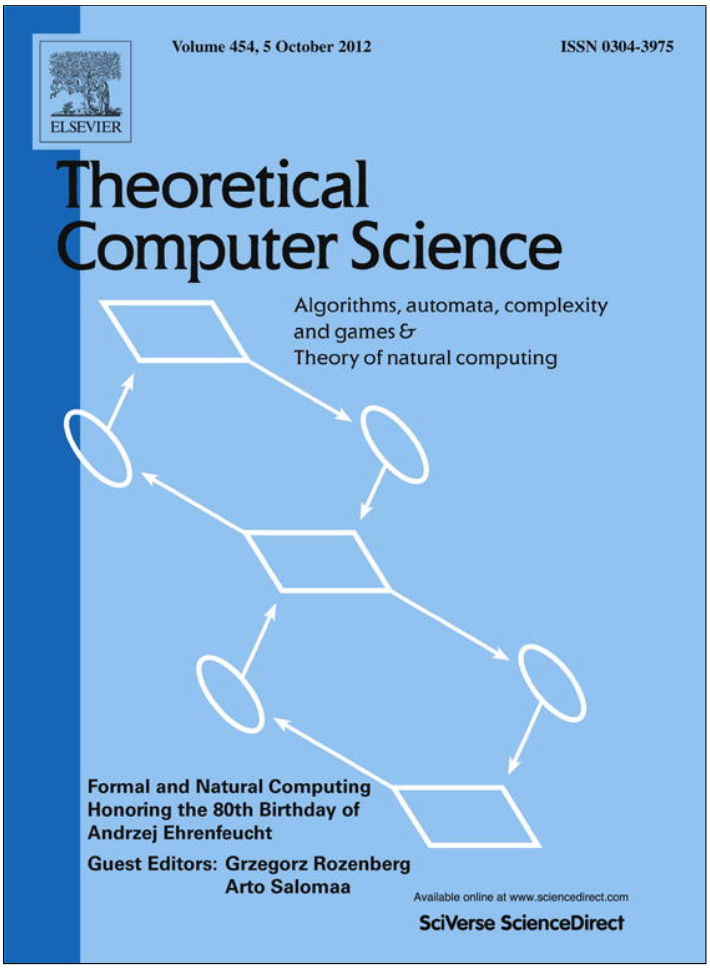

Theoretical Computer Science 454 (2012) 81–87

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

One-reversal counter machines and multihead automata: Revisited

Ehsan Chiniforooshan [a], Mark Daley [b], Oscar H. Ibarra [c,*], Lila Kari [b], Shinnosuke Seki [d]

[a] *Google, Canada*

[b] *Department of Computer Science, University of Western Ontario, London, Ontario, N6A 5B7, Canada*

[c] *Department of Computer Science, University of California, Santa Barbara, CA 93106, USA*

[d] *Department of Systems Bioscience for Drug Discovery, Kyoto University, 46-29, Yoshida-Shimo-Adachi-cho, Sakyo-ku, Kyoto, 606-8501, Japan*

ARTICLE INFO

Keywords:
Reversal-bounded counter machine
Multihead automaton
Bounded language
Inclusion
Incomparability

ABSTRACT

We investigate the power of (1-reversal) counter machines (finite automata with multiple counters, where each counter can "reverse" only once, i.e., once a counter decrements, it can no longer increment) and one-way multihead finite automata (finite automata with multiple one-way input heads) as a language acceptor. They can be non-deterministic as well as augmented with a pushdown stack. First, we prove that adding a pushdown stack properly strengthens the deterministic counter machines. Non-deterministic counter machines with a pushdown stack are then compared with multihead finite automata. The proof of their incomparability involves an interesting technique: an assumption that a language be accepted by a non-deterministic counter machine would bring a contradictory algorithm to decide an undecidable language. Furthermore, we will show that over bounded languages, these two kinds of machines have the same power, and neither non-determinism nor a pushdown stack makes them stronger.

1. Introduction

Ways of strengthening finite state machines include making the computation non-deterministic, allowing the state control read more than one letter on the input tape simultaneously using multiple heads, and augmenting them with read-write unbounded memories such as stack, counter, etc. Non-determinism does not add any power to finite automata (FA) or Turing machines (TM). In contrast, non-determinism increases the power of pushdown automata (a PDA is a finite state machine with a *stack*, that is, semi-infinite tape which operates on a last-in-first-out basis). The significance of deterministic PDA (DPDA) in theory and applications, among others, has motivated research on the power as a language acceptor which non-determinism brings to finite state machines.

