A Sharp Separation Below Log

Žák, Stanislav
1995

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A sharp separation below log

Stanislav Žák

Technical report No. 655

October 28, 1995

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 66414244 fax: (+422) 8585789
e-mail: stan2uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

A sharp separation below log

Stanislav Žák

Technical report No. 655
October 28, 1995

Abstract

A very sensitive measure of space complexity of computations on Turing machines is treated. The space complexity of a computation is given by the amount of the tape required by its simulation on a fixed universal machine.

We use so called demon (Turing) machines where the tape limit is given automatically without any construction [4,7]. By a diagonalization method we obtain the following results: For any arbitrarily small recursive nondecreasing and unbounded function s there is a $k \in N$ and an unary language L such that $L \in SPACE(s(n) + k) - SPACE(s(n-1))$. For a binary L the supposition $lim\ s = \infty$ is sufficient. The witness languages differ from each language from the lower classes on infinitely many words.

The results hold for deterministic and nondeterministic demon machines and also for alternating demon machines with a constant number of alternations, and with unlimited number of alternations.

Keywords
space separation, diagonalization

1 Introduction

Recently a progres has been made in the theory of computations within the sublogarithmic space ([1 - 7] and many others). This paper is inspired by [4] where so called demon machines are treated. The advantage of this model is that the space limit of computations on a Turing machine is given gratis without any need of its construction. Therefore we can work with arbitrarily small limit functions and we can concentrate only on the computability within these bounds. Thus it is easier to prove some separation results also below log.

The new situation enable us to apply a diagonalization method which was used for constructible space bounded Turing machine computations in a cumbersome form in [8], and for time bounded computations in a succint form in [9].

The typical form of separation results is roughly speaking "If $\lim s_1(n)/s_2(n) = 0$ then $SPACE(s_2(n)) - SPACE(s_1(n)) \neq \emptyset$" where $SPACE(s(n))$ means the class of languages which are accepted within the tape complexity $s(n)$. The results of this kind are the best possible since $SPACE(s_1(n)) = \bigcup_{k \in N} SPACE(k.s_1(n))$ because the number of the worktape symbols of TM's is not controlled.

To get stronger results we introduce a new finer more sensitive space complexity measure. The complexity of a computation is given by the amount of the tape required by its simulation on a fixed universal machine. This approach gives us the possibility to prove the results mentioned in the Abstract. The fact that the results also hold for alternating demon machines is not without any interest since the situation with space complexity classes for alternating machines below log is quite different from this one over log [2],[3],[6],[7].

2 The basic computational model

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

Definition 2.1 *([4]) 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.*

Alternating demon s-tape bounded machines may have their computation trees infinite since for small functions $s(n)$ below log the demon machines have difficulties to avoid cycles in its computations. For the evaluation of an infinite computation tree we efectively 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 we mark by 0. Then we evaluate the tree in the usual way.

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 $0, 1, b$ 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. U has the property that for each demon $s(n)$-tape bounded machine M using on the worktape only the symbols $0, 1, b$ and the endmarkers, U simulates M on the tape of the length $s(n) + |p_M|$ where p_M is the program of M. Moreover for each $m \geq 4$ 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 konstant k_m.

For tape bounded deterministic and nondeterministic demon machines and also for alternating machines with a fixed number of alternations and for machines with unlimited number of alternations 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 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 $SPACE(s(n)) =_{df} \{L_s(p) | p \textit{ is a program of a machine}\}$.

Let s, s_1 be functions, $s_1(n) < s(n)$. We say that s_1 is s-constructible iff there is a deterministic $s(n)$-tape bounded demon-machine M such that on the inputs of length n M ends with 1 on the $s_1(n)$-th position (all other positions till $s(n) - 1$ are blank).

3 The diagonalization theorem

The principle of our diagonalization can be formulated without any notions about the computability and the 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, [9]) 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 $L \notin \mathcal{E}(C)$.*

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 results

For deterministic and nondeterministic demon machines and for alternating demon machines with a fixed number of alternations and for machines with unlimited number of alternations the following theorems hold.

