

Blind Counter Automata on ω -Words

Henning Fernau

*Fachbereich IV, Abteilung Informatik, Universität Trier,
D-54286 Trier, Germany
fernau@uni-trier.de*

Ralf Stiebe

*Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg
PF-4120, D-39016 Magdeburg, Germany
stiebe@iws.cs.uni-magdeburg.de*

Abstract. This paper generalizes the concept of blind multicounter languages to infinite words. We introduce two different acceptance modes of blind multicounter machines on ω -words, called synchronous and asynchronous acceptance. These acceptance modes are compared with each other and with families of ω -languages of the form $L = \bigcup_{1 \leq i \leq k} U_i V_i^\omega$, where U_i, V_i are finitary blind multicounter languages.

Keywords: blind counter machines, ω -languages, closure properties

1. Introduction

In this paper, we give some results concerning blind counter automata when used as acceptors for ω -languages. This is interesting, since such devices are in many respects similar to the regular case, which has found many nice applications in the case of ω -languages, as well.

While automata equipped with partially blind counters (a little bit confusingly under the name of blind counters) have been examined as acceptors for ω -languages in the fundamental study of Engelfriet and Hooeboom [2], as well as in topology-oriented research [1, 10], this has not been the case with blind counters, since the empty storage acceptance condition does not fit into their framework.

We distinguish two ways of accepting ω -words by multicounter automata, depending on whether or not we expect the counters to be all synchronized upon reaching a final state. We derive complete, dense, infinite hierarchies (parameterized by the number of counters) in both cases. Furthermore,

we study representations of the form $L = \bigcup_{1 \leq i \leq k} U_i V_i^\omega$, where the U_i and V_i are finitary blind multicounter languages, i.e., the ω -Kleene closure of blind multicounter languages. We also compare these hierarchies *inter alia*. In a separate section, we discuss (Boolean) closure properties of all the ω -language families.

2. Definitions and Examples

For basics in automata theory, we refer the reader to [12, 16]. The theory of ω -languages is nicely covered in [19]. X^* is the free monoid over X , $e \in X^*$ denotes the empty word, X^ω is the set of ω -words over X . \sqsubseteq denotes the prefix relation in X^* , where the right-hand side might also refer to an ω -word. Let $|x|_a$ denote that number of occurrences of a in x . For some tuple t , $\pi_j(t)$ denotes the projection of t on its j th component. $\vec{0}$ is a multidimensional all-zero-vector; \vec{e}_i is the unit-vector whose i th component is one, i.e., $\pi_i(\vec{e}_j) = 1$ if and only if $i = j$. The number of dimensions of $\vec{0}$ and of \vec{e}_i will be seen from the context.

Since definitions of counter automata are not standardized in the literature, we have to make the notions we use precise below, mostly following [11].

A *blind k -counter machine* $\mathfrak{M} = (Q, X, \delta, q_0, Q_f, k)$ consists of a finite set Q of *states*, a designated *initial state* q_0 , a designated subset Q_f of *final or accepting states*, a finite *input alphabet* X and a finite *transition relation*

$$\delta \subseteq Q \times (X \cup \{e\}) \times Q \times \mathbb{Z}^k.$$

Intuitively, this means that δ describes what \mathfrak{M} should do when being in a specific state $q \in Q$, when reading the current input symbol $x \in X$ (or ignoring the input if $x = e$). The last two components of that product describe the reaction of \mathfrak{M} on the observed input; namely, \mathfrak{M} would then go (in the next step) into a specified successor state and add to the i th counter ($1 \leq i \leq k$) the integer given by the i th place of the last component.

A *configuration* c of \mathfrak{M} is a member of $Q \times X^* \times X^* \times \mathbb{Z}^k$. The set of configurations is denoted by $C(\mathfrak{M})$. Especially, $c_0(w) = (q_0, e, w, \vec{0})$ is the *initial configuration for w* and $C_f = Q_f \times \{(w, e, \vec{0})\}$ is the set of *final configurations*.

Observe that we require here that all counters are zero at the end of a computation. In fact, this is the only way that such a machine may test the contents of its counters.

If $(q, a, q', \vec{x}) \in \delta$ and $c = (q, v, aw, \vec{y})$ is a configuration of \mathfrak{M} , then we write

$$(q, v, aw, \vec{y}) \vdash_{\mathfrak{M}} (q', va, w, \vec{y} + \vec{x}) =: c'$$

and say that a *drives* \mathfrak{M} from c into c' . If $a = e$, this is an *e -move*. $\vdash_{\mathfrak{M}}$ is a relation on $Q \times X^* \times X^* \times \mathbb{Z}^k$. Its reflexive transitive closure is denoted by $\vdash_{\mathfrak{M}}^*$. The *language accepted by \mathfrak{M}* is

$$L(\mathfrak{M}) = \{w \in X^* : \exists c_f \in C_f(c_0(w) \vdash_{\mathfrak{M}}^* c_f)\}.$$

For $c_0(w) \vdash_{\mathfrak{M}}^* c$, we also say that w *drives* \mathfrak{M} into c . A sequence $c_0(w) = c_0 \vdash_{\mathfrak{M}} c_2 \vdash_{\mathfrak{M}} \cdots \vdash_{\mathfrak{M}} c_n \in C_f$ is called an *accepting run* of \mathfrak{M} on w .

The family \mathcal{B}_k contains all languages that can be accepted by blind k -counter machines. When neglecting the number of counters, we get $\mathcal{B}_* = \bigcup_{k \geq 0} \mathcal{B}_k$.

When dealing with ω -words, a configuration has to be a tuple in $Q \times X^* \times X^\omega \times \mathbb{Z}^k$. Upon feeding an ω -word $w = a_1 a_2 a_3 \cdots \in X^\omega$ into a blind counter machine, we naturally obtain an infinite

sequence $c_0c_1c_2c_3\dots$ of configurations, with c_0 being the initial configuration and a_i driving the automaton from configuration c_{i-1} into configuration c_i .

There are a number of ways to obtain ω -languages from a blind counter automaton \mathfrak{M} . In the following, we will discuss two modes of acceptance, combining the Büchi-condition for finite automata with blind counters.

