ELSEVIER

Theoretical Computer Science 180 (1997) 139–154

Theoretical Computer Science

On the power of alternation on reversal-bounded alternating Turing machines with a restriction

Hiroaki Yamamoto *

Department of Information Engineering, Faculty of Engineering, Shinshu University, 500 Wakasato, Nagano-shi, 380 Japan

Received July 1995; revised March 1996
Communicated by R.V. Book

Abstract

Whether or not there is a difference of the power among alternating Turing machines with a bounded number of alternations is one of the most important problems in the field of computer science. This paper presents the following result: Let $R(n)$ be a space and reversal constructible function. Then, for any $k \geqslant 1$, we obtain that the class of languages accepted by off-line 1-tape $r\sigma_k$ machines running in reversal $O(R(n))$ is equal to the class of languages accepted by off-line 1-tape σ_1 machines running in reversal $O(R(n))$. An off-line 1-tape σ_k machine M is called an off-line 1-tape $r\sigma_k$ machine if M always limits the non-blank part of the work-tape to at most $O(R(n)\log n)$ when making an alternation between universal and existential states during the computation.

1. Introduction

Whether or not there is a difference of the power among alternating Turing machines (ATM for short) with a bounded number of alternations is one of the most important problems in the field of computer science. The number of alternations is defined as the number of times an ATM makes alternation between existential states and universal states. An ATM is called a σ_k machine (a π_k machine, respectively) if it starts in an existential state (a universal state, respectively) and makes $k-1$ alternations between existential and universal states during its computation. Using this notation, we can state the above problem as follows: for $\forall k \geqslant 1$, consider a σ_k machine with a bounded computational resource such as a time-bounded machine, a space-bounded machine and so on. Then, is a σ_{k+1} machine more powerful than a σ_k machine?

Up to now, many researchers have been studying on this problem. As for time complexity, the problem whether the polynomial time hierarchy by Stockmeyer [9]

* E-mail: yamamoto@cs.shinshu-u.ac.jp.

PII S0304-3975(96)00143-0

and Wrathall [13] collapses or not is widely known as an open problem. That is, we do not know whether or not σ_{k+1} machines running in polynomial time are more powerful than σ_k machines running in polynomial time for any $k \geqslant 1$. On the other hand, by Immerman's result [5] that nondeterministic space bounded classes are closed under complement, the alternation hierarchy for space complexity collapses. That is, the class of languages accepted by σ_k machines running in space $O(S(n))$ is equal to the class of languages accepted by σ_1 machines running in space $O(S(n))$ for any space constructible function $S(n) \geqslant \log n$.

In this paper, we are concerned with reversal complexity, which is defined as the number of times a machine changes the head direction on the work-tape during the computation. This resource has also been intensively studied so far by many researchers. We here consider two reversal-bounded 1-tape models; one is an off-line 1-tape machine which has a read-only two-way input tape and one work-tape. The other is a 1-tape machine which has only one work-tape, and an input is initially given in the work-tape. These work-tapes have a leftmost square but is infinite to the right. Liskiewicz and Lorys [8] investigated the alternation hierarchy for reversal complexity, and showed that such a hierarchy exists for 1-tape machines. In fact, they showed that a 1-tape σ_{k+1} machine running in reversal $O(R(n))$ is more powerful than a 1-tape σ_k machine running in reversal $O(R(n))$ for any reversal constructible function $R(n) \geqslant \log^{(k)} n$. On the other hand, since Baker and Book [1] showed that an NTM with two work-tapes can accept every recursively enumerable set in a constant reversal, it is obvious that the alternation hierarchy collapses for machines with two work-tapes. Thus, depending on the number of tapes of machines, the alternation hierarchy exists or not. Greibach [3] showed that, for any function $R(n)$, the class of languages accepted by $O(R(n)\log n)$ reversal-bounded off-line 1-tape σ_1 machines is equal to the class of languages accepted by $O(R(n))$ space-bounded off-line σ_1 machines. In addition, for 1-tape machines, the similar result is known. Hence, these results and Immerman's result seem to give us some possibility to make the alternation hierarchy on reversal complexity collapse for off-line 1-tape machines and 1-tape machines. By the aforementioned result of Liskiewicz and Lorys, however, there is the alternation hierarchy for conventional 1-tape ATMs. What is a condition to collapse the alternation hierarchy? Thus, it is very interesting to study a condition under which the alternation hierarchy collapses.

We will show that the alternation hierarchy collapses for off-line 1-tape machines with a certain restriction. In fact, the following result is obtained: Let $R(n)$ be a space and reversal constructible function. Then, for any $k \geqslant 1$, we obtain that the class of languages accepted by off-line 1-tape $r\sigma_k$ machines running in reversal $O(R(n))$ is equal to the class of languages accepted by off-line 1-tape σ_1 machines running in reversal $O(R(n))$. The similar result also holds for a 1-tape ATM. An off-line 1-tape σ_k machine M is called an off-line 1-tape $r\sigma_k$ machine if M always limits the non-blank part of the work-tape to at most $O(R(n)\log n)$ when making an alternation between universal and existential states. Here, without loss of generality, we assume that M can write a blank symbol. This is because M can simulate a behavior of the blank symbol by using an additional symbol if it cannot write the blank symbol. Hence, by

the non-blank part, we mean the space from the leftmost square to the rightmost square containing a non-blank symbol. Thus, note that M can use more than $O(R(n)\log n)$ space during the computation from an alternation to the next alternation.

