Tight bounds for the space complexity of nonregular language recognition by real-time machines

Abuzer Yakaryılmaz\textsuperscript{1,*} and A.C. Cem Say\textsuperscript{2,**}

\textsuperscript{1}University of Latvia, Faculty of Computing, Raina bulv. 19, Riga, LV-1586, Latvia
abuzer@lu.lv
\textsuperscript{2}Boğaziçi University, Department of Computer Engineering, Bebek 34342 İstanbul, Turkey
say@boun.edu.tr

August 15, 2011

Abstract. We examine the minimum amounts of useful memory for real-time, as opposed to one-way, computation using several different machine models. In most cases, we are able to show that the lower bounds established using arguments about one-way machines remain tight in the real-time case. It is shown that increasing the number of stacks of real-time pushdown automata can result in exponential improvement in the total amount of space usage for nonregular language recognition.

1 Introduction

The effects of restricting a Turing machine (TM) to a single left-to-right pass of its input string have been well studied since the early years of computational complexity theory. An important distinction in this regard is the one between real-time computation \cite{Rab63b}, in which the machine is required to spend a single computational step on each input symbol, and one-way computation \cite{SHI65}, where the input head is allowed to pause for some steps on the tape. These two modes are known to be equivalent in language recognition power in many setups. An important exception is quantum automata theory, where it has been shown that quantum versions of real-time finite automata \cite{AI99,YS11} and pushdown automata \cite{YFSA} are strictly less powerful than their more general one-way versions in some modes of language recognition.

In this paper, we compare the real-time and one-way modes of computation from the point of view of the minimum amount of memory required by computers to recognize nonregular languages. Several variants of the TM model \cite{Sip06} will be used in this study. All machines are equipped with a read-only input tape and a single worktape with a two-way read/write head. We will distinguish between the strong, middle, and weak modes of space complexity, whose definitions are reviewed below. We assume without loss of generality that none of the TMs we consider ever moves the work tape head to the left of its initial position.

* Yakaryılmaz was partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with grant 108E142 and FP7 FET-Open project QCS.
** Say’s work was partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with grant 108E142.
A TM is said to be strongly $s(n)$ space-bounded if all reachable configurations on any input of length $n$ have the work tape head at not more than $s(n)$ cells away from its initial position.

A TM is said to be middle $s(n)$ space-bounded if all reachable configurations on any accepted input of length $n$ have the work tape head at not more than $s(n)$ cells away from its initial position.

A TM is said to be weakly $s(n)$ space-bounded if there exists at least one accepting computation tree containing no configurations that have the work tape head at more than $s(n)$ cells away from its initial position, for any accepted input of length $n$.

In the rest of the paper, Section 2 considers several specializations of the alternating TM model for both tally languages (i.e. those with unary input alphabets) and general ones, establishing that most minimum space bounds for nonregular language recognition dictated by previous work on one-way versions of these machines remain tight in the real-time case. Similar questions about some other computational models like pushdown and counter automata are examined in Section 3. Section 4 contains a list of open problems.

## 2 Bounds for real-time alternating Turing machines

Table 1 depicts the known tight bounds for the minimum space complexity of nonregular language recognition by various specializations of the alternating TM model for both the general and unary input alphabet cases in the one-way mode. (We refer the reader to [Sze94,MP95] for the proofs of the facts depicted in the table.) These clearly form lower bounds for the corresponding functions for real-time TMs. We will now proceed to prove that most of these bounds are tight, by demonstrating machines of the relevant types recognizing nonregular languages within the corresponding space bound.

### Table 1. Minimum space used by one-way TMs for recognizing nonregular languages. All bounds are tight.

<table>
<thead>
<tr>
<th></th>
<th>General input alphabet</th>
<th>Unary input alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strong</td>
<td>Middle</td>
</tr>
<tr>
<td>Deterministic TM</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Nondeterministic TM</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Alternating TM</td>
<td>$\log n$</td>
<td>$\log \log n$</td>
</tr>
</tbody>
</table>

**Theorem 1.** All the logarithmic bounds in Table 1 remain tight for the real-time versions of the corresponding machines.