- $L^\omega(\mathfrak{M})$: Accept all ω -words that drive \mathfrak{M} infinitely often through a particular final state, having all counters equal zero at the same time.
This mode is called *synchronous acceptance*, since all acceptance conditions have to be met at the same time.
The associated family of ω -languages accepted by blind k -counter automata will be denoted by \mathcal{B}_k^ω .
- $L^{\omega,a}(\mathfrak{M})$: Accept all ω -words that drive \mathfrak{M} infinitely often through a particular final state, having each counter equal zero infinitely often.
This mode is called *asynchronous acceptance*, since the acceptance conditions could be met independently of each other.
The associated family of ω -languages asynchronously accepted by blind k -counter automata will be denoted by $\mathcal{B}_k^{\omega,a}$.

More formally, an ω -word $w = a_1a_2\dots \in X^\omega$, with $a_i \in X \cup \{e\}$ is accepted by the blind k -counter machine $\mathfrak{M} = (Q, X, \delta, q_0, Q_f, k)$ iff there exists a sequence of states q_0, q_1, \dots and a sequence of vectors $\vec{y}_0, \vec{y}_1, \dots$ with $\vec{y}_i \in \mathbb{Z}^k$, such that for $c_0 = c_0(w)$ and

$$c_i = (q_{i-1}, a_1 \dots a_{i-1}, a_i a_{i+1} \dots, \vec{y}_i), \quad i \geq 1$$

such that $c_{i-1} \vdash_{\mathfrak{M}} c_i$ for $i \geq 1$, and

1. in the synchronous acceptance mode, we find an infinite set J of natural numbers (indices) and a final state q_f , such that:
for all $j \in J$, (1) $\pi_1(c_j) = q_f$, and (2) $\pi_4(c_j) = \vec{0}$, i.e., $\forall 1 \leq \kappa \leq k : \pi_\kappa(\pi_4(c_j)) = 0$;
2. in the asynchronous acceptance mode, we find infinite subsets J_0, \dots, J_k , such that:
(1) $\forall j \in J_0 : \pi_1(c_j)$ is final, and (2) $\forall 1 \leq \kappa \leq k \forall j \in J_\kappa : \pi_\kappa(\pi_4(c_j)) = 0$.

Often related is a representation via ω -powers: An ω -language L is said to be in \mathcal{K}_k iff we find blind k -counter languages V_i and W_i such that $L = \bigcup_{i=1}^n V_i \cdot W_i^\omega$. In other words, \mathcal{K}_k is just the ω -Kleene closure of \mathcal{B}_k .

If we neglect the number of counters that are used, we arrive at the classes \mathcal{B}_*^ω , $\mathcal{B}_*^{\omega,a}$ and \mathcal{K}_* .

Example 2.1. For $k \geq 1$, let $X_k = \{a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k\}$. Consider the blind k -counter automaton

$$\mathfrak{M} = (\{q_0\}, X_k, \{(q_0, a_i, q_0, \vec{e}_i), (q_0, b_i, q_0, -\vec{e}_i) : 1 \leq i \leq k\}, q_0, \{q_0\}).$$

The languages and ω -languages accepted by \mathfrak{M} are

$$\begin{aligned} L(\mathfrak{M}) &= \{w \in X_k^* : |w|_{a_i} = |w|_{b_i}, \text{ for } 1 \leq i \leq k\}, \\ L^\omega(\mathfrak{M}) &= \{w \in X_k^\omega : |\{p \sqsubset w : |p|_{a_i} = |p|_{b_i}, \text{ for } 1 \leq i \leq k\}| = \infty\}, \\ L^{\omega,a}(\mathfrak{M}) &= \{w \in X_k^\omega : |\{p \sqsubset w : |p|_{a_i} = |p|_{b_i}\}| = \infty, \text{ for some } 1 \leq i \leq k\}. \end{aligned}$$

We close this section by showing how an accepting run on some word w can be turned into an accepting run on another word w' . This technique will often be used to show that some language or ω -language cannot be accepted by blind multicounter automata.

Consider an accepting run of a blind multicounter automaton on w . Let $p = p_1p_2p_3$ be a finite word such that $w = pt$, and after reading each of the prefixes p_1 , p_1p_2 and $p_1p_2p_3$, the automaton reaches the state z and the vectors \vec{x}_1 , $\vec{x}_1 + \vec{x}_2$ and $\vec{x}_1 + \vec{x}_2 + \vec{x}_3$. Then there is an accepting run on $w' = p't$ with $p' = p_1p_3p_2$: Omit the transitions for p_2 and append them after p_3 . The automaton reaches state z and vector $\vec{x}_1 + \vec{x}_2 + \vec{x}_3$ after reading p' , and then continues as in the run on w . We say that the loop (z, p_2, z, \vec{x}_2) is deleted after p_1 and inserted after p_1p_3 . When dealing with ω -words, this insertion/deletion process is sometimes repeated infinitely often.

3. Hierarchy results

The definitions of ω -languages by blind counter automata imply three natural hierarchies

$$\mathcal{B}_0^\omega \subseteq \mathcal{B}_1^\omega \subseteq \mathcal{B}_2^\omega \subseteq \dots, \quad \mathcal{B}_0^{\omega,a} \subseteq \mathcal{B}_1^{\omega,a} \subseteq \mathcal{B}_2^{\omega,a} \subseteq \dots, \quad \mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \dots.$$

We will show that all these hierarchies are proper, that \mathcal{B}_*^ω and $\mathcal{B}_*^{\omega,a}$ are incomparable, and that $\mathcal{B}_i^\omega \subset \mathcal{K}_i$, for $i \geq 1$. Notice that the properness of the corresponding finitary language hierarchy is well-known, see [11, 22].

In the case of regular ω -languages, that correspond to the case that $k = 0$, we can state as a well-known result:

Lemma 3.1. $\mathcal{B}_0^\omega = \mathcal{B}_0^{\omega,a} = \mathcal{K}_0$.

A slight modification of published proofs [19] (also in [16]) for the mentioned regular case exhibits:

Lemma 3.2. $\forall k \geq 0 (\mathcal{B}_k^\omega \subseteq \mathcal{K}_k)$.

Proof:

Let $\mathfrak{M} = (Q, X, \delta, q_0, Q_f, k)$ be a k -counter-automaton that accepts the ω -language $L = L^\omega(\mathfrak{M})$. Derive from \mathfrak{M} the blind multicounter automata $\mathfrak{M}_p^q = (Q, X, \delta, p, \{q\}, k)$ accepting $L_p^q \subseteq X^*$. We can show that

$$L = \bigcup_{q \in Q_f} L_{q_0}^q (L_q^q)^\omega$$

Namely, any ω -word $w \in L$ drives \mathfrak{M} infinitely often through a specific final state q with all counters zero and can be hence decomposed as $w = uv_1v_2\dots$, where u is taking \mathfrak{M} from its initial state into q , with all counters zero, and v_j is taking \mathfrak{M} from q into q , all counters zero at the beginning and at the end. Therefore, $u \in L_{q_0}^q$ and $v_j \in L_q^q$ for all j .

