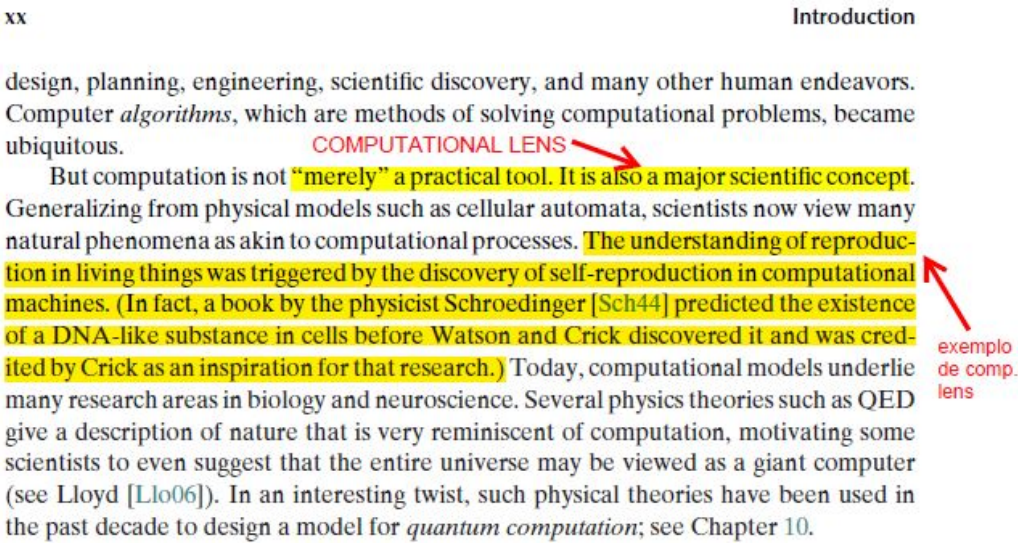
Introdução

Passagem sobre “computational lens”:



xx Introduction

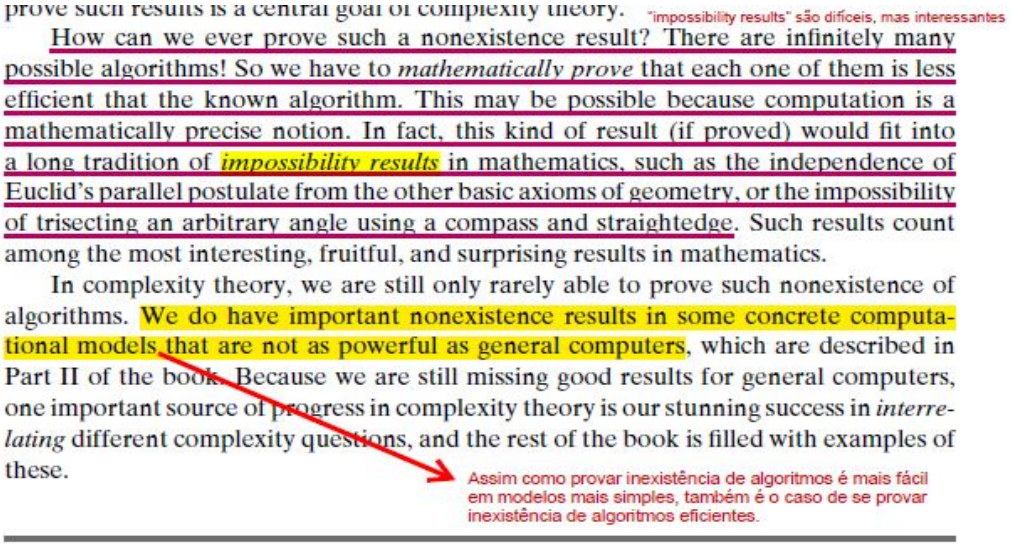
design, planning, engineering, scientific discovery, and many other human endeavors. Computer *algorithms*, which are methods of solving computational problems, became ubiquitous.