As a PDA is obtained by augmenting an FA with a stack, finite state machines can be further reinforced by having more than one stack attached to it. It is well known that two stacks already endow the PDA Turing universality, even if we limit the size of the stack alphabet Γ to 1 (such a stack is called a *counter*) [1]. This universality is not true any more once an upper bound is imposed on the number of reversals which the counters can make during the computation (a reversal is a shift from the non-decreasing mode to the non-increasing mode, or vice versa). Ibarra has introduced the finite state machine with such counters as the *reversal-bounded counter machine* [2]. Non-Turing universality of reversal-bounded counter machines was proved there by showing that if the Parikh image of a language accepted by such a machine is semilinear. This semilinearity makes possible to strengthen FAs while keeping their several useful properties like emptiness decidable. For other related results, see [3–8]. Rosenberg proposed another method to strengthen finite state machines by equipping them with multiple input heads, which move independently on the input tape [9].

* Corresponding author. Tel.: +1 805 893 4171; fax: +1 805 893 8553.
E-mail addresses: daley@csd.uwo.ca (M. Daley), ibarra@cs.ucsb.edu (O.H. Ibarra), lila@csd.uwo.ca (L. Kari), sseki@pharm.kyoto-u.ac.jp (S. Seki).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.04.002

As we shall see in detail in Section 2, unless the number of counters matters, we can assume that they make at most one reversal; such a counter is called 1-*reversal*. Equipping a (non-deterministic: N) FA or PDA with either 1-reversal counters (C) or multiple one-way input heads (MH) brings four classes of finite state machines, that is, the classes of counter machines (NCM), pushdown counter machines (NPCM), one-way multihead finite automata (NMHFA), and one-way multihead pushdown automata (NMHPDA). Their deterministic variants are also of interest. Their abbreviations are obtained by replacing N in those of the non-deterministic classes with D. The investigation on the inclusion hierarchy among these eight classes is the primary purpose of this paper.

More specifically, our questions are of the following types:

Q1. whether a pushdown stack or non-determinism strictly strengthens 1-reversal counter machines;
Q2. whether the machines in a class can be simulated by a machine in another class;
Q3. does there exist a class of languages over which some of these classes possess equivalent power as a language acceptor.

Section 3 addresses Q1 and Q2. First, we propose a language that is accepted by a DPCM but cannot be accepted by any NCM (Theorem 1). This means that the pushdown stack strictly increases the power of deterministic counter machines as a language acceptor. A technique used toward this end deserves special note: on the supposition that an NCM accepted this language, we could design an algorithm to decide the halting problem for Turing machines.

The problem of whether nondeterminism strictly strengthens counter machines remains open in this paper. On this, we have a remark.

Remark 1. In the conference version of this paper, [10], we claimed that

$$L_b = \{a^{i_1}\#a^{i_2}\#\cdots\#a^{i_n} \mid n \geq 2, i_1, \ldots, i_n \geq 0 \text{such that } i_1 + i_2 + \cdots + i_k = i_{k+1} + \cdots + i_n \text{ for some } 1 \leq k < n\}$$

is in NCM \ DPCM. This is true if the input tape of DPCM is not augmented with the right end-marker \$. In the presence of \$, however, L_b can be accepted by a DPCM M. In fact, a counter of M counts the number of a's, while it copies the input onto its stack. Once the input head reaches \$, then it starts popping the stack. When a is popped, the counter decrements by 2. The input is accepted if the stack top is # when the counter becomes 0.

It is more common to assume the existence of \$ for counter machines. In this case, a language in NCM \ DPCM is to be found.

Section 4 is devoted to Q3. We introduce a class of languages over which a weak machine DCM can simulate a strong machine NMHPDA. That is the class of bounded languages. Ibarra and Seki have proved that over this class, DCM = NPCM [11]. We generalize this equation over to the multihead finite state machine classes (Corollary 6) by making use of the simulation of NMHPDA by DCM (Lemma 5) with a simulation of DCM by DMHFA (Theorem 3).

2. Preliminaries

Let Σ be an alphabet and by Σ^* we denote the set of all words over Σ including the empty word ϵ. For a word $w \in \Sigma^*$, $|w|$ denotes its length and w^R denotes its reverse.

Let us recall the definition of reversal-bounded (pushdown) counter machines [2]. A *reversal-bounded counter machine* is a finite automaton augmented with reversal-bounded counters. We can further augment a reversal-bounded counter machine with a pushdown stack to obtain a *reversal-bounded pushdown counter machine*.

