The Turing Machine Space Measure Revised at the Occasion of Sublog Separation

Žák, Stanislav
1997

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

The Turing machine space measure revised at the occasion of sublog separation

Stanislav Žák

Technical report No. 758

February 17, 1997

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+4202) 66 05 36 90 fax: (+4202) 85 85 789
e-mail: stan@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

The Turing machine space measure revised at the occasion of sublog separation

Stanislav Žák[1]

Technical report No. 758
February 17, 1997

Abstract

In the theory of sublog computations on Turing machines a new situation has arised after defining of so called demon-machines [12], whose tape bounds are given gratis without any construction. This alows Geffert [7] to prove separation result of the form $SPACE(s(n)) \supset SPACE(s_1(n)) \neq \emptyset$ under the separation condition $\lim_{n\to\infty} s(n)/s_1(n) = \infty$. The aim of this paper is to find the finest possible separation condition.

To capture the influence of the tape alphabet size and of the capacity of the finite control upon the amount of the tape we use a modified space measure where the (new) space complexity of a computation is given by the (old) complexity of its simulation on an arbitrarily chosen but fixed universal machine (U). We separate $SPACE^U(s(n)) - SPACE^U(s_1(n)) \neq \emptyset$ under the condition $\lim_{n\to\infty} s(n) - s_1(n+1) > K$ where K is a constant. In some special cases the enlarging of the bound in question by adding one strengthens the computational power of machines ($SPACE^U(s(n)+1) \supset SPACE^U(s(n))$).

The results hold for binary and also unary languages, and also for bounds over $\log n$, and for a certain number of types of machines (deterministic, nondeterministic, alternating ...).

Keywords

Turing machine, space complexity, hierarchy, diagonalization

[1]This research was supported by GA CR, Grant. No. 201/95/0976.

1 Introduction

One of the basic tasks of the theory of computational complexity is separating of complexity classes. For a given computational device and for a given complexity measure which controls a computational resource in question - it it necessary to find how small enlarging of the complexity bound stregthens the computational power. In 70's this question was solved for a classical computational device -Turing machines - and for classical time and space complexity measures (e.g. S. Cook, Fischer, Meyer,Seiferas, Sudborough [3,13,14,15] and others). Whenever a new computational device (e.g. RAM) or a new complexity measure (e.g. a number of questions to an oracle) are defined it is quite natural and necessary to solve the question about the separation of complexity classes.

At the beginning of 90's such a situation arised in the theory of computations on Turing machines with sublogarithmic tape. This theory was intensively developped for some years, e.g. [2,4,5,6,8,9,10,11,16]. In the context of this theory there was a disagreable technical difficulty, namely the impossibility to construct the space bound deeply sub log n. This difficulty was solved by Ranjan, Chang and Hartmanis [12] in a radical manner.They introduced so-called demon machines - a new type of Turing machines with sublogaritmic tape where the space bound is given in advance and gratis. The demon-machines are treated also in [16]. By introducing of this new computational device the problem of the separation of complexity classes arises in this case in the combination with the classical space complexity. Geffert solves this problem in [7] and among other he proves some separation results of the classical form : If the separation condition $\lim s_2(n)/s_1(n) = \infty$ holds then $SPACE(s_2(n)) \supset SPACE(s_1(n))$.

This paper presents also some separation results for demon-machines but with a separation condition which is substantially finer than the usual condition used by Geffert. In this sense the presented results correspond to the basic task of the complexity theory mentioned above in a closer way.

The presented results have two basic sources. One of them is a detailed analysis of the problems connected with the space measure which ends by a slightly changed definition (of the space measure) which seems to be very practical, i.e. it is used in the world of the practical computation. The second source is a succint, elegant and effective method of diagonalization which is appropriate in a large set of cases.

First let us discuss the usual space measure when the number of cells of the tape used during the computation of the Turing machine in question is the unique criterium of the needed space. In this case for each space bound s and for each $k \in N$, $SPACE(s(n)) = SPACE(k.s(n))$ holds since we are able to simulate each concrete (off-line) machine by a machine which uses a larger alphabet and for each word of length n it uses only $s(n)/k$ cells of its worktape. It is clear that for the space measure defined in such a way it is impossible to achieve a finer separation condition than $\lim s(n)/s_1(n) = \infty$. If we are not content with such a fat separation condition an idea arises to capture the size of the worktape alphabet.

One possibility is to count $\log m$ for each symbol, where m is the size of the worktape alphabet (see also van Emde Boas [1]). The second possibility is simply to fix the worktape alphabet. Here we have also a physical motivation : in the real world the

ability to distinguish different objects (symbols) within a unit of space (a cell) is limited, hence we may use at most a certain constant number of symbols. In both cases for each $k \in N$, $SPACE(s(n)) = SPACE(s(n)+k)$ holds since it is possible to simulate each segment of the tape of constant length within the finite control. Hence the finest possible separation condition is $lim_{n\to\infty} s(n) - s_1(n) = \infty$.

