AUTOMATA WITH MODULO COUNTERS AND NONDETERMINISTIC COUNTER BOUNDS

Daniel Reidenbach and Markus L. Schmid*

We introduce and investigate Nondeterministically Bounded Modulo Counter Automata (NBMCA), which are two-way multi-head automata that comprise a constant number of modulo counters, where the counter bounds are nondeterministically guessed, and this is the only element of nondeterminism. NBMCA are tailored to recognising those languages that are characterised by the existence of a specific factorisation of their words, e. g., pattern languages. In this work, we subject NBMCA to a theoretically sound analysis.

Keywords: Multi-head automata, Counter automata, Modulo counters, Stateless automata, Restricted nondeterminism

Classification: 68Q45, 68Q05, 68Q10

1. INTRODUCTION

In the present paper we introduce and study a novel automata model, the Nondeterministically Bounded Modulo Counter Automata (NBMCA for short), which comprise several two-way input heads and a number of counters. These NBMCA are algorithmic tools suitable for recognising those languages that are characterised by the existence of a specific factorisation of their words, e. g., pattern languages, and are a generalisation of the Janus automata that have been introduced and applied in [15] in order to investigate the membership problem for pattern languages. In [15], NBMCA with exactly two input heads are used. In the present work we focus on NBMCA with only one head, since by the use of additional counters, an NBMCA can easily simulate several input heads by only one. Every counter of an NBMCA is provided with a counter bound, and can only be incremented and counts modulo its counter bound. The current counter values and counter bounds are hidden from the transition function, which can only check whether a counter has reached its bound. By performing a reset on a counter, the automaton nondeterministically guesses a new counter bound between 0 and $|w|$, where w is the input word, and the actual value of the counter is set back to 0. This guessing of counter bounds is the only possible nondeterministic step of NBMCA, and the transition function is defined completely deterministically. We can interpret the counter bounds as positions of the input and, by means of the counter values, the input head can be moved to these positions.

*Corresponding author.

Two aspects of this approach seem to be particularly worth studying. Firstly, all additional resources the automaton is equipped with, namely the counters, are tailored to storing positions in the input word. We can observe that this aspect is not really new; in fact, the idea of separating the mechanisms of storing positions from the functionality of actually processing the input is formalised in the models of partially blind multi-head automata (see, e. g., Ibarra and Ravikumar [10]), Pebble Automata (see, e. g., Chang et al. [1]) and automata with sensing heads (see, e. g., Petersen [14]). In spite of this similarity between NBMCA and established automata models regarding their emphasis on storing positions in the input word, there is still one difference: the counters of NBMCA are quite limited in their ability to change the positions they represent, since their values cannot be decremented and their bounds cannot be set deterministically. The question arises whether or not, for automata using counters as additional resources, their ability to count in both directions is essential with respect to the expressive power.

The second aspect is that the nondeterminism of NBMCA, which merely allows positions in the input word to be guessed, differs quite substantially from the common nondeterminism of automata, which provides explicit computational alternatives. Nevertheless, automata often use their nondeterminism to actually guess a certain position of the input. For example, a pushdown automaton with one head that recognises $\{ww^R \mid w \in \Sigma^*\}$ needs to perform an unbounded number of guesses even though only one specific position, namely the middle one, of the input needs to be found. Despite this observation, the nondeterminism of NBMCA might be weaker, as it seems to *solely* refer to positions in the input. Hence, we also investigate the question of whether or not it is essential that the nondeterminism is explicitly provided by a nondeterministic transition function in order to exploit it to the full extent, in terms of expressive power.

In order to understand the character of these novel, and seemingly limited, resources NBMCA can use, the present paper compares the expressive power of these automata to that of the well-established, and seemingly less restricted, models of multi-head and counter automata. Furthermore, we study some basic decision problems for NBMCA as well as stateless versions of NBMCA, with and without restricted nondeterminism.

A preliminary version [16] of this paper was presented at the conference CIAA 2012.

Due to space restrictions, some of the proofs are omitted and can be found in a technical report [17].

2. DEFINITIONS AND PRELIMINARY OBSERVATIONS

Let $\mathbb{N}$ denote the set of all positive integers and let $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. The symbols $\subseteq$ and $\subset$ refer to subset and proper subset relation, respectively. For an arbitrary alphabet Σ, a *word* (*over* Σ) is a finite sequence of symbols from Σ, and ε stands for the *empty word*. The symbol Σ^+ denotes the set of all nonempty words over Σ, and $\Sigma^* := \Sigma^+ \cup \{\varepsilon\}$. For the *concatenation* of two words u, v, we write $u \cdot v$ or simply uv, and u^k denotes the k-fold concatenation of u, i. e., $u^k := u_1 \cdot u_2 \cdots u_k$, where $u_i = u$, $1 \leq i \leq k$. We say that a word $v \in \Sigma^*$ is a *factor* of a word $w \in \Sigma^*$ if there are $u_1, u_2 \in \Sigma^*$ such that $w = u_1 \cdot v \cdot u_2$. If $u_1 = \varepsilon$ (or $u_2 = \varepsilon$), then v is a *prefix* of w (or a *suffix*, respectively). The notation $|K|$ stands for the size of a set K or the length of a word K; the term $|w|_a$ refers to the number of occurrences of the symbol a in the word w. If we wish to refer to the symbol at a certain position in a word $w = \mathtt{a}_1 \cdot \mathtt{a}_2 \cdot \cdots \cdot \mathtt{a}_n$, $\mathtt{a}_i \in \Sigma$,

$1 \leq i \leq n$, over some alphabet Σ, we use $w[i] := \mathtt{a}_i$, $1 \leq i \leq n$. Furthermore, for all i, i', $1 \leq i < i' \leq |w|$, let $w[i,i'] := \mathtt{a}_i \cdot \mathtt{a}_{i+1} \cdot \cdots \cdot \mathtt{a}_{i'}$ and $w[i,-] := w[i,|w|]$.

For arbitrary languages L_1, L_2 we define $L_1^+ := \{u_1 \cdot u_2 \cdot \cdots \cdot u_n \mid u_i \in L_1, 1 \leq i \leq n, n \in \mathbb{N}\}$, $L_1^* := L_1^+ \cup \{\varepsilon\}$ and $L_1 \cdot L_2 := \{u \cdot v \mid u \in L_1, v \in L_2\}$.

2.1. Automata Models

For proving our results about NBMCA, we shall apply several variants of multi-head and counter automata.

For every $k \in \mathbb{N}$ let 1DFA(k), 2DFA(k), 1NFA(k) and 2NFA(k) denote the class of *deterministic one-way*, *deterministic two-way*, *nondeterministic one-way* and *nondeterministic two-way automata* with k input heads, respectively. A 1DFA(k), 2DFA(k), 1NFA(k) or 2NFA(k) is given as a tuple $(k, Q, \Sigma, \delta, q_0, F)$ comprising the number of *input heads* k, a set of *states* Q, the *input alphabet* Σ, the *transition function* δ, an *initial state* $q_0 \in Q$ and a set of *accepting states* $F \subseteq Q$. The transition function is a mapping $\delta : Q \times \Sigma^k \to Q \times D^k$ for deterministic and $\delta : Q \times \Sigma^k \to P(Q \times D^k)$ for nondeterministic devices, where $P(A)$ is the power set of a set A, and D, the set of input head movements, is defined by $D := \{0, 1\}$ in case of one-way automata and $D := \{-1, 0, 1\}$ for the two-way versions. Let $C \in Q \times \Sigma^k$ and $S \in Q \times D^k$. Instead of writing transitions in the form $\delta(C) = S$ or, in case of nondeterministic automata, $S \in \delta(C)$, we use the notation $C \to_\delta S$ for both deterministic and nondeterministic automata. If δ is obvious from the context, we simply write $C \to S$. We assume in general that the input of two-way models is bounded by *endmarkers* (¢, \$) and the input head(s) can sense these endmarkers and cannot be moved to the left of the left endmarker or to the right of the right endmarker. For a comprehensive survey on multi-head automata the reader is referred to Holzer et al. [5] and to the references therein.

For some of our proofs, it is also convenient to use so-called *counter automata*, thus, we shall formally define them. For every $l \in \mathbb{N}$ let 1CDFA(l), 2CDFA(l), 1CNFA(l) and 2CNFA(l) denote the class of *deterministic one-way*, *deterministic two-way*, *nondeterministic one-way* and *nondeterministic two-way counter automata* with one input head and l counters. The counters can store only non-negative values. In each transition, these counters can be incremented, decremented or left unchanged and, furthermore, it can be checked on whether or not a certain counter stores value 0. A transition of a counter automaton depends on the state, the currently scanned input symbol and the set of counters currently storing 0. We shall also define more restricted versions of counter automata. To this end, let $k \in \mathbb{N}$ and let $f : \mathbb{N} \to \mathbb{N}$ be a recursive function. An $f(n)$-*bounded nondeterministic* or *deterministic two-way counter automaton with k counters (*$2\text{CNFA}_{f(n)}(k)$ *or* $2\text{CDFA}_{f(n)}(k)$ *for short)* is a 2CNFA(k) (or 2CDFA(k), respectively), whose counters have an upper bound of $f(n)$, where n is the current input length. It can be checked whether a counter stores 0, $f(n)$ or a value in between[1]. For more details on counter automata see, e. g., Ibarra [7], Chiniforooshan et al. [2] or Holzer et al. [5] and the references therein.

For an arbitrary class of automata, such as the set 1DFA(k) of deterministic one-way automata with k input heads, the expression "a 1DFA(k)" refers to any automaton from

[1] In the following, we are mainly concerned with $2\text{CDFA}_{f(n)}(k)$ and $2\text{CNFA}_{f(n)}(k)$ where $f(n) = n$.

1DFA(k). For an arbitrary automaton M, $L(M)$ denotes the set of all words accepted by M and, for an arbitrary class A of automata, let $\mathcal{L}(A) := \{L(M) \mid M \in A\}$.

The following obvious proposition shall be important for some of our proofs:

Proposition 2.1. For every $k \in \mathbb{N}$, $\mathcal{L}(\text{2CNFA}_n(k)) \subseteq \mathcal{L}(\text{2NFA}(k+1))$.

P r o o f. The statement of the proposition can be easily comprehended by observing that we can use k input heads of a 2NFA($k+1$) in such a way that they ignore the input, thus, they behave exactly like counters that are bounded by the input length. □

2.2. Nondeterministically Bounded Modulo Counter Automata

A *Nondeterministically Bounded Modulo Counter Automaton*, NBMCA(k) for short, is a two-way one-head automaton with k counters. More precisely, it is a tuple $M := (k, Q, \Sigma, \delta, q_0, F)$, where $k \in \mathbb{N}$ is the number of *counters*, Q is a finite nonempty set of *states*, Σ is a finite nonempty alphabet of *input symbols*, $q_0 \in Q$ is the *initial state* and $F \subseteq Q$ is the set of *accepting states*. The mapping $\delta : Q \times (\Sigma \cup \{¢, \$\}) \times \{\mathtt{t}_0, \mathtt{t}_1\}^k \to Q \times \{-1, 0, 1\} \times \{0, 1, \mathtt{r}\}^k$ is called the *transition function* (the symbols ¢, \$ (referred to as *left* and *right endmarker*, respectively) are not in Σ). Instead of writing transitions in the form $\delta(C) = S$, we use the notation $C \to_\delta S$. If δ is obvious from the context, we simply write $C \to S$. An input to M is any word of the form $¢w\$$, where $w \in \Sigma^*$. Let $(p, b, s_1, \dots, s_k) \to_\delta (q, r, d_1, \dots, d_k)$. We call the element b the *scanned input symbol* and r the *input head movement.* For each $j \in \{1, 2, \dots, k\}$, the element $s_j \in \{\mathtt{t}_0, \mathtt{t}_1\}$ is the *counter message of counter* j, and d_j is called the *counter instruction for counter* j. The transition function δ of an NBMCA(k) determines whether the input heads are moved to the left ($r_i = -1$), to the right ($r_i = 1$) or left unchanged ($r_i = 0$), and whether the counters are incremented ($d_j = 1$), left unchanged ($d_j = 0$) or reset ($d_j = \mathtt{r}$); a decrement is not possible. In case of a reset, the counter value is set to 0 and a new counter bound is nondeterministically guessed between 0 and the current input length. Hence, every counter is bounded, but these bounds are chosen in a nondeterministic way. In order to define the language accepted by an NBMCA, we need to define the concept of an NBMCA computation.