The paper is organized as follows: In Section 2, some definitions are given. In Sections 3–5, off-line 1-tape models are discussed, and in Sections 4 and 5, the main results are given. In Section 6, the discussion about 1-tape ATMs is given.

2. Preliminaries

In this section, we give some definitions. Throughout the paper, by a function, we mean a non-decreasing function over natural numbers.

The states of an ATM are partitioned into *existential* and *universal* states. We can view a computation of an ATM as a tree of IDs (Instantaneous Description, which consists of the state of the finite control, tape-head positions and contents of tapes). A tree is said to be a *computation tree* on an input w if its nodes are labelled by IDs such that the sons of any non-leaf labelled by a universal ID (an existential ID, respectively) consists of all (exactly one, respectively) of the successors of that ID. If the sons of an existential ID also consists of all of the successors of that IDs, then the tree is called a *full computation tree*. A computation tree is *accepting* if the root is labelled by the initial ID and all the leaves are labelled by accepting IDs. See [2] for the formal definition of ATMs.

Without loss of generality, we may assume that an ATM M can write a blank symbol. This is because M can simulate a behavior of the blank symbol by using an additional symbol if M cannot write the blank symbol.

Definition 1. Let $R(n)$ be a function. It is said that an ATM M runs in reversal $R(n)$ if, for any accepted input of length n, there is an accepting computation tree such that along each computation path, M makes at most $R(n)$ reversals over the work-tape. Such an ATM is also called an $R(n)$ reversal bounded ATM.

Definition 2. It is said that an ATM M is a σ_k machine (π_k machine, respectively) if M starts in an existential state (a universal state, respectively) and makes at most $k-1$ alternations between existential and universal states for any input.

Definition 3. It is said that a function $R(n)$ is reversal constructible (space constructible, respectively) if there is a deterministic off-line 1-tape Turing machine M such that, given an input of length n, M can make a unary representation of the value of $R(n)$ in reversal $O(R(n))$ (space $O(R(n))$, respectively).

We introduce a restricted $R(n)$ reversal-bounded off-line 1-tape ATM M satisfying the following: Let us define an ID of M to be a tuple (q,i,h,α), where q is a state, i is an input head position, h is a work-tape head position, and α is a content of the work-tape from the leftmost square to the rightmost square containing a non-blank

symbol. Then for $\forall w \in L(M)$, there is an accepting computation tree such that for every computation path $(q_0, i_0, h_0, \alpha_0), \ldots, (q_a, i_a, h_a, \alpha_a)$ from the root to a leaf, if q_j is existential (universal, respectively) and q_{j+1} is universal (existential, respectively) for any $0 \leqslant j < a$, then $|\alpha_j|$ [1] is less than or equal to $O(R(n)\log n)$. That is, M always limits the space to $O(R(n)\log n)$ when alternating the states. This time, if M is a σ_k machine (a π_k machine, respectively), then it is called an $r\sigma_k$ machine (an $r\pi_k$ machine, respectively). We should note that M can use more than $O(R(n)\log n)$ space during the computation from an alternation to the next alternation since it can write a blank symbol. Similarly, we define a restricted $R(n)$ reversal-bounded 1-tape ATM M as follows: Let us define an ID of M to be a string $\alpha q \beta$, where q denotes a state, $\alpha\beta$ denotes a content of the work-tape from the leftmost square to the rightmost square containing a non-blank symbol, and the head is on the leftmost symbol of β. Then for $\forall w \in L(M)$, there is an accepting computation tree such that for every computation path $\alpha_0 q_0 \beta_0, \ldots, \alpha_a q_a \beta_a$ from the root to a leaf, if q_j is existential (universal, respectively) and q_{j+1} is universal (existential, respectively) for any $0 \leqslant j < a$, then $|\alpha_j \beta_j|$ is less than or equal to $R(n)$.

We give some notations for classes of languages, which are used in the rest of the paper. Let $R(n)$ be a function. For any $k \geqslant 1$,

off-$\Sigma_k$reve$(R(n))$: The class of languages accepted by $O(R(n))$ reversal-bounded off-line 1-tape σ_k machines.

off-$r\Sigma_k$reve$(R(n))$: The class of languages accepted by $O(R(n))$ reversal-bounded off-line 1-tape $r\sigma_k$ machines.

off-$\Pi_k$reve$(R(n))$: The class of languages accepted by $O(R(n))$ reversal-bounded off-line 1-tape π_k machines.

off-$r\Pi_k$reve$(R(n))$: The class of languages accepted by $O(R(n))$ reversal-bounded off-line 1-tape $r\pi_k$ machines.

The classes for space complexity are also defined similarly. In addition, by removing the prefix "off-", we denote the class of languages for 1-tape machines.

Let $\mathscr{C}$ be a class of languages. Then, $Co\text{-}\mathscr{C} = \{L | \overline{L} \in \mathscr{C}\}$, where $\overline{L}$ means a complement of L.

3. Modifications of basic results

In Sections 3, 4 and 5, we are concerned with off-line 1-tape machines. Hence, we will omit the word "off-line 1-tape" if any confusion does not occur. To get the main result, we need to deal with σ_1 machines and π_1 machines having any string in the work-tape at the starting point, and therefore extend well-known results for NTMs to these machines. For such a machine M, it is said that M accepts (w, α) if M starts with an input w and a work-tape string α, and accepts w. It is also said that M *terminates*

[1] $|\alpha_j|$ denotes the length of string α_j.

on (w,α) if the full computation tree is finite, that is, every computation path from the root to a leaf is finite.

