Bad Foundations and Manipulable Objects

Eduardo Ochs*

February 15, 2026

Abstract

Imagine a student—let's call him 'E', and make him a "he"—that is enrolled in Calculus 2, and who believes that to pass in maths courses he only needs to memorize methods and apply them quickly and without errors. Let's imagine that 'E' is an 'E'xtreme case of "bad foundations" and that he knows how to solve $x + 2 = 5$ by doing $x = 5 - 2 = 3$, but he doesn't know how to substitute the x in $x + 2 = 5$ by 3, and the only way that he knows of "testing the solution" is to apply the same method again and check that he got the same result.

When we are teaching Calculus to classes that have many students that are extreme cases of bad foundations we need new strategies and tools; for example, we can't pretend that "taking a particular case" is an obvious operation anymore — instead we need ways to make these operations easy to visualize. This article shows a way to do that using Maxima.

Para Walter Machado Pinheiro,
que não leu e não vai ler
documento nenhum, e se ler
não vai entender

1 Introduction

Let me start by some stories. I teach Calculus 2 and 3 in a low-prestige campus of a high-prestige federal university in Brazil, and all the characters that I am going to mention are based on real students that I had in Calculus 2 – that I will sometimes abbreviate as "C2".

*eduardoochs@gmail.com

‘B’en is a student that has ‘B’ad foundations and ‘E’ric is a student who has ‘E’extremely bad foundations. They are taking Calculus 2 with me, and in the second test, that has some questions abouts ODEs, they commit the same error: they find $y = \sqrt{25 - x^2} + 3$ as the solution of a certain ODE, and in the item in which they have to draw its graph, they simplify $y = \sqrt{25 - x^2} + 3$ by doing this:

$$\begin{aligned} y &= \sqrt{25 - x^2} + 3 \\ &= \sqrt{5^2 - x^2} + 3 \\ &= 5 - x + 3 \\ &= 8 - x \end{aligned}$$

I tell them that in the third ‘=’ sign they used a rule that is invalid. They panic, and they say: “oh no, sorry, we got distracted – that won’t happen again!!!”. Then I ask them to tell me what rule they used there, and I discover that they don’t have any idea of what I mean... for them Maths is about memorizing methods and applying them; “understanding” is only for geniuses ([SchoenfeldWhenGood, p.7]), and all their knowledge about Maths is *procedural*, not *conceptual* ([EngelbrechtBergsten]). I will explain more about bad foundations on section 7.

‘Q’uentin is a student who believes that the goal of learning Maths is to get answers ‘Q’uickly ([Boaler, p.33]). In a test question that asked to calculate $\int \sqrt{1 - x^2}\, dx$ his answer was just this:

$$\frac{1}{2}\left(\arcsin(x) + x\sqrt{1 - x^2}\right)$$

I tried to explain to him that the “right” answer to that question would a calculation with many steps, in which each ‘=’ would be “easy to understand, to justify, and to verify”, but none of my arguments made any sense to him.

‘F’abian is a student who is more ‘F’riendly and talkative than most. He tells me that he had very good grades in Calculus 1 and that he was very good at calculating derivatives, and I decide that he will be my reference for understanding what the students know about ‘F’unctions. I ask him to show me how he would calculate $\frac{d}{dx}\sin(\cos(\tan(42x))$; he tries but he gets stuck, and my hints don’t help him. It turns out that he learned the Chain Rule by a method similar to the one in [Stewart8, p.154], in which he had to *remember* what are the “outer function”, the “inner function”, and their derivatives; he didn’t have a way to write these four functions down on paper.

The method the Fabian learned is optimized for speed, and in complex cases, like $\frac{d}{dx}\sin(\cos(\tan(42x))$, it needs a lot of working memory to be carried out... and Fabian had a memory overflow.

One day I ask all the students to calculate $\frac{d}{dx} f(\sin(x^4) + \ln x)$. They all drop the ‘$f$’, and the best students get the result $\cos(x^4) \cdot 4x^3 + \frac{1}{x}$. I ask them how they

got there, and they say that the 'f' was "just notation" and meant "function", so they calculated "the derivative of the function $\sin(x^4) + \ln x$".