Let M be an NBMCA and $¢w\$:= a_0 \cdot a_1 \cdot \dots \cdot a_{n+1}$, $a_i \in \Sigma$, $1 \leq i \leq n$. A *configuration of* M *(on input* $¢w\$$*)* is an element of $\widehat{C}_M := \{[q, h, (c_1, C_1), \dots, (c_k, C_k)] \mid q \in Q, 0 \leq h \leq n+1, 0 \leq c_i \leq C_i \leq n, 1 \leq i \leq k\}$. The pair (c_i, C_i), $1 \leq i \leq k$, describes the current configuration of the i^{th} counter, where c_i is the *counter value* and C_i the *counter bound.* The element h is called the *input head position.*

An *atomic move* of M is denoted by the relation $\vdash_{M,w}$ over the set of configurations. Let $(p, b, s_1, \dots, s_k) \to_\delta (q, r, d_1, \dots, d_k)$. Then, for all c_i, C_i, $1 \leq i \leq k$, where $c_i < C_i$ if $s_i = \mathtt{t}_0$ and $c_i = C_i$ if $s_i = \mathtt{t}_1$, and for every h, $0 \leq h \leq n+1$, with $a_h = b$, we define $[p, h, (c_1, C_1), \dots, (c_k, C_k)] \vdash_{M,w} [q, h', (c'_1, C'_1), \dots, (c'_k, C'_k)]$. Here, the elements h' and c'_j, C'_j, $1 \leq j \leq k$, are defined in the following way: $h' := h + r$ if $0 \leq h + r \leq n+1$ and $h' := h$ otherwise. For each $j \in \{1, \dots, k\}$, if $d_j = \mathtt{r}$, then $c'_j := 0$ and, for some $m \in \{0, 1, \dots, n\}$, $C'_j := m$. If $d_j \neq \mathtt{r}$, on the other hand, then $C'_j := C_j$ and $c'_j := c_j + d_j \mod (C_j + 1)$.

To describe a *sequence of (atomic) moves of M (on input w)* we use the reflexive and transitive closure of the relation $\vdash_{M,w}$, denoted by $\vdash^*_{M,w}$. M accepts the word w if and only if $\widehat{c}_0 \vdash^*_{M,w} \widehat{c}_f$, where $\widehat{c}_0 := [q_0, 0, (0, C_1), \dots, (0, C_k)]$ for some $C_i \in \{0, 1, \dots, |w|\}$, $1 \leq i \leq k$, is an *initial configuration*, and $\widehat{c}_f := [q_f, h, (c_1, C_1), \dots (c_k, C_k)]$ for some $q_f \in F$, $0 \leq h \leq n+1$ and $0 \leq c_i \leq C_i \leq n$, $1 \leq j \leq k$, is a *final configuration.* In every computation of an NBMCA, the counter bounds are nondeterministically initialised, and the only nondeterministic step an NBMCA is able to perform during the computation consists in guessing a new counter bound for some counter.

Example 2.2. In order to illustrate the definition of Nondeterministically Bounded Modulo Counter Automata, we sketch how an NBMCA with one counter can recognise the language $L_{\text{rev}} := \{ww^R \mid w \in \Sigma^*\}$, where w^R denotes the reversal of a word w.

In a first step, by moving the input head from the left endmarker to the right endmarker, it is checked whether or not the message of the counter changes from $\mathtt{t}_0$ to $\mathtt{t}_1$ exactly when the input head reaches the right endmarker, i. e., whether or not the counter bound equals the length of the input. Furthermore, at the same time it is checked whether or not the input w has even length. This can be easily done with the finite state control. In case that $|w|$ is odd or the counter bound is not $|w|$, the input is rejected by entering a non-accepting trap state. Now, the counter can be used to execute the following three steps in a loop.

1. Move the input head one step to the right.
2. Move the input head for $|w| + 1$ steps by initially moving it to the right and reversing its direction if the right endmarker is reached.
3. Move the input head for $|w|+1$ steps by initially moving it to the left and reversing its direction if the left endmarker is reached.

This loop is executed until the right endmarker is reached in step 1. It can be easily verified that this exactly happens in the $(|w| + 1)^{\text{th}}$ iteration of the loop. Furthermore, for every i, $1 \leq i \leq |w|$, in the i^{th} iteration of the loop, the position reached after step 1 is i and the position reached after step 2 is $|w| - i + 1$. So in order to check on whether or not $w = uu^R$, $u \in \Sigma^*$, it is sufficient to store the symbol at position i after step 1 in the finite state control and compare it to the symbol at position $|w| - i + 1$ after step 2 in each iteration of the loop. If eventually the right endmarker is reached after step 1, then the automaton accepts its input and if, on the other hand, the symbol stored in the finite state control does not equal the symbol scanned after step 2, then the input is rejected.

3. EXPRESSIVE POWER, HIERARCHY AND DECIDABILITY

In this section, we investigate typical automata theoretical questions with respect to the class of NBMCA. More precisely, we first investigate their expressive power by comparing them to ordinary multi-head and counter automata. The results obtained in this regard are then used in order to conclude a hierarchy result of the class of NBMCA-languages with respect to the number of counters. We conclude this section by a thorough investigation of the decidability of the emptiness, infiniteness, universe, equivalence, inclusion and disjointness problem of the class of languages given by NBMCA.

3.1. Expressive Power

An NBMCA can be regarded as a finite state control with additional resources. Thus, it is quite similar to classical nondeterministic multi-head automata. The essential differences between the models are those addressed in Section 1. Hence, in order to gain insights with respect to the question of whether these differences affect the expressive power, we study the problem of simulating classical nondeterministic multi-head automata by NBMCA and vice versa.

We first address the simulation of NBMCA by nondeterministic multi-head automata. Intuitively, it seems obviously possible, since NBMCA can be interpreted as just a further restricted version of nondeterministic multi-head automata. This intuition is formalised by the following theorem:

Theorem 3.1. For every $k \in \mathbb{N}$, $\mathcal{L}(\text{NBMCA}(k)) \subseteq \mathcal{L}(\text{2NFA}(2k+1))$.

Proof. We prove that, for every $k \in \mathbb{N}$ and for every $M \in \text{NBMCA}(k)$, there exists an $M' \in \text{2CNFA}_n(2k)$ with $L(M) = L(M')$, which, by Proposition 2.1, implies the statement of the theorem. To this end, let $M \in \text{NBMCA}(k)$. The input head of M' is used in exactly the same way M uses its input head. Hence, it is sufficient to illustrate how M' simulates the modulo counters of M. The idea is that the modulo counter i, $1 \leq i \leq k$, of M is simulated by the counters $2i-1$ and $2i$ of M', i. e., counter $2i-1$ represents the counter value and counter $2i$ represents the counter bound of the modulo counter i of M. A reset of modulo counter i is simulated by M' in the following way. First, both counters $2i-1$ and $2i$ of M' are decremented to 0. Then counter $2i-1$ is incremented and after every increment, M' nondeterministically guesses whether it keeps on incrementing or it stops. If the counter reaches value n, it must stop. The value counter $2i-1$ stores after that procedure is interpreted as the new counter bound. The actual counting of the modulo counter i of M is then simulated in the following way. Whenever M increments counter i, then M' increments counter $2i$ and decrements counter $2i-1$. When counter $2i-1$ reaches 0, then this is interpreted as reaching the counter bound. In order to enable a new incrementing cycle of the modulo counter i of M from 0 to its counter bound, the counters $2i-1$ and $2i$ simply change their roles and can then be used again in the same way. □

The converse question, i. e., whether arbitrary multi-head automata, and particularly their unrestricted nondeterminism, can be simulated by NBMCA, is more interesting. One possible way to do this is to use each modulo counter of the NBMCA in order to simulate an input head of the 2NFA(k). To this end, the modulo counter first guesses $|w|$ as counter bound, which is done by resetting it and checking, by means of the input head, whether or not the guessed bound equals $|w|$, and then the counter value can be used in order to store the position of the input head. Since the counter value cannot be decremented, a decrement has to be performed by $|w|-1$ increments.

However, for reasons that shall be explained later, we aim for a simulation that, with respect to the usage of modulo counters, is more economic compared to the construction sketched above. More presicely, we want to use a single modulo counter in order to store the positions of two input heads of a 2NFA(k), i. e., the counter value and the counter bound each represents a distinct input head position. A step of the 2NFA(k)

is then simulated by first moving the input head of the NBMCA successively to all these positions stored by the counters and record the scanned input symbols in the finite state control. After that, all these positions stored by the counters must be updated according to the transition function of the 2NFA(k). It turns out that this is possible, but, since counter values cannot be decremented and counter bounds cannot be changed directly, the constructions are rather involved and require some technical finesse. Furthermore, we need an additional counter which is also used in order to simulate the possible nondeterministic choices of the 2NFA(k).

Theorem 3.2. For every $k \in \mathbb{N}$, $\mathcal{L}(\text{2NFA}(k)) \subseteq \mathcal{L}(\text{NBMCA}(\lceil \frac{k}{2} \rceil + 1))$.

Proof. Given a 2NFA(k) M, we show how an NBMCA($\lceil \frac{k}{2} \rceil + 1$) M' can be designed such that $L(M) = L(M')$. It is a well-known fact that at the cost of an increase in the number of states, we can modify an automaton such that each configuration has at most two next configurations. Therefore, since the number of states does not play any role in the statement of the lemma, we assume that M has this property. Below an input is denoted by $w := a_1 \cdot a_2 \cdots a_n$ for some letters $a_1, \ldots, a_n \in \Sigma$. For the sake of convenience, we assume that k is even; the case that k is odd can be handled analogously. The general idea is that the first $\lceil \frac{k}{2} \rceil$ modulo counters of M' are used to store the positions of the k input heads of M. Thus, one modulo counter of M' stores the positions of two input heads of M, i. e., one position is represented by the counter value and the other one by the counter bound of the modulo counter. In addition to that, M' has an auxiliary counter that is used to store data temporarily. More precisely, if M is able to perform the move $[q, h_1, \ldots, h_k] \vdash_{M,w} [p, h'_1, \ldots, h'_k]$, then M' can perform a sequence of moves $[q, 0, (h_1, h_2), \ldots, (h_{k-1}, h_k), (0, c)] \vdash^*_{M',w} [p, 0, (h'_1, h'_2), \ldots, (h'_{k-1}, h'_k), (0, c')]$, for some c, c', $1 \leq c, c' \leq n$. The role of the counter bounds c and c' of the auxiliary counter are not important right now and shall be explained later on.

We shall now informally explain the basic idea of how a step of M can be simulated by a sequence of moves of M' and formally prove all the technical details afterwards. A transition of M depends on k input symbols and a state. Therefore, M' records in its finite state control all the symbols at the positions determined by the counter values and counter bounds. More precisely, if h_1 and h_2 are the counter value and counter bound of the first counter and M' is in state q right now, then M' moves its input head to position h_1, changes into state $q_{a_{h_1}}$, moves the input head to position h_2 and changes into state $q_{a_{h_1}, a_{h_2}}$. The same procedure is applied to all counters $2, 3, \ldots, \lceil \frac{k}{2} \rceil$ until M' finally reaches a state $q_{a_{h_1}, a_{h_2}, \ldots, a_{h_k}}$. We note that in order to prove that all these steps can be carried out by M', it is sufficient to show that M' can perform the following sequences of moves:

$$\widetilde{c} \vdash^*_{M',w} [q_{a_{h_{2i-1}}}, 0, (h_1, h_2), \ldots, (h_{2i-1}, h_{2i}), \ldots, (h_{k-1}, h_k), (0, c')] \,, \tag{1}$$

$$\widetilde{c} \vdash^*_{M',w} [q_{a_{h_{2i}}}, 0, (h_1, h_2), \ldots, (h_{2i-1}, h_{2i}), \ldots, (h_{k-1}, h_k), (0, c')] \,, \tag{2}$$

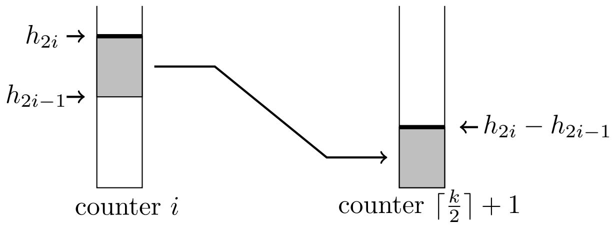
where $\widetilde{c} := [q, 0, (h_1, h_2), \ldots, (h_{k-1}, h_k), (0, c'')]$ is an arbitrary configuration of M' and $0 \leq c', c'' \leq n$ (i. e., M' is able to store the symbol at a position indicated by a counter value (sequence of moves (1)) as well as the symbol at a position indicated by a counter bound (sequence of moves (2))).

