^{k+1}10^{k+1}\} \cup \{0^m1^{k+1}0^{k+2}; 0 \leq m \leq k\}$. Note the dotted arrows denoting 1-transitions from all states present on the path from q_0 to q_k.

Lemma 3.6. Let A be a S-$k\mathsf{R}$-RRW-automaton accepting $L_{\text{sep},k}$. Then there is n_0 such that for all $n > n_0$, the first step in each accepting computation of A on word $w_n = 0^n1^n0^n \in L$ consists in shortening the segment of 1's.

Proof:
By using the standard tools from formal languages theory it can be easily shown that only finitely many words w_n are directly accepted by A. Let k' be the size of the window of A. Suppose we have an accepting computation on w_n such that it consists of at least one reduction and $n > k + k'$. Thus, it holds $w_n \vdash_A w'$ for some $w' \in \{0,1\}^*$. If $w' \in L_2$, then A surely shortened the initial segment of 0's. However, at most k' 0's can be removed at once by A. Therefore, A has to perform a rewriting using a replacement word starting with 1 near the left end of w, but this obviously yields a word $w' \notin L_2$. The only other way is that $w' \in L_1$. As $n > k + k'$, it is not possible to rewrite in both segments of 0's. If exactly one segment of 0's is affected by the rewriting, then due to the fact that A is a single automaton, we obtain that the segment of 0's is shortened. This would lead to $w' \notin L_1$. Therefore, no segment of 0's can be rewritten in the first step. So the rewritten word consists of 1's only. Again, as we have a single automaton, it is not possible for the replacement word to start or to end with the symbol 1. When the symbol 0 is present in the replacement word, then we obtain $w' \notin L_1$. Finally, we see that some 1's were replaced with λ. $\square$

Let us now focus on left constraint language of some meta-instruction. Later, we will be interested in the set of all maximal prefixes consisting of 0's only of all words from that language. The following lemma characterizes this set. The proof is obvious and thus omitted.

Lemma 3.7. Let $L \subseteq \{0,1\}^*$. It holds that $\{0^i;\text{ there is } j \geq 0 \text{ such that } 0^i1^j \in L\} = \mathrm{RQ}(L, \{1\}^*) \cap \{0\}^*$.

It is important to note here that the transformation presented in the above lemma does not preserve k-reversibility as proven below.

Lemma 3.8. There is a 1-reversible language L such that $\mathrm{RQ}(L, \{1\}^*) \cap \{0\}^*$ is not a reversible language.

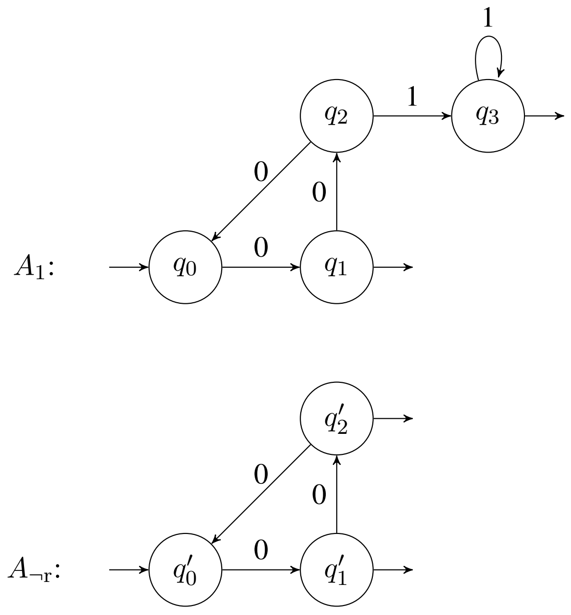

Figure 6. An example of a 1-reversible FSA A_1 accepting language $\{0^{3m+1}; m \geq 0\} \cup \{0^{3m+2}1^n; m \geq 0, n > 0\}$ and a FSA $A_{\neg\mathrm{r}}$ accepting the language $\mathrm{RQ}(\mathrm{L}(A_1), \{1^i; i \geq 0\}) \cap \{0^i; i \geq 0\} = \{0^{3m+1}; m \geq 0\} \cup \{0^{3m+2}; m \geq 0\}$ that is not a reversible language.

Proof:
On Fig. 6 you can find a 1-reversible FSA A_1 accepting the language $\{0^{3m+1}; m \geq 0\} \cup \{0^{3m+2}1^n; m \geq 0, n > 0\}$. The language $\mathrm{RQ}(\mathrm{L}(A_1), \{1\}^*) \cap \{0\}^* = \{0^{3m+1}; m \geq 0\} \cup \{0^{3m+2}; m \geq 0\}$, that is accepted by $A_{\neg\mathrm{r}}$, is not reversible at all. To see this, consider that the two accepting states q'_1 and q'_2 always share a common k-predecessor of any given length k.

□

We will be interested in FSA's accepting subsets of the language $\{0^i1^j; i, j \geq 0\}$. Such words consist of two parts, 0^i and 1^j. It will be very helpful to consider automata such that all their accepting computations meet in a common state right between those two parts. We will call it a 1-bottleneck property.

Definition 3.9. A FSA $A = (Q, \Sigma, \delta, \{q_0\}, F)$ has the *1-bottleneck property*, if there exists $q \in Q$ such that for all $i, j \geq 0$ such that $0^i1^j \in \mathrm{L}(A)$ it holds that $\delta^*(q_0, 0^i) = q$. The state q is called the *bottleneck state*.

In our proof, it will be very useful to represent a given regular language L by a FSA having the 1-bottleneck property. In general, this is not possible. However, for our needs, it will suffice to find a finite number of FSA's $A_1, \ldots, A_n$ for some $n > 0$ having 1-bottleneck property such that $L = \bigcup_{i \in \{1,\ldots,n\}} \mathrm{L}(A_i)$.

Lemma 3.10. For each regular language $L \subseteq \{0^i1^j; i, j \geq 0\}$ there is a set of FSA's $\mathcal{D}$ such that:

- Each $D \in \mathcal{D}$ has the 1-bottleneck property.
- It holds that $L = \bigcup_{D \in \mathcal{D}} \mathrm{L}(D)$.

Moreover, if L is k-reversible for some $k \geq 0$, then all FSA's from $\mathcal{D}$ are k-reversible.

Proof:
Let $A = (Q, \Sigma, \delta, \{q_0\}, F)$ be a minimal deterministic FSA accepting L. W.l.o.g. we suppose that A contains no unnecessary transitions.

