Electronic Colloquium on Computational Complexity, Report No. 25 (2009)

A New Look at Some Classical Results in Computational Complexity

Igor Carboni Oliveira* Arnaldo Vieira Moura†

Abstract

We propose a generalization of the traditional algorithmic space and time complexities. Using the concept introduced, we derive an unified proof for the deterministic time and space hierarchy theorems, now stated in a much more general setting. This opens the possibility for the unification and generalization of other results that apply to both the time and space complexities. As an example, we present a similar approach for the gap theorems.

1 Introduction

Many results in computational complexity theory are based on considerations about the minimum amount of time and space required to solve certain computational problems. This is expected since these two natural complexity measures are intrinsic to the underlying computational model, the Turing machine.

Other abstract theories that have been proposed define the notion of complexity measure in a broader way. One of the most celebrated of these theories was put forward by Manuel Blum in the sixties [1], but the truly most interesting measures are still time and space.

In this paper we present the notion of f-complexity, a simple and natural generalization of the space and time complexity measures. The use of f-complexity allows for the unification of results that have similar proofs for both the time and space complexity measures.

*Institute of Computing - University of Campinas, 13084-971, Campinas, Brazil — igorcarb@gmail.com. Research supported by FAPESP grant 08/07040-0.

†Institute of Computing - University of Campinas, 13084-971 - Campinas, Brazil — arnaldo@ic.unicamp.br. Research supported by CNPq, grant 304363/2008-1, and by FAPESP grant 07/56052-8.

ISSN 1433-8092

We illustrate by proving a generalized and unified version of the time and space hierarchy theorems, two remarkable results originally proved by Hartmanis et al [4, 5]. We show that these hierarchy theorems are in fact inserted in a much wider context, being direct consequences of the generalized version of the hierarchy theorems. As far as we know, this is the first unified proof of the time and space hierarchy theorems.

Finally, applying the same ideas, we present an unified proof for the space and time gap theorems, classical results independently discovered by Trakhtenbrot and Borodin [8, 2].

2 The Space-Time Complexity

To avoid technical details, the Turing machines discussed here are one-tape deterministic machines that halt on all inputs and with a $\{0,1\}$ binary tape alphabet. The only exception occurs when a machine is presented as input, since it is impossible to guarantee that it will always halt. We will not consider sublinear space, therefore it will not be necessary to consider separated input, output and work tapes [7]. We will use the standard big-O and small-o notation, as in [3, 7].

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 $t_A : \{0,1\}^\star \to \mathcal{N}$ 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 $T_A : \mathcal{N} \to \mathcal{N}$ 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 $s_A : \{0,1\}^\star \to \mathcal{N}$ such that started with x on its input tape A scans exactly $s_A(x)$ distinct tape cells. The* space complexity *of A is the function $S_A : \mathcal{N} \to \mathcal{N}$ 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 $f : \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ be an arbitrary function. Now, consider the associated function f-$ts_A : \{0,1\}^\star \to \mathcal{N}$ given by letting f-$ts_A(x) = f(t_A(x), s_A(x))$. Then the* f-complexity *of A is the function f-$TS_A : \mathcal{N} \to \mathcal{N}$ where f-$TS_A(n) = max_{x\in\{0,1\}^n}\{$f-$ts_A(x)\}$.*

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, 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 such that f-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 by [7].

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

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

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

Proof. For the sake of contradiction, suppose that Turing machine A decides $L_{f,g}$ and *f-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 $w \in \{0,1\}^\star$.

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

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

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 $x \in \{0,1\}^\star$ with $n = |x| \geq n_0$. But, clearly, *f-ts*$_A$ = *f-ts*$_{B'}$, and so

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

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 accetps $\langle B' \rangle$ 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 $o(g(n))$. □

As an example, 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 language of all codes $\langle M \rangle$ 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 guarantees 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 obtainig the f-complexity of a specific Turing machine that decides $L_{f,g}$. This can be done by simulating machine M on input $\langle M \rangle$.

Informally, for $L_{f,g}$ to be decidable, the following conditions are sufficient:

1. f and g are computable functions. For a definition of computable functions, see [7];

2. f is increasing function, otherwise M could have very long computations without violating the $g(|\langle M \rangle|)$ bound.

The last item is captured by the following definition.

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

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

Theorem 2 (Space-Time Hierarchy)**.** *Let f be a computable natural complexity measure and let $g : \mathcal{N} \to \mathcal{N}$ 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*

$$T_A(n) \leq c_4\big[T_{M_g}(n)+g(n)\left[T_{M_f}(c_3 g(n)) + S_{M_g}(n) + g(n) + S_{M_f}(c_2 g(n))\right]\big]$$

$$S_A(n) \leq c_1\big[S_{M_g}(n) + g(n) + S_{M_f}(c_2 g(n))\big]$$

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 another step of M on input $\langle M \rangle$, while saving the number of steps t and the amount of space s reached so far.
3. Compute $f(t, s)$. If $f(t, s) > g(n)$, reject and halt.
4. Verify whether $t > g(n)\ 2^{g(n)}$. 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 3.

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 limit 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 $g(n)2^{g(n)}$ possible configurations for machine M on input $\langle M \rangle$. But rejection at step 4 requires $t > g(n)\ 2^{g(n)}$. 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 $\langle M \rangle$. □

Continuing with the theorem, suppose that $\langle M \rangle \in L_{f,g}$. The previous lemma says that $\langle M \rangle$ is not rejected by A at step 4. Since *f*-*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 also M accepts $\langle M \rangle$. Because $\langle M \rangle$ is not rejected by A at step 3, we conclude that *f*-*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 modulo 7 its tape cells, organizing the resulting tracks as follows:

Track 1 will hold $g(n)$.