First we will define two modified versions of machines, *machines with an end-marker* and *segmented machines*, as follows: A machine with an end-marker has an end-marker in the work-tape and during the computation, the machine uses only the space between the leftmost square and the end-marker. It never moves the head to the right of the end-marker. A segmented machine M is a machine such that its work-tape is initially separated into segments consisting of cn squares by a special symbol, that is, M initially has the work-tape separated into segments. Each segment just corresponds to one square of an ordinary TM, and during the computation, M uses only the leftmost square of each segment. It follows that M is said to be an $S(n)$ space-bounded segmented machine if it uses at most $O(S(n))$ segments on the work-tape. Furthermore, when $a_1 \cdots a_m$ is initially given in the work-tape, each a_j of the string is given the leftmost square of the jth segment (see Fig. 1). From the above definition, it is obvious that an $S(n)$ space-bounded segmented ATM is straightforwardly simulated by an $S(n)$ space-bounded ATM and vice versa.

Now we extend Immerman's theorem to segmented TMs having initially any tape string. Since the proof is a straightforward extension of Immerman's proof, it is omitted.

Proposition 1 (Immerman [5]). *Let* $S(n) \geqslant \log n$ *be a space constructible function. Then,*

$$\textit{off-}\Sigma_1 \textit{space}(R(n)) = \textit{Co-off-}\Sigma_1 \textit{space}(R(n)).$$

Proposition 2 (Modified version). *Let* $S(n) \geqslant \log n$ *be a space constructible function. Let* M *be an* $S(n)$ *space-bounded segmented* σ_1 *machine. Then there exists an* $O(S(n))$ *space-bounded segmented* σ_1 *machine* N *satisfying the following: for any input string* w *and any work-tape string* α *with* $|\alpha| \leqslant S(n)$, M *accepts* (w,α) *if and only if* N *rejects* (w,α).

The following propositions, Proposition 4 and Proposition 5, are extensions of Greibach's theorem (Proposition 3). However, the proofs are not straightforward. This

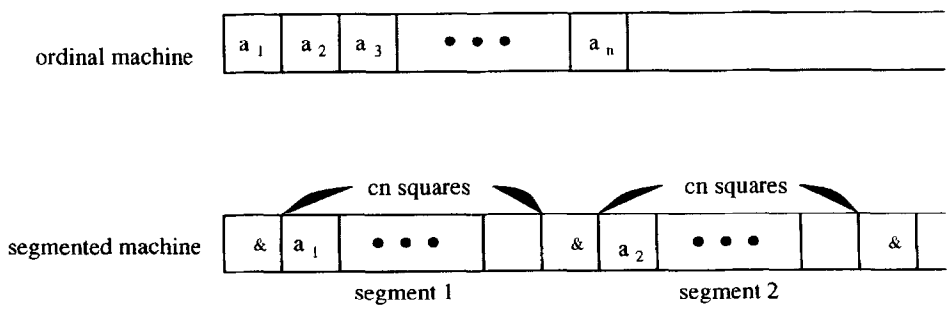

Fig. 1. Correspondence between an ordinal machine and a segmented machine.

is because machines initially may have any string in the work-tape. A segmented machine is introduced to overcome the difficulty and to get a relationship between reversal and space.

Proposition 3 (Greibach [3]). *Let $S(n)$ be a function. Then,*

$$\mathit{off}\text{-}\Sigma_1\,\mathit{reve}(S(n)) = \mathit{off}\text{-}\Sigma_1\,\mathit{space}(S(n)\log n).$$

The next two propositions are modified versions of Proposition 3.

Proposition 4 (Modified version). *Let $S(n)$ be a function. Let M be an $S(n)\log n$ space-bounded segmented σ_1 machine. Then there exists an $O(S(n))$ reversal-bounded σ_1 machine N with an end-marker satisfying the following: for any input string w and any work-tape string α with $|\alpha| \leqslant S(n)\log n$, (1) M accepts (w,α) if and only if N accepts (w,α), (2) N terminates on (w,α).*

Proof. The basic part of the proof is based on Greibach's. However, we cannot straightforwardly extend Greibach's proof to this proposition. The difficulty is that M may initially have any string α on the work-tape. A segmented machine can overcome the difficulty. Without loss of generality, we may assume that the set of the tape symbols of M is $\{0,1\}$.

Now let us describe the detail of N. We regard the work-tape of M as a collection of $S(n)$ blocks, each of which consists of $\log n$ segments. By the property of a segmented machine, $\log n$ segments are simulated by $\log n$ squares. Moreover, as described in [4], $\log n$ squares can be simulated by four $O(n)$ space-bounded counters. Hence, as in [3], a counter machine M_1 with $4S(n)$ counters can simulate M if counters of M_1 are initially set with appropriate values according to string α. Hence, N simulates M_1 instead of M. An ID of this counter machine is defined as

$$q\#i\#a\#\delta\#ACT_1\#count_{1,1}\#count_{1,2}\#count_{1,3}\#count_{1,4}\#\#$$

$$\cdots\#\#ACT_{S(n)}\#count_{S(n),1}\#count_{S(n),2}\#count_{S(n),3}\#count_{S(n),4},$$

where q, i, a and δ are a state, a unary representation of the input head position, the input symbol on the head, and the next transition defined by q, a and the content of the active counters, respectively. ACT_i takes E or A as its value and displays the status of the four counters, $count_{i,1}$, $count_{i,2}$, $count_{i,3}$ and $count_{i,4}$, where E and A mean the inactive status and the active status, respectively. All works that N does is to guess an accepting sequences of IDs, and to check the correctness of the guessed sequence in reversal $O(S(n))$. N has three tracks in the work-tape. The first track, track 1, is used in the same manner as the tape of M, the second track, track 2, is used to store IDs of M_1, and the third track, track 3, is used as an auxiliary counter to make the initial ID. In the proof of Greibach, since the tape of M was initially all empty, N can easily compute the initial ID of M_1. However, our models may have initially any

string in the work-tape. Therefore, N is required to compute the initial ID efficiently. The following procedure does this.