COMPUTATIONAL LENS

But computation is not “merely” a practical tool. It is also a major scientific concept. Generalizing from physical models such as cellular automata, scientists now view many natural phenomena as akin to computational processes. The understanding of reproduction in living things was triggered by the discovery of self-reproduction in computational machines. (In fact, a book by the physicist Schroedinger [Sch44] predicted the existence of a DNA-like substance in cells before Watson and Crick discovered it and was credited by Crick as an inspiration for that research.) Today, computational models underlie many research areas in biology and neuroscience. Several physics theories such as QED give a description of nature that is very reminiscent of computation, motivating some scientists to even suggest that the entire universe may be viewed as a giant computer (see Lloyd [Llo06]). In an interesting twist, such physical theories have been used in the past decade to design a model for *quantum computation*; see Chapter 10.

exemplo de comp. lens

Passagem sobre demonstrações em modelos mais fracos:



prove such results is a central goal of complexity theory.

“impossibility results” são difíceis, mas interessantes

How can we ever prove such a nonexistence result? There are infinitely many possible algorithms! So we have to *mathematically prove* that each one of them is less efficient that the known algorithm. This may be possible because computation is a mathematically precise notion. In fact, this kind of result (if proved) would fit into a long tradition of *impossibility results* in mathematics, such as the independence of Euclid’s parallel postulate from the other basic axioms of geometry, or the impossibility of trisecting an arbitrary angle using a compass and straightedge. Such results count among the most interesting, fruitful, and surprising results in mathematics.

In complexity theory, we are still only rarely able to prove such nonexistence of algorithms. We do have important nonexistence results in some concrete computational models that are not as powerful as general computers, which are described in Part II of the book. Because we are still missing good results for general computers, one important source of progress in complexity theory is our stunning success in *interrelating* different complexity questions, and the rest of the book is filled with examples of these.

Assim como provar inexistência de algoritmos é mais fácil em modelos mais simples, também é o caso de se provar inexistência de algoritmos eficientes.

Capítulo 0

Alfabetos, strings, linguagens

Dado alfabeto S: (tipicamente $S = \{0, 1\}$)

- S^n : conjunto de strings de tamanho n sobre S
- S^* : conjunto de todas as strings sobre S

Dada uma string (ou um vetor) x, denotamos x_i o i-ésimo bit

Dados $x, y \in \{0,1\}^n$, $\quad x \odot y = \sum_i x_i y_i \pmod 2$

(no caso de produto interno de vetores reais, complexos : $\langle u, v \rangle$

Ex $f: \{0,1\}^* \to \{0,1\}$ (função booleana com 1 bit de output)

NOTAÇÃO:

$$L_f = \{x : f(x) = 1\}$$

➡ L_f é um PROBLEMA DE DECISÃO.

(testar se uma string pertence a L_f significa decidir L_f)

OBJETO MATEMÁTICO (e.g., grafo, número, etc) : x

CODIFICAÇÃO EM BINÁRIO DO OBJETO : $\lfloor x \rfloor$

Para um par ordenado x, y as vezes usa-se $\langle x, y \rangle$ ao invés de $\lfloor \langle x, y \rangle \rfloor$

