

Sobre língua, escrita e raciocínio

O que seria a ciência se não a atividade executada pelos cientistas? Podemos então dar uma olhada no que fazem os cientistas. Certamente cada um deles tem lá seu próprio jeito de trabalhar, mas parece que todos eles têm uma forma mais ou menos comum de se comunicar — pelo menos formalmente. Tipicamente os cientistas publicam suas descobertas por escrito. Cada publicação *explica* a sequência de dedução do autor, levando o leitor (se o raciocínio estiver correto) às mesmas conclusões obtidas pelo autor.

O que é explicar? “Explicar” tem radical comum com *explicatrix* do latim, que significa desdobrar, revelar, ou seja, abrir alguma coisa que se encontra fechada.

O *Oxford Latin Dictionary* (1968) define “*explicatrix: explico-trix: unfolding, revealing. Também “explico: ex-plico: [(a)] to free from folds, creases, coils etc. (b) to unroll (books, etc., for the purpose of reading them). (c) to free from surface irregularities, level out, smooth.*

“Explicar” então é trazer algo de dentro pra fora. Se você tem uma certa ideia, explicá-la é retirá-la de dentro de você e colocá-la pra fora pra que não só outros possam ver, mas pra que você também possa vê-la em concreto. Conversamente, as palavras “implícito” e “implicar” têm radical comum com a palavra *implico* do latim, que significa dobrar (pra dentro), ou seja, pegar algo que está explícito e revertê-lo a uma forma implícita.

O *Oxford Latin Dictionary* (1968) define “*implico: in-plico: (a) to fold or twine (a single object) about itself [...].*

Matemáticos às vezes dizem coisas como — A implica B . Por exemplo, considere a afirmação — se n é um inteiro par, então n^2 também é par. É o mesmo que dizer que a paridade de n implica a paridade do seu quadrado. Em outras palavras, é concebível que a afirmação “ n^2 é par” esteja de alguma forma implícita na afirmação “ n é par”, embora não literalmente.

Teorema. Se n é um número inteiro par, então n^2 também é.

Demonstração. Assuma que n seja um número inteiro par. O significado de n ser um inteiro par é que ele pode ser escrito na forma $n = 2k$, sendo k um número inteiro. (É isso que significa ser um número par.) Se multiplicamos n por ele mesmo, obtemos

$$n^2 = (2k)^2 = 2^2 k^2 = 2(2k^2),$$

que também é par, já que $2k^2$ é um número inteiro, que era o que desejávamos demonstrar. $\square$

Observe que o resultado desejado decorre da assunção de que n é inteiro par. Usamos o significado, a definição de “inteiro par” (e mais alguns fatos da álgebra) pra mostrar que n^2 também satisfaz a definição de número inteiro par, que era a conclusão desejada. Não é absurda, portanto, a concepção de que provar uma implicação signifique retirar a afirmação “ n^2 é par” de dentro da premissa “ n é par”¹.

Uma prova matemática é um certo mapa que nos instrui a partir de um ponto inicial e chegar a um ponto final. Uma prova matemática é, portanto, um algoritmo.

A informação de entrada de um algoritmo é um ponto de partida e o resultado do algoritmo é o ponto final. Por exemplo, computar a raiz quadrada de 16 é começar com a informação “16” e terminar com a informação “4”. Quais são os passos que devo executar pra chegar até lá? Em outras palavras, qual é o caminho? O caminho é o algoritmo.

O que faremos neste semestre (e em todos os outros) é partir de pontos iniciais com o objetivo de chegar a destinos. Em outras palavras, desejamos ser habilidosos em explicitar ideias implícitas.

Uma teoria matemática costuma possuir um teorema que usualmente é chamado de teorema fundamental. Por exemplo, em Cálculo, encontramos o “teorema fundamental de Cálculo”, que associa o problema da tangente ao problema da área; na teoria dos números, encontramos o “teorema fundamental da aritmética”, que caracteriza a sequência dos números naturais dizendo-nos que podemos olhar pra cada número natural como um produto de números primos. E para o raciocínio? Existe alguma ferramenta *fundamental*?

A opinião de Leslie Lamport

A ferramenta fundamental para o raciocínio científico é a escrita. Veja a ênfase com que Leslie Lamport expressa o fato.

To think, you have to write. To really think, you have to write. If you're thinking without writing, chances are you're fooling yourself.