A pushdown k-counter machine M is formally represented by a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where Q, Σ, Γ, F are the respective sets of states, input letters, stack symbols, and final states, q_0 is the initial state, and $Z_0 \in \Gamma$ is the particular stack symbol called the start symbol. We define the transition δ in a way that is different from but equivalent to the convention as a relation from $Q \times \Sigma \times \Gamma \times \{0, 1\}^k$ into $Q \times \{\mathtt{S}, \mathtt{R}\} \times \Gamma^* \times \{-1, 0, +1\}^k$. S and R indicate the direction in which M moves its input head (S: stay, R: right). M is said to be *deterministic* if δ is a function. A configuration of M is given by a $(k+3)$-tuples $(q, w\$, x, c_1, \ldots, c_k)$ denoting the fact that M is in the state q, w is the "unexpended" input with the right end-marker \$, the stack contains the word x, and $c_1, c_2, \ldots, c_k$ are the values contained in the k counters. Among configurations, we define a relation $\vdash_M$ as follows: $(q, aw\$, X\alpha, c_1, \ldots, c_k) \vdash_M (p, w'\$, \beta\alpha, c_1 + e_1, \ldots, c_k + e_k)$ if $\delta(q, a, X, \lambda(c_1), \ldots, \lambda(c_k))$ contains $(p, d, \beta, e_1, \ldots, e_k)$, where $d \in \{\mathtt{S}, \mathtt{R}\}$, $e_1, \ldots, e_k \in \{-1, 0, +1\}$,

$$\lambda(c_i) = \begin{cases} 0 & \text{if } c_i = 0 \\ 1 & \text{otherwise} \end{cases} \quad \text{and} \quad w' = \begin{cases} aw & \text{if } d = \mathtt{S} \\ w & \text{if } d = \mathtt{R}. \end{cases}$$

The transition with the indicator S corresponds to the ϵ-transition in the conventional definition of pushdown counter machines, whereas that with R corresponds to the transition which consumes an input symbol. The reflexive and transitive closure of $\vdash_M$ is written as $\vdash_M^*$. The subscript is dropped from $\vdash_M$ and $\vdash_M^*$ whenever the particular M is understood. A word $w \in \Sigma^*$ is accepted by M if $(q_0, w\$, Z_0, 0, \ldots, 0) \vdash^* (q_f, \$, \alpha, c_1, \ldots, c_k)$ for some $q_f \in F$. The set of all words accepted by M is denoted by $L(M)$. By ignoring the pushdown stack through the description so far, we can obtain the analogous definition and notions for counter machines (no pushdown stack).

By a *reversal* of counter, we mean an alternation between non-decreasing mode and non-increasing mode. A counter machine is *reversal-bounded* if there exists a constant c such that any computation involves at most c reversals of each of its

counter. At the cost of increase in the number of states and counters, we can assume $c = 1$. Indeed, we simulate a counter making r-reversals by $\lceil \frac{r+1}{2} \rceil$ counters each of which makes only 1-reversal (1-*reversal counter*) in such a way that the first 1-reversal counter simulates the k-reversal counter until the first two reversals, and the second one simulates until the fourth reversal, and so on. This simulation apparently requires the finite state control. Likewise, we can consider the 1-reversal stack. A machine with a 1-reversal stack is said to be 1-*turn*.

For an integer $k \geq 0$, let NCM(k) be the class of counter machines with k 1-reversal counters, and then let NCM $= \bigcup_{k=0}$ NCM(k). By a slight abuse of notation, we employ NCM(k) to denote also a counter machine with k counters as well as the class of languages accepted by such a machine. Their deterministic subclasses are denoted by DCM(k) and DCM, respectively. Let us denote the class of pushdown machines with k 1-reversal counters (pushdown k-counter machines) by NPCM(k), and NPCM $= \bigcup_{k=0}$ NPCM(k). DPCM(k) and DPCM are their deterministic subclasses. Needless to say, NCM(0) = DCM(0) = FA, while NPCM(0) = PDA and DPCM(0) = DPDA.

A (non-deterministic) multihead finite automaton M with k heads (written as NMHFA(k)) is a machine with k one-way input heads operating on an input string with a right end marker \$. At the start of the computation, the heads are on the leftmost symbol of the input string and M is in its initial state. A move of the machine is a transition of the form $\delta(q, a_1, \ldots, a_k) = \{(p, d_1, \ldots, d_k)\}$, where q is the current state, $a_1, \ldots, a_k$ are the symbols scanned by heads, p is the next state, and $d_1, \ldots, d_k$ are the movements of the heads, which can be R or S (one position to the right, or do not move). Note that the heads are *non-sensing* (i.e., the heads cannot sense the presence of the other heads on the same position). An input is accepted if M eventually enters an accepting state and all heads are on the right end marker. The machine can be augmented with a pushdown stack (NMHPDA) in the obvious way. They can be deterministic (DMHFA, DMHPDA).