For each $q \in Q$ such that there is a 1-transition leaving q and there is $i \geq 0$ such that $\delta^*(q_0, 0^i) = q$, we put $A_q = (Q, \Sigma, \delta_q, \{q_0\}, F_q)$ into $\mathcal{D}$, where δ_q is defined as follows:

- $\delta_q(p, 0) = \delta(p, 0)$ for $p \in Q$ if there is $i > 0$ such that $\delta^*(p, 0^i) = q$,
- $\delta_q(p, 1) = \delta(p, 1)$ for $p \in Q$ if there is $i \geq 0$ such that $\delta^*(q, 1^i) = p$, and
- δ is undefined otherwise,

and $F_q = F \cap \{\delta^*(q, 1^i); i \geq 0\}$. Obviously, if $0^i 1^j \in L$ for some $i \geq 0$ and $j > 0$, then $A_{\delta^*(q_0, 0^i)}$ accepts this word as we preserved all the needed 0-transitions to reach the state $q = \delta^*(q_0, 0^i)$ and also all 1-transitions reachable from that state, including the presence of the accepting state.

It remains to add automata accepting words from $L \cap \{0^i; i \geq 0\}$. Therefore, for all $q \in F$ we create the FSA $A'_q = (Q, \Sigma, \delta', \{q_0\}, \{q\})$ where δ'_q is obtained from δ by removing all 1-transitions, and we add it into $\mathcal{L}$. Obviously, if $0^i \in L$ for some $i \geq 0$, then $A'_{\delta^*(q_0, 0^i)}$ accepts this word.

We already know that $L \subseteq \bigcup_{D \in \mathcal{D}} \mathrm{L}(D)$. As we only removed some transitions and marked some states as non-accepting, we have that each created FSA accepts a subset of L and thus $L = \bigcup_{D \in \mathcal{D}} \mathrm{L}(D)$. It also means that the proposed transformation preserves the k-reversibility.

Also, all FSA's from $\mathcal{L}$ have the 1-bottleneck property where q is the bottleneck state. In the case of A'_q, this is trivial. Let us focus on A_q. For a contradiction, let us suppose that there are $i, j \geq 0$ such that $0^i 1^j \in \mathrm{L}(A_q)$ and $\delta_q^*(q_0, 0^i) \neq q$. Let $q' = \delta_q^*(q_0, 0^i)$. Then

- We know that there is some $i' \geq 0$ such that $\delta^*(q_0, 0^{i'}) = q$.
- As the 1-transition from q is defined, we know that there is $j' > 0$ such that $\delta^*(q, 1^{j'}) \in F$.
- If $j = 0$ then $q' \in F_q$. If $j > 0$ then the 1-transition from q' is defined. In both cases we know that there is $m > 0$ such that $\delta^*(q, 1^m) = q'$.
- If $q' \neq q_0$, then there is a 0-transition entering q' and thus we know that there is some $n > 0$ such that $\delta^*(q', 0^n) = q$. On the other hand, if $q' = q_0$, we put $n = i'$, which is greater than zero as $q \neq q'$, and again we have $\delta^*(q', 0^n) = q$.
- Overall, $\delta^*(q_0, 0^{i'} 1^m 0^n 1^{j'}) \in F$. This is a contradiction.

□

FSA's having the 1-bottleneck property also have a very useful property (we will denote this feature with EXCH) that will be used later.

Lemma 3.11. Let $A = (Q, \Sigma, \delta, \{q_0\}, F)$ be a FSA with the 1-bottleneck property. For all $i, i', j, j' \geq 0$ such that $0^i1^j, 0^{i'}1^{j'} \in \mathrm{L}(A)$, it holds $0^i1^{j'} \in \mathrm{L}(A)$.

Proof:
From $\delta^*(q_0, 0^i) = \delta^*(q_0, 0^{i'})$, it follows that $\delta^*(q_0, 0^i1^{j'}) = \delta^*(\delta^*(q_0, 0^i), 1^{j'}) = \delta^*(\delta^*(q_0, 0^{i'}), 1^{j'}) = \delta^*(q_0, 0^{i'}1^{j'}) \in F$. $\square$

Later, we will cover $\{0\}^*$ except of a finite number of words by a set of k-reversible languages. According to the following lemma, the word 0^k is necessarily covered as well.

Lemma 3.12. Let $\mathcal{L}$ be a finite set of k-reversible languages over $\{0\}$ such that for some $n_0 \geq 0$ it holds that $\{0^n; n \geq n_0\} \subseteq \bigcup_{L \in \mathcal{L}} L$. Then there is $L \in \mathcal{L}$ such that $0^k \in L$.

Proof:
Let us suppose that the opposite holds. W.l.o.g. we suppose that each language from $\mathcal{L}$ is infinite. At first, let A'_i be a FSA accepting one of the languages from $\mathcal{L}$.

Surely, $A'_i = (Q_{A'_i}, \{0\}, \delta_{A'_i}, \{q_{0,A'_i}\}, F_{A'_i})$ does not accept 0^k, by assumption. Moreover, A'_i contains a cycle (because it accepts an infinite language). Because of the k-reversibility, $\delta^*_{A'_i}(q_{0,A'_i}, 0^k)$ is a member of that cycle. Let the length of the cycle be $r_{A'_i}$ symbols. Then even $0^{k+r_{A'_i}s}$ is not accepted for all $s > 0$. For brevity, we denote individual values of $r_{A'_i}$ as $r_1, \ldots, r_z$ (let $z = |\mathcal{L}|$). Thus, words $0^{k+r_1s_1}, \ldots, 0^{k+r_zs_z}$ for all $s_1, \ldots, s_z \geq 0$ are not accepted by individual FSA's.

We would like to show that there are infinitely many words not covered by the union of all languages from $\mathcal{L}$. Let $s_i = h \cdot \Pi_{j \neq i} r_j$ for $h > 0$. Then for all i, j such that $0 < i, j \leq z$, it holds $k + r_is_i = k + r_js_j$ and thus $0^{k+r_1s_1}$ is not a member of any language from $\mathcal{L}$. Therefore, we obtained infinitely many words not covered by $\mathcal{L}$ (it suffices to consider infinitely many different values of h). This is a contradiction. Therefore, 0^k is a member of at least one language from $\mathcal{L}$.

$\square$

Theorem 3.13. For all reversibility levels $k \geq 0$, it holds that $\mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW}) \subset \mathcal{L}(\mathsf{S}\text{-}(k+1)\mathsf{R}\text{-}\mathsf{RRW})$, and $\mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RR}) \subset \mathcal{L}(\mathsf{S}\text{-}(k+1)\mathsf{R}\text{-}\mathsf{RR})$.