To get a separation condition which is finer than the latest one it is necessary to neutralize also the possibilities given by the size of the finite control. In the present paper we do it in such a manner that (new) space complexity of a computation of a Turing machine is given by the (old) space complexity of its simulation on an arbitrarily choosen universal machine U. This definition gives a surprisingly fine separation condition $s(n) - s_1(n+1) > k_0$ which is sufficient for $SPACE^U(s) - SPACE^U(s_1) \neq \emptyset$. The second surprise is the fact that this measure is widely used in the world of practical computations since by the amount of the memory needed by a computation we understand the amount of the memory which is needed when the computation is performed on a computer, which is , in fact, a universal machine.

The diagonalization method is given by a theorem which is stated only by a simple way without any notions concerning computability or complexity. It was applied in the case of the time complexity [19], and in a previous very complicated form it was used for space complexity (as we shall mention) and in the case of computations with oracles when the complexity is given by the number of queries or by their amounts etc. [18].

Due to the generality of our diagonalization method it is not substantial that we work with sublogaritmic tape. Our proofs may be read also as proofs for the superlogaritmic case, hence by the way we prove also already known [17] results but by a simpler manner (in comparison with the original proofs). We work with deterministic, nondeterministic and alternating demon machines with constant or unlimited number of alternations possibly equipped by an auxiliary pushdown store or by an oracle. Our proofs hold simultaneously for sixteen types of computational devices.

Briefly: Before introducing of the demon machines the separation theorem were proven only for $log(n)$ and larger space bounds from the reason of nonconstructability of bounds sub $log(n)$. In this context (overlog) a very sharp separation refining clasical results was proven by using the space measure with a fixed universal machine [17]. After introducing demon-machines (where the problem of constructibility is neutralized) we prove similar very fine results also deeply sub log n. Since our proofs are very general, it is possible to read them (by the way) also as the proofs for the case over log n ; but in comparison with [17] they are substantially shorter and more elegant.

2 The basic computational model

As a standard model (acceptor) of computation we shall consider Turing machines having a two-way read-only input tape and a separate semi-infinite two-way read-write worktape.

Definition 2.1 *([7]) For any function s, a demon s-tape bounded machine begins its computation with a special tape limit marker placed exactly $s(n)$ positions away from*

the initial position of the worktape, for every input of length n. The tape marker can be detected (and cannot be moved). The machines rejects if it ever tries to use more than $s(n)$ tape.

We shall use deterministic, nondeterministic and alternating demon machines with a constant number of alternations or with an unlimited number of alternations.

Alternating demon s-tape bounded machines may have their computation trees infinite since for small functions $s(n)$ below $\log n$ the demon machines have difficulties to avoid cycles in its computations. For the evaluation of an infinite computation tree we effectively construct its reduced finite version. We cut each its branch at the moment when for the first time a configuration appears repeatedly. The leaves of this kind are considered as rejecting configurations. Then we evaluate the tree in the usual way.

We shall use all four types (above) of demon machines also equipped by an oracle. In this case the demon machine in question has an additional tape on which only a special symbol S may be written. The finite control includes three states q, YES, NO. If it enters the state q in the next step it must enter YES or NO in dependance on the fact whether the number of occurences of S on the special tape belongs to the oracle or not. After this action the special tape becomes empty.

Moreover we shall use demon machines with an auxiliary pushdown store which is organized as usually. Hence we shall work with sixteen types of computational devices.

For each type of our computational devices we fix a universal machine U. On the input tape U reads only the symbols $0, 1$ and the endmarkers. On the worktape U works with at least $0, 1, b$ and the endmarkers. The programs of demon machines are well-structured strings of 0's and 1's. U starts its computation in the situation when the program of the simulated machine is placed in the leftmost part of the worktape of U. U has the property that for each demon $s(n)$-tape bounded machine M using on the worktape at most the same number of the symbols as U does, U simulates M on the tape of the length $s(n) + |p_M|$ where p_M is the program of M. Moreover for each m greater than the number of worktape symbols of U there is a constant k_m such that the U-simulation of the machines which use m worktape symbols increases the tape complexity only by the multiplicative constant k_m.

For our sixteen types of computational devices we define complexity classes as follows.

Let p be a program of a machine and s be a function. By $L_s(p)$ we mean the language accepted by the machine M_p within the tape bound s. Also we define $SPACE(s) =_{df} \{L_s(p) | p \textit{ is a program of a machine}\}$ and $SPACE(s, m) =_{df} \{L_s(p) | p \textit{ is a program of a machine with } m \textit{ worktape symbols}\}$.