Conversely, consider an ω -word $w = uv_1v_2\dots$ with $u \in L_{q_0}^q$ and $v_j \in L_q^q$ for all j , where $q \in Q_f$ is fixed. By definition of the Büchi acceptance condition, $w \in L$, since it drives \mathfrak{M} infinitely often into a final configuration.

Hence, $L \in \mathcal{K}_k$. □

Lemma 3.3. $L_1 = \{a^n b^n \mid n \geq 1\}^\omega \notin \mathcal{B}_*^\omega$.

Proof:

Assume to the contrary that there is a blind multicounter machine \mathfrak{M} with N states accepting L_1 . Consider an accepting run on $w = (a^N b^N)^\omega$. After reading the prefix $p_{N+1} = (a^N b^N)^{N+1}$, \mathfrak{M} has reached some configuration (z, \vec{x}) . Let B_i be the i -th block of a 's in p_{N+1} , for $1 \leq i \leq N + 1$. For any $1 \leq i \leq N + 1$, the machine performs a loop reading a non-empty sequence a^{ℓ_i} in B_i , moving from state z_i to itself, and adding \vec{x}_i to the counters. By the pidgeonhole principle, there have to be indices $1 \leq j < k \leq N + 1$ such that $z_j = z_k$. Deleting the loop in block B_j and inserting it in block B_k , we reach the configuration (z, \vec{x}) after reading the prefix

$$p'_{N+1} = (a^N b^N)^{j-1} a^{N-\ell_j} b^N (a^N b^N)^{k-j-1} a^{N+\ell_j} b^N (a^N b^N)^{N+1-k}.$$

Thus, there is an accepting run for the omega-word $p'_{N+1} (a^N b^N)^\omega \notin L_1$, contradicting our assumption. \square

With a similar argument it can be shown that the ω -language

$$L = \{w \in \{a, b\}^\omega : \forall p(p \sqsubseteq w \rightarrow |p|_a \geq |p|_b)\}$$

cannot be accepted by any blind multicounter machine. Note that L is accepted by a deterministic partially blind 1-counter automaton.

We can summarize our findings as follows:

Theorem 3.1. $\forall k \geq 0 (\mathcal{B}_k^\omega \subseteq \mathcal{K}_k)$. The inclusion turns into equality if and only if $k = 0$.

Notice that this result contrasts not only to what is known about regular ω -languages, but also to what is known about context-free ω -languages [19].

It is well known (see, e.g., [21]) that any non-empty regular ω -language contains an ultimately periodic ω -word, and that the emptiness problem is decidable for finite Büchi automata. This can be directly transferred to the class \mathcal{K}_* (and thus to \mathcal{B}_*^ω).

Theorem 3.2. Any non-empty ω -language in \mathcal{K}_* contains an ω -word of the form st^ω , where s and t are (finite) strings.

Proof:

Let $L = \bigcup_{i=1}^n U_i V_i^\omega$, for some $n \in \mathbb{N}$ and U_i, V_i being (finitary) blind k -counter languages. If L is non-empty, there has to be some i such that U_i and V_i are non-empty. Choose some $s \in U_i, t \in V_i$. Then, $st^\omega \in L$. \square

Theorem 3.3. Given a blind multicounter automaton \mathfrak{M} , it is decidable whether $L^\omega(\mathfrak{M})$ is empty.

Proof:

Let \mathfrak{M} be a blind k -counter automaton. According to the proof of Lemma 3.2, we can effectively find blind k -counter automata $\mathfrak{M}_i, \mathfrak{M}'_i$, such that

$$L^\omega(\mathfrak{M}) = \bigcup_{i=1}^n L(\mathfrak{M}'_i) [L(\mathfrak{M}_i)]^\omega.$$

Now, $L^\omega(\mathfrak{M}) \neq \emptyset$ if and only if we can find an index i , $1 \leq i \leq n$, such that $L(\mathfrak{M}'_i) \neq \emptyset$ and $L(\mathfrak{M}'_i) \notin \{\emptyset, \{e\}\}$. It is well-known that blind finitary k -counter languages have a decidable emptiness problem (based on their containment in partially blind finitary k -counter languages and the relation of the latter family to Petri net languages as exhibited in [11]). The second property can be reduced to this problem, as well, by noting that $L = L(\mathfrak{M}'_i) \notin \{\emptyset, \{e\}\}$ for $L \subseteq X^*$ if and only if $L \cap X^+ \neq \emptyset$, employing that blind k -counter languages are effectively closed under intersection with regular sets. \square

Next, we discuss the relations between the acceptance modes. It will turn out that they are equivalent for one counter and incomparable for more than one counter.

Lemma 3.4. $\mathcal{B}_1^{\omega,a} \subseteq \mathcal{B}_1^\omega$.

Proof:

Let $\mathfrak{M} = (Q, X, \delta, q_0, Q_f, 1)$ be some blind 1-counter automaton, which accepts an ω -language L in asynchronous mode. A blind 1-counter automaton $\mathfrak{M}' = (Q \times \{0, 1, 2\}, X, \delta', (q_0, 0), Q \times \{1\}, 1)$ synchronously accepting L is constructed in the following. To shorten our notations, for any $\zeta = (z, a, z', x) \in \delta$ and any $0 \leq i, j \leq 2$, let $\zeta'_{i,j}$ denote the transition $((z, i), a, (z', j), x)$. For such ζ , put $\zeta'_{0,0}$, $\zeta'_{0,1}$, and $\zeta'_{1,2}$ into δ' . Moreover, if $z' \notin Q_f$, then $\zeta'_{2,2} \in \delta'$, and if $z' \in Q_f$, then $\zeta'_{2,0} \in \delta'$.

Now consider an accepting run of \mathfrak{M} on some ω -word w . An accepting run of \mathfrak{M}' starts with $(q_0, 0)$ and proceeds as follows depending on the second component of the state.

