



INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

**A Generalization of the Time and Space
Hierarchy Theorems**

Igor Carboni Oliveira Arnaldo Vieira Moura

Technical Report - IC-09-33 - Relatório Técnico

September - 2009 - Setembro

The contents of this report are the sole responsibility of the authors.
O conteúdo do presente relatório é de única responsabilidade dos autores.

A Generalization of the Time and Space Hierarchy Theorems

Igor Carboni Oliveira*

Arnaldo Vieira Moura†

Abstract

In this paper we introduce a new class of complexity measures and prove a general hierarchy theorem in computational complexity. In addition, we derive as a particular case of our result the traditional time and space hierarchy theorems.

1 Introduction

In their seminal paper, Hartmanis and Stearns [3] proved that if given more time one can solve more problems. For example, there are problems that can be solved in polynomial time but not in time $O(n^3)$. As it was subsequently showed in [2], a similar hierarchy result also holds if we consider space instead of time.

The main argument used to prove such hierarchy theorems is a combination of simulation and diagonalization. In these proofs, a machine simulates other machines with smaller time bounds and then negates their answers, ruling out the possibility that its language is decided by one of the simulated machines.

Since their pioneering work, the same basic idea was successfully applied to prove several other hierarchy results, and these theorems remain one of the mainstays of computational complexity theory. Therefore, a better understanding of the classical hierarchy theorems is of fundamental importance.

In this paper we show that the traditional time and space hierarchy theorems are inserted in a wider context. We prove that they are, in fact, particular cases of a general hierarchy theorem that holds for any reasonable combination of time and space. Furthermore, this is the first time that such an unified treatment is presented in the literature.

2 The Space-Time Complexity

In order to avoid inessential technicalities, the Turing machines discussed here are one-tape deterministic machines with a $\{0, 1\}$ binary tape alphabet. In addition, since we will

*Institute of Computing, University of Campinas. Research supported by FAPESP grant 08/07040-0.

†Institute of Computing, University of Campinas. Research supported by CNPq grant 304363/2008-1, and by FAPESP grant 07/56052-8.

not consider sublinear space, it will not be necessary to consider separated input, output and work tapes [5]. We will use the standard big- O and small- o notations, as in [1, 5]. Further, \mathcal{N} will denote the set of natural numbers.

We start with the usual definitions of time and space complexities.

Definition 1. Let A be a Turing machine. The exact time complexity of A is the function $t_A : \{0, 1\}^* \rightarrow \mathcal{N}$ such that when started with x on its input tape A halts after exactly $t_A(x)$ steps. The time complexity of A is the function $T_A : \mathcal{N} \rightarrow \mathcal{N}$ such that $T_A(n) = \max_{x \in \{0, 1\}^n} \{t_A(x)\}$.

Definition 2. Let A be a Turing machine. The exact space complexity of A is the function $s_A : \{0, 1\}^* \rightarrow \mathcal{N}$ such that when started with x on its input tape A scans exactly $s_A(x)$ distinct tape cells. The space complexity of A is the function $S_A : \mathcal{N} \rightarrow \mathcal{N}$ such that $S_A(n) = \max_{x \in \{0, 1\}^n} \{s_A(x)\}$.

Next, we define a general complexity measure based on the usual time and space complexities, which we call f -complexity.

Definition 3. Let A be a Turing machine and let $f : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ be an arbitrary function. Now, consider the associated function $f\text{-}ts_A : \{0, 1\}^* \rightarrow \mathcal{N}$ given by letting $f\text{-}ts_A(x) = f(t_A(x), s_A(x))$. Then the f -complexity of A is the function $f\text{-}TS_A : \mathcal{N} \rightarrow \mathcal{N}$ where $f\text{-}TS_A(n) = \max_{x \in \{0, 1\}^n} \{f\text{-}ts_A(x)\}$.

Function f should be interpreted as a new complexity measure, the f -complexity, which is based on the time and space complexities of the Turing machine. Obviously, if f is one of the binary projection functions, then f corresponds to the usual time and space complexity measures.

Definition 4. A language L is decidable with f -complexity $O(g(n))$ if there exists a Turing machine A deciding L such that $f\text{-}TS_A$ is $O(g(n))$.

The next result shows that, for any f -complexity measure, there are arbitrarily difficult problems. The notation $\langle M \rangle$ represents a binary string that codifies Turing machine M , as suggested in [5].