Proof:
We will show that $L_{\mathrm{sep},k} \in \mathcal{L}(\mathsf{S}\text{-}(k+1)\mathsf{R}\text{-}\mathsf{RR}) \setminus \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW})$. The relation $L_{\mathrm{sep},k} \in \mathcal{L}(\mathsf{S}\text{-}(k+1)\mathsf{R}\text{-}\mathsf{RR})$ holds thanks to Lemma 3.5. It remains to prove that $L \notin \mathcal{L}(\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW})$.

For a contradiction, let us suppose there is a $\mathsf{S}\text{-}k\mathsf{R}\text{-}\mathsf{RRW}$-automaton $A = (\{0,1\}, \{0,1\}, I)$ accepting L. We will prove that A reduces $0^k1^n0^n \notin L$ (for some $n > k$) to $0^k1^{n'}0^n \in L$ (where $k < n' < n$) and thus $\mathrm{L}(A) \neq L$. In this proof, we will focus on how A works on words $w_n = 0^n1^n0^n \in L$ for $n > k$.

Let $n_{\mathrm{reduce}} = k + k' + n_0$, where k' is the size of the window of A and n_0 is the value used in Lemma 3.6 applied onto A. Thus, for all $n > n_{\mathrm{reduce}}$, the word w_n can be surely reduced by A by shortening the segment of 1's (by using the empty replacement word).

We will use this fact to find a rewriting meta-instruction that reduces a word with the prefix 0^k1. Let I' be the set of all meta-instructions of A that reduce some w_n ($n > n_{\mathrm{reduce}}$) to a word from L. We will collect all words 0^n such that at least one of the considered meta-instructions from I' reduces a word having a prefix 0^n1. If LC is the union of the left contexts of the meta-instructions I', then, according to Lemma 3.7, the language we are looking for is $\mathrm{RQ}(\mathrm{LC}, \{1\}^*) \cap \{0\}^*$.

To show that 0^k belongs to that language, it would be helpful to know that it can be covered with finitely many k-reversible languages. Unfortunately, this language need not be k-reversible, as stated in Lemma 3.8, so this property is not obvious. Therefore, we slightly modify the rewriting meta-instructions from I'. The automaton will not remain single but we do not need this property for the rest of this proof.

We will transform each meta-instruction $(L_\ell, w \to w', L_\mathrm{r}) \in I'$ as follows. We know that $w \in \{1\}^*$. Therefore, each word from L reduced by this meta-instruction can be split into three parts $w_\ell, w,$ and w_r such that $w_n = w_\ell w w_\mathrm{r}$ where $w_\ell \in L_\ell \cap \{0^i1^j; i, j \geq 0\}$, and $w_\mathrm{r} \in L_\mathrm{r}$. We will cover $L_\ell \cap \{0^i1^j; i, j \geq 0\}$ by a finite set of k-reversible languages accepted by FSA's having the 1-bottleneck property (using Lemma 3.10). Let $\mathcal{D}$ be the set of FSA's obtained from Lemma 3.10. We will replace the original meta-instruction with meta-instructions from $\{(\mathrm{L}(D), w \to w', L_\mathrm{r}); D \in \mathcal{D}\}$. Obviously, all words from L reduced by the original meta-instruction $(L_\ell, w \to w', L_\mathrm{r})$ can be reduced by some of the newly created ones in the same way. On the other hand, our new meta-instructions obviously do not perform any reduction not performed by the original meta-instructions. It is important to note that A obviously remains k-reversible, as we only replaced original left constraints with another k-reversible ones.

By I'' we denote the subset of I' containing only rewriting meta-instructions that reduce at least one word w_n for $n > n_\mathrm{reduce}$.

Let $L_{\ell,I''}$ be the set of left constraints of meta-instructions from I'' after the just performed transformation. We will now transform $L_{\ell,I''}$ into $L_{\ell,I'',0}$ in the following way. For each $L_i \in L_{\ell,I''}$, we perform the following steps to obtain $\mathrm{RQ}(L_i, \{1\}^*) \cap \{0\}^*$ (let A_i be a k-reversible FSA accepting L_i). We mark each state of A_i that has an outgoing 1-transition as accepting (the states that were accepting before remain accepting as well), we remove all 1-transitions and then also the non-reachable states, and we denote the resulting FSA as A'_i and we put $\mathrm{L}(A'_i)$ into $L_{\ell,I'',0}$.

Obviously, for each w_n $(n > n_\mathrm{reduce})$, there is a member of $L_{\ell,I'',0}$ containing 0^n. Note that for each $0^r \in \mathrm{L}(A'_i)$, there is some $s \geq 0$ such that $0^r1^s \in \mathrm{L}(A_i) = L_i$. It is very important to note that all members of $L_{\ell,I'',0}$ are k-reversible (remember that the automaton A'_i contains exactly one accepting state and the transitions were present in the k-reversible FSA A_i as well). Thus, $L_{\ell,I'',0}$ covers the language $\{0^n; n > n_\mathrm{reduce}\}$ with finitely many k-reversible languages because each w_n with $n > n_\mathrm{reduce}$ can be reduced by a meta-instruction from I''.

Lemma 3.12 yields that necessarily $L_{\ell,I'',0}$ covers also the word 0^k.

Now we conclude this proof by proving that the automaton A performs a reduction that violates the error preserving property.

Let $w_n = 0^n1^n0^n \in L$ for some $n > n_\mathrm{reduce}$ that can be reduced using the rewriting meta-instruction with left context containing either 0^k or a word with prefix 0^k1. Let it be $(L_{\ell,\mathrm{p}}, 1^i \to \lambda, L_{\mathrm{r,p}})$ for some $i > 0$. So it reduces $0^n1^{i_\ell}1^i1^{i_\mathrm{r}}0^n \in L$ (where $i_\ell + i + i_\mathrm{r} = n$) into $0^n1^{i_\ell + i_\mathrm{r}}0^n$. Therefore, $0^n1^{i_\ell} \in L_{\ell,\mathrm{p}}$ and thus also $0^k1^{i_\ell} \in L_{\ell,\mathrm{p}}$ (thanks to the feature EXCH). Finally, we have that $0^k1^{i_\ell}1^i1^{i_\mathrm{r}}0^n = 0^k1^n0^n \notin L$ can be reduced to $0^k1^{i_\ell}1^{i_\mathrm{r}}0^n \in L$ (note that $i_\ell + i_\mathrm{r} = n - i > n_\mathrm{reduce} - i \geq n_\mathrm{reduce} - k' = k + n_0 > k$ and thus $i_\ell + i_\mathrm{r} > k$).