- $(z, 0)$: If \mathfrak{M} uses the transition $\zeta = (z, a, z', x)$, \mathfrak{M}' applies $\zeta'_{0,0}$, if the counter does not reach the value 0; otherwise it applies $\zeta'_{0,1}$.
- $(z, 1)$: If \mathfrak{M} uses the transition $\zeta = (z, a, z', x)$, \mathfrak{M}' applies $\zeta'_{1,2}$.
- $(z, 2)$: If \mathfrak{M} uses the transition $\zeta = (z, a, z', x)$, \mathfrak{M}' applies $\zeta'_{2,2}$ if z' is not accepting; otherwise it applies $\zeta'_{2,0}$.

If \mathfrak{M} reaches infinitely often an empty counter and an accepting state, then \mathfrak{M}' reaches infinitely often an accepting configuration. On the other hand, if we get from an accepting configuration in \mathfrak{M}' to another accepting one, we have to pass an accepting state and a counter of zero in the corresponding run of \mathfrak{M} . \square

Lemma 3.5. $\mathcal{B}_1^\omega \subseteq \mathcal{B}_1^{\omega,a}$.

Proof:

Let $\mathfrak{M} = (Q, X, \delta, q_0, Q_f, 1)$ be some blind 1-counter automaton, which accepts an ω -language L (in synchronous mode). Without loss of generality assume $q_0 \in Q_f$. A blind 1-counter automaton $\mathfrak{M}' = (Q, X, \delta', q_0, Q_f, 1)$ accepting L asynchronously is constructed as follows. For any transition $(z, a, z', x) \in \delta$, δ' contains the transition $(z, a, z', 2x + d)$, where d is

- 0, if $z \in Q_f$ and $z' \in Q_f$, or $z \notin Q_f$ and $z' \notin Q_f$;
- 1, if $z \in Q_f$ and $z' \notin Q_f$;
- -1, if $z \notin Q_f$ and $z' \in Q_f$.

Obviously, in a run of \mathfrak{M}' , one reaches state z and counter contents $2x + y$ with $y = 0$ if $z \in Q_f$ and $y = 1$ if $z \notin Q_f$, iff a run of \mathfrak{M} reaches state z and counter contents x . \square

Remark 3.1. The reader might have wondered why we also admitted non-synchronization between reaching a final state and an empty counter in our definition of the asynchronous acceptance mode. So, she might have expected in our definition the following lines: ... in the asynchronous acceptance mode, we find infinite subsets J_1, \dots, J_k , such that: $\forall 1 \leq \kappa \leq k \forall j \in J_\kappa : \pi_\kappa(\pi_4(c_j)) = 0$ and $\pi_1(c_j)$ is final.

The idea of having a sort of odd / even mode within the counters (as employed in the proof of the preceding lemma) proves that both definitions yield the same family of ω -languages.

For more than one counter, the acceptance modes are incomparable. The first part of this relation is a consequence of the following remarkable result.

Lemma 3.6. There is a non-empty language $L \in \mathcal{B}_2^{\omega, a}$ such that L does not contain an ω -word of the form st^ω .

Proof:

An example is the ω -language

$$L = \{w \in \{a, b\}^\omega : |\{p : p \sqsubset w \wedge |p|_a = |p|_b\}| = \infty \wedge |\{p : p \sqsubset w \wedge |p|_a = 2|p|_b\}| = \infty\}.$$

Clearly, L is non-empty. Consider the blind 2-counter automaton $\mathfrak{M} = (\{q_0\}, \{a, b\}, \delta, q_0, \{q_0\}, 2)$ with $\delta = \{(q_0, a, q_0, (1, 1)), (q_0, b, q_0, (-1, -2))\}$. After reading a prefix p , the first counter is zero iff $|p|_a = |p|_b$, while the second counter is zero iff $|p|_a = 2 \cdot |p|_b$, i.e., $L^{\omega, a}(\mathfrak{M}) = L$.

Now, suppose for contradiction that L contains some ω -word st^ω . Set $|s| = m$, $|t| = n$, $|t|_a = n_a$, $|t|_b = n_b$. Assume that $n_a > n_b$, and let $N_0 = n_a + m + 1$. Any prefix of length at least $m + n \cdot N_0$ has the form $st^N t'$, where $N \geq N_0$ and t' is a prefix of t . By the chain of inequations

$$|st^N t'|_a \geq |t^N|_a \geq |t^N|_b + N > |t^N|_b + n_b + m \geq |st^N t'|_b$$

it follows that $|p|_a = |p|_b$ holds only for finitely many prefixes p . Similarly, if $n_a \leq n_b$ it can be shown that $|p|_a = 2 \cdot |p|_b$ holds only finitely often. \square

As an immediate consequence, we obtain

Theorem 3.4. $\mathcal{B}_2^{\omega, a} \not\subseteq \mathcal{K}_*$ (and $\mathcal{B}_2^{\omega, a} \not\subseteq \mathcal{B}_*^\omega$).

On the other hand, there are ω -languages that can be accepted synchronously, but not asynchronously.

Theorem 3.5. $\mathcal{B}_2^\omega \not\subseteq \mathcal{B}_*^{\omega, a}$.

Proof:

Consider the ω -language

$$L = \{w \in \{a_1, b_1, a_2, b_2\}^\omega : |\{p : p \sqsubset w \wedge |p|_{a_1} = |p|_{b_1} \wedge |p|_{a_2} = |p|_{b_2}\}| = \infty\}.$$

Obviously, $L \in \mathcal{B}_2^\omega$. Now assume that $L = L^{\omega, a}(\mathfrak{M})$, for some blind k -counter machine \mathfrak{M} . Let N be the number of states of \mathfrak{M} . Consider an accepting run of \mathfrak{M} on

$$w = (a_1^N a_2^N b_1^N b_2^N)^\omega \in L.$$

The basic idea is again to delete/insert loops in the accepting run on w to obtain an accepting run on some $w' \notin L$. For the sake of simplicity, assume for the time being that this run does not contain a loop reading the empty word. (We shall show at the end of the proof how to deal with loops on the empty word.)

For every block a_1^N and every block a_2^N , the accepting run contains a loop. There are loops $loop_1 = (z_1, a_1^{l_1}, z_1, \vec{x})$ and $loop_2 = (z_2, a_2^{l_2}, z_2, \vec{y})$ that appear in infinitely many blocks. Let $\vec{x} = (x_1, x_2, \dots, x_k)$, $\vec{y} = (y_1, y_2, \dots, y_k)$. For $1 \leq i \leq k$, there are integers $(r_i, s_i) \neq (0, 0)$ such that $r_i x_i + s_i y_i = 0$ and $r_i > 0$ or $(r_i = 0 \text{ and } s_i > 0)$.