I discuss that with Fabian, and I draw this diagram:

$$\begin{array}{ccc} \begin{pmatrix} \frac{d}{dx} f(g(x)) = \\ f'(g(x))g'(x) \end{pmatrix} & \to & \begin{pmatrix} \frac{d}{dx} f(42x) = \\ f'(42x) \cdot 42 \end{pmatrix} \\ \downarrow & \searrow & \downarrow \\ \begin{pmatrix} \frac{d}{dx} \sin(g(x)) = \\ \cos(g(x))g'(x) \end{pmatrix} & \to & \begin{pmatrix} \frac{d}{dx} \sin(42x) = \\ \cos(42x) \cdot 42 \end{pmatrix} \end{array}$$

We name its nodes as $\begin{smallmatrix} (1) \to (2) \\ \downarrow \searrow \downarrow \\ (3) \to (4) \end{smallmatrix}$, and Fabian tells me that (4) makes all sense to him and he recognizes (1) as being the formula of the Chain Rule, but he doesn't have any idea of what are (2) and (3).

This reveals that the students know far less than they should, but is not totally surprising – we know that books written in the 13th, 14th and 15th centuries didn't treat unknowns like x as entities with the same algebraic properties as numbers ([FilloyRojano], [PuigRojano, p.216]), and the nodes (1), (2), and (3) in the diagram above talk about unknown *functions*, which are much worse conceptually than unknown *numbers*.

What did these students learn in Calculus 1? My colleagues share practically nothing of their didactic strategies and teaching materials, but they brag that in their courses they present proofs, and that the students wouldn't see any proofs if we let the engineers teach Calculus to them – see [AhmadApplebyEdwards]...

Here's a hypothesis:

> When my students took C1 they saw a version of the Chain Rule "for analysts", in which (1) is a part of the statement of a theorem, and in the full statement f and g are quantified – either as "for any $f, g : \mathbb{R} \to \mathbb{R}$ differentiable everywhere", or as something like that but with different domains and different requirements of differentiability – but the students retained very little from what they saw because they are in a "procedural" stage, and they didn't know how to operate on those quantifiers.

Let me take a paragraph from [CarlsonGulick, p.62]:

> At the same time certain participants emphasized that the fundamental concepts in calculus must remain in the course, along with at least a certain amount of drill work to develop manipulative skill. We must minimize the tedium of working problems where no thought at all is

> necessary. A basic question is the following: What would be an appropriate blend between geometrically-motivated concepts, definitions and theorems, relevant applications, and rigorous proofs?

In this article I will discuss an unusual answer for the question "What would be an appropriate blend...?" above. My proposal is that *when we have lots of students like the ones that I described above* it is a good idea to present expressions and trees as some of our most basic objects, and present this substitution operation

$$\begin{pmatrix} & \frac{d}{dx} f\left(g\left(x\right)\right) \\ = & f'\left(g\left(x\right)\right)\, g'\left(x\right) \end{pmatrix} \begin{bmatrix} g\left(x\right) := 42\, x \\ g'\left(x\right) := 42 \end{bmatrix} = \begin{pmatrix} & \frac{d}{dx} f\left(42\, x\right) \\ = & f'\left(42\, x\right)\, 42 \end{pmatrix}$$

very early in the course, before theorems and quantifiers.

The figure below has a 'L'eft part with particles 'if', 'then', and 'and', a 'M'iddle part with equalities, and a 'R'ight part with justifications:

$$\begin{array}{rrcll} \text{If} & f(g(x)) & \overset{(1)}{=} & x & \\ \text{then} & \frac{d}{dx} f(g(x)) & \overset{(2)}{=} & \frac{d}{dx} x & \text{by (1)} \\ \text{and} & & \overset{(3)}{=} & 1 & \\ \text{and} & \frac{d}{dx} f(g(x)) & \overset{(4)}{=} & f'(g(x))g'(x) & \text{the chain rule} \\ \text{and} & f'(g(x))g'(x) & \overset{(5)}{=} & 1 & \text{by (4), (2), (3)} \\ \text{and} & g'(x) & \overset{(6)}{=} & 1/f'(g(x)) & \text{by (5)} \end{array}$$