This is a contradiction as obviously $\mathrm{L}(A) \neq L$. Thus, L can not be accepted by any S-kR-RRW-automaton. $\square$

It is an open question whether an analogous theorem holds in the case of automata using auxiliary symbols, i.e. in the case of S-kR-RRWW-automata.

SLT-R-automata are very similar to S-kR-RRWW-automata. As both can be inferred from positive samples, it is important to know the relation of classes of languages they can accept. Our model utilizes a richer class of languages in meta-instructions, but we allow only *single* restarting automata. The relation between these two models is therefore not obvious. In what follows, we will compare these two models in the case they do not use auxiliary symbols.

Theorem 3.14. Let $L = L_1 \cup L_2$, where $L_1 = \{0^{2r}1^s0^{2r}1^t; r, s, t > 0\}$, and $L_2 = \{0^{2r+1}1^s0^t1^{s'}; r, s, s', t > 0$ and $s \neq s'\}$. There is no SLT-R-automaton without auxiliary symbols that accepts L, but there is a S-kR-RR-automaton accepting L.

Proof:
At first, we will show that L can not be accepted by any SLT-R-automaton. For a contradiction, let us suppose that there is a k'-SLT-R-automaton M such that $\mathrm{L}(M) = L$. Let us consider the word $w = 0^r1^s0^r1^s \in L_1$ for $r = 4k'$ and $s = 3k'$. Let us analyze the accepting computation on this word (we will often use a very simple idea — let L' be a strictly k'-testable language over an alphabet Γ, let $u \in L'$ and let $v \in \Gamma^*$; then if $\mathrm{P}_{k'}(u) = \mathrm{P}_{k'}(v), \mathrm{S}_{k'}(u) = \mathrm{S}_{k'}(v)$, and $\mathrm{I}_{k'}(u) = \mathrm{I}_{k'}(v)$, then $v \in L'$). It can start in any of the following ways:

- If w is directly accepted by M, then $0w$ is directly accepted as well and this is a contradiction.
- If w is directly reduced to some $w' \in L$, we distinguish two cases:
  - To reduce w into a shorter word $w' \in L_1$, the only possible way is to lower the number of 1's in exactly one of the segments of 1's. If M lowers the number of 1's in one of the segments of 1's in w, then the same can be performed on the word $w'' = 0w \notin L$. However, this would reduce $w'' \notin L$ into $0w' \in L$. This is a contradiction.
  - To reduce w into a shorter word $w' \in L_2$, we need to increase or decrease the number of 0's in the initial segment of 0's by an odd number and also to change the number of 1's in the first segment of 1's. If M reduces w this way, then surely the rewritten subword has the form 0^i1^j where $i, j \geq 0$. Let us consider the word $w'' = 00w \notin L$. Again, this can be reduced analogically into $00w' \in L$. This is a contradiction.

Overall, the word w can not be accepted by M. Thus, there is no SLT-R-automaton without auxiliary symbols accepting L. It remains to prove that L can be accepted by some S-kR-RRW-automaton. Let $B = (\{0,1\}, \{0,1\}, I)$ where I contains the following meta-instructions:

(I1) $(\{0^{2r}; r \geq 0\}, 1 \to \lambda, \{1\} \cdot \{0,1\}^*)$,

(I2) $(\{0^{2r}; r \geq 0\}, 00100 \to 1, \{0\} \cdot \{0,1\}^*)$,

(I3) $(\{0^{2r+1}1^s; r \geq 0, s > 0\}, 0 \to \lambda, \{0\} \cdot \{0,1\}^*)$,

(I4) $(\{0^{2r+1}1^s; r \geq 0, s > 0\}, 101 \to 0, \{1\} \cdot \{0,1\}^*)$, and

(I5) $(\{001001^t; t > 0\} \cup \{0^{2r+1}101^t; r > 0, t \geq 2\} \cup \{0^{2r+1}1^s01; r > 0, s \geq 2\}, \mathrm{Accept})$.

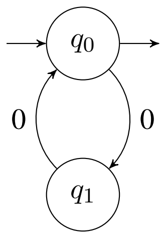

Figure 7. A 0-reversible FSA accepting $\{0^{2k}; k \geq 0\}$.

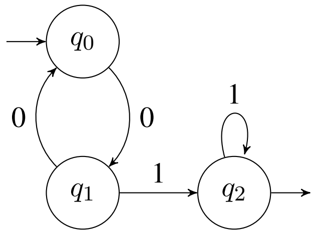

Figure 8. A 1-reversible FSA accepting $\{0^{2k+1}1^{\ell}; k \geq 0, \ell > 0\}$.

We will prove that $\mathrm{L}(B) = L$. We will start with the inclusion $L \subseteq \mathrm{L}(B)$. If $w \in L_1$, (I1) is used to reduce w into $0^{2r}10^{2r}1^t$ for some $r, t > 0$, this word is then reduced to 001001^t by (I2) and then (I5) is used to accept this word. If $w \in L_2$, it is reduced into $0^{2r+1}1^s01^{s'}$ for some $r, s, s' > 0$ using (I3), then (I4) is used to obtain the word $0^{2r+1}101^{s''}$ or $0^{2r+1}1^{s''}01$ for some $s'' > 1$, and the obtained word is then accepted by (I5).

As the words accepted by (I5) obviously belong to L and it can be easily seen that no rewriting meta-instruction reduces a word outside L into a word from L, we have that $\mathrm{L}(B) \subseteq L$ and, in total, $\mathrm{L}(B) = L$. Obviously, B is a single RR-automaton.

It remains to prove that B is k-reversible for some k.

- (I1) is 1-reversible (see Fig. 7 and the proof of Theorem 3.13).
- (I2) is 1-reversible (similarly to (I1)).
- (I3) is 1-reversible (see Fig. 8 and the proof of Theorem 3.13).
- (I4) is 1-reversible (similarly to (I3)).
- (I5) is 3-reversible (see Fig. 9).

Finally, B is a S-3R-RR-automaton accepting L.

□

On the other hand, the restriction to *single* restarting automata prevents accepting some languages despite using a richer class of languages (even regular languages!) in meta-instructions.