Let us first discuss the case $r_1 > 0$. When z_1 appears for the first time in a block of a_1 's, insert r_1 copies of $loop_1$. Continue the run as on w . If $s_1 > 0$, insert s_1 copies of $loop_2$ at the first appearance of z_2 in a block of a_2 's. Otherwise, remove the first $|s_1|$ appearances of $loop_2$, and continue as in the original run, until the next block of b_2 's is read.

If $r_1 = 0$, insert s_1 copies of $loop_2$ the first time z_2 appears in a block of a_2 and continue as in the original run, until the next block of b_2 's is read.

In either case, let p_0 be the prefix of w discussed so far, while p'_0 is the prefix obtained from p_0 by the insertions/deletions. Starting with p_0 and p'_0 , we will construct inductively infinite sequences $p_0 \sqsubset p_1 \sqsubset p_2 \sqsubset \dots \sqsubset p_j \sqsubset \dots \sqsubset w$ and $p'_0 \sqsubset p'_1 \sqsubset p'_2 \sqsubset \dots \sqsubset p'_j \sqsubset \dots$ (the latter sequence defines w') which satisfy the following conditions:

If the accepting run on w reaches after reading p_j the state z and counters' contents (c_1, \dots, c_k) , then for the derived run on p'_j , one finds:

1. $|p'_j|_{a_1} = |p'_j|_{b_1} + r_{(j \bmod k)+1} l_1$, $|p'_j|_{a_2} = |p'_j|_{b_2} + s_{(j \bmod k)+1} l_2$,
2. The state after reading p'_j is z .
3. The contents of the i -th counter is $c'_i = c_i + r_{(j \bmod k)+1} x_i + s_{(j \bmod k)+1} y_i$;
in particular, $c'_{(j \bmod k)+1} = c_{(j \bmod k)+1}$.
4. For any q' with $p'_j \sqsubseteq q' \sqsubseteq p'_{j+1}$, holds $|q'|_{a_1} > |q'|_{b_1}$ or $|q'|_{a_2} > |q'|_{b_2}$.

It is easy to verify that the claims are true for $j = 0$. Now assume that we have transformed a run on prefix p_j to a run on p'_j such that the above conditions are satisfied. First of all, the run after p'_j is continued as the original run on w after p_j , until an accepting state has been passed and counter $(j \bmod k) + 1$ has reached the value 0. As the original run on w is accepting, this is guaranteed to happen. Next, with $d_{r,j} = r_{((j+1) \bmod k)+1} - r_{(j \bmod k)+1}$ and $d_{s,j} = s_{((j+1) \bmod k)+1} - s_{(j \bmod k)+1}$, we insert or delete $|d_{r,j}|$ copies of $loop_1$ and $|d_{s,j}|$ copies of $loop_2$. The operation is insertion if the respective value is positive, and deletion, if it is negative. Moreover, if $r_{((j+1) \bmod k)+1} > 0$, the operations related to $loop_1$ are performed first; otherwise the operations related to $loop_2$ are performed first. Finally, the reading of the block is completed, giving the extended prefixes p_{j+1} and p'_{j+1} .

It is easy to verify that the conditions above are satisfied for $j + 1$. In particular, the order of the insert/delete operations guarantees that the last claim becomes true. Hence, the ω -word w' defined as the limit of the sequence $p'_0 \sqsubset p'_1 \sqsubset p'_2 \sqsubset \dots \sqsubset p'_j \sqsubset \dots$ is accepted by \mathfrak{M} , but not belonging to L .

Let us finally treat e -loops. For the proof, it is only essential that the original run on w contains infinitely many loops of the forms $(z_1, a_1^{l_1}, z_1, \vec{x})$ and $(z_2, a_2^{l_2}, z_2, \vec{y})$ without inner e -loops. This

is achieved as follows. Proceed as in the original accepting run, until all configurations with a state that appears only finitely often in this run are passed. Then, from the first loop of the form $(z_1, a_1^{l_1}, z_1, \vec{x})$ remove iteratively any inner e -loop (z, e, z, \vec{v}) and insert it at the next appearance of z . The removing/inserting of e -loops does not change the prefix read so far. Moreover, the original configuration is reconstructed after the last insertion point. From that point continue the original run, until an accepting state has been met and any counter has reached a zero value. This procedure is continued *ad infinitum*, finally giving an infinite number of the desired loops. \square

Finally, we prove the properness of the three hierarchies.
For a language L , let $\text{Infix}(L) = \{v : \exists u \exists w : uvw \in L\}$.

Lemma 3.7. For $k \geq 1$, let $X_k = \{a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k\}$ and

$$\begin{aligned} L_k &= \{w \in X_k^* : |w|_{a_i} = |w|_{b_i}, \text{ for } 1 \leq i \leq k\}; \\ L'_k &= \{a_1^n \cdots a_k^n : n \geq 0\} \cup \{b_1^n \cdots b_k^n : n \geq 0\}. \end{aligned}$$

If L is a blind $(k-1)$ -counter language such that $\text{Infix}(L)$ contains infinitely many words from L'_k , then L contains words that are not in L_k .

Proof:

Let \mathfrak{M} be a blind $(k-1)$ -counter machine accepting L , and let m be the number of states of \mathfrak{M} . Without loss of generality, we can assume that there are no e -loops of \mathfrak{M} . Any run on a subword of the shape a_i^n , $n \geq m$, contains a loop of the form (z, a_i^l, z, \vec{x}) with $1 \leq l \leq m$, $\vec{x} \in \mathbb{Z}^{k-1}$. For any k -tuple of loops

$$((z_1, a_1^{l_1}, z_1, \vec{x}_1), \dots, (z_k, a_k^{l_k}, z_k, \vec{x}_k)),$$

there is a vector $\vec{s} = (s_1, \dots, s_k) \neq \vec{0}$ such that $\sum_{i=1}^k s_i \vec{x}_i = \vec{0}$. We call \vec{s} the solution vector of the above k -tuple.