The next step of M is now determined by q, the symbols $a_{h_1}, a_{h_2}, \dots, a_{h_k}$ and δ, the transition function of M, which is possibly nondeterministic and can choose among two possible choices. In order to simulate this nondeterministic choice between two options, M' resets counter $\lceil \frac{k}{2} \rceil + 1$ and checks whether or not the newly guessed counter bound equals 0, which is only the case if the counter message is $\mathtt{t}_1$ right after resetting it. The transition function of M' can then be defined such that the first option of the two possible transitions of M is carried out if 0 is guessed as new counter bound and the second option is chosen otherwise. We assume that the transition chosen by M is $(q, a_{h_1}, \dots, a_{h_k}) \rightarrow_\delta (p, d_1, \dots, d_k)$, where p is the new state and $(d_1, \dots, d_k)$ are the input head movements, so next all counter values and counter bounds need to be updated according to $(d_1, \dots, d_k)$. To this end, M' changes into state $p_{d_1,\dots,d_k}$ where the counter value of counter 1 is changed to $h_1 + d_1$ and after that M changes into state $p_{d_2,\dots,d_k}$. Next, the counter bound of counter 1 is changed to $h_2 + d_2$ while the state changes into $p_{d_3,\dots,d_k}$ and so on. Eventually, M' reaches state p and the configurations of the counters are $(h_1 + d_1, h_2 + d_2), \dots, (h_{k-1} + d_{k-1}, h_k + d_k)$. Again, in order to prove that this procedure can be carried out by M', it is sufficient to show that M' can perform the following sequences of moves:

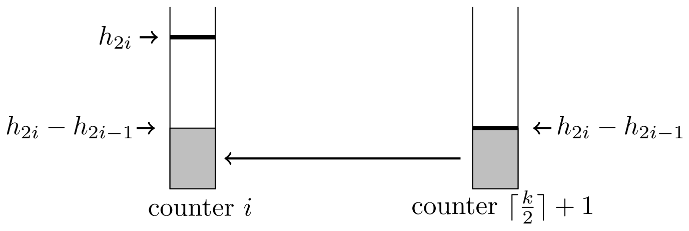
$$\widetilde{c} \vdash^*_{M',w} [q, 0, (h_1, h_2), \dots, (h_{2i-1} + d, h_{2i}), \dots, (h_{k-1}, h_k), (0, c')]\,, \tag{3}$$
$$\widetilde{c} \vdash^*_{M',w} [q, 0, (h_1, h_2), \dots, (h_{2i-1}, h_{2i} + d'), \dots, (h_{k-1}, h_k), (0, c')]\,, \tag{4}$$

where $\widetilde{c} := [q, 0, (h_1, h_2), \dots, (h_{k-1}, h_k), (0, c'')]$ is an arbitrary configuration of M', $0 \leq c', c'' \leq n$, $d, d' \in \{1, -1\}$, $h_{2i-1} + d \leq h_{2i}$ and $h_{2i-1} \leq h_{2i} + d'$.

In order to conclude the proof, it remains to show that the transition function δ' of M' can be defined in a way such that the sequences of moves (1) - (4) can be performed. We begin with the sequences of moves (1) and (2). First, M' resets counter $\lceil \frac{k}{2} \rceil + 1$ and then increments counter $\lceil \frac{k}{2} \rceil + 1$ and counter i simultaneously. If these two counters reach their counter bounds at exactly the same time, then we can conclude that the newly guessed counter bound of counter $\lceil \frac{k}{2} \rceil + 1$ equals $h_{2i} - h_{2i-1}$ and we proceed. In case that a different counter bound is guessed, M' changes into a non-accepting trap state. This procedure is illustrated by the following diagram.

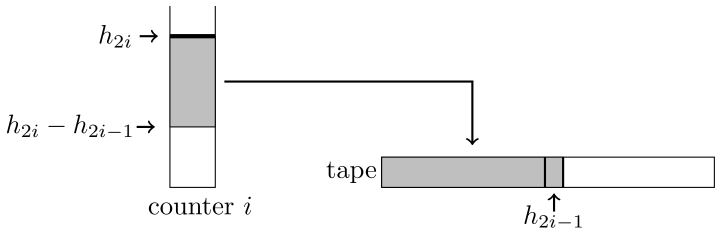


Counters i and $\lceil \frac{k}{2} \rceil + 1$ are then set back to 0 by incrementing them once more. Then they are incremented simultaneously until counter $\lceil \frac{k}{2} \rceil + 1$ reaches its counter bound. After this step, counter i stores value $h_{2i} - h_{2i-1}$ as pointed out by the following illustration.



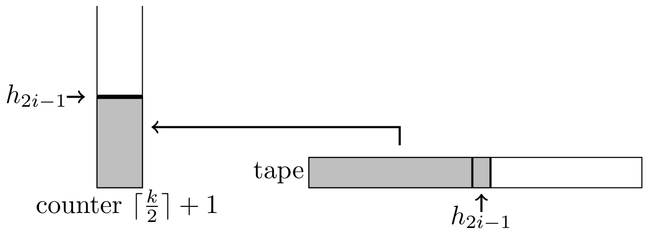
$h_{2i} \rightarrow$
$h_{2i} - h_{2i-1} \rightarrow$
$\leftarrow h_{2i} - h_{2i-1}$
counter i
counter $\lceil \frac{k}{2} \rceil + 1$

Now it is possible to increment counter i and simultaneously move the input head to the right until counter i reaches its bound of h_{2i}. Clearly, this happens after h_{2i-1} increments, so the input head is then located at position h_{2i-1} of the input tape (see the following picture).

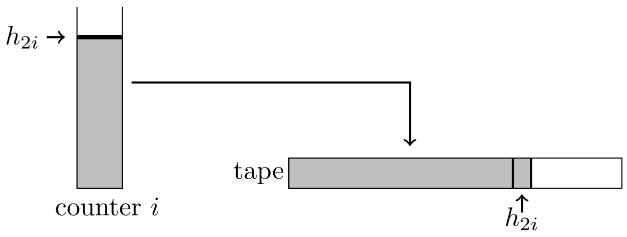


Now, in case of (1), M' changes into state $q_{a_{h_{2i-1}}}$ and sets the value of counter i back to 0. Finally, by moving the input head from position h_{2i-1} to the left until it reaches the left endmarker and simultaneously incrementing counter i, we set the input head back to position 0 and the counter value of counter i back to h_{2i-1}. Furthermore, we set the value of counter $\lceil \frac{k}{2} \rceil + 1$ back to 0.

In case of (2), a few more steps are required. We recall that the input head is located at position h_{2i-1}. M' resets counter $\lceil \frac{k}{2} \rceil + 1$ and checks whether or not the new counter bound equals h_{2i-1}. This is done by moving the input head to the left and simultaneously incrementing counter $\lceil \frac{k}{2} \rceil + 1$.



Next, we set the value of counter i back to 0 and then increment it until the counter bound of h_{2i} is reached and simultaneously move the input head to the right. Obviously, the input head is then located at position h_{2i}. Thus, M' can change into state $q_{a_{h_{2i}}}$.



As counter $\lceil \frac{k}{2} \rceil + 1$ has a counter bound of h_{2i-1}, we can easily set the value of counter i back to h_{2i-1}. Finally, the input head is moved back to position 0 and the value of counter $\lceil \frac{k}{2} \rceil + 1$ is set back to 0.

Next, we consider case (3). If $d = 1$, then we can simply increment counter i. If, on the other hand, $d = -1$, we first move the input head to position h_{2i-1} in the same way we did in case (1), and then one step to the left, i. e., to position $h_{2i-1} + d$. Now we can set counter i to 0, and then increment it and simultaneously move the input head to the left until it reaches the left endmarker. After that step, counter i stores value $h_{2i-1} + d$.

In order to implement case (4), we first move the input head to position h_{2i} in the same way we did in case (2), i. e., we first move it to position h_{2i-1} as done for cases (1) and (2) and then, by resetting counter $\lceil \frac{k}{2} \rceil + 1$, we store h_{2i-1} in the counter bound of counter $\lceil \frac{k}{2} \rceil + 1$ and finally use counter i in order to move the input head to position h_{2i}. Next, we move the input head to position $h_{2i} + d'$, reset counter i and, by moving the input head back to position 0, check whether $h_{2i} + d'$ is guessed as new counter bound. Finally, we use counter $\lceil \frac{k}{2} \rceil + 1$, which has a counter bound of h_{2i-1}, to set the counter value of counter i back to h_{2i-1}.

It remains to show how we can handle the cases where we have $h_{2i-1} = h_{2i}$ and either h_{2i-1} should be incremented or h_{2i} should be decremented. Clearly, this is not possible, so in this case we simply change the roles of the counter bound and counter value to avoid this problem. If we do this, we need to store in the finite state control that from now on the counter value stores the position of input head $2i$ and the counter bound stores the position of input head $2i - 1$.

This shows that M' can perform the sequences of moves (1) - (4), which implies that M' can simulate M in the way described at the beginning of this proof. □

We point out that, by Proposition 2.1, the above proof also implies that an arbitrary $M \in \text{2CNFA}_n(k)$, $k \in \mathbb{N}$, can be simulated by some $M' \in \text{NBMCA}(\lceil \frac{k+1}{2} \rceil + 1)$.

Before we proceed, we discuss the above result in a bit more detail. In the application of NBMCA in [15] every counter bound is interpreted as an anchor on the input tape and the functionality of the counters is merely a mechanism to move the input head to these anchored positions. On the other hand, in the proof of Theorem 3.2 it is vital to overcome the strong dependency between a counter value and its counter bound such that both of them can be fully exploited as mere storages for input positions that can be arbitrarily updated and, thus, the counter value as well as the counter bound are each as powerful as an input head. Considering the substantial differences between NBMCA on the one hand and 2NFA on the other, it seems surprising that this is possible. This means that neither the restrictions on the counters of NBMCA nor the special nondeterminism constitutes a restriction on the expressive power. Thus, NBMCA can be used whenever classical multi-head automata can be applied, but due to their specific counters and nondeterminism they are particularly suitable algorithmic tools for recognising those languages that are characterised by the existence of a certain factorisation for their words, such as pattern languages (see [15]).

3.2. Hierarchy

The tight use of the modulo counters in the simulation used in the proof of Theorem 3.2 turns out to be worth the effort, as it allows us to prove a hierarchy result on the class NBMCA. To this end, we first cite a classical result in automata theory, which states that adding an input head to a 2NFA(k) strictly increases its expressive power (see Holzer et al. [5] for a summary and references of the original papers):

Theorem 3.3. (Monien [13]) For every $k \in \mathbb{N}$, $\mathcal{L}(\text{2NFA}(k)) \subset \mathcal{L}(\text{2NFA}(k+1))$.

Theorem 3.3 together with Theorems 3.1 and 3.2, can be used to prove the following hierarchy result:

Corollary 3.4. For every $k \in \mathbb{N}$, $\mathcal{L}(\text{NBMCA}(k)) \subset \mathcal{L}(\text{NBMCA}(k+2))$.

Proof. By Theorems 3.1, 3.2 and 3.3, we know that, for every $k \in \mathbb{N}$,

- $\mathcal{L}(\text{NBMCA}(k)) \subseteq \mathcal{L}(\text{2NFA}(2k+1))$,
- $\mathcal{L}(\text{2NFA}(2k+1)) \subset \mathcal{L}(\text{2NFA}(2k+2))$ and
- $\mathcal{L}(\text{2NFA}(2k+2)) \subseteq \mathcal{L}(\text{NBMCA}(k+2))$.

Consequently, NBMCA(k) $\subset$ NBMCA($k+2$). □

3.3. Decidability

Next, we investigate the decidability of the emptiness, infiniteness, universe, equivalence, inclusion and disjointness problem with respect to languages given by NBMCA. All these problems are undecidable even for 1DFA(2) (cf., Holzer et al. [5]) and since NBMCA can simulate 1DFA(2) (Theorem 3.2) these negative results carry over to the class of NBMCA. However, it is a common approach to further restrict automata models with undecidable problems in order to obtain subclasses with decidable problems (see, e.g., Ibarra [7]). One respective option is to require the automata to be reversal bounded. The following definitions are according to [7].

In a computation of some two-way automaton model, an *input head reversal* describes the situation that the input head is moved a step to the right (to the left, respectively) and the last time it has been moved it was moved a step to the left (to the right, respectively), so it reverses directions. A *counter reversal* is defined in a similar way just with respect to the increments and decrements of a counter. We say that an automaton is *input head reversal bounded* or *counter reversal bounded* if there exists a constant m such that, for every accepting computation, the number of input head reversals (counter reversals, respectively) is at most m. We now formally define classes of reversal bounded automata.

Definition 3.5. For all $m_1, m_2, k \in \mathbb{N}$, (m_1, m_2)-REV-CNFA(k) denotes the class of 2CNFA(k) and (m_1, m_2)-REV-CDFA(k) denotes the class of 2CDFA(k) that perform at most m_1 input head reversals and every counter performs at most m_2 counter reversals in every accepting computation.

For the reversal bounded automata defined as above, there is no need anymore to distinguish between the one-way and the two-way case as this aspect is covered by the number of input head reversals, i. e., one-way automata coincide with those that are input head reversal bounded by 0. Next, we cite a classical result about reversal bounded counter automata:

Theorem 3.6. (Ibarra [7]) The emptiness, infiniteness and disjointness problems for the class (m_1, m_2)-REV-CNFA(k) are decidable. The emptiness, universe, infiniteness, inclusion, equivalence and disjointness problem for the class (m_1, m_2)-REV-CDFA(k) are decidable.

Our goal is to transfer these results to reversal bounded NBMCA. With respect to NBMCA, a counter reversal is interpreted as an increment of the counter in case that it has already reached its counter bound. Furthermore, we need to bound the number of resets as well.

Definition 3.7. For all $m_1, m_2, l, k \in \mathbb{N}$, let (m_1, m_2, l)-REV-NBMCA(k) denote the class of NBMCA(k) that perform at most m_1 input head reversals, at most m_2 counter reversals and resets every counter at most l times in every accepting computation.