Theorem 3.15. Let $L = L_1 \cup L_2 \cup L_3 \cup L_4$ where

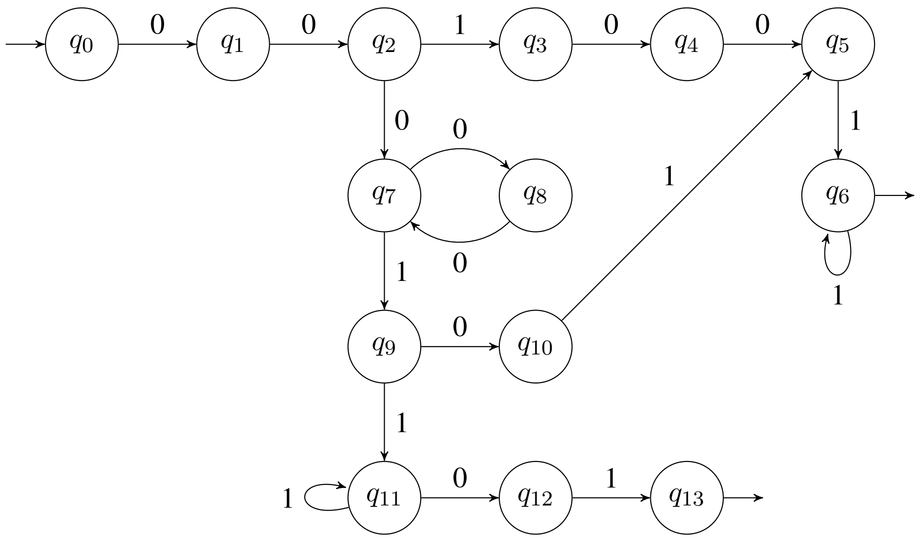

Figure 9. A 3-reversible FSA accepting the language $\{001001^t;\, t > 0\} \cup \{0^{2r+1}101^t;\, r > 0, t \geq 2\} \cup \{0^{2r+1}1^s01; r > 0, s \geq 2\}$.

- $L_1 = \{a0^nc0^nc0^na; a \in \{\oplus, \otimes\}, c \in \{\circ, \bullet\}, n \geq 0\}$,
- $L_2 = \{a0^nc0^{2n}c0^nb; a, b \in \{\oplus, \otimes\}, a \neq b, c \in \{\circ, \bullet\}, n \geq 0\}$,
- $L_3 = \{a0^{n-1}c0^{n-1}d0^na; a \in \{\oplus, \otimes\}, c, d \in \{\circ, \bullet\}, c \neq d, n > 0\}$,
- $L_4 = \{a0^{n-1}c0^{2n-2}d0^nb; a, b \in \{\oplus, \otimes\}, a \neq b, c, d \in \{\circ, \bullet\}, c \neq d, n > 0\}$.

There is no single RRW-automaton (and thus no S-kR-RRW-automaton) accepting L, but there is a SLT-R-automaton using no auxiliary symbols that accepts L.

Proof:
We start with proving that no single RRW-automaton accepts L. For a contradiction, let M be a single RRW-automaton accepting L. Let us analyze processing of particular words from L_2 — let us have a word $w = a0^n \circ 0^{2n} \circ 0^nb$, where $a, b \in \{\oplus, \otimes\}, a \neq b, n \geq 0$. For sufficiently large n, M does not accept this word directly (as can be easily proven using standard tools of the formal languages theory). Thus, a reduction is needed. We suppose n greater than the size of the window of M. Below we analyze all possible ways how to reduce w:

- If the rewritten subword contains no delimiter from $\{\oplus, \otimes, \circ\}$, then the number of symbols present in exactly one of the segments consisting of 0's will be lowered (note that introducing a new delimiter would make the resulting word belong to the complement of L, so we can safely omit this case). Thus, the resulting word falls out of L.
- If the rewritten subword contains $\oplus$ or $\otimes$, then, as we have sufficiently long w, we can suppose it does not contain both of them at once. Also, neither one of the symbols $\circ$ can be changed. The

result of this reduction should fall into L_1 (the outer delimiters must be the same and the inner pair of the delimiters contains the original ones, i.e. $\circ$'s). However, as either the left segment or the right segment of 0's was shortened, the word does not belong to L_1.

- If the rewritten subword contains $\circ$, then we also need either $\circ$ or $\bullet$ in the replacement word to remain in L. As the rewritten subword must be longer than the replacement word and we have a single automaton, we know that at least one symbol 0 is removed. Moreover, the delimiters $\oplus$ and $\otimes$ are not changed by this reduction due to the limited size of the window of M. Therefore, the word obtained by this reduction should belong to $L_2 \cup L_4$. Obviously, it is not possible to stay in L_2 after removing some 0's around either of the delimiters $\circ$. So there remains only one way, i.e. reducing into a word from L_4. Necessarily, the rewritten subword equals to $0^i \circ 0^j$ for some $i, j > 0$. Since we have a single automaton, we know that the replacement word equals to $\bullet$. It follows that $i = 1$ and $j = 2$.

Thus, M contains a rewriting meta-instruction *rw* $= (L_\ell, 0 \circ 00 \to \bullet, L_\mathrm{r})$. We would like to proceed as follows. Let $w_{\overline{L}} = \oplus 0^{2n} 0 \circ 00 0^{2n} \circ 0^{2n+1} \oplus \notin L$. The idea is to show that this word can be reduced by *rw* to $w_L = \oplus 0^{2n} \bullet 0^{2n} \circ 0^{2n+1} \oplus \in L$. This will be a contradiction.