By $L_s^U(p)$ we mean the language of words x which are accepted by U when U starts with the program p in the leftmost part of its worktape and with its worktape endmarker on the position $s(|x|)$. Further we define $SPACE^U(s) =_{df} \{L_s^U(p) | p \textit{ is a program of a machine}\}$ and similarly $SPACE^U(s, m)$ for machines with m worktape symbols.

Let s, s_1 be functions, $s_1(n) < s(n)$. We say that s_1 is $s - \textit{constructible}$, resp. $(s, m) - \textit{constructible}$, iff there is a deterministic $s(n)$-tape bounded demon-machine M , resp. with m worktape symbols, such that on the inputs of length n with blank

worktape cells and with endmarker in the $s(n)$-th cell M ends with 1 in the $s_1(n)$-th position (all other positions with except of the $s(n)$-th one are blank).

3 The diagonalization theorem

The principle of our diagonalization can be formulated without any notions concerning computability or complexity.

For languages over a fixed alphabet we say that L_1 is equivalent to L_2 ($L_1 \sim L_2$) iff L_1, L_2 differ only on a finite number of words. For a class C of languages, by $\mathcal{E}(C)$ we mean the class $\{L' | (\exists L \in C)(L' \sim L)\}$.

Theorem 3.1 *(the d-theorem, [19]) Let L be a language and let C be a class of languages indexed by a set S, $C = \{L_p : p \in S\}$. Let R be a language and let F be a mapping, $F : R \to S$, such that $(\forall p \in S)(\exists^\infty r \in R)(F(r) = p)$. Let z be a mapping, $z : R \to N$, such that for each $r \in R$, $z(r)$ satisfies the following two conditions : (a) $r1^{z(r)} \in L \leftrightarrow r \notin L_{F(r)}$, (b) $(\forall j, 0 \leq j < z(r))(r1^j \in L \leftrightarrow r1^{j+1} \in L_{F(r)})$. Then $L \notin \mathcal{E}(C)$.*

Comment. In our application, C will be the complexity class to be diagonalized over, and S will be the set of programs. Our task will be to construct a diagonalizer M which will accept a language L, $L \notin \mathcal{E}(C)$. The features of M are well described by the conditions (a), (b). On the input $r1^j$, M will derive the program $F(r)$ and then M will try to find whether the input is fully padded ($j = z(r)$); if so, then M will compute according to the condition (a). For inputs which are not fully padded, M will compute according to the condition (b). More details are found in the proof of the theorems below.

Proof: By contradiction. Suppose $L \in \mathcal{E}(C)$. Hence $L \sim L_p$ for some $p \in S$. Moreover, there is an $r \in R$ such that $F(r) = p$ and $L, L_{F(r)}$ $(=L_p)$ differ only on words shorter than r; in particular for each $j \in N, r1^j \in L_{F(r)}$ iff $r1^j \in L$. Hence by condition (b) $r \in L \leftrightarrow r1^{z(r)} \in L$, and then by (a) $r1^{z(r)} \in L \leftrightarrow r \notin L$. A contradiction. □

4 Separation of classes of binary languages

We prove some theorems for 16 types of computational devices simultaneously. We work with demon machines which are deterministic or nondeterministic or alternating with a fixed number of alternations or alternating with unlimited number of alternations, and which may have an auxiliary pushdown store or an oracle. The following theorems and proofs are correct if we replace the word "machine" by the name of one of 16 devices with which we work, for example "nondeterministic demon machines with an auxiliary pushdown store" or "alternating demon machines with unlimited number of alternations and with an oracle". The next two theorems are formulated in a classical manner, it means with a classical separation condition.

Theorem 4.1 *Let s, s_1 be functions, $lim_{n\to\infty} s(n)/s_1(n+1) = \infty$. Let $s_1(n+1)$ be $s(n)$-constructible and recursive. Then there is a language $L \subseteq 1^+0^+1^*$ such that $L \in SPACE(s) - \mathcal{E}\ SPACE(s_1)$.*

Now we try to get a finer separation condition by taking into account also the number of worktape symbols.

Theorem 4.2 *Let $m \in N$, $m \geq 4$. Let s, s_1 be functions, $lim_{n\to\infty} s(n) - s_1(n+1) = \infty$. Let $s_1(n+1)$ be $(s(n), m)$-constructible and recursive. Then there is a language $L \subseteq 1^+0^+1^*$ such that $L \in SPACE(s, m) - \mathcal{E}\ SPACE(s_1, m)$.*

The proofs of both theorems are very similar, so we give them in the following common text.