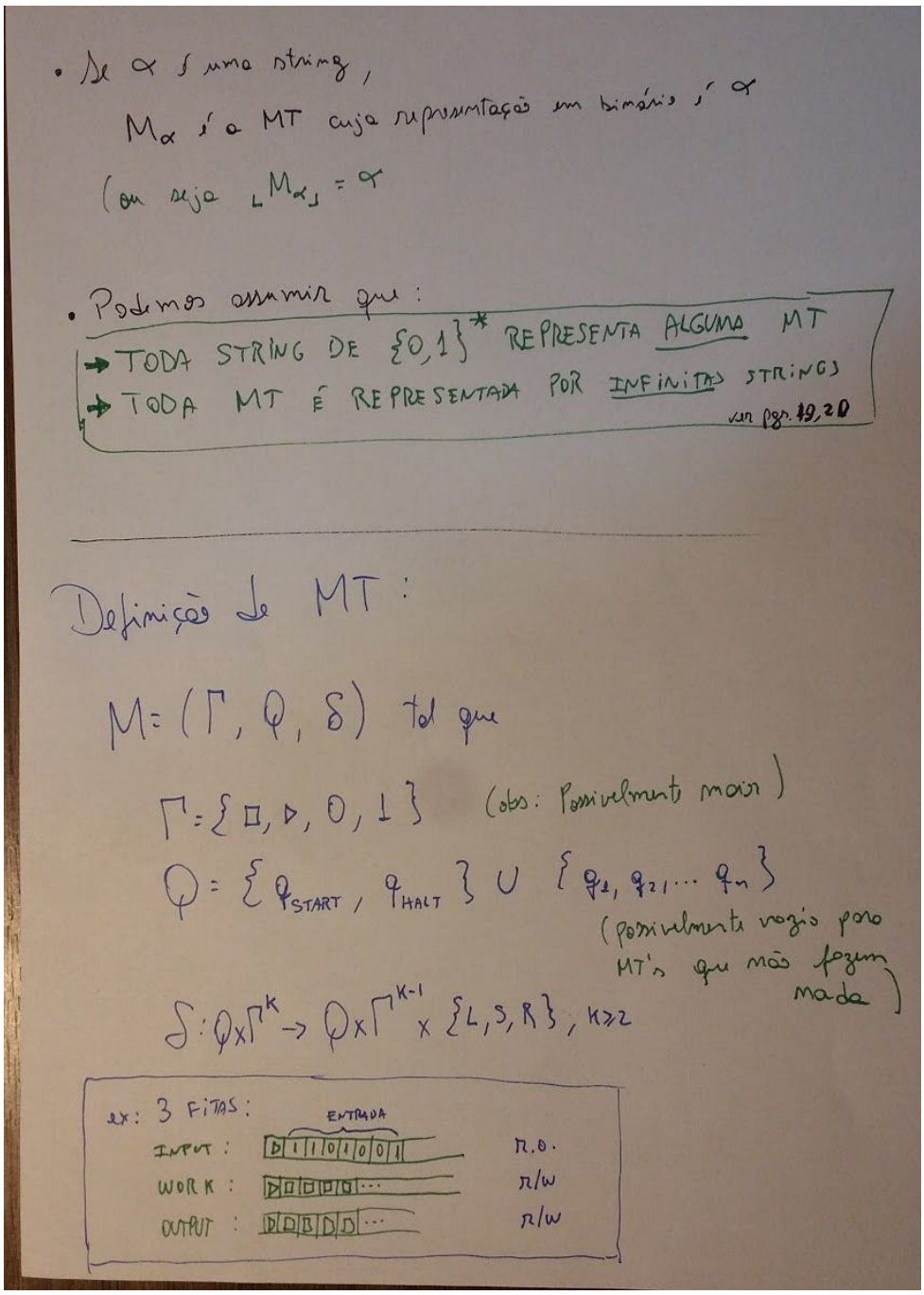
Capítulo 1

Pág.9 (Tese de Church-Turing)

Tese de Church Turing

Surprisingly enough, it turns out there there is a simple mathematical model that suffices for studying many questions about computation and its efficiency—the *Turing machine*. It suffices to restrict attention to this single model since it seems able to *simulate* all physically realizable computational methods with little loss of efficiency.

Seções 1.1 e 1.2



Seção 1.3: Variações de Máquinas de Turing

(Alfabetos diferentes)

Overhead logarítmico para simular alfabetos maiores

Claim 1.5 *For every $f : \{0,1\}^* \to \{0,1\}$ and time-constructible $T : \mathbb{N} \to \mathbb{N}$, if f is computable in time $T(n)$ by a TM M using alphabet Γ, then it is computable in time $4 \log |\Gamma| T(n)$ by a TM $\tilde{M}$ using the alphabet $\{0, 1, \square, \triangleright\}$.* ◇

Moral: tamanho do alfabeto é irrelevante (overhead logarítmico no tamanho do alfabeto, o que significa constante na maioria dos casos, pois MTs tem tamanho fixo)

PROOF SKETCH: Let M be a TM with alphabet Γ, k tapes, and state set Q that computes the function f in $T(n)$ time. We describe an equivalent TM $\tilde{M}$ computing f with alphabet $\{0, 1, \square, \triangleright\}$, k tapes and a set Q' of states. The idea behind the transformation is simple: One can encode any member of Γ using $\log |\Gamma|$ bits.[5] Thus, each of $\tilde{M}$'s work tapes will simply encode one of M's tapes: For every cell in M's tape we will have $\log |\Gamma|$ cells in the corresponding tape of $\tilde{M}$ (see Figure 1.3).