Definition 5. Let $f : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ and $g : \mathcal{N} \rightarrow \mathcal{N}$ be arbitrary functions. Define the language:

$$L_{f,g} = \{ \langle M \rangle \mid \text{Machine } M \text{ accepts } \langle M \rangle \text{ and } f\text{-}ts_M(\langle M \rangle) \leq g(|\langle M \rangle|) \}. \quad (1)$$

Theorem 1. $L_{f,g}$ is not decidable with f -complexity less than $g(n)$, i.e., there is no Turing machine with f -complexity $o(g(n))$ that decides $L_{f,g}$.

Proof. For the sake of contradiction, suppose that a Turing machine A decides $L_{f,g}$ and $f\text{-}TS_A(n)$ is $o(g(n))$. Consider the Turing machine B build from A by swapping accepting and rejecting states of A . Then B accepts w if and only if A rejects w , for all $w \in \{0, 1\}^*$.

Now add irrelevant tuples to B , obtaining a new Turing machine B' . Clearly,

$$L(B') = L(B) = \{0, 1\}^* \setminus L(A). \quad (2)$$

Also, there is some $n_0 \in \mathcal{N}$ such that $f-TS_A(n) \leq g(n)$, for all $n \geq n_0$, since $f-TS_A(n)$ is $o(g(n))$. Thus, $f-ts_A(x) \leq g(n)$, for all $x \in \{0,1\}^*$ with $n = |x| \geq n_0$. But, clearly, $f-ts_A = f-ts_{B'}$, and so

$$f-ts_{B'}(\langle B' \rangle) \leq g(|\langle B' \rangle|), \quad (3)$$

if we add enough tuples to B in order to make $|\langle B' \rangle| \geq n_0$.

Now consider the computation of machine A on input $\langle B' \rangle$:

- A accepts $\langle B' \rangle$ iff (using 1)
- B' accepts $\langle B' \rangle$ and $f-ts_{B'}(\langle B' \rangle) \leq g(|\langle B' \rangle|)$ iff (using 3)
- B' accepts $\langle B' \rangle$ iff (using 2)
- A rejects $\langle B' \rangle$.

We reached a contradiction. It follows that there is no Turing machine that decides $L_{f,g}$ with f -complexity $o(g(n))$. □

Since the function f can be any combination of time and space (f could even be an uncomputable function), the proof of Theorem 1 essentially implies that there is no universal machine that performs better than just a straightforward simulation.

To illustrate, consider the projection $f(x, y) = y$. In this case, the f -complexity measure is just the usual space complexity, and the language $L_{f,g}$ can be described as the set of all strings $\langle M \rangle$ such that M accepts the word $\langle M \rangle$ given as its input while scanning at most $g(|\langle M \rangle|)$ distinct tape cells. Theorem 1 implies that there is no Turing machine that decides $L_{f,g}$ within space $o(g(n))$.

3 The Space-Time Hierarchy

If f is in some sense a natural complexity measure, it is possible to prove the existence of f -complexity hierarchies. We exhibit these hierarchies by obtaining the f -complexity of a specific Turing machine that decides $L_{f,g}$. This can be done by simulating machine M on input $\langle M \rangle$.

Definition 6. Let $f : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$ be a function. We say that f is a natural complexity measure if f is a non-decreasing computable function and, for every pair of integers t and s , we have either $f(t+1, s) > f(t, s)$ or $f(t, s+1) > f(t, s)$.

Now we state the general hierarchy result. Recall that T_M and S_M denote the time and space complexities of a machine M .

Theorem 2 (Space-Time Hierarchy). Let f be a natural complexity measure and let $g : \mathcal{N} \rightarrow \mathcal{N}$ be a computable function with $g(n) \geq n$. Assume that machines M_f and M_g compute f and g ,

respectively. Consider the language $L_{f,g}$ as in Definition 5. Then $L_{f,g}$ is decided by a Turing machine A whose f -complexity is $O(f(T_A(n), S_A(n)))$, where

$$T_A(n) \leq c_4 [T_{M_g}(n) + g(n) [T_{M_f}(c_3g(n)) + S_{M_g}(n) + g(n) + S_{M_f}(c_2g(n))]]$$

$$S_A(n) \leq c_1 [S_{M_g}(n) + g(n) + S_{M_f}(c_2g(n))]$$

and $c_i \in \mathcal{N}$ is a constant, $1 \leq i \leq 4$. Moreover, $L_{f,g}$ cannot be decided by any Turing machine with f -complexity $o(g(n))$.