Proof:

We shall apply our d-theorem. Let S be a recursive set of programs of machines in question. For each $p \in S$ we define $L_p = L_{s_1}(p)$ and further $C = SPACE(s_1)$, resp. $C = SPACE(s_1, m)$. We see that $C = \{L_p | p \in S\}$.

Our diagonalizer M starts its computation with the endmarker in the $s(n)$-th cell. Hence we have $L(M) \in SPACE(s)$, resp. $L(M) \in SPACE(s, m)$.

At first M checks whether its input is of the form $1^k0^l1^j$. Then M puts 1 into $s_1(n+1)$-th cell of its worktape. The other cells are blank.

In the second part of its computation M will operate within the right $s(n) - s_1(n+1)$ cells of its worktape. M tries to construct $bin(k)$ (the binary code of k). If $p = bin(k) \in S$ then M tries to decide whether $1^k0^l \in L_p$. If $s(k+l+j) - s_1(k+l+j+1)$ cells suffice for the decision then M accepts iff $1^k0^l \notin L_p$. (We know that such a j exists since in both cases $lim_{n\to\infty} s(n) - s_1(n+1) = \infty$.)

Otherwise in the last part of its computation M tries to simulate M_p on the input $1^k0^l1^{j+1}$ within $s_1(k+l+j+1)$ cells of the worktape of M_p. M uses at most $k_p.s_1(k+l+j+1)$ cells only (where k_p is an appropriate constant depending on the number of worktape symbols of M_p).

To prove that $L(M) \notin \mathcal{E}\ (C)$ we define R, F, z needed in the d-theorem as follows: R is the set of words of the form 1^k0^l where $bin(k) \in S$ and for each $j \in N$ the following holds: $|bin(k)| \leq s(k+l+j) - s_1(k+l+j+1)$, $s(k+l) - s_1(k+l+1) \leq s(k+l+j) - s_1(k+l+j+1)$ and $s(k+l+j)$ cells are sufficient for simulation of $s_1(k+l+j+1)$ cells of $M_{bin(k)}$ on the input $1^k0^l1^{j+1}$. We see that for each k with $bin(k) \in S$ there is infinitely many l such that $1^k0^l \in R$.

For $r = 1^k0^l \in R$ we define $F(r) = bin(k)$. We see that for each $p \in S$ there are infinitely many $r \in R$ such that $F(r) = p$.

Further for $r = 1^k0^l \in R$ we define $z(r)$ as the minimum j such that the computation of M on $1^k0^l1^j$ ends by the decision whether $1^k0^l \in L_{bin(k)}$ or not.

For each $r \in R$ we have $r1^{z(r)} \in L(M)$ iff $r \notin L_{F(r)}$ (the condition (a) of the d-theorem is satisfied).

According to the description of the last part of the computation of M, for each $r \in R$ and for each $j < z(r)$ we have $r1^j \in L(M)$ iff $r1^{j+1} \in L_{F(r)}$ (the condition (b) of the d-theorem is satisfied).

Hence $L(M) \notin \mathcal{E}\ (C)$.

□

Now we continue our program of searching for the finest separation condition. We will use our modification of the classical space measure which takes into account also the amount of the finite control.

Theorem 4.3 *Let $m \in N$, $m \geq 4$, let U (the universal machine) has m worktape symbols.*

Let s be a function, $lim_{n\to\infty} s(n) = \infty$. Let s_1 be a recursive function, $lim_{n\to\infty} s_1(n) = \infty$ such that $s_1(n+1)$ is $(s(n), m)$-constructible.

Then there is a language $L \subseteq 1^+0^+1^*$ such that
a) $L \in SPACE(s,m) - \mathcal{E}\ SPACE^U(s_1, m)$,
b) there is a constant K such that
$L \in SPACE^U(s+K, m) - \mathcal{E}\ SPACE^U(s_1, m)$.

Immediately we have:

Corollary 4.4 *Let $m \in N$, $m \geq 4$. Let s be a recursive function, $lim_{n\to\infty}\ s(n) = \infty$. Then there is a constant $K \in N$ and a language $L \subseteq 1^+0^+1^*$ such that $L \in SPACE^U(s(n)+K, m) - \mathcal{E}\ SPACE^U(s(n-1), m)$.*

The following theorem shows that our modified measure captures also the number of worktape symbols (m) and that therefore for getting such a tight separation as above it is not necessary to fix m.