3. Comparison among machine classes

In this section, we compare the four counter machine classes (DCM, NCM, DPCM, NPCM) and the four multihead automata classes (DMHFA, NMHFA, DMHPDA, NMHPDA) with respect to their power as language acceptor. By definition, it should be clear that DCM $\subseteq$ NCM, DPCM $\subseteq$ NPCM and DMHFA $\subseteq$ NMHFA, DMHPDA $\subseteq$ NMHPDA.

First of all, we will prove that there exists a language accepted by a DPCM that cannot be accepted by any NCM (Theorem 1). An example is the marked palindrome language $L_{\text{pal}} = \{x\#x^R \mid x \in \{0, 1\}^+\}$ (recall that R denotes reverse). It is obvious that this can be accepted by a 1-turn DPDA. Suppose L_{pal} were accepted by $M \in$ NCM(k) for some $k \geq 0$. Using a result by Baker and Book [12], we may assume that M operates in linear time. Consider an input $u\#u^R$, where $|u| = n$. Clearly, the number of possible configurations when the input head of M reaches # is $O(n^k)$. Now consider another input $v\#v^R$, where $|v| = n$ and $v \neq u$. It follows that since there are 2^n binary strings of length n, $u\#v^R$ would be accepted by M for n large enough. This is a contradiction.

Now Theorem 1 has been proved. However, we would like to mention another proof, which employs an interesting technique and can be used also for proving the existence of a language that is in DMHFA \ NPCM (Theorem 2). This technique allows us to construct a contradictory algorithm to decide an undecidable on the assumption that a specific language be accepted by an NCM.

Theorem 1. *There is a language accepted by a* 1*-turn* DPDA *that cannot be accepted by any* NCM*.*

Proof. For $a \in \Sigma$, let $L \subseteq \{a\}^*$ be a unary recursively enumerable language that is not decidable (such L exists) and M be a Turing machine accepting L. Let Q and Σ be the state set and worktape alphabet of M and let $q_0 \in Q$ be the initial state of M. Let $\Sigma' = Q \cup \Sigma \cup \{\#\}$. The halting computation of M on the input a^d can be represented by the string $ID_1\#ID_3 \cdots \#ID_{2k-1}\#\#ID_{2k}^R \cdots \#ID_4^R\#ID_2^R$ for some $k \geq 2$ (without loss of generality, we can assume that the length of a computation is even), where $ID_1 = q_0a^d$ and ID_{2k} are the initial and halting configurations of M, and $(ID_1, ID_2, \ldots, ID_{2k})$ is a sequence of configurations of M on input a^d, i.e., configuration ID_{i+1} is a valid successor of ID_i.

Now consider the languages

$$\begin{aligned} L_1 = \{&ID_1\# \cdots \#ID_{2k-1}\#\#ID_{2k}^R \cdots \#ID_2^R \mid ID_{2k} \text{ is a halting configuration,} \\ &k \geq 2, ID_1 = q_0a^p (p \geq 1), \text{ and } ID_{i+1} \text{ is a valid successor of } ID_i \text{ for odd } i\}, \\ L_2 = \{&ID_1\# \cdots \#ID_{2k-1}\#\#ID_{2k}^R \cdots \#ID_2^R \mid ID_{2k} \text{ is a halting configuration,} \\ &k \geq 2, ID_1 = q_0a^p (p \geq 1), \text{ and } ID_{i+1} \text{ is a valid successor of } ID_i \text{ for even } i\}. \end{aligned}$$

Clearly, L_1 and L_2 can be accepted by 1-turn DPDAs. Note that their intersection is a set of all halting computations of M.

Now we will prove that L_1 or L_2 cannot be accepted by any NCM, and this suffices for our purpose. Suppose they were accepted by NCMs M_1, M_2 with n_1, n_2 1-reversal counters, respectively. From these NCMs, we can construct an NCM M accepting $L_1 \cap L_2$ [2]. Using M, a contradictory algorithm to decide L could be constructed in the following way:

1. On an input a^d, construct a finite automaton accepting $q_0a^d\#\Sigma'^*$.
2. From the finite automaton and the NCM M (accepting $L_1 \cap L_2$), construct an NCM M' which accepts $q_0a^d\#\Sigma'^* \cap L_1 \cap L_2$.

3. Test if the language accepted by M' is empty. This is possible since the emptiness problem for NCMs (or even for NPCMs) is decidable [2].

Note that $a^d \notin L$ if and only if the language accepted by M' is empty. □

Corollary 1. DPCM *is strictly more powerful than* DCM.