To simulate one step of M, the machine $\tilde{M}$ will (1) use $\log |\Gamma|$ steps to read from each tape the $\log |\Gamma|$ bits encoding a symbol of Γ, (2) use its state register to store the symbols read, (3) use M's transition function to compute the symbols M writes and M's new state given this information, (4) store this information in its state register, and (5) use $\log |\Gamma|$ steps to write the encodings of these symbols on its tapes.

ou seja, guarda a informação nos estados de M'

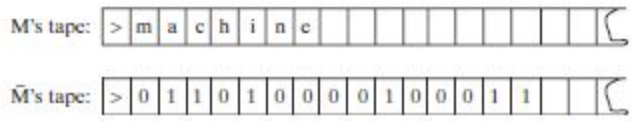


Figure 1.3. We can simulate a machine M using the alphabet $\{\triangleright, \square, \mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$ by a machine M' using $\{\triangleright, \square, 0, 1\}$ via encoding every tape cell of M using five cells of M'.

One can verify that this can be carried out if $\tilde{M}$ has access to registers that can store M's state, k symbols in Γ, and a counter from 1 to $\log |\Gamma|$. Thus, there is such a machine $\tilde{M}$ utilizing no more than $c|Q||\Gamma|^{k+1}$ states for some absolute constant c. (In general, we can always simulate several registers using one register with a larger state space. For example, we can simulate three registers taking values in the sets A, B and C, respectively, with one register taking a value in the set $A \times B \times C$, which is of size $|A||B||C|$.)

It is not hard to see that for every input $x \in \{0,1\}^n$, if on input x the TM M outputs $f(x)$ within $T(n)$ steps, then $\tilde{M}$ will output the same value within less than $4 \log |\Gamma| T(n)$ steps. ■

??

Mais ou menos fitas

Overhead quadrático para simular k fitas em apenas 1 fita

Claim 1.6 *Define a single-tape Turing machine to be a TM that has only one read-write tape, that is used as input, work, and output tape. For every $f : \{0,1\}^* \to \{0,1\}$ and time-constructible $T : \mathbb{N} \to \mathbb{N}$, if f is computable in time $T(n)$ by a TM M using k tapes, then it is computable in time $5kT(n)^2$ by a single-tape TM $\bar{M}$.* ◇

Obs: quadrático **para todo k**

Prova:

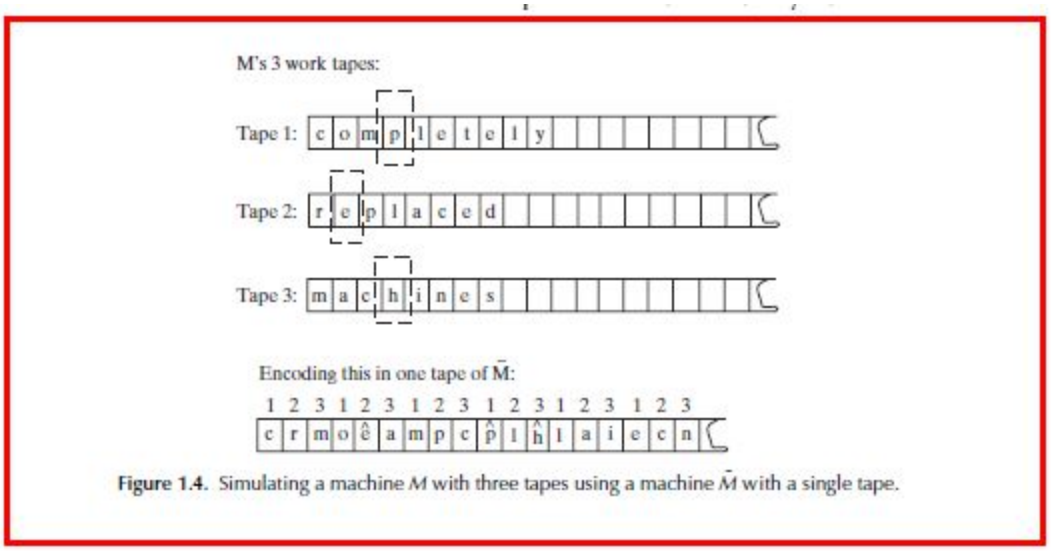


Figure 1.4. Simulating a machine M with three tapes using a machine $\bar{M}$ with a single tape.

Fita infinita nas duas direções

overhead linear

Claim 1.8 *Define a bidirectional TM to be a TM whose tapes are infinite in both directions. For every $f : \{0,1\}^* \to \{0,1\}^*$ and time-constructible $T : \mathbb{N} \to \mathbb{N}$, if f is computable in time $T(n)$ by a bidirectional TM M, then it is computable in time $4T(n)$ by a standard (unidirectional) TM $\bar{M}$.* ◇

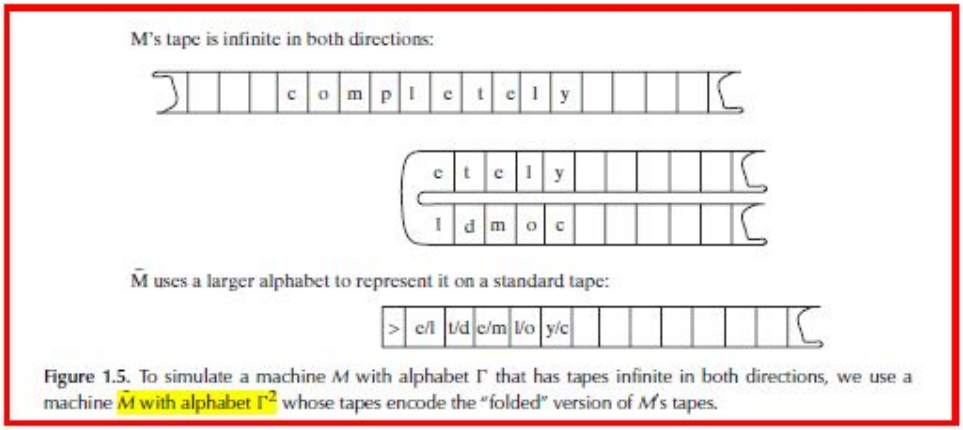


Figure 1.5. To simulate a machine M with alphabet Γ that has tapes infinite in both directions, we use a machine $\bar{M}$ with alphabet Γ^2 whose tapes encode the "folded" version of M's tapes.

Seção 1.4: MÁQUINAS DE TURING UNIVERSAIS

Theorem 1.9 *(Efficient universal Turing machine)*
There exists a TM $\mathcal{U}$ such that for every $x, \alpha \in \{0,1\}^*$, $\mathcal{U}(x,\alpha) = M_\alpha(x)$, where M_α denotes the TM represented by α.
Moreover, if M_α halts on input x within T steps then $\mathcal{U}(x,\alpha)$ halts within $CT \log T$ steps, where C is a number independent of $|x|$ and depending only on M_α's alphabet size, number of tapes, and number of states.

PROOF OF RELAXED VERSION OF THEOREM 1.9: Our universal TM $\mathcal{U}$ is given an input x, α, where α represents some TM M, and needs to output $M(x)$. A crucial observation is that we may assume that M (1) has a single work tape (in addition to the input and output tape) and (2) uses the alphabet $\{\triangleright, \square, 0, 1\}$. The reason is that $\mathcal{U}$ can transform a

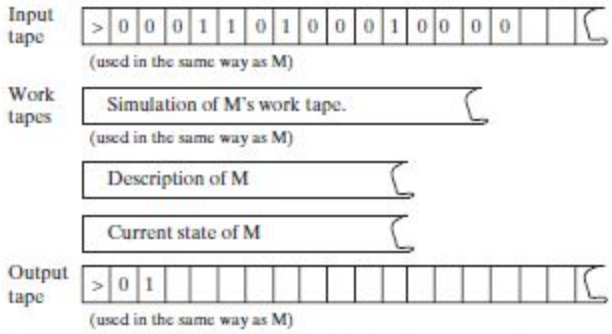


Figure 1.6. The universal TM $\mathcal{U}$ has in addition to the input and output tape, three work tapes. One work tape will have the same contents as the simulated machine M, another tape includes the description M (converted to an equivalent one-work-tape form), and another tape contains the current state of M.