Procedure INITIAL

{Comment: for each block β_i $(1 \leqslant i \leqslant S(n))$ of M, N processes every β_i in parallel. This procedure changes the content of each block into a unary representation.}

Let $\beta_i = b_{i,1}b_{i,2}\cdots b_{i,\log n}$, where $b_{i,j}$ is the symbol of the leftmost square of the jth segment of the ith block.

For $j = 1$ to $\log n$, do the following:

Step 1: N sweeps β_1 to $\beta_{S(n)}$ once, and inspects whether $b_{i,j} = 0$ or 1. If $b_{i,j} = 0$, then it stores a 0 in the leading square of the block, and if $b_{i,j} = 1$ then it stores a 1. And then N marks $b_{i,j}$.

Step 2: {Comment: N computes a unary representation of 2^{j-1}. In this step, suppose that there already exists a unary representation of 2^{j-2} in track 3.}

(2-1) N sweeps β_1 to $\beta_{S(n)}$ once, and does the following: N moves the input head two squares right each time it sees a 1 on track 3. After seeing 2^{j-2} 1's, the input head is just on position 2^{j-1}. N moves the head to the $(i+1)$th block, and writes the unary representation of 2^{j-1} using the input tape. Thus in one sweep over the work tape, N can replace 2^{j-2} in track 3 with 2^{j-1} for blocks with even numbers.

(2-2) To replace for odd number blocks, N sweeps over the tape once more. This time, N copies the content of the $2j$th block to the $(2j+1)$th block using the input tape. At this point, the counter of every block has become 2^{j-1}.

(2-3) Finally, N makes another sweep from the left to the right, and if the leading square of blocks has 1 then N adds the value of track 3 to track 2. As N can store the value of track 3 in the input tape, this work is done in O(1) reversals.

Thus, N can change a binary representation of $\log n$ bits to a unary representation in $O(\log n)$ reversals, because it requires only O(1) reversals per one bit.

End of INITIAL

The above procedure sets the initial ID of M_1. The main part of N is as follows:

Step 1: N computes the initial ID α_1 by using INITIAL, and then guesses an accepting sequence of IDs, $\alpha_1, \alpha_2, \ldots, \alpha_m$, and stores $\alpha_1\$\alpha_1\$\$\cdots\$\$\alpha_m\$\alpha_m$ in the work-tape. After that, N checks whether or not the first α_j equals the second α_j for $1 \leqslant j \leqslant m$. Since most of this work is to check counters, this can be done in reversal $O(S(n))$ using the input tape.

Step 2: N checks the following two:

(2-1) For each ID α_j, checking whether or not the input symbol at the position i equals symbol a written in α_j.

Since the number i is represented in unary notation, this is done in O(1) reversals.

(2-2) Checking the correctness of the transition δ.
To check the transition δ in α_j, N knows the next transition by seeing the first α_j. After that, it compares the transition with δ in the second α_j. If not equal, then N enters the rejecting state. This work is also done in O(1) reversals.

Step 3: For $\alpha_j\$\alpha_j$ $(1 \leqslant i \leqslant m)$, N changes the second α_j to the next ID according to the transition δ. After that, N checks whether the updated α_j equals α_{j+1} or not. If equals for every j, then N accepts the input; otherwise rejects. This work is done in $O(S(n))$ reversals, because N needs to check the equality of $4S(n)$ counters.

As seen from the simulation, N runs in $O(S(n))$ reversals. □

Proposition 5 (Modified version). *Let $S(n)$ be a function. Let M be an $S(n)$ reversal-bounded σ_1 machine. Then there exists an $O(S(n)\log n)$ space-bounded segmented σ_1 machine N satisfying the following: for any input string w and any work-tape string α with $|\alpha| \leqslant O(S(n)\log n)$, M accepts (w,α) if and only if N accepts (w,α).*

Proof. By the same technique as Lemma 7 in [11] using crossing sequences, it is easy to show this proposition. Thus, the proof is easy, and it is omitted. □

Remark. We do not know whether the restriction $|\alpha| \leqslant O(S(n)\log n)$ in the above proposition can be removed or not. Since α is any string, the crossing sequence argument requires at least space $|\alpha|$. This is why we introduced a restricted machine for an off-line 1-tape machine.

4. *off-r*Σ_k*reve*$(R(n))$ collapses

First, to get the main result, we provide some lemmas.

Lemma 1. *Let $R(n)$ be a reversal constructible function, and let M be an $R(n)$ reversal-bounded segmented π_1 machine with an end-marker. Then there exists an $O(R(n))$ reversal-bounded π_1 machine N with an end-marker satisfying the following: for any input string w and any work-tape string α, (1) M accepts (w,α) if and only if N accepts (w,α), (2) N terminates on (w,α).*

Proof. There are two difficulties to obtain N. One is that M may continue to move right making no reversal, and the other is that M may continue to move on the same square making no head-movement, that is, it only moves the input head and changes the internal state. Since M has an end-marker, the former is easily overcome; namely M reverses the head if reaching the end-marker. The latter is overcome as follows. The number of stationary moves M makes successively is at most kn, where k is the number of states. Since M is a segmented machine, N can count the number of stationary moves using each segment. Indeed, N is constructed as follows: N has three tracks; track 1, track 2 and track 3.