The construction of contradictory algorithm has a broad range of applications. Let us employ it in order to prove that the following language:

$$L_{\mathrm{ID}} = \{ID_1\#ID_2\#\cdots\#ID_k \mid ID_1 = q_0a^p \text{ for some } p \geq 1, ID_k \text{ is a halting configuration, and } ID_{i+1} \text{ is a valid successor of } ID_i \text{ for } i \geq 1\}$$

can be accepted by a DMHFA(2) but cannot be accepted by any NPCM. Note that, unlike in L_1 or L_2, even IDs are not reversed in L_{ID}. It is easy to observe that $L_{\mathrm{ID}} \in$ DMHFA(2). By replacing NCM and $L_1 \cap L_2$ by NPCM and L_{ID}, Step 2 produces an NPCM that accepts $q_0a^d\#\Sigma'^* \cap L_{\mathrm{ID}}$. Since the emptiness problem for NPCMs is decidable, this algorithm is contradictory that decides the halting problem for TMs on blank tape.

Theorem 2. $L_{\mathrm{ID}} \in$ DMHFA(2) $\setminus$ NPCM.

We will further strengthen Theorem 2 up to the incomparability between DMHFA and NPCM. As mentioned above, L_{pal} can be accepted by a 1-turn DPDA. Let us prove that this language cannot be accepted by any DMHFA. This may have already been known, but we have not been able to find an appropriate reference. We give a proof below for completeness. The proof uses an idea in [13], which is based on the Kolmogorov complexity.

For a word $w \in \Sigma^*$, let $K(w)$ be the Kolmogorov complexity of w, and let $K(w|y)$ be the conditional Kolmogorov complexity of w with respect to a given extra information y. A word w is said to be *random* if $K(w) \geq |w|$, and it is known that there exist random words. We state a simple well-known fact that if a word uvw is random, then $K(v|uw) \gg |v| - O(\log|uvw|)$ (see [13]).

Lemma 1. $L_{\mathrm{pal}} \notin$ DMHFA(2).

Proof. Suppose that there were a DMHFA(2) M such that $L(M) = L_{\mathrm{pal}}$. Let h_r, h_l be the rightmost and leftmost heads of M, respectively.

Let us consider a random word $w = w_1w_2$ satisfying $|w_1| = |w_2| \gg \log|w| + |M|$, where $|M|$ denotes the (program) size of M. Note that $w_1 \neq w_2$ because of the randomness of w. Then as an input, we put $I_1 = *w_1 * w_2 * \$ * w_2^R * w_1^R*$ into M. For this input, we say that w_i $(i = 1, 2)$ is *checked* if there is a time t when h_r is on w_i^R while h_l is on w_i.

We claim that both w_1 and w_2 have to be checked. Indeed, suppose that w_1 were not checked. Consider the computation of K on I_1. Let $ID_{\mathrm{in}}(h_r)$ ($ID_{\mathrm{out}}(h_r)$) be the configuration of M when h_r first reaches the first (resp. second) * sign in $*w_1^R*$ of I_1. The analogous notation is defined for h_l. Note that these IDs can be described of length $O(\log|w|)$. Given these IDs and w_2, we can reconstruct w_1 by a short program as follows: For a word x with $|x| = |w_1|$, let $P(x) = *0^{|w_1|} * w_2 * \$ * w_2^R * x*$. We simulate M on this program $P(x)$ starting with $ID_{\mathrm{in}}(h_r)$ and $ID_{\mathrm{in}}(h_l)$. If $ID_{\mathrm{in}}(h_r) \vdash^*_{P(x)} ID_{\mathrm{out}}(h_r)$ and $ID_{\mathrm{in}}(h_l) \vdash^*_{P(x)} ID_{\mathrm{out}}(h_l)$ hold, then M accepts $I_1(x) = *w_1 * w_2 * \$ * w_2^R * x*$. This can happen if and only if $x = w_1^R$ (otherwise, an input not in L_{pal} would be accepted by M). Therefore, $K(w_1|w_2) = O(\log|w|)$, but this contradicts the above-mentioned fact on conditional Kolmogorov complexity.

Now, the claim has been verified so that both w_1 and w_2 have to be checked, but clearly it cannot be done by one-way multihead FA (if w_2 is checked, then h_l is on w_2 so that it cannot return back to w_1 in order to check w_1.) □

Theorem 2 and Lemma 1 provide the next result.