We can show that any $M \in (m_1, m_2, l)$-REV-NBMCA(k) can be simulated by some $M' \in (m'_1, m'_2)$-REV-CNFA(k'), which implies that the results of Theorem 3.6 carry over to (m_1, m_2, l)-REV-NBMCA(k).

Lemma 3.8. For every automaton $M \in (m_1, m_2, l)$-REV-NBMCA(k), there exists an automaton $M' \in (m_1 + 2, m_2 + l + 1)$-REV-CNFA$(4k)$ such that $L(M) = L(M')$.

P r o o f. Let $M \in (m_1, m_2, l)$-REV-NBMCA(k). First, we recall that, by Theorem 3.1, an NBMCA(k) can be simulated by a CNFA$_n(2k)$. Furthermore, in this simulation, the input head of the CNFA$_n(2k)$ is used in the same way as the input head of NBMCA(k), and every counter reversal and reset of a modulo counter of the NBMCA(k) causes each of the two corresponding counters of the CNFA$_n(2k)$ to perform a reversal. Consequently, in the simulation of an NBMCA(k) by a CNFA$_n(2k)$, the input head reversals of the NBMCA(k) are preserved and the counter reversals of the CNFA$_n(2k)$ are bounded by $m_2 + l$. We conclude that there exists an $(m_1, m_2 + l)$-REV-CNFA$_n(2k)$ M', i. e., an $(m_1, m_2 + l)$-REV-CNFA$(2k)$ whose counters are bounded by the input length, with $L(M) = L(M')$. This M' can be simulated by an $(m_1 + 2, m_2 + l + 1)$-REV-CNFA$(4k)$ M'' in the following way. At the beginning of the computation M'' increments the first $2k$ counters to the input length by moving the input head over the input. After that step, for every i, $1 \leq i \leq 2k$, counter i stores the input length n and counter $i + 2k$ stores 0. Counters i and $i + 2k$ of M'' can simulate counter i of M' by decrementing (or incrementing) counter i and incrementing (or decrementing, respectively) counter

$i + 2k$ for every increment (or decrement, respectively) of counter i of M'. Hence, when counter i of M'' reaches 0, then counter i of M' reaches n and when counter $i + 2k$ of M'' reaches 0, then counter i of M' reaches 0 as well. This requires two additional input head reversals and an additional counter reversal for the first k counters. □

With Theorem 3.6, we can conclude the following:

Theorem 3.9. For the class (m_1, m_2, l)-REV-NBMCA, the emptiness, infiniteness and disjointness problems are decidable.

In the following, we study the question of whether it is possible to ease the strong restriction of (m_1, m_2, l)-REV-NBMCA a little without losing the decidability results. More precisely, we investigate the decidability of the emptiness, infiniteness, universe, equivalence, inclusion and disjointness problems for the class (m, ∞, l)-REV-NBMCA, i. e., the number of counter reversals is not bounded anymore. We shall explain our motivation for this in a bit more detail. To this end we cite the following result.

Theorem 3.10. (Ibarra [7]) The emptiness, infiniteness, universe, equivalence, inclusion and disjointness problems are undecidable for $(1, \infty)$-REV-CDFA(1).

Consequently, with respect to CDFA (and, thus, CNFA) the typical decision problems remain undecidable when the restriction on the counter reversals is abandoned. However, regarding (m, ∞, l)-REV-NBMCA we observe a slightly different situation. While a counter reversal of a counter automaton can happen anytime in the computation and for any possible counter value, a counter reversal of an NBMCA strongly depends on the current counter bound, i. e., as long as a counter is not reset, all the counter reversals of that counter happen at exactly the same counter value. So while for $(1, \infty)$-REV-CDFA the counters are not restricted at all, the modulo counters of (m, ∞, l)-REV-NBMCA can still be considered as restricted, since the number of resets is bounded. Intuitively, this suggests that the restrictions of (m, ∞, l)-REV-NBMCA are still stronger than the restrictions of $(1, \infty)$-REV-CDFA.

In the following, we give a negative answer to the question of whether or not the class (m, ∞, l)-REV-NBMCA has the same positive decidability results as the class (m_1, m_2, l)-REV-NBMCA. To this end, we first need another way to simulate counter automata by NBMCA. The simulation used to prove Theorem 3.2 has the advantage of requiring a relatively small number of modulo counters, but pays the price of a large number of input head reversals and counter resets. In fact, in the simulation of Theorem 3.2 even if the 2CNFA(k) is input head reversal bounded and counter reversal bounded, the number of counter resets as well as the input head reversals of the NBMCA($\lceil \frac{k+1}{2} \rceil + 1$) are not necessarily bounded anymore. Hence, it is our next goal to find a simulation of counter automata by NBMCA that preserves the number of input head reversals and requires only a constant number of resets. Before we give such a simulation, we need the following technical lemma, which shows that we can transform an arbitrary 2CNFA$_{f(n)}(k)$ or 2CDFA$_{f(n)}(k)$ into an equivalent 2CNFA$_{f(n)}(k)$ (or 2CDFA$_{f(n)}(k)$, respectively) that only reverses counters at value 0 or $f(n)$.

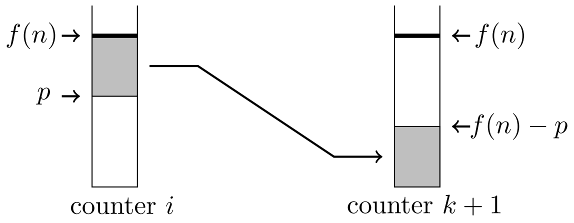
Lemma 3.11. For every $M \in$ 2CNFA$_{f(n)}(k)$ (or $M \in$ 2CDFA$_{f(n)}(k)$) there exists an $M' \in$ 2CNFA$_{f(n)}(k + 2)$ (or $M' \in$ 2CDFA$_{f(n)}(k + 2)$, respectively) such that $L(M) =$

$L(M')$ and every counter of M' reverses only at value 0 or $f(n)$. Furthermore, for every $w \in \Sigma^*$, if M reverses the input head m times and reverses every counter at most q times on input w, then M' reverses the input head m times and reverses every counter at most $2kq$ times on w.

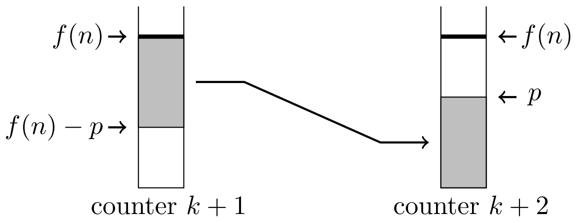
Proof. We shall show how M can be changed such that all counters only reverse at value 0 or $f(n)$. All the following constructions are deterministic, so determinism of M is preserved. We can assume that, for every counter, M stores in its finite state control whether this counter is in incrementing or decrementing mode. Thus, for any counter, M can identify a change from incrementing to decrementing and vice versa. Furthermore, by using additional states, every $\text{2CNFA}_{f(n)}(k)$ (or $\text{2CDFA}_{f(n)}(k)$, respectively) can be transformed into one that increments or decrements at most one counter in any transition. Hence, we can assume M to have this property.

We define how an $M' \in \text{2CNFA}_{f(n)}(k+2)$ (or $M' \in \text{2CDFA}_{f(n)}(k+2)$, respectively), whose counters reverse only at values 0 or $f(n)$, can simulate M. In this simulation, the counters 1 to k of M' exactly correspond to the counters 1 to k of M, and the counters $k+1$ and $k+2$ of M' are auxiliary counters. We now consider a situation where counter i of M is decremented from value p to $p-1$ and this decrement constitutes a reversal. We show how M' simulates this step such that its counter i reverses at $f(n)$. The main idea is to use the auxiliary counters $k+1$ and $k+2$ to temporarily store values, but, since these counters are required to reverse at values 0 or $f(n)$ as well, the construction is not straightforward.

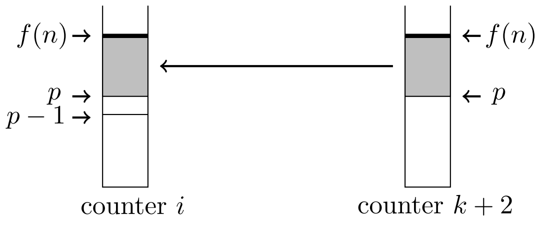
M' simulates the above described step in the following way. Instead of decrementing counter i from p to $p-1$, M' performs further dummy increments until value $f(n)$ is reached and simultaneously increments counter $k+1$. Hence, counter $k+1$ stores exactly $f(n)-p$ when counter i reaches $f(n)$. This situation is illustrated below.



Next, we increment counters $k+1$ and $k+2$ simultaneously until counter $k+1$ reaches $f(n)$. This implies that counter $k+2$ stores now p.



We can now decrement counter i, thus performing the reversal at value $f(n)$, and simultaneously increment counter $k+2$ until it reaches $f(n)$. After these steps, counter i stores value p again, but is now in decrementing mode. M finally decrements counter i to value $p-1$.



Both counters $k+1$ and $k+2$ store now value $f(n)$ and therefore are simply decremented until value 0 is reached. We note that in the above described procedure, counter i is incremented from p to $f(n)$ and then decremented from $f(n)$ to $p-1$. Furthermore, both auxiliary counters $k+1$ and $k+2$ are incremented from 0 to $f(n)$ and then again decremented from $f(n)$ to 0, so they reverse only at 0 or $f(n)$. We conclude that M satisfies the required conditions.

A reversal from decrementing to incrementing can be handled in an analogous way. The only difference is that counter i keeps on decrementing until 0 is reached and is then incremented again. The two auxiliary counters $k+1$ and $k+2$ can be used in exactly the same way.

We assume that M reverses the input head m times and every counter reverses at most q times on some input w. Obviously, M' also reverses the input head m times on input w. Furthermore, for every reversal of a counter of M that is not done at either value 0 or value $f(n)$, M' reverses counters $k+1$ and $k+2$ twice. Hence, the two auxiliary counters reverse at most $2kq$ times on input w. □

We are now ready to show how $\text{CDFA}_n(k)$ can be simulated by NBMCA such that the number of input head reversals is preserved and no counter is reset.

Lemma 3.12. For every $M \in 2\text{CDFA}_n(k)$ there exists an $M' \in \text{NBMCA}(k+2)$ with $L(M) = L(M')$. Furthermore, M' resets none of its counters and if M reverses the input head m times on some input w, then M' reverses the input head $m+2$ times on w.

Proof. We show how to define M' so that it simulates M. First, we transform M into an equivalent $2\text{CDFA}_n(k+2)$ $\widehat{M}$ that reverses counters only at value 0 or value n. By Lemma 3.11 we know that such an automaton exists and, furthermore, on any input w, $\widehat{M}$ reverses the input head as many times as M does on input w. We shall now show how this automaton $\widehat{M}$ is simulated by M'. At the beginning of a computation, M' checks whether or not *all* modulo counters are initialised with a counter bound of n, where n is the current input length. This can be done by moving the input head from the left endmarker to the right endmarker and simultaneously incrementing all $k+2$ modulo counters. After that, the input head needs to be moved back to the left end of the input, so M' makes two additional input head reversals. Next, we show how M' simulates a step of $\widehat{M}$. The input head of M' is used in exactly the same way $\widehat{M}$ uses its input head, and every counter of M' simulates one counter of $\widehat{M}$. Since the modulo counters of M' have bound n, we can simulate both, an incrementing sequence from 0 to n and a decrementing sequence from n to 0 of a counter of $\widehat{M}$ by an incrementing cycle from 0 to n of a modulo counter of M'. However, we need to keep track in the finite state control on whether the counters of M' simulate an incrementing or a decrementing cycle of a counter of $\widehat{M}$ at the moment, i. e., M' keeps track on whether reaching the counter bound with some modulo counter is interpreted as the situation that the corresponding counter of $\widehat{M}$ reaches 0 or it reaches n.

From our considerations above, we can conclude that, on any input, M' reverses the input head exactly two times more often than M. Furthermore, none of the modulo counters is reset. □

So far, we have demonstrated that NBMCA can simulate $2\mathrm{CDFA}_n(k)$ in such a way that the number of input head reversals is preserved and the counters are not reset. However, in order to transfer the undecidability results stated by Theorem 3.10 from $(1,\infty)$-REV-CDFA(1) to (m,∞,l)-REV-NBMCA, we need to find a way to simulate deterministic two-way counter automata with only one counter and *without* a bound on the counter values by NBMCA. Furthermore, this simulation should preserve the number of input head reversals and none of the counters of the NBMCA should be reset. We shall implicitly show that such a simulation is possible, by showing how 2CDFA(1) can be simulated by $2\mathrm{CDFA}_n(1)$. To this end, we first need the following technical lemma, which, informally speaking, states that in accepting computations, a 2CDFA(1) cannot reach arbitrarily large values with its counter.[2]

Lemma 3.13. Let M be an arbitrary 2CDFA(1) with n states. During the computation of M on an arbitrary $w \in L(M)$, the counter never reaches a value $m \geq 2\,n\,(|w|+2)$.