It is easy to see that $\oplus 0^{2n} \in L_\ell$ as the reductions $(\oplus 0^m, 0 \circ 00 \to \bullet, 0^{2m} \circ 0^{m+1} \otimes)$ must be performed (for sufficiently large m) by *rw* as shown above. It remains to prove that $0^{2n} \circ 0^{2n+1} \oplus \in L_\mathrm{r}$. At first, it may not be so obvious why this should hold. For a contradiction, let us suppose the opposite holds. Thanks to the closure properties of regular languages, we know that the complement of the language L_r, i.e. the language L_r^c, is regular as well. Let us suppose that for all $m > m_0$ (for some constant m_0), it holds $0^{2m} \circ 0^{2m+1} \oplus \in L_\mathrm{r}^\mathrm{c}$. It is clear that a FSA A accepting L_r^c steps through some cycles while reading the segments of 0's in such words. We can pass such cycles several times and obtain new words belonging to L_r^c. Let us start with the word $0^{2n} \circ 0^{2n+1} \oplus \in L_\mathrm{r}^\mathrm{c}$ for sufficiently large n. We obtain words $0^{2n+k_1\ell_1} \circ 0^{2n+1+k_2\ell_2} \oplus \in L_\mathrm{r}^\mathrm{c}$ where ℓ_1, ℓ_2 are the lengths of corresponding cycles in A and $k_1, k_2 \geq 0$ are variables we can choose arbitrarily. We would like to find k_1 and k_2 such that $2(2n + k_2\ell_2) = 2n + k_1\ell_1$ (note that the value ℓ_2 may depend on n). Note that we can also choose $n > m_0$. Let $n = m_0 \cdot P! \cdot C$ where P is the number of states of A and $C > 0$ is an arbitrary number that assures that n is large enough to satisfy all restrictions on n introduced earlier in this proof. Now the values of ℓ_1, ℓ_2 are fixed. It is easy to obtain that $k_1 = \frac{2n}{\ell_1} + \frac{2k_2}{\ell_1} \cdot \ell_2$ and thus it suffices to put $k_2 = \ell_1$ to obtain k_1 as an integer. Thus, we have that $0^{2n+k_1\ell_1} \circ 0^{2n+1+k_2\ell_2} \oplus \in L_\mathrm{r}^\mathrm{c}$, then $0^{2n+2n+2\ell_1\ell_2} \circ 0^{2n+1+\ell_1\ell_2} \oplus \in L_\mathrm{r}^\mathrm{c}$, then $0^{2(2n+\ell_1\ell_2)} \circ 0^{(2n+\ell_1\ell_2)+1} \oplus \in L_\mathrm{r}^\mathrm{c}$, and finally $0^{2s} \circ 0^{s+1} \oplus \in L_\mathrm{r}^\mathrm{c}$ for $s = 2n + \ell_1\ell_2$. Thus, the word $\otimes 0^{2n+\ell_1\ell_2} 0 \circ 00 0^{2(2n+\ell_1\ell_2)} \circ 0^{(2n+\ell_1\ell_2)+1} \oplus = \otimes 0^{s+1} \circ 0^{2(s+1)} \circ 0^{s+1} \oplus \in L_2 \subseteq L$ will not be reduced (we consider s as above) — and as n is large enough, it will not be accepted at all.

Thus, we have a contradiction. Therefore, $0^{2n} \circ 0^{2n+1} \oplus \in L_\mathrm{r}$. Thus, M reduces the word $w_{\overline{L}} \notin L$ into the word $w_L \in L$. This is a contradiction. Therefore, $\mathrm{L}(M) \neq L$.

It remains to prove that there is a SLT-R-automaton M' not using auxiliary symbols and accepting L. Let $\Sigma = \{0, \oplus, \otimes, \circ, \bullet\}$ and $M' = (\Sigma, \Sigma, I)$ where I contains the following meta-instructions (let $\Sigma_\circ = \Sigma \setminus \{\bullet\}, \Sigma_\bullet = \Sigma \setminus \{\circ\}$):

(I1) $(\oplus 0^*, 0 \circ 0 \to \bullet, \Sigma_\circ^* \oplus)$	(I5) $(\oplus 0^*, 0 \circ 00 \to \bullet, \Sigma_\circ^* \otimes)$	(I9) $(\Sigma_\circ^*, \bullet 0 \to \circ, 0^*(\oplus + \otimes))$
(I2) $(\otimes 0^*, 0 \circ 0 \to \bullet, \Sigma_\circ^* \otimes)$	(I6) $(\otimes 0^*, 0 \circ 00 \to \bullet, \Sigma_\circ^* \oplus)$	(I10) $(\Sigma_\bullet^*, \circ 0 \to \bullet, 0^*(\oplus + \otimes))$
(I3) $(\oplus 0^*, 0 \bullet 0 \to \circ, \Sigma_\bullet^* \oplus)$	(I7) $(\oplus 0^*, 0 \bullet 00 \to \circ, \Sigma_\bullet^* \otimes)$	(I11) $(\{accb; a, b \in \{\oplus, \otimes\},$
(I4) $(\otimes 0^*, 0 \bullet 0 \to \circ, \Sigma_\bullet^* \otimes)$	(I8) $(\otimes 0^*, 0 \bullet 00 \to \circ, \Sigma_\bullet^* \oplus)$	$c \in \{\circ, \bullet\}\}, \mathrm{Accept})$

It is easy to see that M' is strictly 4-testable (note that every strictly k-testable language is also a strictly $k+1$ testable language for all $k > 0$ [19]):

- Languages $a0^*$ where $a \in \{\oplus, \otimes\}$ correspond to the triple $(\{a0\}, \{00\}, \{a0, 00\})$.
- Languages Σ_c^* where $c \in \{\circ, \bullet\}$ correspond to the triple $(\Sigma_c, \Sigma_c, \Sigma_c)$.
- Languages $\Sigma_c^* a$ where $c \in \{\circ, \bullet\}$ and $a \in \{\oplus, \otimes\}$ correspond to the triple $(\Sigma_c^2, \Sigma_c^2, \Sigma_c \cdot \{a\})$.
- Language $0^*(\oplus + \otimes)$ corresponds to the triple $(\{0\oplus, 0\otimes, 00\}, \{00\}, \{0\oplus, 0\otimes\})$.
- Language $\{accb; a, b \in \{\oplus, \otimes\}, c \in \{\circ, \bullet\}\}$ corresponds to the triple $(\{\oplus \circ \circ, \oplus \bullet \bullet, \otimes \circ \circ, \otimes \bullet \bullet\}, \emptyset, \{\circ \circ \oplus, \circ \circ \otimes, \bullet \bullet \oplus, \bullet \bullet \otimes\})$.
- All rewritten subwords are shorter than 5 symbols.

Let us prove that $L = \mathrm{L}(M')$. We will start with $L \subseteq \mathrm{L}(M')$. The words shorter than 5 symbols are accepted by the accepting meta-instruction and they will not be considered below. The words from L_1 are reduced by (I1)–(I4) to a word from L_3. The words from L_2 are reduced by (I5)–(I8) to a word from L_4. Finally, the words from $L_3 \cup L_4$ are reduced by (I9) and (I10) to a word from $L_1 \cup L_2$. Thus, $L \subseteq \mathrm{L}(M')$. On the other hand, it holds that $\mathrm{L}(M') \subseteq L$ as obviously no rewriting meta-instruction reduces a word outside L to a word from L and the accepting meta-instruction accepts exactly the listed words that all belong to L. This concludes our proof. □

Overall, we obtained that the classes of languages accepted by S-kR-RRW-automata and SLT-R-automata not using auxiliary symbols are incomparable. Therefore, both should be considered when learning languages from positive samples.