Corollary 2. DMHFA(2) *and* NPCM *are incomparable.*

This result is now improved to the incomparability between DMHFA and NPCM. In order to prove that $L_{\mathrm{pal}} \notin$ DMHFA(n), in the proof of Lemma 1, we split the random string w into n substrings of the same length as $w = w_1w_2\cdots w_n$.

Corollary 3. DMHFA *and* NPCM *are incomparable.*

Actually, even non-determinism does not help for multihead automata to accept L_{pal}. If a language L is accepted by an NMHFA(k) M and $w \in L$, then there must exist an accepting computation of M for w and it can be regarded as a sequence of transition rules. By appending an encoding of such a sequence to each word in L, one can generate a language L' and this language can be accepted by a DMHFA($k+1$). Using this idea, we can prove that $L_{\mathrm{pal}} \notin$ NMHFA.

Corollary 4. NMHFA *and* NPCM *are incomparable.*

Having related the group of the counter machine classes with that of the multihead automata classes, now we prove that DCM $\subseteq$ DMHFA in order to strengthen this connection further.

It is known (see, e.g., [14]) that one can prevent a DPDA from falling into an infinite loop. Using a similar technique, an analogous result can be proved for DCM.

Lemma 2. *Any* DCM M *can be converted to an equivalent* DCM M' *with the property that, for every input,* M' *does not go into an infinite loop on a symbol on the input tape.*

Proof. Let M have s states. M' is constructed as follows:

1. M' simulates M.
2. If M has made more than s moves while remaining on the symbol without at least one of the following happening:
   (a) a positive counter decrementing,
   (b) a zero counter becoming positive (note that when a positive counter becomes zero from being positive, it can no longer become positive again),

then M is looping on the symbol. M' then has two cases to handle:

Case 1: During the looping, M does not enter any accepting state. In this case, M' enters a special (new) reject state r and scans the remaining input in state r until the head falls off the input.

Case 2: During the looping, M enters an accepting state. In this case, M' enters a special (new) accepting state f (thus accepting the input if there is no more symbol to the right of the head). Then M', in state f, on any input enters a special rejecting state r and scans the remaining input in state r until the head falls off the input. □

Lemma 3. *Any* DCM M *can be converted to an equivalent* DCM M' *such that there exist constants* c_1, c_2*, and for any input* w*, the values of counters are at most* $c_1|w| + c_2$ *during the computation by* M' *on* w*.*

Proof. We concern the value of counters so that it makes sense to assume $k \geq 1$. Let M' be the DCM converted from the DCM M in Lemma 2. Recall that M has the s states, and let $c = s \times 2^k$. M' is designed in a way that if M' makes more than c moves without moving its head at all and $c \geq s$, then during that time either a positive counter is decrementing or a zero counter becomes positive.

We claim that during the computation by M' on an input of length n, the value of each counter can be at most $(c+2)^{k-1}(c+1)(n+k)$, i.e., $O(n)$, by induction on the number of counters. Without loss of generality, we can assume that at the point when a counter reaches its highest value during the computation, the other counters are zero because otherwise while decrementing another counter with value n' up to 0, M' can further increment the counter at least by cn'. In order to make the value of a counter as large as possible, we employ the following strategies:

1. at each transition, M' will increase the values of all counters which are in non-decreasing mode;
2. M' never decrease two counters at the same transition.

For the case $k = 1$, more than c transitions should not occur uninterruptedly without moving its head. According to the first strategy, M should increment the counter from the very beginning transition, and once being decremented, M cannot increment the counter any more. Thus the value of this counter can be at most $(c+1)n + c \leq (c+1)(n+1)$.

Now suppose that the claim holds for some $k-1$ and consider the case when M' has k counters. Let us assume that M decrements $(k-1)$-th counter for the first time while M' moving its head from the $(m-1)$-th input symbol to the m-th one. At the point, the value of $(k-1)$-th or k-th counter can reach at most $(c+2)^{k-2}(c+1)(m+k-1)$ according to the induction hypothesis, even if M' makes the other counters to be 0. While decrementing the $(k-1)$-th counter up to 0, the k-th counter can increase at most by $(c+1)\cdot(c+2)^{k-2}(c+1)(m+k-1)+c$. By expending the rest of the input ($n-m$ symbols), we can increase at most $(c+1)(n-m)+c$. Thus, the k-th counter can become largest when $m = n$ and the value is

$$(c+2)^{k-2}(c+1)(n+k-1) + (c+1)\cdot(c+2)^{k-2}(c+1)(n+k-1) + c \leq (c+2)^{k-1}(c+1)(n+k).$$

Thus, the induction step is verified and the claim holds. By letting $c_1 = (c+2)^{k-1}(c+1)$ and $c_2 = k(c+2)^{k-1}(c+1)$, this lemma holds. □

Theorem 3. *For any* $k \geq 0$, $\mathrm{DCM}(k) \subseteq \mathrm{DMHFA}(2k+1)$.