1 We use the head position, rather than the number of nonblank cells, [Sze94] for measuring space usage, since real-time quantum TMs are known [YPSA] to be able to recognize nonregular languages by a technique that involves just moving the work head ($O(n)$ steps) to the right, without writing any nonblank symbols.
Proof. We present a real-time deterministic TM named $D$ that recognizes a nonregular tally language using only logarithmic space. The single input symbol is $a$. The input string is assumed to be followed by the endmarker $\$ on the tape. The nonblank work tape symbols are $\langle$, $\rangle$, 0, and 1. $D$ will maintain a counter in reverse binary representation between the delimiters $\langle$ and $\rangle$ on its work tape. At the beginning of the computation, $D$ expects to read four $a$'s from the input, (rejecting otherwise,) and moves the work tape head right and then left during these four steps to write the string $\langle 0 \rangle$, and position the head on the symbol $\langle$. Having initialized its counter, $D$ enters the following loop:

- the work tape head goes from symbol $\langle$ to symbol $\rangle$ while incrementing the counter by one, and then,
- the work tape head goes from symbol $\rangle$ to symbol $\langle$, while checking whether the counter value is a power of two or not.

During the travel of the work tape head, we assume that it cannot be stationary on any symbol. Note that, if required, the place of symbol $\rangle$ on the work tape is shifted one square to the right. If $D$ reads $\$ at any time when the work tape head is not placed on the symbol $\langle$, the input is rejected. When the work tape head is placed on symbol $\langle$ and the currently scanned symbol from the input tape is $\$$, the input is accepted if the counter is a power of two, and rejected otherwise.

Let $L_D$ be the language recognized by $D$. Let $a_i$ be the $i^{th}$ shortest element of $L_D$, where $i > 0$. It is easy to see that the value of $|a_{i+1}| - |a_i|$ increases as $i$ increases. This leads to a trivial proof, using the pumping lemma for regular languages, of the nonregularity of $L_D$.

Since $D$ belongs to the most specialized machine class under consideration, and obviously uses $O(\log n)$ space to use a unary nonregular language, we conclude that all the logarithmic lower bounds in Table 1 are tight for real-time machines as well.

We note that Freivalds and Karpinski use a real-time logarithmic-size counter in order to demonstrate a lower time bound for probabilistic TMs in [FK95]. Such a counter could also be used to prove at least some of the bounds handled by Theorem 1 to be tight. To the best of our knowledge, the minimum space requirements for nonregular language recognition by these real-time machines have not been studied before.

We will now turn our attention to the double logarithmic bounds for machines with general input alphabets in Table 2. Our strategy is to modify the one-way machines which are used in demonstrating these results to obtain real-time machines with the same space complexity. The languages recognized by these new real-time machines will be “padded” versions of their counterparts in the one-way setup. Essentially, the real-time machine will consume a padding symbol ($\kappa$) for each step of the one-way machine in which the input head pauses on the tape.

**Definition 1.** Given a one-way deterministic, nondeterministic or alternating TM $D$, the real-time TM $D_{\kappa}$ of the same model variant is constructed as follows:

- The input alphabet of $D_{\kappa}$ is $\Sigma_{\kappa} = \Sigma \cup \{\kappa\}$, where $\Sigma$ is the input alphabet of $D$, and $\kappa \notin \Sigma$;
- The work tape alphabets of the two machines are identical;
- $D_{\kappa}$ skips over any prefix of $\kappa$'s in its input until it sees the first non-$\kappa$ symbol, at which point it starts its emulation of $D$;
- On each input symbol other than $\kappa$, $D_{\kappa}$ emulates the state transition and work tape action of $D$ for that symbol;
– After implementing a stationary transition (that is, one which does not move the input head,) of \( D \) in the manner described above, \( D_\kappa \) checks whether the next input symbol is \( \kappa \). If this is the case, \( D_\kappa \) implements the transition that \( D \) would be making while scanning the symbol that is the last non-\( \kappa \) symbol scanned by \( D_\kappa \), otherwise, \( D_\kappa \) rejects the input;
– After implementing a transition of \( D \) that moves \( D \)'s input head to the right, \( D_\kappa \) skips over any \( \kappa \)'s in the input without any state or work tape change, until it sees a non-\( \kappa \) symbol.

At the end, \( D_\kappa \) accepts or rejects as \( D \) would, unless it has already rejected in the eventualities described above.

Let \( h_\kappa \) be a homomorphism such that

– \( h_\kappa(x) = x \) if \( x \neq \kappa \), and,
– \( h_\kappa(\kappa) = \varepsilon \).