MT universal com contador de tempo:

Universal TM with time bound
It is sometimes useful to consider a variant of the universal TM $\mathcal{U}$ that gets a number T as an extra input (in addition to x and α), and outputs $M_\alpha(x)$ if and only if M_α halts on x within T steps (otherwise outputting some special failure symbol). By adding a time counter to $\mathcal{U}$, the proof of Theorem 1.9 can be easily modified to give such a universal TM. The time counter is used to keep track of the number of steps that the computation has taken so far.

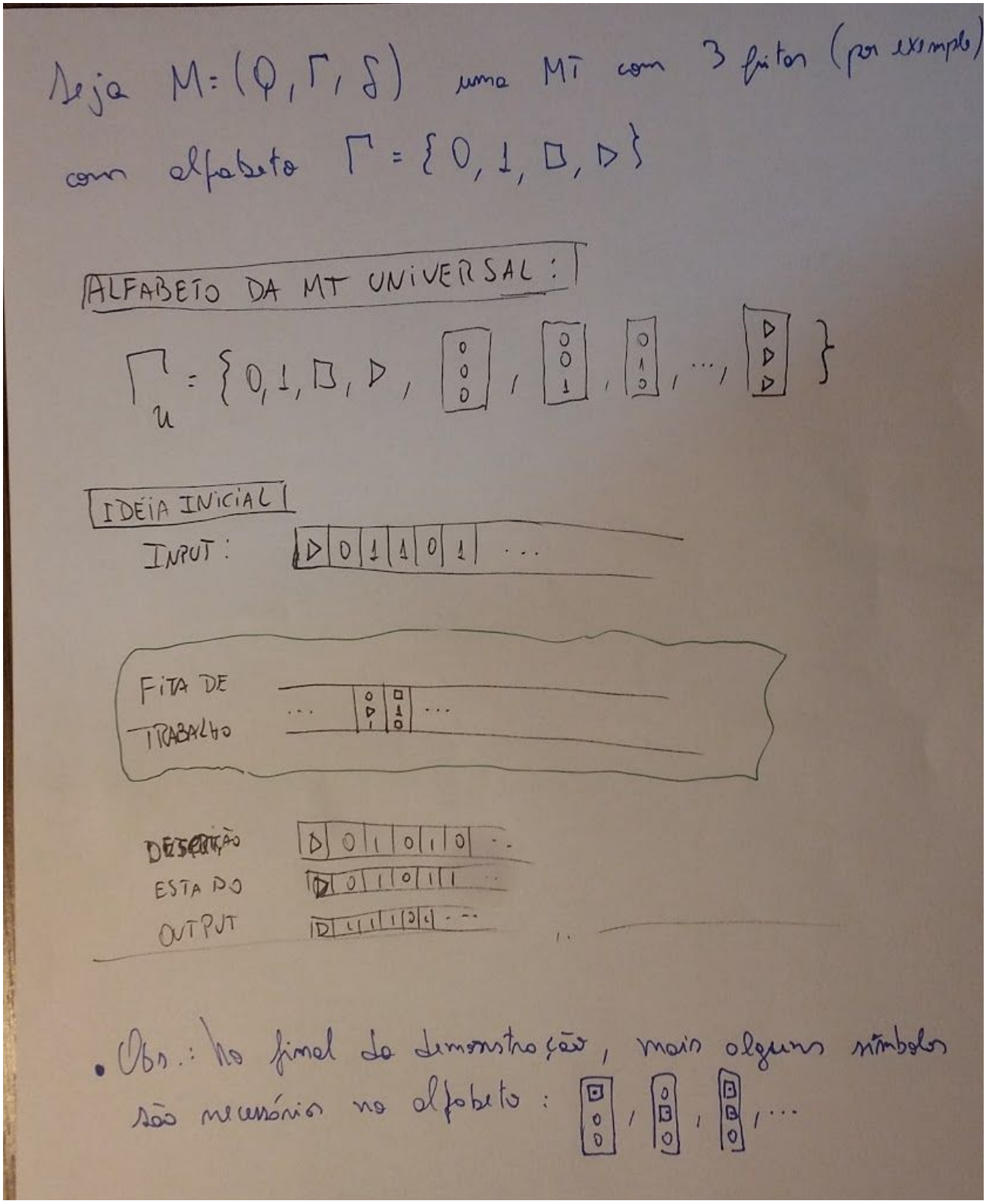
MTs abstraídas (Exercício)