The number of loops of the above form is finite, say r , and the number of k -tuples of loops is finite, too. Let S be the maximal absolute value among the coefficients of solutions \vec{s} for the k -tuples of loops. We choose $N_0 = m \cdot r \cdot S$. Now, L contains a word w with a subword $a_1^N \cdots a_k^N$ or with a subword $b_1^N \cdots b_k^N$, for some $N > N_0$. By symmetry, it suffices to consider the first case. If $w \notin L_k$, we are done. Otherwise, in an accepting run on w , any block a_i^N , $1 \leq i \leq k$, contains at least $r \cdot S$ disjoint loops. Among these loops, there have to be S of the same type $loop_i = (z_i, a_i^{l_i}, z_i, \vec{x}_i)$. Let $\vec{s} = (s_1, \dots, s_k) \neq \vec{0}$ be the solution vector for the k -tuple. By definition, $|s_i| \leq S$. Now the accepting run on w can be modified to an accepting run on some $w' \notin L$ as follows. If $s_i \geq 0$, one inserts s_i copies of $loop_i$. If $s_i < 0$, one deletes $|s_i|$ copies of $loop_i$. As $s_i \neq 0$, for some i , the number of a_i^l 's is changed, while the number of b_i 's remains the same. Hence, the modified word is not in L_k . On the other hand, after reading the modification of the subword $a_1^N \cdots a_k^N$, one has the same state and the same contents of counters. Hence, the modified word w' is accepted. \square

Theorem 3.6. For $k \geq 1$, $\mathcal{B}_k^\omega \setminus \mathcal{K}_{k-1}$ is non-empty.
(Hence, $\mathcal{B}_{k-1}^\omega \subset \mathcal{B}_k^\omega$ and $\mathcal{K}_{k-1} \subset \mathcal{K}_k$.)

Proof:

For $k \geq 1$, let $X_k = \{a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k\}$. As shown in Example 2.1, the ω -language L_k^ω , where

$$L_k = \{w \in X_k^* : |w|_{a_i} = |w|_{b_i}, \text{ for } 1 \leq i \leq k\},$$

belongs to \mathcal{B}_k^ω .

Now consider some language $K \in \mathcal{K}_{k-1}$ that contains the subset L_k^ω . Clearly, K contains all ω -words of the form $(a_1^n \cdots a_k^n b_1^n \cdots b_k^n)^\omega$. By definition of \mathcal{K}_{k-1} , there have to be (finitary) blind $(k-1)$ -counter languages U, V such that UV^ω contains infinitely many ω -words of this form.

Suppose that $V \subseteq L_k$, then $\text{Infix}(V)$ contains infinitely many words of the forms $a_1^n a_2^n \cdots a_k^n$ or $b_1^n b_2^n \cdots b_k^n$. By Lemma 3.7, V would then contain words not in L_k , which is a contradiction. Consequently, V contains some word $v \notin L_k$, say $|v|_{a_i} > |v|_{b_i}$. Then UV^ω contains uv^ω , for some $u \in U$. Any prefix longer than $|u| + |v|(|u| + |v| + 1)$ (i.e., which has the prefix $uv^{|u|+|v|+1}$) contains more a_i 's than b_i 's. This implies that $K \neq L_k^\omega$. \square

Theorem 3.7. For $k \geq 1$, $\mathcal{B}_{k-1}^{\omega,a} \subset \mathcal{B}_k^{\omega,a}$.

Proof:

For $k \geq 1$, let $X_k = \{a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k\}$. As shown in Example 2.1, the ω -language

$$L_k = \{w \in X_k^\omega : |\{p \sqsubset w : |p|_{a_i} = |p|_{b_i}\}| = \infty, \text{ for } 1 \leq i \leq k\}$$

is in $\mathcal{B}_k^{\omega,a}$.

Let \mathfrak{M} be a blind $(k-1)$ -counter machine such that $L_k \subseteq L^{\omega,a}(\mathfrak{M})$. Similar to the proof of Lemma 3.7, an accepting run of \mathfrak{M} on some ω -word $w = (a_1^n \cdots a_k^n b_1^n \cdots b_k^n)^\omega$ can be transformed to an accepting run on an ω -word $w' = p(a_1^n \cdots a_k^n b_1^n \cdots b_k^n)^\omega$, where $|p|_{a_i} > |p|_{b_i}$ for some i , i.e., $w' \notin L_k$. \square

4. Closure Properties

As regards union, the standard construction for finite automata on languages (and for Büchi automata on ω -languages) can be extended to blind multicounter automata. Moreover, this property is trivial for the classes \mathcal{K}_k . We summarize:

Theorem 4.1. The families $\mathcal{B}_k^\omega, \mathcal{B}_k^{\omega,a}, \mathcal{K}_k, 1 \leq k$, and $\mathcal{B}_*^\omega, \mathcal{B}_*^{\omega,a}, \mathcal{K}_*$ are closed under union.

The well-known product construction can be adapted to show closure under intersection with regular ω -languages.

Theorem 4.2. The families $\mathcal{B}_k^\omega, \mathcal{B}_k^{\omega,a}, 1 \leq k$, and $\mathcal{B}_*^\omega, \mathcal{B}_*^{\omega,a}$ are closed under intersection with regular ω -languages.

Proof:

We give the proof only for the synchronous acceptance mode. For the asynchronous mode, a similar proof will work. Let $\mathfrak{M}_1 = (Q_1, X, \delta_1, q_{0,1}, Q_{f,1}, k)$ be a blind k -counter automaton and let $\mathfrak{M}_2 = (Q_2, X, \delta_2, q_{0,2}, Q_{f,2})$ be a finite Büchi automaton. The transition relation δ_2 is a subset of $Q_2 \times (X \cup \{e\}) \times Q_2$. Without loss of generality, assume that $(q_1, e, q_1, \vec{0}) \in \delta_1$ for all $q_1 \in Q_1$ and $(q_2, e, q_2) \in \delta_2$ for all $q_2 \in Q_2$.

We construct $\mathfrak{M} = (Q, X, \delta, q_0, Q_f, k)$ with $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$, $q_0 = q_{0,1} \times q_{0,2} \times \{0\}$, $Q_f = Q_{f,1} \times Q_2 \times \{2\}$, and the transition relation δ is defined as follows:

For $(q_1, a, q_1', \vec{x}) \in \delta_1$ and $(q_2, a, q_2') \in \delta_2$, δ contains the transitions

- $((q_1, q_2, 0), a, (q'_1, q'_2, 0), \vec{x})$, if $q'_2 \notin Q_{f,2}$;
- $((q_1, q_2, 0), a, (q'_1, q'_2, 1), \vec{x})$, if $q'_2 \in Q_{f,2}$;
- $((q_1, q_2, 1), a, (q'_1, q'_2, 1), \vec{x})$;
- $((q_1, q_2, 1), a, (q'_1, q'_2, 2), \vec{x})$, if $q'_1 \in Q_{f,1}$;
- $((q_1, q_2, 2), a, (q'_1, q'_2, 0), \vec{x})$.