Theorem 4.5 *Let s be a function, $lim_{n\to\infty} s(n) = \infty$. Let s_1 be a recursive function, $lim_{n\to\infty} s_1(n) = \infty$ such that $s_1(n+1)$ is $(s(n), m)$-constructible for an $m \in N$. Let U has m symbols.*

Then there is a language $L \subseteq 1^+0^+1^*$ such that
a) $L \in SPACE(s,m) - \mathcal{E}\ SPACE^U(s_1)$,
b) there is a constant K such that
$L \in SPACE^U(s+K, m) - \mathcal{E}\ SPACE^U(s_1)$.

Corollary 4.6 *Let s be a recursive function, $lim_{n\to\infty}\ s(n) = \infty$. Then there is a constant $K \in N$ and a language $L \subseteq 1^+0^+1^*$ such that $L \in SPACE^U(s(n)+K) - \mathcal{E}$ $SPACE^U(s(n-1))$.*

The common proof of both theorems is done according to a similar scheme as in the previous case.
Proof:

We shall apply our d-theorem. Let S be a recursive set of programs of machines, resp. of programs of machines with m worktape symbols, in question. For each $p \in S$ we define $L_p = L^U_{s_1}(p)$ and further $C = SPACE^U(s_1)$, resp. $C = SPACE^U(s_1, m)$. We see that $C = \{L_p | p \in S\}$.

Our diagonalizer M starts its computation with the endmarker in the $s(n)$-th cell. Hence we have $L(M) \in SPACE(s,m)$.

At first M checks whether its input is of the form $1^k0^l1^j$. Then M puts 1 into $s_1(n+1)$-th cell of its worktape. The other cells are blank.

In the second part of its computation M will operate within the first $s_1(n+1)$ cells of its worktape. M tries to construct $bin(k)$. If $p = bin(k) \in S$ then M tries to decide whether $1^k0^l \in L_p$. If $s_1(k+l+j+1)$ cells suffice for the decision then M accepts iff $1^k0^l \notin L_p$. (We know that such a j exists since in both cases $lim_{n\to\infty} s_1(n) = \infty$.)

Otherwise in the last part of its computation M tries to compute as U does and to simulate M_p on the input $1^k0^l1^{j+1}$ within $s_1(k+l+j+1)$ cells of the worktape of M.

To prove that $L(M) \notin \mathcal{E}\ (C)$ we define R, F, z needed in the d-theorem as follows: R is the set of words of the form 1^k0^l where $bin(k) \in S$ and for each $j \in N$ the following holds: $|bin(k)| \leq s_1(k+l) \leq s_1(k+l+j)$. We see that for each k with $bin(k) \in S$ there is infinitely many l such that $1^k0^l \in R$.

For $r = 1^k0^l \in R$ we define $F(r) = bin(k)$. We see that for each $p \in S$ there are infinitely many $r \in R$ such that $F(r) = p$.

Further for $r = 1^k0^l \in R$ we define $z(r)$ as the minimum j such that the computation of M on $1^k0^l1^j$ ends by the decision whether $1^k0^l \in L_{bin(k)}$.

For each $r \in R$ we have $r1^{z(r)} \in L(M)$ iff $r \notin L_{F(r)}$ (the condition (a) of the d-theorem is satisfied).

According to the description of the last part of the computation of M, for each $r \in R$ and for each $j < z(r)$ we have $r1^j \in L(M)$ iff $r1^{j+1} \in L_{F(r)}$ (the condition (b) of the d-theorem is satisfied).

Hence $L(M) \notin \mathcal{E}\ (C)$.

The parts b) of the Theorems are trivial consequences of parts a).

□

In the next section we shall prove similar results for languages over one-letter alphabet.

5 Separation of classes of unary languages

As in the previous section we work with 16 types of computational devices.

Theorem 5.1 *Let s, s_1 be functions, $lim_{n\to\infty} s(n)/s_1(n+1) = \infty$. Let s_1 be nondecreasing, unbounded and recursive. Let $s_1(n+1)$ be $(s(n), m)$-constructible for an $m \in N$, $m \geq 4$,let U have m symbols.*

Then there is a language $L \subseteq 1^*$ such that
a) $L \in SPACE(s, m) - \mathcal{E}\ SPACE(s_1)$,
b) there is a K such that $L \in SPACE^U(s + K, m) - \mathcal{E}\ SPACE(s_1)$.

Also in the case of unary languages for machines with any fixed worktape alphabet we get a finer separation condition.

Theorem 5.2 *Let $m \in N$, $m \geq 4$, let U have m symbols. Let s, s_1 be functions, $lim_{n\to\infty} s(n) - s_1(n+1) = \infty$. Let s_1 be nondecreasing, unbounded and recursive, let $s_1(n+1)$ be $(s(n), m)$-constructible.*

Then there is a language L such that
a) $L \in SPACE(s,m) - \mathcal{E}\ SPACE(s_1,m)$
b) there is a $K \in N$ *such that* $L \in SPACE^U(s+K,m) - \mathcal{E}\ SPACE(s_1,m)$.