Next, we will show that there are even quite simple languages that require the use of auxiliary symbols to be accepted by a restarting automaton. We will need the following two auxiliary lemmas.

The first one states that each restarting automaton can be transformed into an equivalent one that accepts directly only short words.

Lemma 3.16. ([15])
Let $X \in \{\mathsf{RR}, \mathsf{RRW}, \mathsf{RRWW}\}$. For each X-automaton M, there is an X-automaton M' and $k > 0$ such that $\mathrm{L}(M) = \mathrm{L}(M')$ and no word longer than k symbols is accepted by M' directly (i.e. without a restart).

The following lemma is the well-known pumping lemma.

Lemma 3.17. Let $A = (Q, \Sigma, \delta, I, F)$ be a FSA. Then there is n such that for every word $w \in \mathrm{L}(A)$ such that $|w| \geq n$, there are $x, y, z \in \Sigma^*$ such that:

- $w = xyz$,
- $y \neq \lambda$,
- $|xy| \leq n$, and
- for all $k \geq 0$ it holds $xy^k z \in \mathrm{L}(A)$.

Now we can proceed with the theorem announced above.

Theorem 3.18. Let $L = \{ww' | w \in \{0,1\}^*, |w| = |w'|, w^{\mathrm{R}} \neq w'\} \cup \{w \in \{0,1\}^*; |w| \text{ is an odd number}\}$. It holds:

- $L \in \mathsf{EL}$, and
- $L \in \mathcal{L}(\mathsf{RRWW}) \setminus \mathcal{L}(\mathsf{RRW})$.

Proof:
Let us prove that L is an even linear language. We will show that L can be generated by the even linear grammar $G = (\{0,1\}, \{\mathrm{S}, \mathrm{T}\}, \mathrm{S}, P)$, where P contains the following rules:

(i) $\mathrm{S} \to a\mathrm{S}a$ for $a \in \{0,1\}$,

(ii) $\mathrm{S} \to a\mathrm{T}b$ for $a, b \in \{0,1\}, a \neq b$,

(iii) $\mathrm{S} \to a$ for $a \in \{0,1\}$,

(iv) $\mathrm{T} \to a\mathrm{T}b$ for $a, b \in \{0,1\}$,

(v) $\mathrm{T} \to \lambda$, and

(vi) $\mathrm{T} \to a$ for $a \in \{0,1\}$.

We will first prove that $L \subseteq \mathrm{L}(G)$. Let $w \in L$. Let x be the longest prefix of the left half of w (in the case of w having odd length, we do not consider the symbol in the middle to be part of either half) such that w has a suffix x^{R}. We start from the non-terminal S by using $|x|$-times the rule (i). This way we obtain $x\mathrm{S}x^{\mathrm{R}}$. Then, if only 1 symbol is missing to obtain w, we finish the derivation by the rule (iii). Otherwise, we use the rule (ii) to obtain $xa\mathrm{T}bx^{\mathrm{R}}$ for $a, b \in \{0,1\}$ such that $a \neq b$. Then the rule (iv) is used until at most 1 symbol is missing to obtain w. Then we finish the derivation by the rule (v) or (vi).

Now we will prove that $\mathrm{L}(G) \subseteq L$. There are three ways to finish a derivation of some word — using either the rule (iii), (v), or (vi). In the case of the rules (iii) and (vi), it is clear that the obtained word belongs to L because it has odd length. In the case of the rule (v), the resulting word belongs to L as well because in the derivation of that word, there must be a switch from the non-terminal symbol S to the non-terminal symbol T (i.e. the use of the rule (ii)) that introduced two different terminal symbols on the corresponding positions in the left half and in the right half of the obtained word.

Thus, L is an even linear language. It is therefore clear that $L \in \mathcal{L}(\mathsf{RRWW})$ because all context-free languages can be accepted by an RRWW-automaton [8].

Next, we will show that L can not be accepted by any RRW-automaton. For a contradiction, let us suppose that an RRW-automaton $M = (\Sigma, \Gamma, I)$ accepts L. According to Lemma 3.16, we can suppose that all words accepted by M directly (without a restart) are shorter than ℓ' symbols, for some constant $\ell' > 0$.

Let I_{RW} be the set of all rewriting meta-instructions of M and let $L_{\mathrm{rd}} = \bigcup_{(L_\ell, x \to y, L_{\mathrm{r}}) \in I_{\mathrm{RW}}} (L_\ell \cdot \{x\} \cdot L_{\mathrm{r}})$ be the set of all words reduced by M. This is obviously a regular language. Similarly, $L_{\mathrm{nr}} = \{0,1\}^* \setminus L_{\mathrm{rd}}$ is the regular language consisting of all words which can not be reduced by M.

Let $w_{n,k} = (0^{2k}1)^n 0^{2k} (10^{2k})^n$ for $n > 0$ and $k > 0$. Obviously, $w_{n,k} \notin L$ as $w_{n,k}$ has even length and it is of the form ww^{R} for $w = (0^{2k}1)^n 0^k$. In general, M need not to reduce long words not belonging to L. However, we will prove that the automaton M can reduce the words $w_{n,k}$ for all $n > \ell'$ and $k > k_0$, where k_0 is the constant denoted as n within the pumping lemma for regular languages (Lemma 3.17) applied to L_{nr}. For a contradiction, let us suppose that for some $k > k_0$ and $n > \ell'$, the word $w_{n,k}$ can not be reduced by M. Hence, $w_{n,k} \in L_{\text{nr}}$ and we can pump within the prefix 0^{2k} of $w_{n,k}$. We obtain the word $w' = (0^{2k+2i}1)(0^{2k}1)^{n-1}0^{2k}(10^{2k})^n$ for some $i > 0$ also from L_{nr}. However, w' is in L as w' has even length and starts by 0^{2k+1}, but ends by 10^{2k}. This is a contradiction to the assumption that all words accepted by M directly are shorter than ℓ' symbols.

Thus all words $w_{n,k}$ for $n > \ell'$ and $k > k_0$ can be reduced by M. We will additionally suppose that k_0 is greater than the size of the window of M. We will further prove that those words are always reduced by a rewriting in the innermost subword 0^{2k}.

Let us suppose that the rewriting does not occur in the innermost subword 0^{2k}. As the result of the rewriting, no more than $k-1$ symbols can be removed (because k is greater than the size of the window of M). Thus, the center of the word moves by less than $k/2$ symbols to either side. Before the rewriting, there was the word 0^k before the center and 0^k following the center — in total, there was the innermost subword 0^{2k}. After the rewriting, there will surely be a pair of corresponding symbols around the center of the word that will not be the same and thus the resulting word will belong to L. This is a contradiction to the error preserving property of M. Thus, the rewriting has to occur in the innermost subword 0^{2k}.