Proof. Without loss of generality, we may assume that M stops as soon as an accepting state is reached. Let Q be the set of states of M, let F be the set of accepting states and let $C_{M,w} := \{[q,h,d] \mid q \in Q, 0 \leq h \leq |w|+1, d \in \mathbb{N}\}$ be the set of possible configurations of M on input $w \in L(M)$. Furthermore, for every $[q,h,d] \in C_{M,w}$ let the mapping g be defined by

$$g([q,h,d]) := \begin{cases} [q,h,0] & \text{if } d = 0, \\ [q,h,1] & \text{else.} \end{cases}$$

In order to prove the statement of the lemma, we assume to the contrary that the counter of M reaches a value $m \geq 2|Q|(|w|+2)$ in the computation of M on w. This implies that in the computation of M on w there must be a sequence of at least $m+1$ configurations such that the first of these configurations has a counter value of 0, the last configuration has a counter value of m and all the configurations in between have a counter value strictly between 0 and m. More precisely, there exists a sequence of configurations $c_1, c_2, \ldots, c_{m'}$, $m' > m$, where $c_i := [q_i, h_i, d_i]$, $1 \leq i \leq m'$, $d_1 = 0$, $d_{m'} = m$ and $1 \leq d_i \leq m-1$, $2 \leq i \leq m'-1$. Furthermore, $q_i \notin F$, $1 \leq i < m'$, as otherwise the

[2] We wish to point out that [2] contains an analogous result with respect to deterministic 1-reversal one-way counter automata.

automaton stops in a configuration c_i, $1 \leq i < m'$. As $|\{g(c) \mid c \in C_{M,w}\}| = 2|Q|(|w|+2)$ and $m' > 2|Q|(|w|+2)$, we can conclude that there exist j, j', $1 \leq j < j' \leq m'$, with $g(c_j) = g(c_{j'})$. Since M is deterministic, this implies that in the computation for c_j and $c_{j'}$ the transition function applies the same transition; thus, the computation may enter a loop. We consider two possible cases depending on the counter values d_j and $d_{j'}$ of the configuration c_j and $c_{j'}$:

- $d_j \leq d_{j'}$: This implies that M has entered an infinite loop and all states in this loop are non-accepting. Thus, $w \notin L(M)$, which is a contradiction.

- $d_j > d_{j'}$: This implies that M decrements the counter to value 0 before reaching value m, which contradicts the fact that $1 \leq d_i \leq m-1$, $2 \leq i \leq m'-1$, and $d_{m'} = m$.

Consequently, the assumption that the counter reaches a value $m \geq 2|Q|(|w|+2)$ implies $w \notin L(M)$, which clearly contradicts $w \in L(M)$. □

This result can now be used to simulate 2CDFA(1) by 2CDFA$_n$(1). Intuitively, this is done by counting the constant part of $2|Q|(|w|+2)$ with the finite state control and only the part that depends on the current input by the actual counter, which can then be bounded by the current input length.

Lemma 3.14. For every $M \in$ 2CDFA(1), there exists an $M' \in$ 2CDFA$_n$(1), such that $L(M) = L(M')$ and if M reverses the input head m times on some input w, then M' reverses the input head m times on w.

P r o o f. Let Q_M be the set of states of M. By Lemma 3.13, we can assume that, in every accepting computation of M on any input w, the counter value does not reach the value $2\,|Q_M|\,(|w|+2)$. This implies that there exists a constant c_M depending on Q_M such that in every accepting computation of M on any w the counter value does not reach the value $c_M\,|w|$. Hence, we can construct an $M' \in$ 2CDFA(1) with a set of states $Q_{M'} := \{q_i \mid q \in Q_M, 1 \leq i \leq c_M\}$ and $L(M) = L(M')$. More precisely, whenever the counter of M is in state q and has a counter value of $k\,c_M + k'$, $k, k' \in \mathbb{N}$, then M' is in state $q_{k'}$ and has a counter value of k. In other words, M' uses the subscript in the states as a counter bounded by c_M and increments the actual counter only every c_M increments. This implies that in every accepting computation of M' on some w the counter value does not reach the value $|w|$. Hence, its counter is bounded by $|w|$. Consequently, we can simulate M' by an $M'' \in$ 2CDFA$_n$(1): On any input w, we simulate M' by M'' and abort the current computation in a non-accepting state in the case that the counter reaches a value of $|w|$. □

Finally, we can state that all the problems considered in Theorem 3.10 are also undecidable even for NBMCA with only 3 counters that do not reset any of the counters and perform at most 3 input head reversals in every accepting computation:

Theorem 3.15. The emptiness, infiniteness, universe, equivalence, inclusion and disjointness problems are undecidable for $(3, \infty, 0)$-REV-NBMCA(3).

P r o o f. Let $M \in (1,\infty)$-REV-CDFA(1). By Lemma 3.14, we can conclude that there exists an $M' \in (1,\infty)$-REV-CDFA$_n$(1) with $L(M) = L(M')$. Furthermore, by Lemma 3.12, there exists an $M'' \in (3,\infty,0)$-REV-NBMCA(3) with $L(M') = L(M'')$. Hence, $\mathcal{L}((1,\infty)$-REV-CDFA$(1)) \subseteq \mathcal{L}((3,\infty,0)$-REV-NBMCA$(3))$, which, with Theorem 3.10, implies the statement of the theorem. □

4. NBMCA WITHOUT STATES

In this section, we consider NBMCA without states. Stateless versions of automata have first been considered by Yang et al. [18], where they are compared to P Systems. This comparison is appropriate, as it is a feature of P Systems that they are not controlled by a finite state control. Ibarra et al. [9] and Frisco and Ibarra [3] mainly investigate stateless multi-head automata, whereas Ibarra and Eğecioğlu [8] consider stateless counter machines. In Kutrib et al. [12] stateless restarting automata are studied and stateless versions of multi-head automata with pebbles have been recently investigated by Kutrib et al. in [11]. Intuitively, the lack of states results in a substantial loss of possible control mechanisms for the automaton. For instance, the task to recognise exactly the singleton language $\{a^k\}$ for some fixed constant k, which is easily done by any automaton with states, suddenly seems difficult, as we somehow need to count k symbols without using any states. In [9] an example of a stateless multi-head automaton that recognises $\{a^k\}$ can be found.

We now define stateless NBMCA and we shall illustrate this model with an example. Then we investigate the question of whether or not stateless NBMCA can simulate NBMCA with a finite state control. In the following Section 4.1, we further restrict the model of stateless NBMCA in order to investigate a more general question in automata theory regarding limited nondeterminism.

A *stateless* NBMCA (SL-NBMCA for short) can be regarded as an NBMCA with only one internal state that is never changed. Hence, the component referring to the state is removed from the transition function and transitions do not depend anymore on the state. As a result, the acceptance of inputs by accepting state is not possible anymore. So for stateless NBMCA we define the input to be accepted by a special accepting transition, i. e., the transition that does not change the configuration of the automaton anymore. On the other hand, if the automaton enters a configuration for which no transition is defined, then the input is rejected and the same happens if an infinite loop is entered. For example, $(b, s_1, \dots, s_k) \rightarrow (r, d_1, \dots, d_k)$ is a possible transition for an SL-NBMCA(k) and $(b, s_1, \dots, s_k) \rightarrow (0, 0, 0, \dots, 0)$ is an accepting transition. For the sake of convenience we shall denote an accepting transition by $(b, s_1, \dots, s_k) \rightarrow \mathbf{0}$. An SL-NBMCA($k$) can be given as a tuple (k, Σ, δ) comprising the number of counters, the input alphabet and the transition function.

As already mentioned, in Ibarra et al. [9] an example of a stateless multi-head automaton that recognises $\{a^k\}$ can be found. We shall now consider a similar example with respect to SL-NBMCA, i. e., we show how the following languages can be recognised by SL-NBMCA.

Definition 4.1. For every $k \in \mathbb{N}$, let $S_k := \{a^k, \varepsilon\}$.

We introduce an SL-NBMCA(5) that recognises S_3.

Definition 4.2. Let $M_{S_3} := (5, \{a\}, \delta) \in \text{SL-NBMCA}(5)$, where δ is defined by

1. $(¢, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{1}, \mathtt{1}, \mathtt{1}, \mathtt{1}, \mathtt{r})$,
2. $(\mathtt{a}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_0}) \rightarrow_\delta (\mathtt{-1}, \mathtt{1}, \mathtt{1}, \mathtt{1}, \mathtt{1}, \mathtt{1})$,
3. $(¢, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{1}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0})$,
4. $(\mathtt{a}, \mathtt{t_1}, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{0}, \mathtt{1}, \mathtt{0}, \mathtt{0}, \mathtt{0})$,
5. $(\mathtt{a}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_0}, \mathtt{t_0}, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{0}, \mathtt{0}, \mathtt{1}, \mathtt{0}, \mathtt{0})$,
6. $(\mathtt{a}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_0}, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{1}, \mathtt{1})$,
7. $(\$, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_1}, \mathtt{t_0}) \rightarrow_\delta \mathbf{0}$.

Proposition 4.3. $L(M_{S_3}) = S_3$.

We shall only give an intuitive explanation of how M_{S_3} recognises S_3 and the details are left to the reader. The first 4 counters are used to count the 4 steps that are necessary to move the input head from the left to the right endmarker in case that aaa is the input. However, this is possible only if all counter bounds of these counters are 1. So initially M_{S_3} checks whether or not all the first 4 counters are initialised with counter bounds of 1. To this end, the input head is moved one step to the right while the first 4 counters are incremented. After that it is checked whether all these counters have reached their bounds after this increment and then the input head is moved back to the left endmarker. Then, M_{S_3} uses the counters in order to count the occurrences of symbols a on the input tape. Hence, the computations of M_{S_3} comprise two parts: a first part of checking whether or not the right counter bounds are guessed and a second part where the symbols on the tape are counted. However, since there are no states, the automaton is not able to distinguish between these two parts. This is mainly due to the fact that in the first part we move the input head and, thus, necessarily scan an input symbol. So it is possible that the counter bounds are initialised in a way such that in the first part of the computation M_{S_3} accidentally enters a configuration that is also reached in the second part. In order to separate these two parts of the computation, we need an additional counter.

By generalising Definition 4.2 and Proposition 4.3 it can be shown that for every $k \in \mathbb{N}$, there exists an $M_{S_k} \in \text{SL-NBMCA}(k+2)$ with $L(M_{S_k}) = S_k$:

Proposition 4.4. For every $k \in \mathbb{N}$, $S_k \in \mathcal{L}(\text{SL-NBMCA}(k+2))$.

The question of whether or not states are indispensable for a model, i. e., whether it is possible to simulate automata by their stateless counterparts, is probably the most fundamental question about stateless automata. Obviously, the models of DFA and NFA degenerate if the number of states is restricted to at most one. On the other hand, we know that the power of nondeterministic PDA is not dependent on the number of states and, thus, every PDA can be turned into a PDA with only a single state (see, e. g., Hopcroft and Ullman [6]). Intuitively, the pushdown store compensates for the loss of states. Regarding *deterministic* pushdown automata, we find a different situation; here, the expressive power strictly increases with the number of states (see, e. g., Harrison [4]).

Our first main result shows that every NBMCA with states can be turned into an equivalent one without states. Hence, the loss of the finite state control does not lead to a reduced expressive power of the model.

Theorem 4.5. For every $M \in \text{NBMCA}(k)$, $k \in \mathbb{N}$, with a set of states Q, there exists an $M' \in \text{SL-NBMCA}(k + \lceil \log(|Q| + 1) \rceil + 2)$ with $L(M) = L(M')$.

A formal proof of Theorem 4.5 can be found in [17]. Here, we shall only outline its main ideas. It has been shown in [9] that two-way stateless multi-head automata can easily simulate a finite state control by using $\log(n)$ additional input heads, where n is the number of states, and interpreting these input heads as a binary number, which represents the index of a state. Regarding SL-NBMCA it is not completely obvious how this idea of simulating states can be applied. This is mainly due to the fact that the counters of an SL-NBMCA can have any counter bound, and it is not possible to control these bounds. So it seems more difficult to use a counter as some sort of a bit that can be flipped between 0 and 1. However, it is possible to define an SL-NBMCA such that in an accepting computation certain counters must be initialised with a counter bound of 1. Informally speaking, this is done by simply using the input head to check whether or not certain counters have counter bounds of 1. In order to do this, the input head has to be moved in the input; hence, as we cannot use any states, the problem is how to separate this initial phase from the main part of the computation.