Proof. Let $M \in \mathrm{DCM}(k)$. Each 1-reversal counter C of M can be simulated by two input heads H_1 and H_2, which are initially positioned on the leftmost symbol of the input. When C increments by 1, H_1 is moved one position to the right. When C reverses, both heads are moved simultaneously to the right until H_1 reaches the end marker before continuing with the simulation of C. After that, decrementing C would correspond to moving H_2 one position to the right. When H_2 reaches the end marker, this indicates that C has the value zero. This simulation works if the counter values are bounded by n, where n is the length of the input. If the counter values are bounded by $c_1 n + c_2$ for some constant c_1, c_2 (i.e., the counter values are linearly bounded), then H_1 (resp. H_2) operates modulo c_1, i.e., H_1 (resp. H_2) is moved one position to the right for every c_1 increments (resp. decrements) of C. The existence of such c_1, c_2 is guaranteed by Lemma 3. □

Due to Theorems 2 and 3, the following proper inclusion holds.

Corollary 5. $\mathrm{DCM} \subsetneq \mathrm{DMHFA}$.

4. Counter machines and multihead automata over bounded languages

The languages classes over which these machine classes become comparable or become equally powerful are of interest. The class of bounded languages is one of such. A language L is *bounded* if it is a subset of $w_1^* \cdots w_k^*$ for some integer $k \geq 1$ and (not necessarily distinct) words $w_1, \ldots, w_k \in \Sigma^*$. We can assume that w_i is different from w_{i+1} for $1 \leq i \leq k-1$. If all of $w_1, \ldots, w_k$ are letters (elements of Σ), then L is especially called *letter-bounded*. For example, any language that is a subset of $a_1^* a_2^* a_1^* a_2^* a_3^* a_1^*$ is (letter-)bounded, while $(a_1 + a_2)^*$ is not even bounded, where $a_1, a_2, a_3 \in \Sigma$. The balanced language L_b is not bounded, either.

For a bounded language $L \subseteq w_1^* \cdots w_k^*$, there corresponds a subset $Q(L)$ of $\mathbb{N}^k$ defined by $Q(L) = \{(i_1, \ldots, i_k) \mid w_1^{i_1} \cdots w_k^{i_k} \in L\}$. A subset Q of $\mathbb{N}^k$ is a *semilinear* set if

$$Q = \bigcup_{i=1}^{m} \left\{ v_0^i + \sum_{j=1}^{r_i} t_j v_j^i \mid t_j \geq 0 \right\}$$

for some $m \geq 1$, $r_i \geq 1$ $(1 \leq i \leq m)$, and $v_j^i \in \mathbb{N}^k$ $(1 \leq i \leq m$ and $0 \leq j \leq r_i)$. If $Q(L)$ is semilinear, then L is *bounded semilinear*.

Theorem 4 ([11]). *Over bounded languages,* DCM = NPCM.

The proof of this theorem given in [11] consists of proving that if a bounded language L is accepted by an NPCM, then L is bounded semilinear, and that, from $Q(L)$, we can construct a DCM accepting L. This means that if a class of machines has a property that all bounded languages accepted by a machine in this class are semilinear, then this is a subclass of DCM.

Remark 2. The actual result proved in [11] is stronger than Theorem 4. The equation holds even on some of the machine classes that are obtained from respective DCM, NCM, DPCM, NPCM by letting their input head move two-ways under restrictions on the number of turn (finite-turn input head) or on the number of crossing each input letter (finite-crossing input head). Ibarra and Seki proved that if a bounded language is accepted by a finite-crossing two-way DPCM, then it is bounded semilinear, and hence, can be accepted by a one-way DCM. The analogous problem for finite-crossing two-way NPCM remains open to our knowledge.

We will expand this equation so as to include classes of multihead automata of various kinds. Due to Theorems 3 and 4, it suffices to prove that, over bounded languages, NMHPDA $\subseteq$ DCM holds.

Lemma 4. *Over letter-bounded languages,* NMHPDA $\subseteq$ DCM.

Proof. It is known that, given an NMHPDA accepting a letter-bounded language L, we can effectively construct a semilinear set Q such that $Q = Q(L)$ [15]. From Q, we can construct a DCM for L, as mentioned above. □

Lemma 5. *Over bounded languages,* NMHPDA $\subseteq$ DCM.