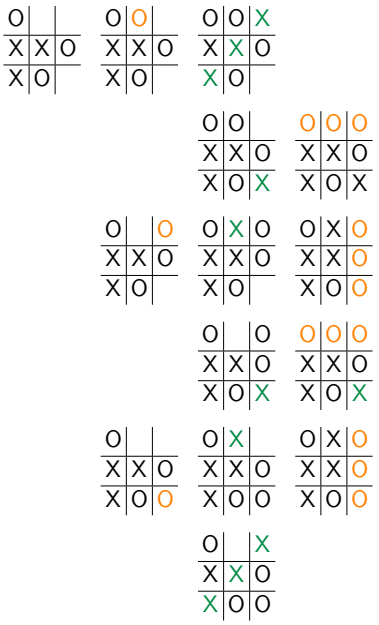
One day I write the part 'M' on the blackboard and I ask the students a) if that is how they learned that $\frac{d}{dx} \ln x = \frac{1}{x}$ in Calculus 1, and b) if all the steps are "obvious" to them. 'ST'eve says that that's not obvious at all, and asks me if I can make that clearer. I show that we can write the columns 'L' and 'R', and I ask him what is the first step that doesn't make sense to him. He answers that it's '$\overset{(5)}{=}$', and I write this expanded version:

$$\begin{array}{rrcll} \text{If} & f(g(x)) & \overset{(1)}{=} & x & \\ \text{then} & \frac{d}{dx} f(g(x)) & \overset{(2)}{=} & \frac{d}{dx} x & \text{by (1)} \\ \text{and} & & \overset{(3)}{=} & 1 & \\ \text{and} & \frac{d}{dx} f(g(x)) & \overset{(4)}{=} & f'(g(x))g'(x) & \text{the chain rule} \\ \text{and} & f'(g(x))g'(x) & \overset{(5)}{=} & \frac{d}{dx} f(g(x)) & \text{by (4)} \\ \text{and} & & \overset{(6)}{=} & \frac{d}{dx} x & \text{by (2)} \\ \text{and} & & \overset{(7)}{=} & 1 & \\ \text{and} & f'(g(x))g'(x) & \overset{(8)}{=} & 1 & \text{by (5), (6), (7)} \\ \text{and} & g'(x) & \overset{(9)}{=} & 1/f'(g(x)) & \text{by (5)} \end{array}$$

We discover that 'ST'eve did not know that the equality is 'S'ymmetric and 'T'ransitive. He only knew three meanings for the '=' sign: "compute", "simplify", and "solve this" – see for example [FischbeinTacit, p.10] and [ThomasRethinking, p.179].

2 Expandable objects

Game trees. The tree below is *essentially the same* as the one in [Hut16, Section 11.8],



but the one in Hutton's book has the root node at the top, has lines from each parent node to its child nodes, and is in black and white; our tree draws child nodes to the right, like this, $\begin{bmatrix} 1 & 1.1 & 1.1.1 \\ & & 1.1.2 \\ & 1.2 & 1.2.1 \end{bmatrix}$, and uses colors to indicate the most recent move and the winning lines.

Modern students are able to see immediately that the tree above is something that could be drawn by a computer interface, and it is easy for them to imagine that the interface has buttons that turn colors on and off, that turns arrows on and off, that switches between that layout and Hutton's, and that displays or hides other parts of the tree. *The full tree would be so big that we don't want to see all of it* – lots of objects in Mathematics are like that, and I will call them *expandable objects*.