Let us create a language similar to L_{rd} that will mark the exact places of rewritings. We define it as $L'_{\text{rd}} = \bigcup_{(L_\ell, x \to y, L_{\text{r}}) \in I_{\text{RW}}} (L_\ell \cdot \{x\bullet\} \cdot L_{\text{r}})$ where $\bullet$ is a new symbol. Again, L'_{rd} is a regular language. Let $L_1 = \{w_{n,k'_0}; n > n_0\}$ for $k'_0 = k_0 + 1$ — note that this is a regular language as well because $L_1 = \{(0^{2k'_0}10^{2k'_0}1)^i 0^{2k'_0}; i \geq 0\} \cap \Sigma^{\geq 2(n_0+1)(2k'_0+1)+2k'_0}$. We define a homomorphism h by $h(0) = 0, h(1) = 1$, and $h(\bullet) = \lambda$. Now let us consider $L_2 = L'_{\text{rd}} \cap h^{-1}(L_1)$. We know that all words from L_1 are reduced by M. The words in L_2 contain a special mark $\bullet$ denoting where exactly rewritings of words from L_1 occur. Remember that we already proved that the rewriting changes the innermost subword 0^{2k} in all those $w_{n,k}$ and thus also the mark $\bullet$ occurs at the corresponding place of each word from L_2 — but note that the mark can be placed either in the innermost subword 0^{2k} or in the following one (consider e.g. the reduction $0010\underline{010}0 \Rightarrow 0010100$, replacing the subword 010 by the word 10, where the corresponding word in L_2 would be $0010010\bullet0$). Still, L_2 is a regular language (the class of regular languages is closed under inverse homomorphism [6]).

However, consider the homomorphism $g(0) = \lambda, g(1) = 1$, and $g(\bullet) = \bullet$ applied to L_2. We obtain the infinite language $g(L_2) \subseteq \{1^n \bullet 1^n; n > n_0\} \cup \{1^{n+1} \bullet 1^{n-1}; n > n_0\}$ that is not a regular language (use Lemma 3.17). This is a contradiction because the class of regular languages is closed under homomorphism [6].

We conclude that there is no RRW-automaton accepting L.

□

4. Conclusions

We analyzed S-kR-RRWW-automata. They are quite powerful as they can represent all GCSL. In the case of S-kR-RRW-automata the reversibility level can be used to tune their power. This helps in grammatical inference as lowering the power of a model can significantly reduce the search space and amount of training samples needed. [5] shows that S-kR-RRW-automata can be successfully inferred from positive data even in the case of random targets. However, we showed that S-kR-RRW-automata and

SLT-R-automata without auxiliary symbols characterize incomparable classes of languages. Thus both have their place in the grammatical inference field. Nevertheless, the relation between S-kR-RRWW-automata and SLT-R-automata is open.

References

[1] Amar, V., Putzolu, G.: On a family of linear grammars, *Information and Control*, **7**, 1964, 283–291.

[2] Angluin, D.: Inference of reversible languages, *JACM*, **29**(3), 1982, 741–765.

[3] Buntrock, G.: *Wachsende kontextsensitive Sprachen*, Habilitation thesis, Universität Würzburg, 1996.

[4] Dahlhaus, E., Warmuth, M. K.: Membership for growing context-sensitive grammars is polynomial, *Journal of Computer System Sciences*, **33**(3), 1986, 456–472.

[5] Hoffmann, P.: *Machine learning of analysis by reduction*, Phd-thesis, Charles University in Prague, 2013.

[6] Hopcroft, J. E., Ullman, J. D., Motwani, R.: *Introduction to automata theory, languages, and computation*, 2nd edition, Reading : Addison-Wesley, 2001, ISBN 0201441241.

[7] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata, *FCT'95, Proceedings* (H. Reichel, Ed.), 965, Springer, 1995.

[8] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: On monotonic automata with a restart operation, *Journal of Automata, Languages and Combinatorics*, **4**(4), 1999, 287–311.

[9] Kobayashi, S., Yokomori, T.: Learning concatenations of locally testable languages from positive data, *AII '94, ALT '94, Proceedings* (S. Arikawa, K. P. Jantke, Eds.), 872, Springer, 1994.

[10] Lautemann, C.: One pushdown and a small tape, *Dirk Siefkes zum 50. Geburtstag* (K. W. Wagner, Ed.), Technische Universität Berlin and Universität Augsburg, 1988.

[11] Linz, P.: *An introduction to formal languages and automata*, Lexington : D. C. Heath, 1990.

[12] Lopatková, M., Plátek, M., Kuboň, V.: Modelling syntax of free word-order languages: Dependency analysis by reduction, *TSD 2005, Proceedings* (V. Matoušek, P. Mautner, T. Pavelka, Eds.), 3658, Springer, 2005.

[13] McNaughton, R.: The loop complexity of pure-group events, *Information and Control*, **11**(1–2), 1967, 167–176.

[14] Mráz, F., Otto, F., Plátek, M.: Learning analysis by reduction from positive data, *ICGI 2006, Proceedings* (Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, E. Tomita, Eds.), 4201, Springer, 2006.

[15] Mráz, F., Plátek, M., Procházka, M.: On Special Forms of Restarting Automata, *Grammars*, **2**(3), 1999, 223–233.

[16] Otto, F.: Restarting automata and their relation to the Chomsky hierarchy, *DLT 2003, Proceedings* (Z. Ésik, Z. Fülöp, Eds.), 2710, Springer, 2003.

[17] Otto, F.: Restarting automata, in: *Recent Advances in Formal Languages and Applications* (Z. Ésik, C. Martín-Vide, V. Mitrana, Eds.), vol. 25 of *Studies in Computational Intelligence*, Springer, Berlin, 2006, 269–303.

[18] Plátek, M., Lopatková, M., Oliva, K.: Restarting automata: motivations and applications, *Workshop 'Petrinetze' and 13. Theorietag 'Formale Sprachen und Automaten', Proceedings* (M. Holzer, Ed.), Institut für Informatik, Technische Univ. München, 2003.

[19] Yokomori, T., Kobayashi, S.: Learning local languages and their application to DNA sequence analysis, *IEEE Transactions on Pattern Analysis Machine Intelligence*, **20**(10), 1998, 1067–1079.