4.1. SL-NBMCA **with a Bounded Number of Resets**

In this section, we use the model of stateless NBMCA in order to investigate a more general question in automata theory regarding limited nondeterminism. Usually, the nondeterminism is mainly controlled by the finite state control, i. e., certain states allow nondeterministic steps whereas others enforce a deterministic transition. Hence, nondeterminism can be switched on and off and, thus, it is a resource the automaton may use, but it is not forced to. These considerations suggest that the finite state control plays an important role regarding restricted nondeterminism and it is not obvious what consequences, in this regard, an abolishment of the finite state control may have. In the following we try to answer this question by employing SL-NBMCA. As shown in the previous section, if we allow an unbounded number of modulo counters, a finite state control can be simulated and, thus, nondeterminism can be controlled in the usual way. Therefore we consider SL-NBMCA with only one counter and, furthermore, we assume the input head to operate in a one-way manner, i. e., for every transition $(b, x) \rightarrow_\delta (y, z)$, we have $y \in \{\mathtt{0}, \mathtt{1}\}$. In order to restrict the nondeterminism of the model, we simply limit the number of possible resets for the modulo counter. More precisely, in any computation the first k applications of a transition of form $(b, x) \rightarrow (y, \mathtt{r})$, $b \in \Sigma$, $x \in \{\mathtt{t}_0, \mathtt{t}_1\}$, $y \in \{\mathtt{0}, \mathtt{1}\}$, reset the counter in accordance with the definition of NBMCA. Every further application of a transition of that form simply ignores the counter, i. e., if in a computation a transition $(a, x) \rightarrow (y, \mathtt{r})$ is applied after the counter has already been reset for at least k times, then this transition is interpreted as $(a, x) \rightarrow (y, \mathtt{0})$. We shall refer to this model by $\text{1SL-NBMCA}_k(1)$, where k stands for the number of possible resets.

This way of restricting automata is unusual compared to the common restrictions that are found in the literature. This can be illustrated by recalling input head reversal bounded automata as an example (see, e. g., Ibarra [7] and Section 3.3): an input head reversal bounded automaton is an automaton that recognises each word of a language in such a way that the number of input head reversals is bounded. There is no need to require the input head reversals to be bounded in the non-accepting computations as well, as this does not constitute a further restriction. This is due to the fact that we can always use the finite state control to count the number of input head reversals in order to interrupt a computation in a non-accepting state as soon as the bound of input head reversals is exceeded. However, regarding stateless automata this is not possible anymore, and there seems to be a difference whether a restriction is defined for all possible computations or only for the accepting ones. Our definition of bounds on the number of resets introduced above avoids these problems by slightly changing the model itself, i. e., in every computation it loses the ability to reset the counter after k resets.

We recall that in a computation of an NBMCA, the counter bounds are already nondeterministically initialised. Hence, in a computation of a 1SL-NBMCA$_k(1)$ the counter can have $k+1$ distinct counter bounds. Since the input head of 1SL-NBMCA$_k(1)$ is a one-way input head, we require all accepting transitions to be of form $(\$, x) \rightarrow \mathbf{0}$, $x \in \{\mathtt{t}_0, \mathtt{t}_1\}$.

The main question is whether or not the classes $\mathcal{L}$(1SL-NBMCA$_k(1)$), $k \in \mathbb{N}$, describe a hierarchy with respect to k. First, for every $k \in \mathbb{N}$, we separate the classes $\mathcal{L}$(1SL-NBMCA$_k(1)$) and $\mathcal{L}$(1SL-NBMCA$_{k+1}(1)$) by identifying a language L that can be recognised by an 1SL-NBMCA$_{k+1}(1)$, but by no $M \in$ 1SL-NBMCA$_k(1)$. The words of such L are basically concatenations of $k+2$ words u_i, $1 \leq i \leq k+2$, where each u_i comprises unary factors of the same length n_i, $1 \leq i \leq k+2$. A 1SL-NBMCA$_{k+1}(1)$ can recognise this language by using the initial counter bound and the $k+1$ counter bounds guessed in the computation in order to check the unary factors for equality. Intuitively, a 1SL-NBMCA$_k(1)$, which can only use at most $k+1$ different counter bounds in any computation, is not able to recognise this language, since it is possible that $n_i \neq n_{i+1}$; thus, at least $k+2$ distinct counter bounds are required. Next, we shall formally define these languages and use them in the above illustrated way in order to separate the classes 1SL-NBMCA$_k(1)$, $k \in \mathbb{N}$.

In the remainder of this paper we exclusively consider languages and automata defined over the alphabet $\Sigma := \{a, \#_1, \#_2\}$. Next, for every $k \in \mathbb{N}$, we define a language over Σ that shall then be shown to be in $\mathcal{L}$(1SL-NBMCA$_k(1)$) but not in $\mathcal{L}$(1SL-NBMCA$_{k-1}(1)$).

Definition 4.6. For every $n \in \mathbb{N}_0$ let $\widetilde{\mathbf{L}}_n := \{a^n\} \cdot \{\#_1 \cdot a^n\}^*$, and let $\widetilde{\mathbf{L}} := \bigcup_{n \in \mathbb{N}_0} \widetilde{\mathbf{L}}_n$. Furthermore, for every $k \in \mathbb{N}$, let

$$\mathbf{L}_{k,1} := \{u_1 \#_2 u_2 \#_2 \cdots \#_2 u_{k'} \mid u_i \in \widetilde{\mathbf{L}}, 1 \leq i \leq k' \leq k\},$$

$$\mathbf{L}_{k,2} := \{u_1 \#_2 u_2 \#_2 \cdots \#_2 u_{k-1} \mid u_i \in \widetilde{\mathbf{L}}, 1 \leq i \leq k-1\} \left((\#_2 \widetilde{\mathbf{L}}_0)^+ \cup \bigcup_{n \geq 1} (\#_2 \#_1 \widetilde{\mathbf{L}}_n)^+ \right)$$

and let $\mathbf{L}_k := \mathbf{L}_{k,1} \cup \mathbf{L}_{k,2}$.

Thus, the words of language $\widetilde{\mathbf{L}}$ consist of concatenations of factors over $\{a\}$ of the same length that are separated by occurrences of $\#_1$. The words of the language $\mathbf{L}_k$ are basically concatenations of words in $\widetilde{\mathbf{L}}$ separated by occurrences of $\#_2$. However, in a word from $\mathbf{L}_k$, only the first k of these elements from $\widetilde{\mathbf{L}}$ can be arbitrarily chosen, for all the others the length of the factors over $\{a\}$ must be the same as for the k^{th} word, with the only difference that they start with an additional occurrence of $\#_1$. For example,

$$aa \cdot \#_1 \cdot aa \cdot \#_2 \cdot \#_2 \cdot aaa \cdot \#_1 \cdot aaa \cdot \#_1 \cdot aaa \cdot \#_2 \cdot \#_1 \cdot aaa \cdot \#_1 \cdot aaa \in \mathbf{L}_3\,,$$
$$aaaaaa \cdot \#_2 \cdot a \cdot \#_1 \cdot a \cdot \#_1 \cdot a \cdot \#_2 \cdot aaa \cdot \#_1 \cdot aaa \cdot \#_2 \cdot \#_2 \cdot \#_1 \cdot \#_1 \cdot \#_1 \in \mathbf{L}_4\,,$$
$$aaaaaaaa \cdot \#_1 \cdot aaaaaaaa \cdot \#_1 \cdot aaaaaaaa \in \mathbf{L}_6\,.$$

For every $k \in \mathbb{N}$, we now define a 1SL-NBMCA$_{k-1}(1)$ that recognises exactly the language $\mathbf{L}_k$.

Definition 4.7. Let $M_{\mathbf{L}} := (1, \{a, \#_1, \#_2\}, \delta) \in$ SL-NBMCA(1), where

$$\delta := \{(\text{¢}, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{0}), (\text{¢}, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{0}), (a, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{1}),$$
$$(\#_1, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{1}), (\#_2, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{r}), (\$, \mathtt{t_1}) \rightarrow_\delta \mathbf{0}\}\,.$$

For every $k \in \mathbb{N}$, let $M_{\mathbf{L}_k}$ be $M_{\mathbf{L}}$ interpreted as a 1SL-NBMCA$_{k-1}(1)$.

The following considerations explain why $M_{\mathbf{L}_k}$ recognises $\mathbf{L}_k$: $M_{\mathbf{L}_k}$ uses its counter to count the occurrences of a on the input tape. Whenever an occurrence of $\#_1$ is scanned, the counter must have reached its counter bound, which then implies that the length of the factor over $\{a\}$ corresponds to the counter bound. When an occurrence of $\#_2$ is scanned, the counter message must be $\mathtt{t_1}$ as well. Furthermore, in case that the input is a word from $\mathbf{L}_k$, a new sequence of possibly different factors over $\{a\}$ follows. Thus, the counter is reset in order to guess a new counter bound. When all $k-1$ resets have been used, the counter bound does not change anymore; hence, the remaining factors over $\{a\}$ must all have the same length. We note that $k-1$ resets are sufficient as the counter is already nondeterministically initialised.

Theorem 4.8. For every $k \in \mathbb{N}$, $\mathbf{L}_k \in \mathcal{L}(\text{1SL-NBMCA}_{k-1}(1))$.

A formal proof of Theorem 4.8 is omitted and can be found in [17].

As described above, our next goal is to state that $\mathbf{L}_k$ cannot be accepted by any 1SL-NBMCA$_{k-2}(1)$. To this end we first observe that for every $M \in$ 1SL-NBMCA$_k(1)$, $k \in \mathbb{N}$, that accepts a language $\mathbf{L}_{k'}$, a certain property related to the fact that, by definition, a word $w \in \mathbf{L}_{k'}$ can have $k'-1$ factors of form $c \cdot a^n \cdot \#_2 \cdot a^{n'} \cdot c'$, $n, n' \in \mathbb{N}$, $c \neq a \neq c'$, must be satisfied. The next lemma states that M must reset its counter at least once in the process of moving the input head over any such factor.

Lemma 4.9. Let $k, k' \in \mathbb{N}$, $k' \geq 2$ and $M \in$ 1SL-NBMCA$_k(1)$ with $L(M) = \mathbf{L}_{k'}$. Let furthermore $w := u_1 \cdot \#_2 \cdot u_2 \cdot \#_2 \cdot \cdots \cdot \#_2 \cdot u_{k'}$ with $|u_i|_{\#_1} \geq 1$, $1 \leq i \leq k'$, such that $u_i \in \widetilde{\mathbf{L}}_{n_i}$ and $n_i \geq 2k+1$, $1 \leq i \leq k'$, and $n_i \neq n_{i+1}$, $1 \leq i \leq k'-1$. In an accepting computation of M on w, for every j, $1 \leq j < k'$, M resets its counter while scanning $\#_1 a^{n_j} \#_2 a^{n_{j+1}} \#_1$.

P r o o f. Let δ be the transition function of M and let $C := (c_1, c_2, \ldots, c_m)$ be an arbitrary accepting computation of M on w. We shall prove the statement of the lemma by first proving two claims establishing certain properties of M, a 1SL-NBMCA$_k(1)$ that recognises $\mathbf{L}_{k'}$. The first claim concerns the way how M scans occurrences of a.

Claim (1): In C, the transitions

T$_1$ $(a, \mathtt{t}_1) \rightarrow_\delta (\mathtt{0}, \mathtt{1})$,

T$_2$ $(a, \mathtt{t}_1) \rightarrow_\delta (\mathtt{1}, \mathtt{0})$,

T$_3$ $(a, \mathtt{t}_1) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$,

T$_4$ $(a, \mathtt{t}_0) \rightarrow_\delta (\mathtt{0}, \mathtt{1})$,

T$_5$ $(a, \mathtt{t}_0) \rightarrow_\delta (\mathtt{1}, \mathtt{0})$,

are not applied and the transition $(a, \mathtt{t}_0) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$ is applied.

P r o o f. (*Claim* (1)) We shall first show that none of the transitions T$_1$ to T$_5$ is applied in C. To this end, we observe that since $n_i \geq 2k+1$, $1 \leq i \leq k'$, and there are at most k resets possible in C, we can conclude that there are at least two consecutive occurrences of a in w such that only non-resetting transitions are performed while these occurrences are scanned by the input head. Let us denote these positions by $\widehat{p}$ and $\widehat{p}+1$. Next, we show that it is not possible that any of the transitions T$_1$ to T$_5$ apply when the input head scans position $\widehat{p}$. To this end, we assume to the contrary that one of these transitions is applied in configuration $[\widehat{p}, (p, q)]$, $p, q \in \mathbb{N}$, $p \leq q$, and then show that a word is accepted by M that is not an element of $\mathbf{L}_{k'}$, which is a contradiction.

If transitions T$_2$ or T$_5$ apply in configuration $[\widehat{p}, (p, q)]$, then the input head is moved over the occurrence of a at position $\widehat{p}$ without changing the counter value. Thus, the counter message does not change. This directly implies that the word $w[1, \widehat{p}] \cdot a \cdot w[\widehat{p}+1, -] \notin \mathbf{L}_{k'}$ is accepted by M as well, which is a contradiction.