Theorem 4.1 *Let s be a recursive function, $\lim s(n) = \infty$. Then there is a constant $k \in N$ and a language $L \subseteq 1^+0^+1^*$ such that $L \in SPACE(s(n)+k) - \mathcal{E}\ SPACE(s(n-1))$.*

Proof: We shall apply our d-theorem. Let S be a recursive set of programs of the machines in question.We put $C =_{df} SPACE(s(n-1))$. For $p \in S$ we define $L_p = L_{s(n-1)}(p)$. Then $C = \{L_p | p \in S\}$. Further we define $R =_{df} \{1^k0^l |\ |bin(k)| \leq s(k+l) - 1 \wedge bin(k) \in S \wedge (\forall j)(s(k+l) \leq s(k+l+j))\}$, and for $r = 1^k0^l \in R$, $F(r) =_{df} bin(k) \in S$ (where $bin(k)$ is the binary code of k). We see that $(\forall p \in S)(\exists^\infty r \in R)(F(r) = p)$.

Our diagonalizer M checks whether the input is of the form $1^k0^l1^j$. Then M tries to construct $bin(k)$ on its worktape. If $bin(k) = p \in S$ then M tries to decide whether $1^k0^l \in L_p$. If $s(k+l+j) - 1$ cells suffice and M is succesful then M accepts iff $1^k0^l \notin L_p = L_{F(r)}$, otherwise M simulates p on $1^k0^l1^{j+1}$ (as U). Let $r = 0^k1^l \in R$. We define $z(r) =_{df} min\{j | on\ r1^j\ M\ decides\ whether\ r \in L_{F(r)}\}$. We see that $r1^{z(r)} \in L(M)$ iff $r \notin L_{F(r)}$ (the condition (a) of the d-theorem). Further for $j < z(r)$ $r1^j \in L(M)$ iff $r1^{j+1} \in L_{s(n-1)}(F(r)) = L_{F(r)}$ (the condition (b) of the d-theorem). Hence $L(M) \notin \mathcal{E}(C)$.

Since the universal machine U uses only the symbols $0, 1, b$ and an endmarker we are able to construct our diagonalizer M in such a way that M uses also only these symbols. Hence the U-simulation of the segment of length $s(n)$ of the worktape of M requires $s(n) + |p_M|$ cells only. Therefore $L \in SPACE(s(n) + k)$ for some $k \in N$. □

Theorem 4.2 *Let s be a recursive function such that there is an s-constructible non-decreasing and unbounded function s_1. Then there is a constant $k \in N$ and a language $L \subseteq 1^+$ such that $L \in SPACE(s(n) + k) - \mathcal{E}\ SPACE(s(n-1))$.*

Proof: We prepare the situation for an application of the d-theorem. Let S be a recursive set of programs of the machines in question. We put $C =_{df} SPACE(s(n-1))$. For $p \in S$ we define $L_p =_{df} L_{s(n-1)}(p)$. 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. On its worktape M constructs s_1 - on the position $s_1(n)$ M gives the symbol 1.

Within the first $s_1(n) - 1$ 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 $n_1 =_{df} min\{n | s_1(n) - 1 \geq a_1\}$. 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 non-used cells) P generates p_{i+1}, P constructs $n_{i+1} =_{df} min\{n | s_1(n) - 1$ 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.

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

If this decision is achieved then the input is of the form $r_i 1^j$, $j \geq z(r_i)$. M accepts iff $r_i \notin L_{p_i}$. Let L be the language accepted by M. We have $r_i 1^{z(r_i)} \in L \leftrightarrow r_i \notin L_{p_i} = L_{F(r_i)}$ - the condition (a) of the d-theorem is satisfied.

If the decision in question is not achieved then M computes as the universal machine U on the input $1^{n+1} (= r_i 1^{j+1})$ with the program p_i on the leftmost part of the worktape. We have proven $r_i 1^j \in L \leftrightarrow r_i 1^{j+1} \in L_{p_i} = L_{F(r_i)}$ (the condition (b) of the d-theorem is satisfied). We have $L \notin \mathcal{E}\ SPACE(s(n-1))$.

Since the universal machine U uses only the symbols $0, 1, b$ and an endmarker we are able to construct our diagonalizer M in such a way that M uses only $0, 1, b$, and the endmarker, 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 $L \in SPACE(s(n)+k)$ for some $k \in N$. $\square$

Remark. It does not seem to be possible to convert the separation results into the hierarchy results of the form $SPACE(s(n)+k) \supset SPACE(s(n-1))$ 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.

Acknowledgment. I thank to V. Geffert for his helpful comments.

Bibliography

[1] 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

[2] 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.

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

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

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

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

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

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

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