Proof. Let $L \subseteq w_1^* w_2^* \cdots w_k^*$ be bounded language accepted by an NMHPDA. For a new alphabet $\Delta = \{a_1, a_2, \ldots, a_k\}$, we define a homomorphism $h : \Delta^* \to \Sigma^*$ as $h(a_i) = w_i$ for $1 \leq i \leq k$. Then, let $L' = h^{-1}(L) \cap a_1^* a_2^* \cdots a_k^*$. It is easy to see that $L' \in$ NMHPDA. Since being letter-bounded, L' can be accepted by a DCM. A way to convert this DCM into a DCM that accepts L is given in [11]. □

Combining this lemma with Theorem 3, now we can strengthen Theorem 4 as follows.

Corollary 6. *Over bounded languages,* DCM = NPCM = DMHFA = NMHPDA.

It is known that the universe problem, determining if a given machine accepts all words in Σ^*, is undecidable for NCM(1). As a corollary, the containment and equivalence problems for this class are also undecidable. On the other hand, the containment and equivalence problems for DCM are decidable [2]. Thus, Corollary 6 leads us to the following result on decidability for NPCM and NMHPDA over bounded languages.

Theorem 5. *Both containment and equivalence problems for* NPCM *and* NMHPDA *over bounded languages are decidable.*

5. Conclusion

In this paper, we investigated the relative computational power of various classes of machines, which can be obtained by augmenting FA with a stack, 1-reversal counters, multiheads, or non-determinism. Among the eight machine classes thus obtained, we proved inclusion, incomparability, and equivalence relations.

Acknowledgments

We wish to thank anonymous referees for their critical comments and suggestions on the earlier version of this paper.

This research was supported by the National Science Foundation Grant CCF-0524136 to Oscar H. Ibarra, Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant R2824A01 and the Canada Research Chair Award to Lila Kari, Funding Program for Next Generation World-Leading Researchers (NEXT program) to Professor Yasushi Okuno, and Kyoto University Start-up Grant-in-Aid for Young Scientists 021530 to Shinnosuke Seki.

References

[1] M.L. Minsky, Recursive unsolvability of Post's problem of tag and other topics in theory of Turing machines, Ann. Math. 74 (1961) 437–455.
[2] O.H. Ibarra, Reversal-bounded multicounter machines and their decision problems, J. ACM 25 (1978) 116–133.
[3] P. Duris, Z. Galil, On reversal-bounded counter machines and on pushdown automata with a bound on the size of the pushdown store, Inform. Control 54 (1982) 217–227.
[4] O. Eğecioğlu, O.H. Ibarra, On stateless multicounter machines, in: K. Ambos-Spies, B. Löwe, W. Merkle (Eds.), Proc. of CiE 2009, in: LNCS, vol. 5635, Springer, 2009, pp. 178–187.
[5] S.Z. Fazekas, H. Ito, Y. Okuno, S. Seki, K. Taneishi, On computational complexity of graph inference from counting, submitted for publication.
[6] T. Harju, O.H. Ibarra, J. Karhumäki, A. Salomaa, Some decision problems concerning semilinearity and commutation, J. Comput. Syst. Sci. 65 (2002) 278–294.
[7] O.H. Ibarra, O. Eğecioğlu, Hierarchies and characterizations of stateless multicounter machines, in: H.Q. Ngo (Ed.), Proc. COCOON 2009, in: LNCS, vol. 5609, Springer, 2009, pp. 408–417.
[8] O.H. Ibarra, S. Woodworth, F. Yu, A. Păun, On spiking neural P systems and partially blind counter machines, Nat. Comp. 7 (2008) 3–19.
[9] A.L. Rosenberg, On multi-head finite automata, IBM J. Res. Dev. 10 (1966) 388–394.
[10] E. Chiniforooshan, M. Daley, O.H. Ibarra, L. Kari, S. Seki, One-reversal counter machines and multihead automata: Revisited, in: Proc. SOFSEM 2011, in: LNCS, vol. 6543, Springer, 2011, pp. 166–177.
[11] O.H. Ibarra, S. Seki, Characterizations of bounded semilinear languages by one-way and two-way deterministic machines, in: Proc. AFL 2011, 2011, pp. 211–224.
[12] B.S. Baker, R.V. Book, Reversal-bounded multipushdown machines, J. Comput. Syst. Sci. 8 (1974) 315–332.
[13] M. Li, Y. Yesha, String-matching cannot be done by a two-head one-way deterministic finite automaton, Inform. Process. Lett. 22 (1986) 231–235.
[14] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.
[15] O.H. Ibarra, A note on semilinear sets and bounded-reversal multihead pushdown automata, Inform. Process. Lett. 3 (1) (1974) 25–28.