Remark 1.7 *(Oblivious Turing machines)*
With a bit of care, one can ensure that the proof of Claim 1.6 yields a TM $\tilde{M}$ with the following property: Its head movements do not depend on the input but only depend on the input length. That is, every input $x \in \{0,1\}^*$ and $i \in \mathbb{N}$, the location of each of M's heads at the ith step of execution on input x is only a function of $|x|$ and i. A machine with this property is called *oblivious*, and the fact that every TM can be simulated by an oblivious TM will simplify some proofs later on (see Exercises 1.5 and 1.6 and the

Obs: overhead logarítimico

UNIVERSALIDADE EFICIENTE

Ideia da prova



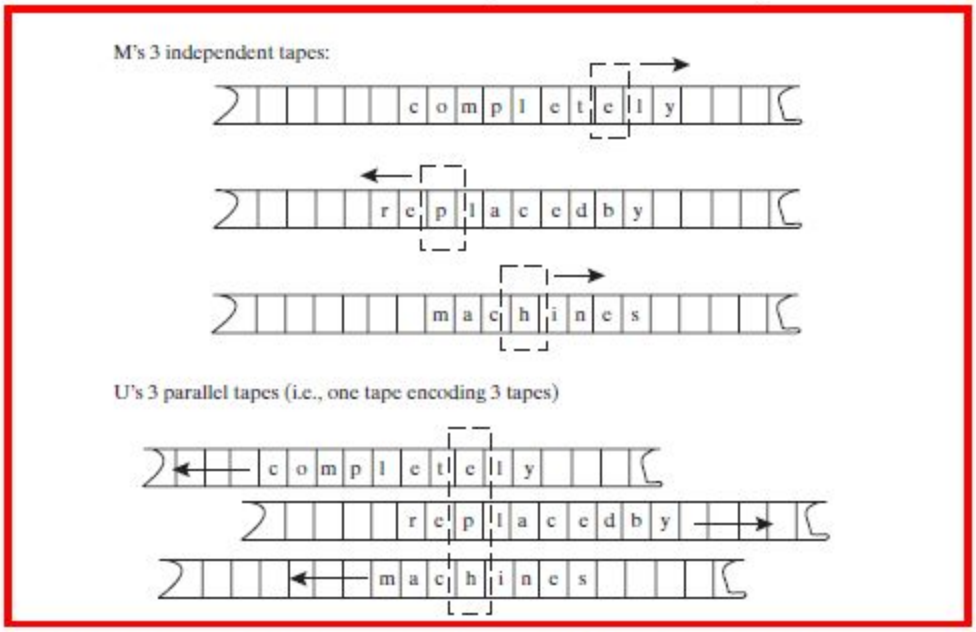


Figure 1.8. Packing k tapes of M into one tape of $\mathcal{U}$. We consider $\mathcal{U}$'s single work tape to be composed of k parallel tapes, whose heads move in unison, and hence we shift the contents of these tapes to simulate independent head movement.