Proof. Theorem 1 implies that $L_{f,g}$ is not decidable by any Turing machine with f -complexity $o(g(n))$. The following Turing machine A decides $L_{f,g}$. On input $\langle M \rangle$, A computes as follows, where we let $n = |\langle M \rangle|$:

1. Compute $g(n)$.
2. Simulate one step of M on input $\langle M \rangle$, while saving the number of steps t and the amount of space s reached so far by M .
3. Compute $f(t, s)$. If $f(t, s) > g(n)$, reject and halt.
4. Verify whether $t > ng(n)2^{g(n)}$. Reject and halt if this is the case.
5. If this is the last step of M , then A accepts and halts if M accepts, otherwise A rejects and halts.
6. Return to step 2.

Step 4 is necessary because for some functions f , machine M may get into an infinite loop while using only a finite amount of space, keeping the value $f(t, s)$ constant. First we prove that $L(A) = L_{f,g}$ and then we bound the f -complexity of A .

Lemma 1. *If Turing machine M halts on input $\langle M \rangle$, then $\langle M \rangle$ is not rejected by A at step 4.*

Proof. Suppose that $\langle M \rangle$ is rejected by A at step 4. Since $\langle M \rangle$ was not just rejected at step 3, we know that $f(t, s) \leq g(n)$.

If $s > g(n)$, then clearly $t > g(n)$, and so $f(t, s) > g(n)$, because f is a natural complexity measure. Hence, $s \leq g(n)$.

Therefore there exist no more than $ng(n)2^{g(n)}$ possible configurations for machine M on input $\langle M \rangle$. But rejection at step 4 requires $t > ng(n)2^{g(n)}$. This establishes that if A rejects M at step 4 then M never finishes its computation on input $\langle M \rangle$, contradicting the fact that M halts on input $\langle M \rangle$. \square

Continuing with the theorem, suppose that $\langle M \rangle \in L_{f,g}$. The previous lemma implies that $\langle M \rangle$ is not rejected by A at step 4. Since $f\text{-ts}_M(\langle M \rangle) \leq g(n)$ and f is non-decreasing,

then $\langle M \rangle$ will not be rejected by A at step 3. Therefore, the simulation will halt at step 5 and, since M accepts $\langle M \rangle$, so must A . This shows that $\langle M \rangle \in L(A)$.

Now let $\langle M \rangle \in L(A)$. Then $\langle M \rangle$ is accepted at step 5, and so machine M also accepts $\langle M \rangle$. Because $\langle M \rangle$ is not rejected by A at step 3, we conclude that $f\text{-}ts_M(\langle M \rangle) \leq g(|\langle M \rangle|)$. Therefore $\langle M \rangle \in L_{f,g}$. Thus, $L(A) = L_{f,g}$.

We now turn to the simulation done in step 2. Machine A divides its tape into 7 tracks, organizing the resulting tracks as follows:

- Track 1 will hold $g(n)$.
- Track 2 will hold $ng(n)2^{g(n)}$.
- Track 3 will save the t counter.
- Track 4 will store the s counter.
- Track 5 will be used to compute $f(t, s)$.
- Track 6 will hold the code for M and its current state.
- Track 7 will be the same as the tape of M .

Machine A simulates machine M and always keeps the information on the tracks close together.

First, let us determine the space complexity $S_A(n)$ of A . The value $g(n)$ is computed in space $S_{M_g}(n)$, since M_g computes g . The value $ng(n)2^{g(n)}$ can easily be computed and stored in space $O(g(n))$. The counter s is limited by counter t which, in turn, is bounded by the value in track 2. Therefore tracks 3 and 4 are asymptotically irrelevant. The value $f(t, s)$ can be computed in space $S_{M_f}(c_2g(n))$ because the sizes of t and s are asymptotically bounded by $g(n)$. The description of M in track 6 has size n , and because $g(n) \geq n$ this space is also asymptotically irrelevant. Track 7 is also bounded by $g(n)$ (as in the proof of the lemma). Therefore $S_A(n)$ satisfies