If transition T$_4$ applies in configuration $[\widehat{p}, (p, q)]$, then the transition $[\widehat{p}, (p+1, q)]$ is reached and, thus, T$_4$ is repeated until the counter reaches its bound, i. e., M enters configuration $[\widehat{p}, (q, q)]$. Since the computation is accepting, a next transition must be defined and this transition must move the head as otherwise no transition is defined that moves the head while an occurrence of a is scanned. Furthermore, by assumption, this transition is non-resetting. Since we have already ruled out transition T$_2$, the only possible next transition is T$_3$. This implies that configuration $[\widehat{p}+1, (0, q)]$ is reached. Consequently, the word $w[1, \widehat{p}] \cdot a \cdot w[\widehat{p}+1, -] \notin \mathbf{L}_{k'}$ is accepted as well and, again, a contradiction is obtained.

Next we assume that transition T$_1$ applies in configuration $[\widehat{p}, (p, q)]$. If q, the current counter bound, equals 0, then the counter message cannot change and the automaton is in an infinite loop, which contradicts the fact that the computation C is accepting. So we assume that $q \geq 1$ which implies that configuration $[\widehat{p}, (0, q)]$ is reached by applying T$_1$. Since C is accepting a next transition must be applicable that is non-resetting and moves the input head. We have already ruled out transition T$_5$. Thus, the only possible next transition is $(a, \mathtt{t}_0) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$ and therefore the configuration $[\widehat{p}+1, (1, q)]$

is reached. We observe that this implies that $w[1,\widehat{p}] \cdot a^q \cdot w[\widehat{p}+1,-] \notin \mathbf{L}_{k'}$ is accepted by M as well, which is a contradiction.

It remains to consider the case that T_3 is applied in configuration $[\widehat{p},(p,q)]$. If $q = 0$, then the counter message does not change by applying transition T_3. This implies that the effect of transition T_3 is the same as of transition T_2. Hence, we can show in a similar way as before that $w[1,\widehat{p}] \cdot a \cdot w[\widehat{p}+1,-] \notin \mathbf{L}_{k'}$ is accepted by M. Since this is a contradiction, we conclude that $q \geq 1$ and observe that configuration $[\widehat{p}+1,(0,q)]$ is reached by applying T_3. Again, as C is accepting, a next configuration must be defined. We recall that we assume that no resetting transition is applied while the input head scans positions $\widehat{p}$ and $\widehat{p}+1$. Therefore, the next transition is non-resetting, but does not necessarily move the input head. The only possible transitions of that kind are T_4, T_5 and $(a,\mathtt{t_0}) \rightarrow_\delta (\mathtt{1},\mathtt{1})$. Since $w[\widehat{p}+1] = a$, we can conclude that T_4 and T_5 cannot be applied in configuration $[\widehat{p}+1,(0,q)]$ in exactly the same way as we have already shown above that T_4 and T_5 cannot be applied in configuration $[\widehat{p},(p,q)]$. Therefore the only possible transition left is transition $(a,\mathtt{t_0}) \rightarrow_\delta (\mathtt{1},\mathtt{1})$. Similarly as before, we can conclude that in this case M accepts $w[1,\widehat{p}] \cdot a^{q+1} \cdot w[\widehat{p}+1,-] \notin \mathbf{L}_{k'}$ and, thus, we obtain a contradiction.

We conclude that the transition $(a,\mathtt{t_0}) \rightarrow_\delta (\mathtt{1},\mathtt{1})$ is the only possible transition that can be applied when configuration $[\widehat{p},(p,q)]$ is reached. This proves Claim (1). $\square$

The next claim states that M cannot move the input head over an occurrence of $\#_1$ or $\#_2$ by a non-resetting transition if the counter message is $\mathtt{t_0}$.

Claim (2): For every $b \in \{\#_1, \#_2\}$, the transitions

T_6 $(b,\mathtt{t_0}) \rightarrow_\delta (\mathtt{1},\mathtt{0})$,

T_7 $(b,\mathtt{t_0}) \rightarrow_\delta (\mathtt{1},\mathtt{1})$

are not defined.

Proof. (*Claim* (2)) We assume to the contrary that, for some $b \in \{\#_1, \#_2\}$, transition T_6 or T_7 is defined and use our accepting computation C of M on w to obtain a contradiction, i. e., we show that M accepts a word that is not an element of $\mathbf{L}_{k'}$.

As we have already shown in Claim (1), there must be a position $\widehat{p}$, $1 \leq \widehat{p} \leq |w|$, such that $c_i := [\widehat{p},(p,q)]$ is converted into $c_{i+1} := [\widehat{p}+1,(p+1,q)]$ by transition $(a,\mathtt{t_0}) \rightarrow_\delta (\mathtt{1},\mathtt{1})$. Furthermore, we can conclude that $p < q$.

We now assume that transition T_6 is defined and consider the input $w' := w[1,\widehat{p}-1] \cdot b \cdot w[\widehat{p},-]$, i. e., we insert an occurrence of b to the left of the occurrence of a at position $\widehat{p}$. It is not possible that in configuration c_{i-1} the input head is located at position $\widehat{p}$ as well, as this implies the application of a transition other than $(a,\mathtt{t_0}) \rightarrow_\delta (\mathtt{1},\mathtt{1})$ which, by Claim (1), must be resetting. Hence, there is a computation of M on w' that is identical to C up to the first i elements. So configuration $[\widehat{p},(p,q)]$ is reached and, as $w'[\widehat{p}] = b$ and $p < q$, T_6 applies and changes M into configuration $[\widehat{p}+1,(p,q)]$. Now, as $w'[\widehat{p}+1,-] = w[\widehat{p},-]$, it is possible that the computation terminates with the last $m-i$ elements of C, where the first component of each configuration has increased by 1. Hence, w' is accepted by M.

Next, we assume that transition T_7 is defined and consider the input $w'' := w[1, \widehat{p} - 1] \cdot b \cdot w[\widehat{p}+1, -]$, i. e., we substitute the occurrence of a at position $\widehat{p}$ by an occurrence of b. There is a computation of M on w'' that is identical to C up to the first i elements. So configuration $[\widehat{p}, (p, q)]$ is reached and, as $w''[\widehat{p}] = b$ and $p < q$, T_7 applies and changes M into configuration $[\widehat{p}+1, (p+1, q)]$. Now, as $w''[\widehat{p}+1, -] = w[\widehat{p}+1, -]$, it is possible that the computation terminates with the last $m - (i + 1)$ elements of C. Hence, w'' is accepted by M.

In order to conclude the proof, it remains to show that $w' \notin \mathbf{L}_{k'}$ and $w'' \notin \mathbf{L}_{k'}$ for every $b \in \{\#_1, \#_2\}$. We recall that w' is obtained from w by inserting an occurrence of b to the left of an occurrence of a and w'' is obtained from w by substituting an occurrence of a by an occurrence of b. If $b = \#_1$, then there exists a factor $c \cdot a^n \cdot \#_1 \cdot a^{n'} \cdot c'$ in w' (or w''), where $n \neq n'$ and $c \neq a \neq c'$; hence $w' \notin \mathbf{L}_{k'}$ (or $w'' \notin \mathbf{L}_{k'}$, respectively). If $b = \#_2$ and $\widehat{p}$ is such that $w[\widehat{p} - 1] \notin \{¢, \#_2\}$, then there also exists a factor $c \cdot a^n \cdot \#_1 \cdot a^{n'} \cdot c'$ in w' (or w'', respectively), where $n \neq n'$ and $c \neq a \neq c'$. Thus, $w' \notin \mathbf{L}_{k'}$ (or $w'' \notin \mathbf{L}_{k'}$, respectively).

It remains to consider the case where $w[\widehat{p}-1] \in \{¢, \#_2\}$ and $b = \#_2$. First, we observe that in this case w'' must have a factor $c \cdot \#_2 \cdot a^{n-1} \cdot \#_1 \cdot a^n \cdot c'$, where $c \in \{¢, \#_2\}$ and $c \neq a$. Consequently, $w'' \notin \mathbf{L}_{k'}$. Regarding w', we do not substitute an occurrence of a by an occurrence of $\#_2$, but we insert it to the left of the occurrence of a at position $\widehat{p}$. So if $w[\widehat{p} - 1] \in \{¢, \#_2\}$, then it is possible that $w' \in \mathbf{L}_{k'}$. However, we can show that there must exist a position $\widehat{p}'$, $1 \leq \widehat{p}' \leq |w|$, such that $w[\widehat{p}' - 1] \notin \{¢, \#_2\}$ and in the computation C a configuration $[\widehat{p}', (p, q)]$ is changed into $[\widehat{p}' + 1, (p + 1, q)]$ by transition $(a, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$. To this end, we assume that there exists no $\widehat{p}'$, $1 \leq \widehat{p}' \leq |w|$, such that $w[\widehat{p}' - 1] \neq \{¢, \#_2\}$ and in the computation C a configuration $[\widehat{p}', (p, q)]$ is changed into $[\widehat{p}' + 1, (p + 1, q)]$ by transition $(a, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$. This implies that the input head is moved over all the occurrences of a at a position $\widehat{p}''$ with $w[\widehat{p}'' - 1] = a$ by a transition of form $(a, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, x)$, and by Claim (1) of this lemma we can conclude that $x = \mathtt{r}$. Since there are $n_1 - 1$ such occurrences of a that require an application of the transition $(a, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{r})$ in the prefix a^{n_1} of w and since $n_1 \geq 2k + 1$, we can conclude that all possible k resets are performed in the process of moving the input head over the prefix a^{n_1} of w. Furthermore, after these k applications of $(a, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{r})$, the transition $(a, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{r})$ will be interpreted as $(a, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{0})$ for all further occurrences of a in the prefix a^{n_1}. This implies that in the computation C the transition $(a, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{0})$ is applied which is a contradiction according to Claim (1). This shows that there must exist a position $\widehat{p}'$, $1 \leq \widehat{p}' \leq |w|$, such that $w[\widehat{p}' - 1] \notin \{¢, \#_2\}$ and in the computation C a configuration $[\widehat{p}', (p, q)]$ is changed into $[\widehat{p}' + 1, (p + 1, q)]$ by transition $(a, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$. Consequently, we can construct a w' with respect to that position $\widehat{p}'$ in the way described above and there exists an accepting computation of M on w', but $w' \notin \mathbf{L}_{k'}$. This concludes the proof of Claim (2). $\square$

We can now prove the statement of the lemma. To this end, let j, j', $1 \leq j < j' \leq |w|$, be such that $w[j, j'] = \#_1 \cdot a^{n_i} \cdot \#_2 \cdot a^{n_{i+1}} \cdot \#_1$ with $1 \leq i \leq k' - 1$ and, for some l, l', $1 \leq l < l' \leq m$, and some $p_i, q_i \in \mathbb{N}_0$, $1 \leq i \leq 4$, let

$$c_l, \ldots, c_{l'} = [j, (p_1, q_1)], [j + 1, (p_2, q_2)], \ldots, [j' - 1, (p_3, q_3)], [j', (p_4, q_4)]\,,$$

such that, for every i, $l + 1 \leq i \leq l' - 1$, c_i is converted into c_{i+1} by a transition of form $(b, x) \rightarrow_\delta (y, z)$, $b \in \Sigma$, $x \in \{\mathtt{t_0}, \mathtt{t_1}\}$, $y, z \in \{\mathtt{0}, \mathtt{1}\}$. Since the input head is moved from position j to position $j+1$, we know that the transition that converts $[j,(p_1,q_1)]$ into $[j+1,(p_2,q_2)]$ is of form $(\#_1, x) \rightarrow_\delta (\mathtt{1}, y)$. If $y \neq \mathtt{r}$, then, by Claim (2), $x = \mathtt{t_1}$ is implied and if furthermore $y = 0$, then the occurrence of a at position $j+1$ is reached with counter message $\mathtt{t_1}$. Now, using Claim (1), we can conclude that the only possible next transition must reset the counter, which contradicts our assumption. Consequently, the transition that converts $[j,(p_1,q_1)]$ into $[j+1,(p_2,q_2)]$ is either $(\#_1, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$ or a transition of form $(\#_1, x) \rightarrow_\delta (\mathtt{1}, \mathtt{r})$. We note that regardless of which of the possible transitions apply, the input head is moved one step to the right and the counter configuration changes to $(0, q_2)$. We define $v := w[j+1, j'-1] = a^{n_i} \cdot \#_2 \cdot a^{n_{i+1}}$. By Claims (1) and (2) and the assumption that the input head is moved over v without counter resets, we conclude that the input head is moved over all the occurrences of a in v by applying transition $(a, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$. So this transition applies until either the input head scans $\#_2$ or the counter message changes to $\mathtt{t_1}$. If the counter message changes to $\mathtt{t_1}$ while still an occurrence of a is scanned by the input head, then, by Claim (1), the next transition would reset the counter, which is not possible; so we can conclude that $q_2 \geq n_i$. If the input head reaches the occurrence of $\#_2$ with a counter message of $\mathtt{t_0}$, then the transition $(\#_2, \mathtt{t_0}) \rightarrow_\delta (\mathtt{0}, \mathtt{1})$ applies, since we assume that the counter is not reset and a non-resetting transition that moves the input head while $\#_2$ is scanned and the counter message is $\mathtt{t_0}$ is not possible, according to Claim (2). However, this implies that the counter is incremented without moving the input head until the counter message changes to $\mathtt{t_1}$. We conclude that the occurrence of a to the left of the occurrence $\#_2$ could be deleted and the computation would still be accepting. This is clearly a contradiction. Therefore the input head reaches $\#_2$ exactly with counter message $\mathtt{t_1}$ and, thus, $q_2 = n_i$. We have now reached the configuration where the input head scans $\#_2$ and the counter message is $\mathtt{t_1}$. If the next transition does not move the input head, then it must increment the counter, as otherwise the transition would be accepting which, by definition, is not possible. This results in the configuration where still $\#_2$ is scanned but with a counter message of $\mathtt{t_0}$. In the same way as before, by applying Claim (2), we can conclude that for such a configuration no non-resetting transition that moves the input head is defined. Hence, the automaton stops, which is a contradiction. Consequently the next transition that applies when $\#_2$ is scanned is transition $(\#_2, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, z)$. Furthermore, $z = \mathtt{1}$, as otherwise the first occurrence of a to the right of $\#_2$ is reached with counter message $\mathtt{t_1}$, which, as already shown above, is not possible. So transition $(\#_2, \mathtt{t_1}) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$ applies and then again several times transition $(a, \mathtt{t_0}) \rightarrow_\delta (\mathtt{1}, \mathtt{1})$. For the same reasons as before we can conclude that the counter message must not change to $\mathtt{t_1}$ as long as occurrences of a are scanned and; thus, $q_2 \geq n_{i+1}$. If we reach the occurrence of $\#_1$ at position j' with a counter message of $\mathtt{t_0}$, we have several possibilities. If a transition applies that does not reset the counter, then, by Claim (2), it must be $(\#_1, \mathtt{t_0}) \rightarrow_\delta (\mathtt{0}, \mathtt{1})$. On the other hand, since M is now in configuration $c_{l'}$, it is also possible that a transition of form $(\#_1, \mathtt{t_0}) \rightarrow_\delta (x, \mathtt{r})$ applies. However, for all these cases we observe that if we would delete the occurrence of a to the left of the occurrence of $\#_1$ at position j', then the changed input would still be accepted, which is a contradiction. So we conclude that the input head reaches the occurrence of $\#_1$ exactly with counter message $\mathtt{t_1}$. This implies $q_2 = n_{i+1}$ and, hence, $n_i = n_{i+1}$, which is a contradiction. This concludes the proof Lemma 4.9. □