Proof: We shall apply our d-theorem.

Let S be a recursive set of programs of the machines in question. For $p \in S$ we define $L_p =_{df} L_{s_1}(p)$. We put $C =_{df} SPACE(s_1)$, resp. $C =_{df} SPACE(s_1,m)$. We see that $C = \{L_p | p \in S\}$.

Our diagonalizer M will compute as follows. M checks whether the input is an unary string and then M computes within $s(n)$ worktape cells, hence $L(M) \in SPACE(s,m)$. On its worktape M constructs $s_1(n+1)$ - on the position $s_1(n+1)$ M gives the symbol 1.

Within the first $s_1(n+1)-1$ cells of its worktape, M will perform an initial part of a recursive process P which we shall describe. (P will give us the possibility to define R, F and z which we need for the application of the d-theorem.)

P contains a generator of the programs from S such that if $\{p_i\}$ is the generated sequence then $(\forall p \in S)(\exists^\infty i)(p_i = p)$.

P starts with the generation of p_1. At this moment some amount a_1 of the worktape has been used. Then P constructs $n_1 =_{df} min\{n | s_1(n+1) - 1 \geq a_1\}$. Then P decides whether $1^{n_1} \in L_{p_1}$ or not.

If P has generated p_i, constructed 1^{n_i} and decided whether $1^{n_i} \in L_{p_i}$ then (on the cells which P has not used till now) P generates p_{i+1}, P constructs $n_{i+1} =_{df} min\{n | s_1(n+1) - 1$ *cells suffice for* P *till the generation of* $p_{i+1}\}$ and P decides whether $1^{n_i+1} \in L_{p_{i+1}}$ or not.

When P is stopped because of the lack of the space on the worktape- P has used all $s_1(n+1)-1$ cells - then by the result of P we mean the last generated p_i and also the decision whether $1^{n_i} \in L_{p_i}$ if this decision is achieved.

If this decision is achieved then M accepts iff $1^{n_i} \notin L_{p_i}$.

If the decision in question is not achieved then in the last part of its computation M tries to reorganize its worktape in such a manner that in the leftmost cells p_i is written followed by $s_1(n+1)$ blank cells. Then M tries to simulate (as U) M_{p_i} on the input 1^{n+1} and on $s_1(n+1)$ cells of the worktape of M_{p_i}.

Let $\{p_{i_k}\}$ be a subsequence of $\{p_i\}$ such that for all k whenever the result of P is p_{i_k} (without the decision) then in the last part of its computation M has enough of tape for the simulation. Due to the condition $s(n)/s_1(n+1)$ resp. $s(n) - s_1(n+1)$ *tends to infinity* each p occurs infinitely many times not only in $\{p_i\}$ but also in $\{p_{i_k}\}$.

Now , we define $r_k =_{df} 1^{n_{i_k}}$, $R =_{df} \{r_k | k \in N\}$, $F(r_k) =_{df} p_{i_k}$. Further $z(r_k)$ is the first m such that $s_1(|r_k| + m) - 1$ cells suffices for P to generate p_{i_k}, to construct r_k and to decide whether $r_k \in L_{p_{i_k}}$.

We have $r_k 1^{z(r_k)} \in L(M) \leftrightarrow r_k \notin L_{p_{i_k}} = L_{F(r_k)}$ - the condition (a) of the d-theorem is satisfied.

We have proven $r_k 1^j \in L(M) \leftrightarrow r_k 1^{j+1} \in L_{p_{i_k}} = L_{F(r_k)}$ (the condition (b) of the d-theorem is satisfied).

We have $L(M) \notin \mathcal{E}\ (C)$.

Since the universal machine U uses only m symbols we are able to construct our diagonalizer M in such a way that M uses only m symbols, too. Hence the U-simulation of the segment of length $s(n)$ of the worktape of M requires $s(n) + |p_M|$ cells only. Therefore $L(M) \in SPACE^U(s + K, m)$ for some $K \in N$. $\square$

The next two theorems demonstrate that our modified space measure which takes into account also the capacity of the finite control gives the possibility of a very fine separation condition also in the case of unary languages. The first of them is formulated for machines with any fixed worktape alphabet, the second one shows that this restriction on the size of the worktape alphabet is not necessary.

Theorem 5.3 *Let $m \in N$, $m \geq 4$, let U have m worktape symbols. Let s be a function, let s_1 be a recursive, nondecreasing and unbounded function and let $s_1(n+1)$ be $(s(n), m)$-constructible. Then for each $m \geq 4$ there is a language $L \subseteq 1^*$ such that*
a) $L \in SPACE(s, m) - \mathcal{E}\ SPACE^U(s_1, m)$,
b) there is a constant K such that
$L \in SPACE^U(s + K, m) - \mathcal{E}\ SPACE^U(s_1, m)$.