Step 1: N makes a block of $R(n)$ squares on track 3. This is done in $O(R(n))$ reversals, because $R(n)$ is reversal constructible.

Step 2: By a right sweep and a left sweep, we mean the movement of the head of M from the left to the right and from the right to the left, respectively. N simulates M using track 1 for the work-tape of M. Therefore, track 1 is also partitioned into segments as the tape of M. N behaves until M makes a reversal as follows: During the simulation, N has an end-marker at the same position as M, and never moves the head right than the end-marker. Moreover, N counts the number of reversals made by M using track 3, and if the count is over $R(n)$ then N rejects the input and halts.

(2-1) For moves other than a stationary move,
  (2-1-1) If it is in a right sweep, then N moves in the same manner as M.
  (2-1-2) If it is in a left sweep, then N moves in the same manner as M except for using the rightmost square in each segment.

(2-2) If M enters a stationary move, then N counts the number of successive stationary moves using track 2. In other words, N writes the same symbol as M writes, and moves the head right (if in a right sweep) or left (if in a left sweep). Since M is a segmented machine, each segment has kn squares for one square of ordinary TMs. Hence, N can count the number of successive stationary moves in the segment. If the count is over kn then N rejects the input.

(2-3) If M makes a reversal, then go to Step 3.

Step 3: This step is done when M reverses the head. If its reversal is from a right sweep to a left sweep, then the rightmost symbol in each segment on track 2 is the latest symbol for the corresponding square of M. Hence, N writes the symbol in the rightmost square in each segment on track 1 and clears track 2. Similarly, if its reversal is from a left sweep to a right sweep, then the leftmost symbol in each segment on track 2 is the latest symbol for the corresponding square. Hence, N writes the symbol in the leftmost square in each segment on track 1 and clears track 2. After this processing, N again goes to Step 2.

It is clear that the number of reversals made by N is $O(R(n))$ and N satisfies the conditions. □

Lemma 2. *Let $R(n)$ be a function, and let M be an $R(n)$ reversal-bounded σ_1 machine (π_1 machine, respectively) with an end-marker that terminates on any pair (w, α). Then there exists an $O(R(n))$ reversal-bounded π_1 machine (σ_1 machine, respectively) N with an end-marker satisfying the following: for any input string w and any work-tape string α, M accepts (w, α) if and only if N rejects (w, α).*

Proof. We consider the case that M is a σ_1 machine. Then N simulates M by replacing existential states, accepting states and rejecting states of M with universal states, rejecting states and accepting states, respectively. Note that since M always terminates

on all computation paths, the states other than the accepting states can be regarded as rejecting states. Thus the lemma holds. □

Lemma 3. *Let $R(n)$ be a space and reversal constructible function, and let M be an $R(n)$ reversal-bounded σ_1 machine with an end-marker. Then there exists an $O(R(n))$ reversal-bounded π_1 machine N satisfying the following: for any input string w and any work-tape string α with $|\alpha| \leqslant O(R(n)\log n)$, M accepts (w,α) if and only if N accepts (w,α).*

Proof. The desired N can be obtained as follows. For $\forall w$ and $\forall\alpha$ with $|\alpha| \leqslant O(R(n)\log n)$, by Proposition 5, there exists an $O(R(n)\log n)$ space-bounded segmented σ_1 machine N_1 such that M accepts (w,α) iff N_1 accepts (w,α). By Proposition 2, there exists an $O(R(n)\log n)$ space-bounded segmented σ_1 machine N_2 such that N_1 accepts (w,α) iff N_2 rejects (w,α), and therefore M accepts (w,α) iff N_2 rejects (w,α). By Proposition 4, there exists an $O(R(n))$ reversal-bounded σ_1 machine N_3 with an end-marker such that N_2 accepts (w,α) iff N_3 accepts (w,α), and therefore M accepts (w,α) iff N_3 rejects (w,α). By Lemma 2, there exists an $O(R(n))$ reversal-bounded π_1 machine N with an end-marker such that N accepts (w,α) iff N_3 rejects (w,α), and therefore M accepts (w,α) iff N accepts (w,α). □

Lemma 4. *Let $R(n)$ be a space and reversal constructible function, and let M be an $R(n)$ reversal-bounded π_1 machine with an end-marker. Then there exists an $O(R(n))$ reversal-bounded σ_1 machine N satisfying the following: for any input string w and any work-tape string α with $|\alpha| \leqslant O(R(n)\log n)$, M accepts (w,α) if and only if N accepts (w,α).*

Proof. This is proved in the similar way to Lemma 3 by using Lemma 2, Proposition 5, Proposition 2, and Proposition 4 in this order. □

Theorem 1. *Let $R(n)$ be a space and reversal constructible function. Then, for any $k \geqslant 1$,*

$$\mathit{off}\text{-}r\Sigma_k\mathit{reve}(R(n)) = \mathit{off}\text{-}\Sigma_1\mathit{reve}(R(n)).$$

Proof. Since an $r\sigma_1$ machine and a σ_1 machines are the same, it suffices to show that an $O(R(n))$ reversal-bounded $r\sigma_k$ machine ($k \geqslant 1$) is simulated by an $O(R(n))$ reversal-bounded σ_1 machine. It is proved by induction on the number k of alternations.