— Leslie Lamport: Thinking Above the Code

Microsoft Faculty Summit, 2014

Fonte: https://youtu.be/-4Yp3j_jk8Q?t=176s

Já aconteceu com você de achar que uma certa ideia funciona até colocá-la em prática? Já viu algum programa de computador produzindo algum resultado absurdo?

When I started writing something down... Well, when you write something down, that really forces you to do it precisely. And, when I started doing it precisely, I realized it didn't work. It just didn't hold together.

— Oral History of Leslie Lamport Part 2

Fonte: <https://youtu.be/uK9yGNuGWKE?t=2155>

¹É certo que usamos mais do que a premissa do teorema pra obter a conclusão desejada. O mais exato é dizer que retiramos a conclusão de todo o conjunto de elementos que nos permitam executar as deduções desejadas. Por exemplo, a conclusão de que n^2 é par decorre não apenas da assunção de que n é par, mas também do teorema da álgebra que afirma que $(ab)^2 = a^2b^2$, entre outros.

A escrita nos encoraja a escrever o mapa em alta precisão — experimente. Alta precisão demanda clareza. Portanto, um antídoto contra a falta de clareza é a escrita. Detectar as dificuldades é *um* primeiro passo em direção à facilidade.

A opinião de Donald Knuth

Donaldo Knuth — autor da monumental obra chamada “The Art of Computer Programming” — revolucionou a indústria tipográfica com seu software chamado T_EX. (Falaremos sobre T_EX mais à frente.) Curiosamente, a revolução não é o que mais o tenha impressionado. Pra ele, o maior benefício da concepção de T_EX foi a ideia do que ele chamou de “programação letrada”, que é a ideia de redigir programas de computador como se obras literárias fossem.

In retrospect, I think the greatest benefit to me personally out of all this work in typography was the idea of literate programming. I decided to call it literate programming, which is a way of treating computer programs as literature. A computer program is something that human beings are supposed to read and you write it for people to read rather than for a computer to read.

— Donald Knuth: Literate programming (66/97)

Fonte: <https://youtu.be/bTkXg2LZIMQ?t=13s>

Eis o entusiasmo de Knuth em 1984:

[T]he impact of this new approach on my own style has been profound, and my excitement has continued unabated for more than two years. I enjoy the new methodology so much that it is hard for me to refrain from going back to every program that I’ve ever written and recasting it in ‘literate’ form. I find myself unable to resist working on programming tasks that I would ordinarily have assigned to student research assistants; and why? Because it seems to me that at last I’m able to write programs as they should be written. My programs are not only explained better than ever before; they also are better programs, because the new methodology encourages me to do a better job.

— Donald Knuth, “Literate programming.”

The Computer Journal 27.2, 1984: páginas 97–111.

O que Knuth percebe é que o trabalho sobe em qualidade porque a percepção, o entendimento dele sobre o programa aumenta consideravelmente em virtude da ênfase que a programação letrada coloca na escrita.

Leslie Lamport e Donald Knuth não são os únicos. Muitas outras observam a importância da escrita. Eis as palavras de Paul Graham num dos últimos que ensaio que escreveu.

Talking about your ideas with other people is a good way to develop them. But even after doing this, you’ll [...] still discover new things when you sit down to write. There is a kind of thinking *that can only be done by writing*.

— Paul Graham, “The Need to Read”, November 2022.

<http://paulgraham.com/read.html>

Nesse ensaio, Paul Graham argumenta que leitura nunca ficará obsoleta porque a leitura, além de nos ensinar sobre certa matéria, ensina-nos também a escrever — e a escrita é necessária pra se fazer certos tipos de descoberta.

Por que a escrita é tão fundamental?

Por exemplo, por que não é a dança a ferramenta fundamental? Não seria possível que o Tango fosse a melhor ferramenta pra se analisar um fenômeno intelectual qualquer?

Em vez de escrita, façamos a pergunta — por que o uso da *língua* é tão fundamental pro raciocínio? Se a língua foi desenvolvida primariamente pra comunicação, por que seria a comunicação tão importante para o raciocínio? É comum a concepção de que a língua tenha a comunicação entre pessoas como sua função primária, mas o que estamos fazendo é exatamente colocando essa concepção em cheque.