**Definition 2.** \( L_{-\kappa} \subseteq \Sigma^* \) is the language recognized by \( D_\kappa \), where \( D \) is a one-way TM recognizing the language \( L \subseteq \Sigma^* \).

**Lemma 1.** \( h_\kappa(L_{-\kappa}) = L \).

**Proof.** It is obvious that \( h_\kappa(L_{-\kappa}) \subseteq L \). Assume that a string \( w \in \Sigma^* \) is recognized by the one-way alternating TM \( D \). \( D \) may spend different amounts of time, pausing on different symbols, in different branches of its accepting computation tree. Let \( t \) be the maximum number of steps that \( D \) pauses for on any particular symbol in this tree. The string \( w' \), obtained by inserting \( t \) \( \kappa \)'s after every symbol of \( w \), is accepted by \( D_\kappa \).

**Remark 1.** If \( L \) is not a member of class \( \mathcal{C} \) that is closed under homomorphism, then \( L_{-\kappa} \) is also not a member of \( \mathcal{C} \). In particular, if \( L \) is nonregular, then \( L_{-\kappa} \) is also nonregular.

**Theorem 2.** If \( L \) is recognized by a strongly space \( s(n) \)-bounded one-way TM \( D \), then for any nondecreasing function \( t(n) \) such that \( s(n) \in O(t(n)) \), the real-time TM \( D_\kappa \) recognizing \( L_{-\kappa} \) is strongly \( O(t(n)) \)-space bounded.

**Proof.** For any \( w \in \Sigma^* \), the space used by \( D_\kappa \) on any input which is a member of \( \{ w_\kappa \in \Sigma^*_\kappa \mid h_\kappa(w_\kappa) = w \} \) is at most equal to the space used by \( D \) on \( w \).

**Theorem 3.** If \( L \) is recognized by a middle space \( s(n) \)-bounded one-way TM \( D \), then for any nondecreasing function \( t(n) \) such that \( s(n) \in O(t(n)) \), the real-time TM \( D_\kappa \) recognizing \( L_{-\kappa} \) is middle \( O(t(n)) \)-space bounded.

**Proof.** Similar to the proof of Theorem 2.

Consider the nonregular language
\[
L_{\text{gcm}} = \{ a^m b^M \mid M \text{ is a common multiple of all } i \leq m \}.
\]
Szepietowski \cite{Szepietowski} showed that \( L_{\text{gcm}} \) is recognized by a middle space \( \log \log(n) \)-bounded one-way alternating TM that we will call \( \mathcal{A} \).

**Corollary 1.** The real-time alternating TM \( \mathcal{A}_\kappa \) recognizing the nonregular language \( L_{\text{gcm} - \kappa} \) is middle \( O(\log \log(n)) \)-space bounded.
We are therefore able to state that the middle and weak space bounds for alternating TMs with general input alphabets in Table 1 are also tight for the real-time versions of these machines.

**Remark 2.** There is no analogue of Theorems 2 and 3 for weak space, because some accepting path may use more space but use less time (i.e. perform fewer stationary moves) than the accepting path with the best space usage in the one-way machine, and the padding process described above may then yield some strings whose only accepting paths use unacceptably large amounts of space in the real-time machine.

To examine the weak space bound case for nondeterministic machines with general input alphabets, we focus on the one-way machine $\mathcal{N}$ [Sze94] used to establish the corresponding lower bound in Table 1: The work tape of $\mathcal{N}$ is divided into three tracks. At the beginning of the computation, $\mathcal{N}$ nondeterministically selects a number $l > 1$, and writes it in binary on the top track. Any string containing an $a$ after a $b$ is rejected. For strings of the form $a^rb^s$, $\mathcal{N}$ calculates and stores the value of $r \mod (l)$ (resp., $s \mod (l)$) on the middle track (resp., bottom track). When the end-marker is read, the numbers in the middle and bottom tracks are compared. If they are not equal, the input is accepted. The nondeterministic path corresponding to $l$, namely $\text{npath}_l$, needs only $\Theta(\log(l))$ space.

If $r = s$, then $r \equiv s \mod (l)$ for all $l > 1$, and the input would be rejected. Using a number-theoretical fact ([Sze94], page 22) which states that, whenever $r \neq s$, there exists a number $l \in O(\log(r + s))$ such that $r \not\equiv s \mod (l)$, one concludes that $\mathcal{N}$ is weakly log log($n$)-space bounded, and recognizes the nonregular language $L_{j \neq k} = \{a^jb^k \mid j \neq k\}$.