Case $k = 1$: Obvious.

Case $k > 1$: Suppose that $r\sigma_{k-1}$ machine is simulated by σ_1 machine. Then we can construct an $r\sigma_{k-1}$ machine N simulating an $r\sigma_k$ machine M as follows.

Step 1: N guesses the number m of squares which M uses, and makes m segments of length cn for a constant c. Moreover, N guesses space enough to simulate M, puts an end-marker on the work-tape, and computes the value of $dR(n)$ for a constant d. This value is used by machines in Step 3.

Step 2: N behaves as a segmented machine when simulating M, that is, N uses one segment per a square of M. During the simulation, N counts the number of alternations made by M. If M makes the $(k-1)$th alternation by a transition, then N also executes the same transition. This time, however, N does not make the alternation, that is, if the old state is universal (existential, respectively), then the new state also remains universal (existential, respectively). After that, N marks the current head position, and returns the head to the leftmost square. Furthermore N stores the input head position in the work-tape.

Step 3: Just after Step 2, the rest of the computation of M can be viewed as either a segmented σ_1 machine or a segmented π_1 machine with an end-marker having a work-tape string α with $|\alpha| \leqslant O(R(n)\log n)$. This is because M is an $r\sigma_k$ machine. Hence if it is a segmented σ_1 machine, then N uses the π_1 machine in Lemma 3. On the other hand, if it is a segmented π_1 machine, then N can use the σ_1 machine in Lemma 4. Because, by Lemma 1, we can construct a π_1 machine with an end-marker which always terminates. Thus N becomes an $r\sigma_{k-1}$ machine, and accepts the same languages as M.

It is easy to show that the number of reversals of N are $O(R(n))$. Because $R(n)$ is a space and reversal constructible function, in Step 1, N requires only $O(R(n))$ reversals. And in Step 2 and 3, so does N. □

Corollary 1. *Let $R(n)$ be a space and reversal constructible function. Then, for $k \geqslant 2$, off-$r\Pi_k$ reve$(R(n))$ = off-Σ_1 reve$(R(n))$.*

Proof. It follows from Theorem 1, because $r\pi_k$ machine can be simulated by $r\sigma_{k+1}$ machine. □

5. Equality of *off-$\Sigma_1$reve$(R(n))$* and *off-$\Pi_1$reve$(R(n))$*

In the previous section, we showed that the class of languages accepted by $O(R(n))$ reversal-bounded off-line 1-tape $r\sigma_k$ machines or $O(R(n))$ reversal-bounded off-line 1-tape $r\pi_k$ machines equals the class of languages accepted by $O(R(n))$ reversal-bounded σ_1 machines. It follows from this result that *off-$\Pi_1$reve$(R(n))$* $\subseteq$ *off-$\Sigma_1$reve$(R(n))$* since π_1 machines and $r\pi_1$ machines are the same. However, we do not know whether or not *off-$\Sigma_1$reve$(R(n))$* $\subseteq$ *off-$\Pi_1$reve$(R(n))$*. In this section, we will show that this inclusion holds. Hence, it follows from this result that *off-$\Sigma_1$reve$(R(n))$=off-$\Pi_1$reve$(R(n))$*.

Now, we modify a machine M with an end-marker as follows: for any input of length n, the end-marker is initially put on the $n^{cR(n)}$th square, where c is a constant. The action of M is the same as in the previous definition. Throughout this section, we consider that machines with an end-marker all have the end-marker on the $n^{cR(n)}$th square. Then the following holds.

Lemma 5. *Let $R(n)$ be a function. Then the class of languages accepted by $O(R(n))$ reversal-bounded σ_1 machines with an end-marker is equal to the class of languages accepted by $O(R(n))$ reversal-bounded σ_1 machines.*

Proof. Let M and N be an $O(R(n))$ reversal-bounded σ_1 machine with an end-marker and an $O(R(n))$ reversal-bounded σ_1 machine, respectively. Then it is obvious that M is simulated by N, since N can guess the position of the end-marker and simulates M. Hence, it suffices to show that N is simulated by M. This can be shown by crossing sequence argument. The crossing sequences of N are defined as a sequence of (ihp, q), where ihp and q denote the input head position and the state, respectively, at crossing a boundary between squares. Since N is $O(R(n))$ reversal-bounded, the length of the crossing sequence of each boundary is at most $O(R(n))$, where the length of a crossing sequence means the number of pairs (ihp, q) in the crossing sequence. Hence the number of different crossing sequences of length $O(R(n))$ is at most $n^{cR(n)}$ for a constant c. Let $w \in L(N)$. Then there is an accepting computation of N on w such that N uses only $n^{cR(n)}$ squares. Because if N uses more than $n^{cR(n)}$ squares, there is always at least two boundaries having the same crossing sequence. Therefore, N can accept w not using the squares between these two boundaries. It follows that N is simulated by M since the position of the end-marker of M is $n^{cR(n)}$. □

This lemma says that it is sufficient to show that σ_1 machines with an end-marker is simulated by a π_1 machine. In what follows, we will show this.