Now we are able to show that $\mathbf{L}_k$ can be recognised by a 1SL-NBMCA$_{k-1}(1)$ (Theorem 4.8), but cannot be recognised by any 1SL-NBMCA$_{k-2}(1)$.

Theorem 4.10. For every $k \in \mathbb{N}$ with $k \geq 2$, $\mathbf{L}_k \notin \mathcal{L}(\text{1SL-NBMCA}_{k-2}(1))$.

P r o o f. We assume to the contrary that there exists an $M \in \text{1SL-NBMCA}_{k-2}(1)$ with $L(M) = \mathbf{L}_k$. Let $w := a^{n_1} \cdot \#_1 \cdot a^{n_1} \cdot \#_2 \cdot a^{n_2} \cdot \#_1 \cdot a^{n_2} \cdot \#_2 \cdot \cdots \cdot \#_2 \cdot a^{n_k} \cdot \#_1 \cdot a^{n_k}$, with $n_1, n_2, \ldots, n_k \geq 2k+1$ such that $n_i \neq n_{i+1}$ for all $1 \leq i \leq k-1$. Obviously, $w \in \mathbf{L}_k$ and w satisfies the conditions of Lemma 4.9. We observe that in w, there are $k-1$ factors of form $\#_1 \cdot a^{n_i} \cdot \#_2 \cdot a^{n_{i+1}} \cdot \#_1$, but in an accepting computation of M on w, there are at most $k-2$ resets possible. Hence, there must be an i, $1 \leq i \leq k-1$, such that the input head is moved over the factor $a^{n_i} \cdot \#_2 \cdot a^{n_{i+1}}$ without performing a reset. According to Lemma 4.9, however, this is not possible, so we obtain a contradiction. □

This proves that for every $k \in \mathbb{N}$ there exists a language that can be recognised by a 1SL-NBMCA$_k(1)$, but cannot be recognised by a 1SL-NBMCA$_{k-1}(1)$. Next, we consider the converse question, i. e., whether or not there are languages that can be recognised by a 1SL-NBMCA$_k(1)$, but cannot be recognised by any 1SL-NBMCA$_{k+1}(1)$. It turns out that the existence of such languages can be shown in a non-constructive way by applying Theorems 4.8 and 4.10 and a simple reasoning about the following subsets of the classes 1SL-NBMCA$_k(1)$, $k \in \mathbb{N}$:

Definition 4.11. For every $k \in \mathbb{N}$, let 1SL-NBMCA$^{\Sigma}_k(1)$ be the class of all automata in 1SL-NBMCA$_k(1)$ that are defined over Σ.

By definition, all $M \in \text{1SL-NBMCA}^{\Sigma}_k(1)$ have just one counter and are defined over the same alphabet Σ. Hence, for all $k \in \mathbb{N}$, the sets 1SL-NBMCA$^{\Sigma}_k(1)$ have the same constant cardinality:

Proposition 4.12. There exists a constant $\widehat{m} \in \mathbb{N}$ such that $|\,\text{1SL-NBMCA}^{\Sigma}_k(1)| = \widehat{m}$, for every $k \in \mathbb{N}$.

It is not always the case that $|\,\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_k(1))| = |\,\text{1SL-NBMCA}^{\Sigma}_k(1)|$, thus, $|\,\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_k(1))| \leq \widehat{m}$, $k \in \mathbb{N}$. However, it can be shown with little effort that there are infinitely many $k \in \mathbb{N}$, such that $|\,\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_k(1))| = |\,\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_{k+1}(1))|$ and then Theorems 4.8 and 4.10 imply that these classes are incomparable. This result can be easily extended to the classes $\mathcal{L}(\text{1SL-NBMCA}_k(1))$, $k \in \mathbb{N}$.

Theorem 4.13. There exist infinitely many $k \in \mathbb{N}$ such that $\mathcal{L}(\text{1SL-NBMCA}_k(1))$ and $\mathcal{L}(\text{1SL-NBMCA}_{k+1}(1))$ are incomparable.

P r o o f. We first observe that if, for some $k \in \mathbb{N}$, the classes $\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_k(1))$ and $\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_{k+1}(1))$ are incomparable, then also the classes $\mathcal{L}(\text{1SL-NBMCA}_k(1))$ and $\mathcal{L}(\text{1SL-NBMCA}_{k+1}(1))$ are incomparable. This is due to the fact that, for all $k \in \mathbb{N}$, all the languages over Σ in $\mathcal{L}(\text{1SL-NBMCA}_k(1))$ are also contained in $\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_k(1))$. Hence, we shall prove the theorem by showing that there exist infinitely many $k \in \mathbb{N}$ such that the classes $\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_k(1))$ and $\mathcal{L}(\text{1SL-NBMCA}^{\Sigma}_{k+1}(1))$ are incomparable. For the sake of convenience, for every $k \in \mathbb{N}$, we define $\Gamma_k := \mathcal{L}(\text{1SL-NBMCA}_k^{\Sigma}(1))$. We note that it is sufficient to show $|\Gamma_k| \geq |\Gamma_{k+1}|$ in order to conclude that Γ_k and Γ_{k+1} are incomparable. This is due to the fact that by Theorems 4.8 and 4.10 there is a language L with $L \in \Gamma_{k+1}$ and $L \notin \Gamma_k$. Hence, $|\Gamma_k| \geq |\Gamma_{k+1}|$ implies the existence of a language L' with $L' \in \Gamma_k$ and $L' \notin \Gamma_{k+1}$.

Now let $k \in \mathbb{N}$ be arbitrarily chosen. We assume that for each k', $k \leq k' \leq k' + \widehat{m} - 1$, we have $|\Gamma_{k'}| < |\Gamma_{k'+1}|$. Since, for every k' with $k \leq k' \leq k + \widehat{m}$, $|\Gamma_{k'}| \leq \widehat{m}$, this is not possible. Hence, we conclude that there exists a k', $k \leq k' \leq k + \widehat{m} - 1$, such that $|\Gamma_{k'}| \geq |\Gamma_{k'+1}|$, which implies that $\Gamma_{k'}$ and $\Gamma_{k'+1}$ are incomparable. This concludes the proof. □

The above results yield the following conclusions: For every $k \in \mathbb{N}$, there is a language that can be recognised by a 1SL-NBMCA(1) with k, but not with $k-1$ resets. This meets our expectation of nondeterminism being a useful resource enhancing the expressive power of automata. Theorem 4.13, on the other hand, does not fit with the usual results on restricted nondeterminism, as it shows that expressive power is lost by increasing the nondeterminism, i. e., for infinitely many $k \in \mathbb{N}$, there is a language that can be recognised by a 1SL-NBMCA(1) with k, but not with $k + 1$ resets. Considering the strong restrictions of $\text{1SL-NBMCA}_k(1)$, it is maybe not surprising that without any states the nondeterminism cannot be controlled anymore and, thus, a result of the sort mentioned above can be obtained. However, proving this behaviour is quite involved and, to the knowledge of the authors, it is the first result in the literature that formally establishes such a connection between finite state control and nondeterminism.

ACKNOWLEDGEMENTS

The authors wish to thank the anonymous referees for their thorough work and their valuable comments, which improved the readability of this paper significantly.

REFERENCES

[1] J. H. Chang, O. H. Ibarra, M. A. Palis, and B. Ravikumar. On pebble automata. *Theoretical Computer Science*, 44:111–121, 1986.

[2] E. Chiniforooshan, M. Daley, O. H. Ibarra, L. Kari, and S. Seki. One-reversal counter machines and multihead automata: Revisited. In *Proc. 37th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2011*, volume 6543 of *Lecture Notes in Computer Science*, pages 166–177, 2011.

[3] P. Frisco and O. H. Ibarra. On stateless multihead finite automata and multihead pushdown automata. In *Proc. Developments in Language Theory 2009*, volume 5583 of *Lecture Notes in Computer Science*, pages 240–251, 2009.

[4] M. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, MA, 1978.

[5] M. Holzer, M. Kutrib, and A. Malcher. Complexity of multi-head finite automata: Origins and directions. *Theoretical Computer Science*, 412:83–96, 2011.

[6] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[7] O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25:116–133, 1978.

[8] O. H. Ibarra and Ö. Eğecioğlu. Hierarchies and characterizations of stateless multicounter machines. In *Computing and Combinatorics*, volume 5609 of *Lecture Notes in Computer Science*, pages 408–417, 2009.

[9] O. H. Ibarra, J. Karhumäki, and A. Okhotin. On stateless multihead automata: Hierarchies and the emptiness problem. *Theoretical Computer Science*, 411:581–593, 2010.

[10] O. H. Ibarra and B. Ravikumar. On partially blind multihead finite automata. *Theoretical Computer Science*, 356:190–199, 2006.

[11] M. Kutrib, A. Malcher, and M. Wendlandt. One-way multi-head finite automata with pebbles but no states. In *Proc. 17th International Conference on Developments in Language Theory, DLT 2013*, volume 7907 of *Lecture Notes in Computer Science*, pages 313–324, 2013.

[12] M. Kutrib, H. Messerschmidt, and F. Otto. On stateless two-pushdown automata and restarting automata. *International Journal of Foundations of Computer Science*, 21:781–798, 2010.

[13] B. Monien. Two-way multihead automata over a one-letter alphabet. *RAIRO Informatique Théorique*, 14:67–82, 1980.

[14] H. Petersen. Automata with sensing heads. In *Proc. 3rd Israel Symposium on Theory of Computing and Systems*, pages 150 – 157, 1995.

[15] D. Reidenbach and M. L. Schmid. A polynomial time match test for large classes of extended regular expressions. In *Proc. 15th International Conference on Implementation and Application of Automata, CIAA 2010*, volume 6482 of *Lecture Notes in Computer Science*, pages 241–250, 2011.

[16] D. Reidenbach and M. L. Schmid. Automata with modulo counters and nondeterministic counter bounds. In *Proc. 17th International Conference on Implementation and Application of Automata, CIAA 2012*, volume 7381 of *Lecture Notes in Computer Science*, pages 361–368, 2012.

[17] D. Reidenbach and M. L. Schmid. Automata with modulo counters and nondeterministic counter bounds. Internal Report 1110, Department of Computer Science, Loughborough University, 2013. `https://dspace.lboro.ac.uk/2134/13438`.

[18] L. Yang, Z. Dang, and O. H. Ibarra. On stateless automata and P systems. *International Journal of Foundations of Computer Science*, 19:1259–1276, 2008.

Daniel Reidenbach, Department of Computer Science, Loughborough University,
Loughborough, Leicestershire, LE11 3TU, United Kingdom
e-mail: D.Reidenbach@lboro.ac.uk

Markus L. Schmid, Fachbereich IV – Abteilung Informatikwissenschaften,
Universität Trier, D-54296 Trier, Germany
e-mail: MSchmid@uni-trier.de