We will modify $\mathcal{N}$ to obtain a real-time TM recognizing a padded version of $L_{j \neq k}$, taking extra care to handle the additional complications of weak space bounds mentioned in the above remark. We do this by making sure that nondeterministic paths with larger values of $l$ have greater runtimes than those with small values of $l$.

We rewrite the program of $\mathcal{N}$ such that $d_l$, that is, the number of stationary steps performed during the processing of a single input symbol by the nondeterministic path containing $l$ on the top track depends solely on $l$. For example, we can use the following strategy:

- The work tape head is always placed on the leftmost nonblank symbol before reading the next input symbol, and,
- The operations on the work tape are performed by shuttling the work tape head, which operates with the speed of one cell per step, between the leftmost and rightmost nonblank symbols $c > 0$ times.

By selecting a sufficiently large value for $c$, $d_l$ can be set to $c[\log l] + k$ for nondeterministic path $\text{npath}_l$, where the exact value of $k \in \mathbb{Z}$ depends on the way the numbers are stored on the work tape. Thus, we can guarantee that for any $l' > l$, the number of the stationary steps (on the input tape) of $\text{npath}_{l'}$ cannot be less than that of $\text{npath}_l$. Let $\mathcal{N}'$ be the one-way nondeterministic TM implementing this modified algorithm, and $L'_{j \neq k-\kappa}$ be the language recognized by $\mathcal{N}'$.

**Lemma 2.** The real-time nondeterministic TM $\mathcal{N}'_\kappa$ recognizing the nonregular language $L'_{j \neq k-\kappa}$ is weakly $O(\log \log(n))$-space bounded.

**Proof.** Let $w = a^rb^s$ be a member of $L_{j \neq k}$, and $l$ be the smallest number satisfying $r \neq s \mod (l)$. Then, for all $w_\kappa \in L_{j \neq k-\kappa}$ satisfying $h_\kappa(w_\kappa) = w$, the nondeterministic path of $\mathcal{N}'_\kappa$ corresponding to $l$ accepts the input. \qed
Table 2 summarizes the findings presented in this section.

### Table 2. Lower space bounds for nonregular language recognition by real-time TMs. The bounds we have shown to be tight are in boldface.

<table>
<thead>
<tr>
<th></th>
<th>General input alphabet</th>
<th>Unary input alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strong</td>
<td>Middle</td>
</tr>
<tr>
<td>Deterministic TM</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Nondeterministic TM</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Alternating TM</td>
<td>$\log n$</td>
<td>$\log \log n$</td>
</tr>
</tbody>
</table>

### 3 Other real-time computational models

The question of the minimum amount of useful space is also interesting for more restricted models, like pushdown or counter automata. The following facts are known about (one-way) pushdown automata (PDAs) with unrestricted pushdown alphabets:

**Fact 1.** All deterministic PDAs which recognize nonregular languages are weakly $\Theta(n)$-space bounded [Gab84].

**Fact 2.** Nondeterministic PDAs that use sublogarithmic space recognize only finite languages [Rei07].

**Fact 3.** There exists a weakly $\log n$-space bounded nondeterministic PDA that recognizes a nonregular language [Rei07].

Counter automata [FMR67] (essentially, PDAs with a unary pushdown alphabet,) do not seem to have been investigated deeply in this regard. As an easy corollary of Fact 1 we have

**Fact 4.** $o(n)$-space bounded real-time deterministic counter automata cannot recognize nonregular languages.

We now show that weakly $\log n$-space bounded nondeterministic real-time counter automata can recognize some nonregular languages. This bound is tight, due to Fact 2.

Let $(i)_2^r$ be the reverse of the binary representation of $i \in \mathbb{N}$. Consider the nonregular language

$$L_{rev-bins} = \{a(1)_2^r a(2)_2^r a \cdots a(k)_2^r a \mid k \in \mathbb{Z}^+\}.$$

In the following, substrings of the form $\{0, 1\}^*$ delimited by a’s will be called “blocks,” and the term $\text{block}_i$ will denote the $i$’th block to be encountered by the one-way input head. If $\text{block}_i$ is nonempty, the number it represents in reverse binary will be denoted $\text{num}(i)$.