The first two components of a state in Q simulate the states in runs of \mathfrak{M}_1 and \mathfrak{M}_2 . The third component of the state in \mathfrak{M} changes from 0 to 1 iff \mathfrak{M}_2 reaches an accepting state, it can nondeterministically decide to change from 1 to 2 iff an accepting state of \mathfrak{M}_1 is reached, and returns from 2 to 0. Hence, an accepting configuration is reached infinitely often by \mathfrak{M} , iff both \mathfrak{M}_1 and \mathfrak{M}_2 reach an accepting configuration infinitely often. \square

One of the striking facts about the family of languages that can be accepted by blind counter automata is its closure under intersection, a rare closure property in formal language theory. More precisely, one can accept the intersection of a k -counter language with an ℓ -counter language by a $(k + \ell)$ -counter machine. This property cannot be extended to ω -languages of the \mathcal{B}^ω - and \mathcal{K} -hierarchies. For the asynchronous mode, however, the product construction will work similar to Theorem 4.2.

Theorem 4.3. None of the classes $\mathcal{B}_k^\omega, \mathcal{K}_k, 1 \leq k, \mathcal{B}_*^\omega, \mathcal{K}_*$ is closed under intersection.

Proof:

Note that the ω -language L in the proof of Lemma 3.6 is the intersection of the omega-languages L_1, L_2 with

$$\begin{aligned} L_1 &= \{w \in \{a, b\}^\omega : |\{p : p \sqsubset w \wedge |p|_a = |p|_b\}| = \infty\}, \\ L_2 &= \{w \in \{a, b\}^\omega : |\{p : p \sqsubset w \wedge |p|_a = 2|p|_b\}| = \infty\}. \end{aligned}$$

It is easy to see that $L_1, L_2 \in \mathcal{B}_1^\omega$. \square

Theorem 4.4. The family $\mathcal{B}_*^{\omega,a}$ is closed under intersection.

Proof:

Let $\mathfrak{M}_1 = (Q_1, X, \delta_1, q_{0,1}, Q_{f,1}, k_1)$ and $\mathfrak{M}_2 = (Q_2, X, \delta_2, q_{0,2}, Q_{f,2}, k_2)$ be two blind multi-counter automata. Without loss of generality, assume that $(q_1, e, q_1, \vec{0}) \in \delta_1$ for all $q_1 \in Q_1$ and $(q_2, e, q_2, \vec{0}) \in \delta_2$ for all $q_2 \in Q_2$.

We construct $\mathfrak{M} = (Q, X, \delta, q_0, Q_f, k_1 + k_2)$ with $Q = Q_1 \times Q_2 \times \{0, 1, 2\}$, $q_0 = q_{0,1} \times q_{0,2} \times \{0\}$, $Q_f = Q_1 \times Q_2 \times \{2\}$, and the transition relation δ is defined as follows:

For $(q_1, a, q'_1, \vec{x}) \in \delta_1$ and $(q_2, a, q'_2, \vec{y}) \in \delta_2$, δ contains the transitions

- $((q_1, q_2, 0), a, (q'_1, q'_2, 0), \vec{x} \times \vec{y})$, if $q_1 \notin Q_{f,1}$;
- $((q_1, q_2, 0), a, (q'_1, q'_2, 1), \vec{x} \times \vec{y})$, if $q_1 \in Q_{f,1}$;
- $((q_1, q_2, 1), a, (q'_1, q'_2, 1), \vec{x} \times \vec{y})$, if $q_2 \notin Q_{f,2}$;
- $((q_1, q_2, 1), a, (q'_1, q'_2, 2), \vec{x} \times \vec{y})$, if $q_2 \in Q_{f,2}$;

- $((q_1, q_2, 2), a, (q'_1, q'_2, 0), \vec{x} \times \vec{y})$.

It is easy to see that \mathfrak{M} simulates one step of both automata simultaneously. Moreover, the third component of the state in \mathfrak{M} changes from 0 to 1 for an accepting first component, from 1 to 2 for an accepting second component and returns from 2 to 0. Hence, an accepting state is reached infinitely often by \mathfrak{M} , iff both \mathfrak{M}_1 and \mathfrak{M}_2 reach an accepting state infinitely often. \square

Theorem 4.5. None of the families $\mathcal{B}_i^\omega, \mathcal{B}_i^{\omega,a}, \mathcal{K}_i, 1 \leq i$, and $\mathcal{B}_*^\omega, \mathcal{B}_*^{\omega,a}, \mathcal{K}_*$ are closed under complementation.

Proof:

As regards the \mathcal{B}^ω - and \mathcal{K} -families, the non-closure simply follows from the closure under union and the non-closure under intersection.

For the asynchronous case, let

$$L = \{w \in \{a, b\}^\omega : |\{p : p \sqsubset w, |p|_a = |p|_b\}| = \infty\}.$$

Obviously, L is in $\mathcal{B}_1^{\omega,a}$. Now assume that there is a blind multicounter automaton \mathfrak{M} accepting the complement L^c of L in asynchronous mode, and let N be the number of states of \mathfrak{M} . Consider the ω -word $w = a^{N+1}b^N(a^N b^N)^\omega$. For any prefix $p \neq e$, $1 \leq |p|_a - |p|_b \leq N + 1$, and thus $w \in L^c$. We can also write w as $w = aB_1B_2 \dots$, where $B_i = a^N b^N$, and call B_i a block in w . Now regard an accepting run of \mathfrak{M} on w . We will construct an accepting run on a certain ω -word $w' \in L$. In some block B_i , there is a loop (z, a^l, z, \vec{x}) such that state z appears in the substring a^N of a later block B_j . We remove this loop in B_i and insert it in B_j . Note that the number of a 's and b 's is equal after reading the first $(N_l + 1)$ b 's of (the modified) block B_i . After reading the (modified) block B_j , \mathfrak{M} has the same state and contents of counters as in the run on w . Then the run is continued as on w , until an accepting state has been seen and all counters have reached the value zero. This insert/delete/continue procedure is repeated infinitely often. It gives us the desired run on an ω -word in L . \square

5. Conclusions and Prospects

We started a formal-language oriented study on natural extensions of regular ω -languages. This contrasts a little bit with the multitude of topology-oriented studies performed in this area before. It might be worthwhile continuing this line of research, somewhat in parallel to the theory developed for word languages. A natural question would be to find more general criteria that ensure the existence (or, more difficult, the non-existence) of representations of the form $L = \bigcup_{i=1}^n U_i V_i^\omega$, (ω -Kleene closure) where U_i and V_i are word languages described by a certain mechanism, and L is a language supposed to be described by the same mechanism type, now seen as a device that describes ω -languages.