A opinião de Noam Chomsky

A tese que defendemos aqui é a de Noam Chomsky — a língua foi desenvolvida primariamente pro raciocínio e secundariamente pra comunicação. Por clareza, podemos talvez exagerar um pouco e afirmar algo ainda mais forte — onde há língua, há raciocínio e *vice-versa*, ou seja, língua e raciocínio são dois lados da mesma moeda. (É útil levar conjecturas a seus extremos pra ver com facilidade até onde elas resistem.)

“We use [language] for communication, of course, but it’s basically an instrument of thought. [This is] a long tradition — I mentioned Galileo, but many others just assumed that language is what was sometimes called “audible thought”: it’s the instrument of thought. Now, if you think about our own language in use, that’s... Almost all we do is language — construct thoughts in our mind. Sometimes we externalize them; sometimes we communicate, but that’s a kind of a side-property of language. Now we can pin that down much more precisely. When you look at the actual design and structure of language, what you find systematically is that when there are instances where there are conflicts between communicative efficiency and computational efficiency, computational efficiency always wins. There are very simple cases like simple structural ambiguities. So, if I say “flying planes can be dangerous” — that’s structurally ambiguous. [Does that] mean the act of flying or the planes that fly? Well, the point is that if you just let the rules run freely, computational efficiency [wins] [...], but they harm communication because [it] means [we] can’t understand what somebody’s saying. Do they mean to fly planes or planes that fly? And there are many things like this. Some of them pretty serious which impede communication, but mother nature didn’t care. When evolution moves along, some change takes place [and] you get the simplest solution; simplest most elegant solution. [It] may happen to be dysfunctional, [but that’s] not mother nature’s problem. [...] That’s the way — it seems — [that] language develop[ed].

— Noam Chomsky, “Mind Your Language. Thought, metaphor and imagination.” World Science Festival, September 30th 2021. (Entrevistado por Brian Greene.) Fonte: <https://youtu.be/6LXHtDUXkS0?t=863s>

As regras a que Chomsky se refere em “*let the rules run freely*” são as regras que governam como a língua funciona ou talvez como ele provavelmente diria — as regras oriundas da faculdade linguística do cérebro. Ele defende que seres humanos nascem com capacidade linguística em vez de adquiri-la ao longo dos anos. O exemplo dele argumenta que a evolução da língua não incluiu uma estrutura gramatical que pudesse resolver a ambiguidade na frase. A afirmação é de que a língua *poderia* acomodar uma forma de fazer a distinção naquela frase, evitando a ambiguidade, mas a evolução simplesmente se mostrou desinteressada disso — *mother nature didn’t care* —, o que sugere que não é precisão de comunicação que mais parece ter importado pra ela. É possível que a eficiência computacional da interpretação tome a precedência.

Ferramentas de escrita

Dada a importância da escrita, é sensato que caminhemos na direção de torná-la confortável e fácil, o que significa que precisamos investir em ferramentas. Ferramentas para escrita são leitura, gramática, um editor de textos, um software de composição tipográfica como L^AT_EX de Leslie Lamport, que é o estado da arte em composição tipográfica — junto com T_EX. L^AT_EX é software derivado de T_EX — obra de Donald Knuth —, que é a ferramenta original. Quando falamos de L^AT_EX, estamos falando necessariamente de T_EX.

Existe um serviço na Internet chamado [overleaf.com](https://www.overleaf.com). O serviço lhe oferece toda a parafernália que produz um documento PDF com o seu texto. Usar um serviço assim pode não ser o ideal, mas é um jeito fácil de conhecer L^AT_EX e *começar*.

Você vai precisar compreender a linguagem L^AT_EX, o que pode ser feito inicialmente com exemplos. O tutorial do Overleaf em

https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes
serve como um começo.

O Overleaf também lhe permite compartilhar o documento pra que você possa mostrar a um colega ou trabalhar em conjunto.

Como se pronuncia T_EX?

Palavras como “tecnologia” originam-se de um radical grego que começa com as letras gregas τ, ε, χ, isto é, *tau*, *epsilon* e *ki*. A pronúncia, portanto, é *ték* e não *téx*. Essa mesma palavra grega também representa a palavra “arte”, além de “tecnologia”. Dado que T_EX foi projetado pra produzir documentos técnicos de altíssima qualidade, o que envolve tecnologia e principalmente arte e *design*, o nome T_EX foi escolhido. (Veja “The TeXbook”, Donald Knuth, 1970, Addison-Wesley, capítulo 1, página 1. ISBN: 0-201-13447-0.)