Lemma 6. *Let $R(n)$ be a reversal constructible function. Let M be an $R(n)$ reversal-bounded σ_1 machine with an end-marker. Then, there exists an $O(R(n))$ reversal-bounded σ_1 machine N with an end-marker such that N accepts the same language as M and terminates for any input.*

Proof. This lemma can be proved in the same way as Lemma 1. That is, N makes $n^{cR(n)}$ segments of length $O(n)$ on the work-tape, and simulates M using one segment for one square of M. The way to use the segments is the same as in Lemma 1. Furthermore, since $R(n)$ is a reversal constructible function, N can count the number of reversals made by M during the simulation. Thus the lemma holds. □

Lemma 7. *Let $R(n)$ be a function. Let M be an $R(n)$ reversal-bounded σ_1 machine with an end-marker, which does terminate for every input. Then, $\overline{L(M)}$ is accepted by an $O(R(n))$ reversal-bounded π_1 machine.*

Proof. This lemma is proved in the same way as Lemma 2. □

Theorem 2. *Let $R(n)$ be a space and reversal constructible function. Then,*

$$\textit{off-}\Pi_1\,\textit{reve}(R(n)) = \textit{off-}\Sigma_1\,\textit{reve}(R(n)).$$

Proof. It suffices to show that for any $L \in \Sigma_1 reve(R(n))$, $L \in \Pi_1 reve(R(n))$. Let $L \in \Sigma_1 reve(R(n))$. From Proposition 3, there exists an $O(R(n)\log n)$ space-bounded σ_1 machine which accepts L. From Proposition 1, there exists an $O(R(n)\log n)$ space-bounded σ_1 machine which accepts $\overline{L}$. From Proposition 3, there exists an $O(R(n))$ reversal-bounded σ_1 machine which accepts $\overline{L}$. From Lemmas 5 and 6, there exists an $O(R(n))$ reversal-bounded σ_1 machine with an end-marker which accepts $\overline{L}$. From Lemma 7, there exists an $O(R(n))$ reversal-bounded π_1 machine which accepts L. Thus, it is shown that $L \in \Pi_1 reve(R(n))$. Hence, the theorem holds. □

The next corollary immediately follows from Theorems 1 and 2.

Corollary 2. *Let $R(n)$ be a space and reversal constructible function. Then, for any $k \geqslant 1$,*

$$off\text{-}r\Sigma_k reve(R(n)) = off\text{-}\Pi_1 reve(R(n)), \text{ and}$$

$$off\text{-}r\Pi_k reve(R(n)) = off\text{-}\Pi_1 reve(R(n)).$$

6. 1-tape models

In this section, we deal with 1-tape machines, but not off-line. Hence, we will omit the word "1-tape" for σ_k machines and π_k machines if any confusion does not occur. The results for 1-tape machines can be obtained more easily than that of off-line 1-tape machines. In this case, we do not need segmented machines and machines having initially any string. Therefore, we can use the existing results as follows.

6.1. Basic propositions

Proposition 6 (Immerman). *Let $S(n) \geqslant n$ be a space constructible function. Let M be an $S(n)$ space-bounded σ_1 machine. Then there exists an $O(S(n))$ space-bounded σ_1 machine N satisfying the following: for any input string w, M accepts w if and only if N rejects w.*

The following two propositions play the similar role to Propositions 4 and 5 of off-line 1-tape machines. These are modified versions of Proposition 3 in [12], and the proofs are almost the same as in [12]. Hence, we omit the proofs.

Proposition 7. *Let $S(n) \geqslant n$ be a function. Let M be an $S(n)$ space-bounded σ_1 machine. Then there exists an $O(S(n))$ reversal-bounded σ_1 machine N with an end-marker satisfying the following: for any input string w, (1) M accepts w if and only if N accepts w, (2) N terminates on w.*

Proposition 8. *Let $S(n) \geqslant n$ be a function. Let M be an $S(n)$ reversal-bounded σ_1 machine. Then there exists an $O(S(n))$ space-bounded σ_1 machine N satisfying the following: for any input string w, M accepts w if and only if N accepts w.*

6.2. Main results

Lemma 8. *Let $R(n)$ be a reversal constructible function, and let M be an $R(n)$ reversal-bounded π_1 machine with an end-marker. Then there exists an $O(R(n))$ reversal-bounded π_1 machine N with an end-marker satisfying the following: for any input string w, (1) M accepts w if and only if N accepts w, (2) N terminates on w.*

Proof. The proof is similar to Lemma 1, but we do not need segmented machines. As in Lemma 1, there are two difficulty to obtain N. One is that M may continue to move right making no reversal, and the other is that M continues to move on the same square making no head-movement, that is, it only changes the internal state and the tape symbol. Since M has an end-marker, the former is easily overcome; namely M reverses the head if reaching the end-marker. The latter is overcome as follows. The number of stationary moves M makes successively is at most $k \times s$, where k is the number of states and s is the number of tape symbols. Since $k \times s$ is a constant, N can count the number of stationary moves with the finite control. The rest of the proof is the same as Lemma 1. □

Lemma 9. *Let $R(n)$ be a function, and let M be an $R(n)$ reversal-bounded σ_1 machine (π_1 machine, respectively) with an end-marker that terminates on any input w. Then there exists an $O(R(n))$ reversal-bounded π_1 machine (σ_1 machine, respectively) N with an end-marker satisfying the following: for any input string w, M accepts w if and only if N rejects w.*

Proof. Similar to Lemma 2. □

Lemma 10. *Let $R(n) \geqslant n$ be a space and reversal constructible function, and let M be an $R(n)$ reversal-bounded σ_1 machine with an end-marker. Then there exists an $O(R(n))$ reversal-bounded π_1 machine N satisfying the following: for any input string w, M accepts w if and only if N accepts w.*

Proof. Similar to Lemma 3, except for using Propositions 6–8, and Lemma 8. □

Lemma 11. *Let $R(n) \geqslant n$ be a space and reversal constructible function, and let M be an $R(n)$ reversal-bounded π_1 machine with an end-marker. Then there exists an $O(R(n))$ reversal-bounded σ_1 machine N satisfying the following: for any input string w, M accepts w if and only if N accepts w.*

Proof. Similar to Lemma 4, except for using Propositions 6–8, and Lemmas 8 and 9. □

Now we are ready to show the results for 1-tape machines.