Set comprehensions. Set comprehensions are also expandable objects. We can calculate the results of simple set comprehensions, like $\{\, 10a \mid a \in \{2, 3, 4\} \,\}$, by "reading them aloud" in the right way, i.e., by translating the mathematical

notation into English; but if we need to explain our mental process, or to handle more complex cases, we need other techniques – like 1) annotating the parts of the comprehension as "generators", "filters", and "resulting expression", 2) converting the two standard notations for comprehensions, $\{\,10a \mid a \in \{2,3,4\}\,\}$ and $\{\,a \in \{2,3,4\} \mid a^2 < 5\,\}$, to a unified notation in which the resulting expression always appears at the right, 3) converting them to programs, and 4) using trees to calculate the result. The figure below shows (1), (2), and (3):

$$\begin{aligned}
\{\,a \in \{2,3,4\} \mid a^2 < 10\,\} &= \{\,\underbrace{a \in \{2,3,4\}}_{\text{gen}} \mid \underbrace{a^2 < 10}_{\text{filt}}\,\} \\
&= \{\,\underbrace{a \in \{2,3,4\}}_{\text{gen}}, \underbrace{a^2 < 10}_{\text{filt}}\,;\, \underbrace{a}_{\text{expr}}\,\} \\
&= \{4,9\}
\end{aligned}$$

```
for a=2,4 do
  if a^2<10 then
    print(a)
  end
end
```

$$\begin{aligned}
\{\,10a \mid a \in \{2,3,4\}\,\} &= \{\,\underbrace{10a}_{\text{expr}} \mid \underbrace{a \in \{2,3,4\}}_{\text{gen}}\,\} \\
&= \{\,\underbrace{a \in \{2,3,4\}}_{\text{gen}}\,;\, \underbrace{10a}_{\text{expr}}\,\} \\
&= \{20,30,40\}
\end{aligned}$$

```
for a=2,4 do
  print(10*a)
end
```

Here is an example of (4):

$$\{\,\underbrace{x \in \{1,\ldots,5\}}_{\text{gen}}, \underbrace{y \in \{x,\ldots,6-x\}}_{\text{gen}}\,;\, \underbrace{(x,y)}_{\text{expr}}\,\} = \quad$$

```
for x=1,6 do
  for y=6,6-x do
    print(x,y)
  end
end
```

x	y	(x,y)
1	1	$(1,1)$
	2	$(1,2)$
	3	$(1,3)$
	4	$(1,4)$
	5	$(1,5)$
2	2	$(2,2)$
	3	$(2,3)$
	4	$(2,4)$
3	3	$(3,3)$
4	|	
5	|	

x	$6-x$	$\{x,\ldots,6-x\}$	y	(x,y)
1	5	$\{1,2,3,4,5\}$	1	$(1,1)$
			2	$(1,2)$
			3	$(1,3)$
			4	$(1,4)$
			5	$(1,5)$
2	4	$\{2,3,4\}$	2	$(2,2)$
			3	$(2,3)$
			4	$(2,4)$
3	3	$\{3\}$	3	$(3,3)$
4	2	$\{\}$	|	
5	1	$\{\}$	|	

Note that the two tree/tables above are "simple" in different ways: in the one in the center it is obvious how we chose its columns, but the one at the right is simpler to follow. See [OchsNSC2026] for more details and some class activities using set comprehensions.

Proofs with justifications. For example:

$$
\begin{array}{lll}
((6x^3)(7x^4))' & \overset{(1)}{=} & \frac{d}{dx}((6x^3)(7x^4)) \\
& \overset{(2)}{=} & (6x^3)\frac{d}{dx}(7x^4) + (7x^4)\frac{d}{dx}(6x^3) \\
& \overset{(3)}{=} & (6x^3)\frac{d}{dx}(7x^4) + (7x^4)\cdot 6\cdot \frac{d}{dx}x^3 \\
& \overset{(4)}{=} & (6x^3)\frac{d}{dx}(7x^4) + (7x^4)\cdot 6\cdot 3x^2 \quad \text{por} \quad \underbrace{\text{[RPot]}}_{\left(\begin{array}{l}\frac{d}{dx}x^n \\ = \; n\,x^{n-1}\end{array}\right)} \quad [n := 3] \\
& \overset{(5)}{=} & (6x^3)\frac{d}{dx}(7x^4) + (7x^4)(18x^2) \qquad \underbrace{\qquad\qquad\qquad}_{\left(\begin{array}{l}\frac{d}{dx}x^3 \\ = \; 3\,x^{3-1}\end{array}\right)} \\
& \overset{(6)}{=} & (6x^3)\cdot 7\frac{d}{dx}x^4 + (7x^4)(18x^2) \\
& \overset{(7)}{=} & (6x^3)\cdot 7\cdot 4x^3 + (7x^4)(18x^2) \\
& \overset{(8)}{=} & (6x^3)(28x^3) + (7x^4)(18x^2) \\
& \overset{(9)}{=} & (6x^3)(28x^3) + 126x^6 \\
& \overset{(10)}{=} & 168x^6 + 126x^6 \\
& \overset{(11)}{=} & 294x^6
\end{array}
$$