Without proof (and without formal definitions), we mention that the classes $\mathcal{B}_k^\omega, \mathcal{B}_k^{\omega,a}$ can be also characterized as ω -languages that can be generated by regular valence grammars, see [9, 8, 15]. Furthermore, some of our proofs also transfer to the case of partially blind multicounter automata, see [11].

Besides questions of hierarchies of families of ω -languages, algebraic (closure) properties and decidability questions could be of interest. Here, we only mention that known closure properties of

word languages could transfer to the “corresponding” ω -languages if representations as ω -Kleene closures are known, see [17, 19]. For example, one can see the closure of $\mathcal{B}_k^{\omega,a}$ and of \mathcal{B}_k^ω under inverse morphisms and e -free morphisms. However, \mathcal{K}_k appears to be more tricky in this respect, possibly just because the underlying word languages do not form AFLs.

Moreover, one could think of quite a number of variants of blind counter machines working on ω -words that could be interesting to explore from a formal language point of view. For example, besides the Büchi acceptance condition that we chose, one could likewise examine other acceptance conditions as introduced in the case of regular ω -languages.

Let us finally sketch one application of such a representation theorem, if it exists: As exhibited in a sequence of papers by Staiger [6, 7, 14, 18, 20], ω -languages can be used to describe fractal images; in fact, equipped with the “right” metric on X^ω , ω -languages themselves form interesting topological and measure-theoretic objects. Then, units like the Hausdorff measure and the Hausdorff dimension of ω -languages and fractals can be asked for and possibly computed. When dealing with an ω -language $L = \bigcup_{i=1}^n U_i V_i^\omega$, it is sufficient to be able to compute those units for the V_i^ω , which in turn can be often reduced to computations on V_i alone, see [3, 4, 6, 7, 13]. To ensure computability, the decidability of certain code properties turns out to be important, see [5]. Moreover, it is interesting whether one could assume that those V_i are (e.g.) prefix codes, without restricting the class of languages. This seems to be unknown for \mathcal{K}_k .

References

- [1] Duparc, J., Finkel, O., Ressayre, J.-P.: Computer science and the fine structure of Borel sets, *Theoretical Computer Science*, **257**, 2001, 85–105.
- [2] Engelfriet, J., Hoogeboom, H. J.: X -automata on ω -words, *Theoretical Computer Science*, **110**, 1993, 1–51.
- [3] Fernau, H.: *Iterierte Funktionen, Sprachen und Fraktale*, Mannheim: BI-Verlag, 1994.
- [4] Fernau, H.: Valuations, regular expressions, and fractal geometry, *Applicable Algebra in Engineering, Communication and Computing*, **7**(1), 1996, 59–75.
- [5] Fernau, H., Reinhardt, K., Staiger, L.: Decidability of code properties, *Developments in Language Theory DLT: Foundations, Applications, and Perspectives* (G. Rozenberg, W. Thomas, Eds.), World Scientific, 2000, pages 153–163.
- [6] Fernau, H., Staiger, L.: Valuations and unambiguity of languages, with applications to fractal geometry, *Automata, Languages and Programming, 21st International Colloquium, ICALP* (S. Abiteboul, E. Shamir, Eds.), Lecture Notes in Computer Science **820**, Springer, 1994, pages 11–22.
- [7] Fernau, H., Staiger, L.: Iterated function systems and control languages, *Information and Computation*, **168**, 2001, 125–143.
- [8] Fernau, H., Stiebe, R.: Sequential grammars and automata with valences, *Theoretical Computer Science*, **276**, 2002, 377–405.
- [9] Fernau, H., Stiebe, R.: Valuated and valence grammars: an algebraic view, *Proceedings DLT* (W. Kuich, G. Rozenberg, A. Salomaa, Eds.), Lecture Notes in Computer Science **2295**, Springer, 2002, pages 281–292.
- [10] Finkel, O.: An effective extension of the Wagner hierarchy to blind counter automata, *Computer Science Logic CSL* (L. Fribourg, Ed.), Lecture Notes in Computer Science **2142**, Springer, 2001, pages 369–383.

- [11] Greibach, S.: Remarks on blind and partially blind one-way multicounter machines, *Theoretical Computer Science*, **7**, 1978, 311–324.
- [12] Hopcroft, J. E., Ullman, J. D.: *Introduction to Automata Theory, Languages, and Computation*, Reading (MA): Addison-Wesley, 1979.
- [13] Mauldin, R. D., Williams, S. C.: On the Hausdorff dimension of some graphs, *Transactions of the American Mathematical Society*, **298**(2), December 1986, 793–803.
- [14] Merzenich, W., Staiger, L.: Fractals, dimension, and formal languages, *RAIRO Informatique théorique et Applications/Theoretical Informatics and Applications*, **28**(3–4), 1994, 361–386.
- [15] Păun, G.: A new generative device: valence grammars, *Rev. Roumaine Math. Pures Appl.*, **XXV**(6), 1980, 911–924.
- [16] Rozenberg, G., Salomaa, A., Eds.: *Handbook of Formal Languages (3 volumes)*, Springer, 1997.
- [17] Staiger, L.: Empty-storage-acceptance of omega-languages, *Fundamentals of Computation Theory FCT* (M. Karpinski, Ed.), Lecture Notes in Computer Science **56**, Springer, 1977, pages 516–521.
- [18] Staiger, L.: Kolmogorov complexity and Hausdorff dimension, *Information and Computation (formerly Information and Control)*, **103**, 1993, 159–194.
- [19] Staiger, L.: ω -languages, Technical Report 9, Martin-Luther-Universität Halle-Wittenberg, Fachbereich Mathematik und Informatik, 1997.
- [20] Staiger, L.: Infinite iterated function systems in Cantor space and the Hausdorff measure of ω -power languages, *International Journal of Foundations of Computer Science*, **16**, 2005, 787–802.
- [21] Thomas, W.: Automata on infinite objects, in: *Handbook of Theoretical Computer Science*, Elsevier, 1990, 133–191.
- [22] Vicolov, S.: Hierarchies of valence languages, *Developments in Theoretical Computer Science / International Meeting of Young Computer Scientists IMYCS* (J. Dassow, A. Kelemenová, Eds.), Gordon and Breach, 1994, pages 191–196.