Theorem 5.4 *Let $m \in N$, $m \geq 4$, let U have m worktape symbols. Let s be a function, let s_1 be a recursive, nondecreasing and unbounded function and let $s_1(n+1)$ be $(s(n), m)$-constructible. Then there is a language $L \subseteq 1^*$ such that*
a) $L \in SPACE(s, m) - \mathcal{E}\ SPACE^U(s_1)$,
b) there is a constant K such that
$L \in SPACE^U(s + K, m) - \mathcal{E}\ SPACE^U(s_1)$.

The following corollary gives a very tight separation.

Corollary 5.5 *Let s be a recursive, nondecreasing and unbounded function. Then there is a constant $K \in N$ and a language $L \subseteq 1^+$ such that $L \in SPACE^U(s(n) + K) - \mathcal{E}\ SPACE^U(s(n-1))$, for U with an appropriate number of worktape symbols.*

First we prove our corollary.
Proof: We see that if we put $s_1(n) = s(n-1)$ the assumptions of the theorem concerning s_1 are satisfied. $\square$

Now we prove our theorems.
Proof: We shall apply the d-theorem.

Let S be a recursive set of programs of the machines in question. For $p \in S$ we define $L_p =_{df} L_{s_1}(p)$. We put $C =_{df} SPACE(s_1, m)$, resp. $C =_{df} SPACE(s_1)$. We see that $C = \{L_p | p \in S\}$.

Now the proof continues by seven paragraphs of the description of our diagonalizer M from the previous case.

If the decision in question is not achieved then in the last part of its computation M computes as the universal machine U on the input 1^{n+1} with the program p_i on the leftmost part of the worktape of length $s_1(n+1)$ and M uses only $s_1(n+1)$ worktape cells.

Now , we define $r_i =_{df} 1^{n_i}$, $R =_{df} \{r_i | i \in N\}$, $F(r_i) =_{df} p_i$. Further $z(r_i)$ is the first m such that $s_1(|r_i| + m) - 1$ cells suffices for P to generate p_i, to construct r_i and to decide whether $r_i \in L_{p_i}$.

We have $r_i 1^{z(r_i)} \in L(M) \leftrightarrow r_i \notin L_{p_i} = L_{F(r_i)}$ - the condition (a) of the d-theorem is satisfied.

We have proven $r_i 1^j \in L(M) \leftrightarrow r_i 1^{j+1} \in L_{p_i} = L_{F(r_i)}$ (the condition (b) of the d-theorem is satisfied).

We have $L(M) \notin \mathcal{E}\ (C)$.

Since the universal machine U uses only m symbols we are able to construct our diagonalizer M in such a way that M uses only m symbols, too. Hence the U-simulation of the segment of length $s(n)$ of the worktape of M requires $s(n) + |p_M|$ cells only. Therefore $L(M) \in SPACE^U(s + K, m)$ resp. $L(M) \in SPACE^U(s + K)$ for some $K \in N$. □

6 Conclusions and remarks

First let us demonstrate that our modified space complexity measure induces a structure within the classical complexity classes $SPACE(s, m)$.

Lemma 6.1 *Let s be a function , K be an integer and m be a natural number, $m \geq 4$. Then*
$SPACE^U(s + K + 1, m) \supseteq SPACE^U(s + K, m)$.

Proof: Let $L \in SPACE^U(s + K, m)$. Then for a program p $L = L^U_{s+K}(p)$. There is a program p' of the same machine such that $|p'| = |p|+1$. Hence $L \in SPACE^U(s+K+1)$. □

Proposition 6.2 *Let s be a function, let m be a natural number, $m \geq 4$ and let U have m symbols. Then*
$SPACE(s, m) = \bigcup_{K \in Z} SPACE^U(s + K, m)$
where Z is the set of integers.

Proof: Let $L \in SPACE(s, m)$. Then $L = L_s(p)$ for a program p and we have $L = L^U_{s+|p|}(p) \in SPACE^U(s + |p|, m)$. On the other hand if $L \in \bigcup_{K \in N} SPACE^U(s + K, m)$ then $L = L^U_{s+K}(p)$ for an integer K and for a program p. Therefore $L = L_{s+K-|p|}(p)$ and there is a program p' such that $L = L_s(p') \in SPACE(s, m)$. □

We see that the class $SPACE(s, m)$ is the union of the chain of the classes ... $\subseteq SPACE^U(s - 1, m) \subseteq SPACE^U(s, m) \subseteq SPACE(s + 1, m) \subseteq$ ... which is infinite in both directions.