Figure 1.9. Performing a shift of the parallel tapes. The left shift of the first tape involves zones $R_0, L_0, R_1, L_1, R_2, L_2$, the right shift of the second tape involves only R_0, L_0, while the left shift of the third tape involves zones R_0, L_0, R_1, L_1. We maintain the invariant that each zone is either empty, half-full, or full and that the total number of nonempty cells in $R_i \cup L_i$ is $2 \cdot 2^i$. If before the left shift zones $L_0, \ldots, L_{i-1}$ were full and L_i was half-full (and so $R_0, \ldots, R_{i-1}$ were full and R_i half-full), then after the shift zones $R_0, L_0, \ldots, R_{i-1}, L_{i-1}$ will be half-full, L_i will be full and R_i will be empty.

Detalhes da prova:

shall always maintain the following invariants:

- Each of the zones (1) is either *empty*, *full*, or *half-full* with non-⊠ symbols. That is, the number of symbols in zone R_t that are not ⊠ is either 0, 2^t, or $2 \cdot 2^t$ and the same holds for L_t. (We treat the ordinary □ symbol the same as any other symbol in Γ, and in particular a zone full of □'s is considered full.)
  We assume that initially all the zones are half-full. We can ensure this by filling half of each zone with ⊠ symbols in the first time we encounter it.
- The total number of non-⊠ symbols in $R_t \cup L_t$ is $2 \cdot 2^t$. That is, (2) either R_t is empty and L_t is full, or R_t is full and L_t is empty, or they are both half-full.
- Location 0 always contains a non-⊠ symbol. (3)

Performing a shift

The advantage in setting up these zones is that now when performing the shifts, we do not always have to move the entire tape, but we can restrict ourselves to only using some of the zones. We illustrate this by showing how $\mathcal{U}$ performs a left shift on the first of its parallel tapes (see also Figure 1.9):

1. $\mathcal{U}$ finds the smallest i_0 such that R_{i_0} is not empty. Note that this is also the smallest i_0 such that L_{i_0} is not full. We call this number i_0 the *index* of this particular shift.
2. $\mathcal{U}$ puts the leftmost non-⊠ symbol of R_{i_0} in position 0 and shifts the remaining leftmost $2^{i_0} - 1$ non-⊠ symbols from R_{i_0} into the zones $R_0, \ldots, R_{i_0-1}$ filling up exactly half the symbols of each zone. Note that there is exactly room to perform this since all the zones $R_0, \ldots, R_{i_0-1}$ were empty and indeed $2^{i_0} - 1 = \sum_{j=0}^{i_0-1} 2^j$.
3. $\mathcal{U}$ performs the symmetric operation to the left of position 0. That is, for j starting from $i_0 - 1$ down to 0, $\mathcal{U}$ iteratively moves the $2 \cdot 2^j$ symbols from L_j to fill half the cells of L_{j+1}. Finally, $\mathcal{U}$ moves the symbol originally in position 0 (modified appropriately according to M's transition function) to L_0.
4. At the end of the shift, all of the zones $R_0, L_0, \ldots, R_{i_0-1}, L_{i_0-1}$ are half-full, R_{i_0} has 2^{i_0} fewer non-⊠ symbols, and L_t has 2^t additional non-⊠ symbols. Thus, our invariants are maintained.
5. The total cost of performing the shift is proportional to the total size of all the zones involved $R_0, L_0, \ldots, R_{i_0}, L_{i_0}$. That is, $O(\sum_{j=0}^{i_0} 2 \cdot 2^j) = O(2^{i_0})$ operations.

After performing a shift with index i the zones $L_0, R_0, \ldots, L_{t-1}, R_{t-1}$ are half-full, which means that it will take at least $2^t - 1$ left shifts before the zones $L_0, \ldots, L_{t-1}$ become empty or at least $2^t - 1$ right shifts before the zones $R_0, \ldots, R_{t-1}$ become empty. In any case, once we perform a shift with index i, the next $2^t - 1$ shifts of that particular parallel tape will all have index less than i. This means that for every one of the parallel tapes, at most a $1/2^t$ fraction of the total number of shifts have index i. Since we perform at most T shifts, and the highest possible index is $\log T$, the total work spent in shifting $\mathcal{U}$'s k parallel tapes in the course of simulating T steps of M is

$$O(k \cdot \sum_{t=1}^{\log T} \frac{T}{2^{t-1}} 2^t) = O(T \log T) . \blacksquare$$