Track 2 will hold $g(n)2^{g(n)}$.

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 $g(n)2^{g(n)}$ 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 $S_{M_f}(c_2 g(n))$ 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

$$S_A(n) \leq c_1 \big[S_{M_g}(n) + g(n) + S_{M_f}(c_2 g(n)) \big]. \tag{4}$$

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 1 can be easily carried out in time $O(g(n)^2)$. During the simulation, A needs to compute $f(t, s)$. We always have $s \leq t \leq g(n)2^{g(n)} + 1$ which in binary has length $O(g(n))$.

Hence $f(t,s)$ is computed in time $T_{M_f}(c_3 g(n))$, 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 tapes, i.e., $c_1[S_{M_g}(n) + g(n) + S_{M_f}(c_2 g(n))\ ]$. Finally, no more than $O(g(n))$ steps are simulated. Therefore, $T_A(n)$ satisfies

$$\begin{aligned} T_A(n) \leq c_4\big[&T_{M_g}(n) \\ &+ g(n)\big[T_{M_f}(c_3 g(n)) + S_{M_g}(n) + g(n) + S_{M_f}(c_2 g(n))\big]\big]. \end{aligned} \tag{5}$$

□

The question about the existence of a more efficient simulation is inessential to our purposes. The previous theorem readily implies versions of the time and space hierarchy theorems. Different formulations and proofs of these theorems can be found in [6, 7].

Definition 7. *A function $g : \mathcal{N} \to \mathcal{N}$ 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 $g : \mathcal{N} \to \mathcal{N}$, 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, $S_A(n) \leq c_1[S_{M_g}(n) + g(n) + S_{M_f}(c_2 g(n))\ ]$. 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 $o(g(n))$.

□

Definition 8. *A function $g : \mathcal{N} \to \mathcal{N}$ 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 $g : \mathcal{N} \to \mathcal{N}$, 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 $o(g(n))$.

□

4 The Gap Theorem

In this section we modify the proof presented in [6], unifying and generalizing the so called gap theorems to any computable natural f-complexity measure.

Definition 9. *A language L is in the class f-COMPLEXITY$(g(n))$ if there is a Turing machine A that decides L and for all integers n, it holds that f-TS$_A(n) \leq g(n)$.*

Theorem 3 (The Gap Theorem)**.** *Let f be a computable natural complexity measure. There is a recursive function $g : \mathcal{N} \to \mathcal{N}$ such that if $L \in f$-COMPLEXITY$(2^{g(n)})$, then L is decidable with f-complexity $O(g(n))$.*

Proof. We consider all Turing machines ordered lexicographically: M_0, M_1, M_2, ... For $i, k \geq 0$, we define property $P(i, k)$:

> For all j, $0 \leq j \leq i$, machine M_j, when started on any input x of length i, will satisfy either *f-ts*$_{M_j}(x) < k$ or *f-ts*$_{M_j}(x) > 2^k$ or it will have *f-ts*$_{M_j}(x)$ undefined (because M_j may not halt on some inputs).

Note that $P(i, k)$ can be decided by simulation applying the same ideas presented in the proof of theorem 2.

Now we define the desired function g. Consider the following sequence of values for k: $k_1 = 0$, and, for $j \geq 2$, $k_j = 2^{k_{j-1}} + 1$. Now, fix some $i \geq 0$. Given a pair of machine M_j and input x, with $j \leq i$ and $|x| = i$, there is at most one $p \geq 0$ for which $P(i, k_p)$ is false. Hence, there must be an integer l such that $P(i, k_l)$ is true. Select the least such integer and define $g(i) = k_l$. Clearly, g is a computable function and we have $P(i, g(i))$ true for all i.

Now suppose that $L \in f$-COMPLEXITY$(2^{g(n)})$. Then L is decided by some Turing machine M_j within f-complexity $2^{g(n)}$. Now consider any input x with $|x| \geq j$. By construction, $P(|x|, g(|x|))$ is true. We have $j \leq |x|$, therefore for M_j: *f-ts*$_{M_j}(x) < g(|x|)$ or *f-ts*$_{M_j}(x) > 2^{g(|x|)}$ or *f-ts*$_{M_j}(x)$ is undefined. Since M_j decides L within f-complexity $2^{g(n)}$, it follows that *f-ts*$_{M_j}(x) < g(|x|)$. Hence L is decidable with f-complexity $O(g(n))$.

□

When we let f be any of the two binary projection functions we get, again, classical time and space gap theorems.

5 Conclusion

The level of abstraction introduced by f-complexity measures suggests a conceptual framework in which to present unified proofs in computational complexity theory. It also allows for the generalization of important results in the theory of complexity. Here, the traditional hierarchy theorems have been generalized and unified in a much wider context. A similar approach for the gap theorems is also presented. We believe that other important results that involve both the space and time complexity measures can be unified and generalized in a similar way.

References

[1] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14 (2):322–336, 1967.

[2] A. Borodin. Computational complexity and the existence of complexity gaps. *Journal of the ACM*, 19 (1):158–174, 1972.

[3] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[4] 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.

[5] J. Hartmanis and R.E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117 (5):285–306, 1965.

[6] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Co., 1994.

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

[8] B. A. Trakhtenbrot. Turing computations with logarithmic delay. *Algebra i Logika*, 3 (4):33–48, 1964.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/