**Theorem 4.** The complement of $L_{rev-bins}$ can be recognized by a weakly $\log n$-space bounded real-time nondeterministic counter automaton.
Proof. Along with its other tasks, the machine $M_N$ that we will build for recognizing this language will check whether the input string $w$ is generated by the regular expression \((a\{0,1\}^+)^+a\), and accept whenever it sees that $w$ is not of this form. $M_N$ splits to two main nondeterministic paths at the beginning of the computation. One path will check whether $num(1)$ fails to be 1, if not, it will try to find a pair of blocks which fail to represent consecutive numbers in the set $\{(block_1, block_2), (block_3, block_4), (block_5, block_6), \ldots\}$. When such a failure is detected, the input will be accepted. The second main path performs the same check for pairs of blocks in the set $\{(block_2, block_3), (block_4, block_5), (block_6, block_7), \ldots\}$.

At the beginning of each such check for a pair $(block_i, block_{i+1})$, the counter value is zero. Upon scanning the $k$'th bit of $block_i$, the counter is incremented to $k$, the value $B$ that the $k$'th bit of $block_{i+1}$ at this position would have if $num(i+1)$ were indeed one greater than $num(i)$ is computed, and a nondeterministic split is performed. One branch of this split, $P_{\text{first}}$, continues with the next bit of $block_i$, whereas the other branch, $P_{\text{second}}$, skips over the rest of $block_i$, and uses the counter to find the $k$'th corresponding bit of $block_{i+1}$, accepting if this bit is not $B$, or if the input ends prematurely, and rejecting otherwise. Upon reaching the $a$ at the end of $block_i$, $P_{\text{first}}$ increments the counter, computes the symbol that should not be in the corresponding position of the next block, and decrements the counter all the way through $block_{i+1}$, starting the scan of the pair $(block_{i+2}, block_{i+3})$ with an empty counter if it has not accepted earlier due to $block_{i+1}$ being too short or too long for a proper successor of $num(i)$.

If the input string is not in $L_{\text{rev-bins}}$, it should be accepted in at least one of the two main computational subtrees. Consider the earliest (i.e. shortest) accepting path, and say that it accepts when scanning $block_i$. The space used by the counter up to that point cannot exceed $\log i$, and the length of the scanned input prefix is clearly more than $i$. We conclude that this path uses $\log n$ space in the counter.

When alternating machines are considered, we can obtain a similar statement for middle space bounds:

**Theorem 5.** $L_{\text{rev-bins}}$ can be recognized by a middle log $n$-space bounded real-time alternating counter automaton.

Proof. It is easy to see that this machine need perform only universal branchings, at precisely the points corresponding to the existential branchings of the machine $M_N$ of the proof of Theorem 4. At the start, two main paths, corresponding to precisely the same sets of pairs of blocks as in $M_N$, are created. We now insist that all bits of $block_{i+1}$ match the corresponding bits of the number that is one more than $num(i)$, for all $i$. Since the counter space can never exceed $\log n$ for members of $L_{\text{rev-bins}}$, we have a middle space bound.

We have considered only a single work tape in our real-time models until now. It is known that increasing the number of work tapes does increase the computational power of these machines. We show that deterministic PDAs and counter automata with two work tapes can recognize nonregular languages using sublinear space, doing better that the bound of Fact 1 for single-stack machines:

Consider the language

$$L_{\text{even-rev-bins}} = \{a(0)2a(1)2a(2)a(3)2a \cdots a(2k)2a(2k+1)2^k \mid k > 0\}.$$  \hspace{1cm} (1)
Theorem 6. $L_{\text{even-rev-bins}}$ can be recognized by a deterministic real-time PDA with two stacks, and the total amount of space used on the stacks for accepted strings is $O(\log n)$.

Proof. Using the terminology of the previous proofs, one stack is used for checking that the pairs in $\{(\text{block}_1, \text{block}_2), (\text{block}_3, \text{block}_4), \ldots\}$ are related according to the language definition, whereas the other stack is used for the set $\{(\text{block}_2, \text{block}_3), (\text{block}_4, \text{block}_5), \ldots\}$. The space bound is calculated as in Theorem 4.

Theorem 7. For any $j > 1$, there exists a nonregular language that can be recognized by a deterministic real-time automaton with $j$ (unary) counters, such that the total amount of space used on the counters is $O(n^\frac{1}{j})$ for all input strings.