Theorem 3. *Let $R(n) \geqslant n$ be a space and reversal constructible function. Then, for any $k \geqslant 1$,*

$$r\Sigma_k reve(R(n)) = \Sigma_1 reve(R(n)).$$

Proof. Since an $r\sigma_1$ machine and a σ_1 machines are the same, it suffices to show that an $O(R(n))$ reversal-bounded $r\sigma_k$ machine ($k \geqslant 1$) is simulated by an $O(R(n))$ reversal-bounded σ_1 machine. it is proved by induction on the number k of alternations.

Case $k = 1$: Obvious.

Case $k > 1$: Suppose that $r\sigma_{k-1}$ machine is simulated by σ_1 machine. Then we can construct an $r\sigma_{k-1}$ machine N simulating an $r\sigma_k$ machine M as follows.

Step 1: N guesses space enough to simulate M, puts an end-marker on the work-tape, and computes the value of $R(n)$. This value is used by machines in Step 3.

Step 2: N behaves as M when simulating M. During the simulation, N counts the number of alternations made by M. If M makes the $(k-1)$th alternation by a transition δ, then N also performs the same transition. This time, however, N does not make the alternation, that is, if the old state is universal (existential, respectively), then the new state also remains universal (existential, respectively). After that, N marks the current head position, and returns the head to the leftmost square.

Step 3 Just after Step 2, the rest of the computation of M can be viewed as either a σ_1 machine with an end-marker or a π_1 machine with an end-marker having an input $|\alpha| \leqslant R(n)$. This is because M is an $r\sigma_k$ machine. Hence if it is a σ_1 machine, then N uses the π_1 machine in Lemma 10, and if it is a π_1 machine, then N can use the σ_1 machine in Lemma 11. Because, by Lemma 8, we can construct a π_1 machine with an end-marker which always terminates. Thus N becomes an $r\sigma_{k-1}$ machine, and accepts the same languages as M.

It is easy to show that the reversals of N are $O(R(n))$. Because, in Step 1, N requires only $O(R(n))$ reversals, and in Steps 2 and 3, so does N. □

Remark. Lemmas 10 and 11 have the condition $R(n) \geqslant n$. Hence, to use these lemmas in the proof of the above theorem, we need to have the condition $|\alpha| \leqslant R(n)$ in Step 3. This is why we introduced a restrict machine for a 1-tape machine.

Corollary 3. *Let $R(n) \geqslant n$ be a space and reversal constructible function. Then, for $k \geqslant 2$,*

$$r\Pi_k reve(R(n)) = \Sigma_1 reve(R(n)).$$

Proof. It follows from Theorem 3, because $r\pi_k$ machine can be simulated by $r\sigma_{k+1}$ machine. □

Similarly, we can get the similar results to Theorem 2 and Corollary 2 as follows. We omit these proofs because they are almost the same as Theorem 2 and Corollary 2.

Theorem 4. *Let $R(n) \geqslant n$ be a space and reversal constructible function. Then,*

$$\Pi_1 reve(R(n)) = \Sigma_1 reve(R(n)).$$

Corollary 4. *Let $R(n) \geqslant n$ be a space and reversal constructible function. Then, for any $k \geqslant 1$,*

$$r\Sigma_k reve(R(n)) = \Pi_1 reve(R(n)),$$

and

$$r\Pi_k reve(R(n)) = \Pi_1 reve(R(n)).$$

References

[1] B.S. Baker and R.V. Book, Reversal bounded multipushdown machines, *J. Comput. System Sci.* **8** (1974) 315–322.

[2] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, *J. Assoc. Comput. Mach.* **28** (1981) 114–133.

[3] S.A. Greibach, Visits, crosses and reversal for nondeterministic off-line machines, *Inform. and Control* **36** (1978) 174–216.

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

[5] N. Immerman, Nondeterministic space is closed under complementation, *SIAM J. Comput.* **17** (1988) 935–938.

[6] M. Kutylowski, M. Liskiewicz and K. Lorys, Reversal complexity classes for alternating Turing machines, *SIAM J. Comput.* **19** (1990) 207–221.

[7] M. Liskiewicz and K. Lorys, On reversal complexity for alternating Turing machines, *Proc. 30th IEEE FOCS* (1989) 618–623.

[8] M. Liskiewicz, K. Lorys and M. Piotrow, Note on reversal bounded alternating Turing machines, *Theoret. Comput. Sci.* **54** (1987) 331–339.

[9] L.J. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977) 1–22.

[10] S. Toda, $\Sigma_2 SPACE(n)$ is closed under complement, *J. Comput. System Sci.* **35** (1987) 145–152.

[11] H. Yamamoto, Reversal-space trade-offs for simultaneous resource-bounded nondeterministic Turing machines, *ICALP'93 Proc.*, Lecture Notes in Computer Science, Vol. 700 (Springer, Berlin 1993) 203–214.

[12] H. Yamamoto and S.Noguchi, Comparison of the power between reversal-bounded ATMs and reversal-bounded NTMs, *Inform. and Comput.* **75** (1987) 144–161.

[13] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977) 23–33.