$$S_A(n) \leq c_1 [S_{M_g}(n) + g(n) + S_{M_f}(c_2g(n))] . \quad (4)$$

It remains to find an upper bound on $T_A(n)$. We know that A computes $g(n)$ in time $T_{M_g}(n)$. The multiplication at step 4 can easily be carried out in time $O(g(n)^2)$. During the simulation, A needs to compute $f(t, s)$. We always have $s \leq t \leq ng(n)2^{g(n)} + 1$ which in binary has length $O(g(n))$. Hence $f(t, s)$ is computed in time $T_{M_f}(c_3g(n))$, for some integer constant c_3 . In each step of the simulation, the tracks need to be shifted and some values must be compared. This can be done in time proportional to the size of the tracks, i.e., $c_1[S_{M_g}(n) + g(n) + S_{M_f}(c_2g(n))]$. Finally, no more than $O(g(n))$ steps are simulated. Therefore, $T_A(n)$ satisfies

$$T_A(n) \leq c_4 [T_{M_g}(n) + g(n) [T_{M_f}(c_3g(n)) + S_{M_g}(n) + g(n) + S_{M_f}(c_2g(n))]] . \quad (5)$$

□

The classical time and space hierarchy theorems can be derived as a particular case of the general hierarchy theorem. Different formulations and proofs of these theorems can be found in [4, 5].

Definition 7. A function $g : \mathcal{N} \rightarrow \mathcal{N}$ is called *space constructible* if the function that maps 1^n to the binary representation of $g(n)$ is computable in space $O(g(n))$.

Corollary 1 (Space Hierarchy). Let $g : \mathcal{N} \rightarrow \mathcal{N}$, with $g(n) \geq n$, be a space constructible function. Then there exists a language A that is decidable in space $O(g(n))$ but not in space $o(g(n))$.

Proof. Let $f(x, y) = y$. Then Theorem 2 can be applied and we have that $L_{f,g}$ is decidable within f -complexity $O(S_A(n))$. But, again by Theorem 2, $S_A(n) \leq c_1[S_{M_g}(n) + g(n) + S_{M_f}(c_2g(n))]$. Because g is space constructible, we have that $S_{M_g}(n)$ is $O(g(n))$, since $g(n) \geq n$. By the definition of f , we can assume that $S_{M_f}(n)$ is $O(n)$. Hence, $L_{f,g}$ is decidable in space $O(g(n))$. By Theorem 1, $L_{f,g}$ cannot be decided in space $o(g(n))$. \square

Definition 8. A function $g : \mathcal{N} \rightarrow \mathcal{N}$ is called *time constructible* if the function that maps 1^n to the binary representation of $g(n)$ is computable in time $O(g(n))$.

Corollary 2 (Time Hierarchy). For any time constructible function $g : \mathcal{N} \rightarrow \mathcal{N}$, with $g(n) \geq n$, there exists a language A that is decidable in time $O(g(n)^2)$ but not in time $o(g(n))$.

Proof. Let $f(x, y) = x$. Then, using Theorem 2 and the definition of f , $L_{f,g}$ is decidable in time $O(T_A(n))$, where $T_A(n)$ satisfies (5). We also have that $S_{M_g}(n) \leq T_{M_g}(n)$ and $T_{M_g}(n)$ is $O(g(n))$ because g is a time constructible function. By the definition of f , we can assume that $T_{M_f}(n)$ and $S_{M_f}(n)$ are $O(n)$. Therefore, $L_{f,g}$ is decidable in time $O(g(n)^2)$. By Theorem 1, $L_{f,g}$ is not decidable in time $o(g(n))$. \square

4 Conclusion

In this note we generalized the deterministic time and space hierarchy theorems, and derived the traditional versions of these results as a particular case of our general theorem. Our result shows that the hierarchy theorems are in fact inserted in a wider context. In particular, the standard notions of time and space do not need to be considered apart or remain restricted in order to prove the existence of complexity hierarchies.

References

- [1] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.
- [2] J. Hartmanis, P.L. Lewins II, and R.E. Stearns. Hierarchies of memory-limited computations. In *Proc. 6th Annual IEEE Symp. on Switching Circuit Theory and Logic Design*, pages 179–190, 1965.
- [3] J. Hartmanis and R.E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117 (5):285–306, 1965.
- [4] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Co., 1994.

- [5] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1996.