Proof. We start by considering the language

$$L_2 = \{a_1a_0a_1^2a_1a_0^3a_1 \cdots a_1a_0^k \mid k \in \mathbb{N}\},$$

(inspired by [ABBS00,Gab84]), on the alphabet $\{a_0, a_1\}$. $L_2$ is recognized by a deterministic real-time automaton with two counters. The algorithm, say, $A_2$, is similar to, in fact simpler than, that of Theorem 6. As for the space usage, assume that the machine arrives at a decision to accept or reject after scanning the $i$’th block of $a_0$’s. Since the length of the scanned input prefix is more than $\sum_{i=1}^{i-1}(i+1) = \Theta(i^2)$, and the counters have used $O(i)$ space up to that point, the machine is strongly $O(\sqrt{n})$-space bounded. Let $w_{j,k}$ denote the $k$’th shortest element of $L_j$. Note that $A_2$ accepts $w_{2,k}$ with one counter containing zero, and the other counter containing $k$.

We define

$$L_3 = \{a_2w_{2,1}a_2w_{2,2}a_2w_{2,3}a_2 \cdots a_2w_{2,k} \mid k \in \mathbb{N}\}$$

on the alphabet $\{a_0, a_1, a_2\}$. A deterministic real-time machine with three counters can recognize $L_3$ as follows. The automaton processes each block delimited by $a_2$’s by using two of the counters for implementing $A_2$. If the input head arrives at the $i$’th $a_2$ ($i > 1$), two of the counters contain zero, whereas the third counter contains $i - 1$. The two counters containing zero are now used to run $A_2$ on the $i$’th block, while the remaining counter is used to check that the number of $a_1$’s in the $i$’th block is indeed $i$. Two counters have value zero, and the third one contains $k$ when this machine accepts $w_{3,k}$. By a reasoning similar to the one about $L_2$ above, the space usage is $O(n^\frac{1}{3})$. The generalization to

$$L_{j+1} = \{a_jw_{j,1}a_jw_{j,2}a_jw_{j,3}a_j \cdots a_jw_{j,k} \mid k \in \mathbb{N}\}$$

is now simple.

Even with one work tape, the probabilistic versions of these machines [HS10,Fre79] also require less space than the deterministic versions for recognizing some languages. The language $L_{\text{even-rev-bins}}$ of Equation 1 can be recognized by a real-time probabilistic PDA with error bound $\frac{1}{4}$ as follows: Reject the input at the start with probability $\frac{1}{4}$. With the remaining probability, toss a fair coin to split to two computational paths, each of which uses its single stack to mimic the corresponding stack of Theorem 6. If the input is in $L_{\text{even-rev-bins}}$, both paths accept, yielding an overall acceptance probability of $\frac{2}{3}$. Otherwise, at least one path rejects, resulting in an acceptance probability of at most $\frac{1}{3}$. The space requirement for members of the language is $O(\log n)$, as in Theorem 6.
With a similar approach, real-time probabilistic automata with one counter that can recognize any language of the $L_j$ family from Theorem 7 using $O(n^{\frac{4}{j}})$ space for member strings, albeit with error bounds that increase with $j$, can be constructed.

4 Open questions

We conclude with a list of problems for future work.

An obvious question left open in Section 2 is whether the double logarithmic lower bounds for the recognition of nonregular tally languages by real-time nondeterministic and alternating TMs are tight.

How does alternation effect the amount of useful space for (both one-way and real-time) PDAs and counter automata?

What are the lower total space bounds for nonregular language recognition for machines with multiple work tapes, as exemplified in Section 3?

Real-time “nondeterministic” quantum Turing machines (i.e., those that accept with nonzero probability if and only if the input is a member of the language.) are known [YS10] to be able to recognize nonregular languages even with constant space, whereas no such language can be recognized by small-space probabilistic Turing machines in this mode, since these are synonymous with nondeterministic TMs (Table 2). When two-sided unbounded error is allowed, probabilistic finite automata can recognize nonregular languages [Rab63a]. What are the minimum space requirements for real-time nonregular language recognition of probabilistic TMs in the bounded-error regime [Fre85]? Can one improve on the space usage of the probabilistic machines mentioned at the end of Section 3?

Finally, what is the minimum amount of space required for a real-time quantum Turing machine to recognize a nonregular language with bounded error?

Acknowledgement. We thank Stefan D. Bruda, Giovanni Pighizzini, Juraj Hromkovič, and Rūsiņš Freivalds for their helpful answers to our questions.

References