Let us try to find the finest difference between two complexity bounds which causes the separation of the respective complexity classes.

We are able to achieve such a result only for the case of linear complexity bounds. We shall apply our separation theorems.

Proposition 6.3 *Let s_1 be a linear function. Then there is a K such that $SPACE^U(s_1 + K, m) - \mathcal{E}\ SPACE^U(s_1, m) \neq \emptyset$.*

This follows from Corollary 4.4. We have the following theorem.

Theorem 6.4 *Let $m \in N$, $m \geq 4$. For each linear s there is a constant l_s such that $SPACE^U(s + l_s + 1, m) \supset SPACE^U(s + l_s, m)$.*

We see that the increasing of the bound by 1 strengthens the complexity power.

A similar result can be proven also for the classes $SPACE^U(s)$.

Remark. For bounds deeply sub $\log n$ it does not seem to be possible to convert our separation results into the hierarchy results of the form $SPACE^U(s(n) + k) \supset SPACE^U(s(n-1))$ since there are such recursive functions s that the sets $\{n | s(n) \neq s(n-1)\}$ are of very high complexities. This is a negative consequence of the elegance of the definition of the demon machines, and of their non-constructiveness. There is an open question whether there is a reasonable class of small recursive functions such that the result conversion mentioned above is possible.

Remark. For the case over $\log n$ it is possible to prove similar results for usual (non demon-) machines with bounds which are constructible.

Acknowledgment. I thank to V. Geffert for his very helpful comments and to J. Wiedermann for discussions.

Bibliography

[1] P. van Emde Boas, Machine models and simulations, in Algorithms and Complexity, pp. 3 - 66, 2.1.3., Vol. A, Handbook of Theoretical Computer Science, ed. Jan van Leeuwen, Elsevier 1990

[2] B. von Braunműhl, R. Gengler, and R. Rettinger, The alternation hierarchy for machines with sublogarithmic space is infinite, Proc. of STACS'94, LNCS 775, Springer-Verlag, 85-96, 1994.

[3] S. Cook, A hierarchy for nondeterministic time complexity, JCSS 7, 4, (1973), pp. 343 - 353.

[4] V. Geffert, Tally versions of the Savitch and Immerman-Szelepcsényi theorems for sublogarithmic space, SIAM J. Comput., vol. 22, no. 1, 102-113, February 1993

[5] V. Geffert, Sublogarithmic Σ_2-Space is not closed under complement and other separation results, Informatique théorique et Applications, vol. 27, no. 4, 1993, 349-366.

[6] V. Geffert, A hierarchy that does not collapse: Alternations in low level space, Informatique théorique et Applications, vol. 28, no. 5, 1994, 465-512.

[7] V. Geffert, Bridging Across the log(n) Space Frontier, Proc. MFCS'95 Prague, LNCS 969, 50-65.

[8] K. Iwama, ASPACE(o(loglog(n)) is regular, SIAM J. Comput.., 22, 136-46,1993.

[9] M. Liśkiewicz, R. Reischuk , The Complexity World below Logarithmic Space, Proc. the Structure in Complexity Theory, 1994, 64-78.

[10] M. Liśkiewicz, R. Reischuk, The Sublogarithmic Alternating Space World, to appear in SIAM J. Comp.

[11] B. Monien and H. Sudborough, On eliminating nondeterminism from Turing machines which use less than logarithm work tape space, Theoret. Comput. Sci., 21, 237-53, 1982.

[12] D. Ranjan,R. Chang, and J. Hartmanis, Space bounded computations: Review and new separation results, Theoret. Comput. Sci., 80, 289-302, 1991.

[13] J. I. Seiferas, Relating refined space complexity classes, JCSS 14 (1977), 1, pp. 100 - 129.

[14] J. I. Seiferas, M. J. Fischer and A. R. Meyer, Separating Nondeterministic Time Complexity Classes, J. of ACM 25 (1978), 1, pp. 146 - 167.

[15] I. H. Sudborough, Separating tape bounded auxiliary pushdown automata classes, Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, Boulder, Colorado, May 2 - 4, 1977, pp. 208 - 217.

[16] A. Szepietowski, Turing Machines with Sublogarithmic Space, Springer-Verlag, LNCS 843.

[17] S. Žák, A Turing machine space hierarchy, Kybernetika(Prague) 15(1979), no. 2, 100-121.

[18] A Turing machine oracle hierarchy I,II, Comm. Math. Univ.Carol. 21 (1980), pp. 11 - 39.

[19] S. Žák, A Turing machine time hierarchy, Theoretical Computer Science 26(1983), 327-333.