Imagine that this is a proof of $((6x^3)(7x^4))' = 294x^6$ that has a computer interface. We could toggle off the display of the intermediate steps, and we would get just $((6x^3)(7x^4))' = 294x^6$, but we told the computer a) to display it as a series of steps that are easy to justify, b) to number its equalities, c) to highlight what changed between the left hand side and the right hand side of '$\overset{(4)}{=}$', d) to show the name of the rule that was used there e) to show that rule "as a formula", f) to show what particular case we used there ($n = 3$), and g) to show what that formula becomes in that particular case.

3 Free and dependent variables

Some people find dependent variables confusing. I was an extreme case of that; I am not anymore, but I do believe that the students who find dependent variables confusing deserve support – like translations.

This is the first example of the chain rule in [ApexCalculus4, p.101], slightly reordered:

$$
\begin{array}{lrcl}
\text{If:} & y & = & f(g(x)) \\
\text{then:} & y' & = & f'(g(x))g'(x). \\
\text{If:} & f(x) & = & x^2 \\
\text{and:} & g(x) & = & 1-x \\
\text{then:} & f'(x) & = & 2x, \\
 & g'(x) & = & -x, \\
 & y & = & (1-x)^2, \\
 & y' & = & 2(1-x)\cdot(-1).
\end{array}
$$

We can use set comprehensions to make this more concrete to students who don't understand equations well because they have problems with free variables.

The last two lines above say that the derivative of the function with this graph

$$\{\,(x,y)\in\mathbb{R}^2 \mid y=(1-x)^2\,\}$$

is the function with this graph,

$$\{\,(x,y')\in\mathbb{R}^2 \mid y'=2(1-x)\cdot(-1)\,\}$$

and we can change the y' to y.

In the simplest examples, in which we can ignore the conditions on smoothness and on domains, the chain rule is just $\frac{d}{dx}f(g(x))=f'(g(x))g'(x)$, and the derivative of the function with this graph

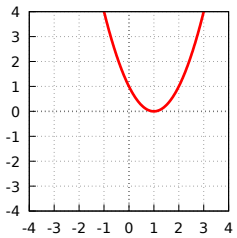
$$\{\,(x,y)\in\mathbb{R}^2 \mid y=f(g(x))\,\}$$

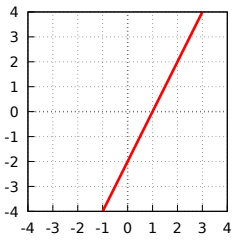
is the function with this graph:

$$\begin{array}{rl} & \{\,(x,y)\in\mathbb{R}^2 \mid y=\frac{d}{dx}f(g(x))\,\} \\ = & \{\,(x,y)\in\mathbb{R}^2 \mid y=f'(g(x))g'(x)\,\}\,. \end{array}$$

Here is another way to write, and draw, that. If $f(x)=x^2$ and $g(x)=1-x$ then $f'(x)=2x$ and $g'(x)=-1$, and:

$$\frac{d}{dx}\quad \underbrace{f(g(x))}_{(1-x)^2} \quad = \quad \underbrace{f'(g(x))g'(x)}_{2(1-x)\cdot(-1)}$$





4 Maxima

Maxima is a computer algebra system that is very extensible, and in which we can define new operators and set lots of properties for any operators – for example we can set how they are LaTeXed, as in the examples in the first column

below. Note that '`matrix`' is standard, '`bmatrix`' uses '`\begin{bmatrix}`' and '`\end{bmatrix}`', that use square brackets, '`barematrix`' doesn't draw any outer brackets, and '`verbatimmatrix`' uses a kind of verbatim box.

The internal representation of `f(a,b)` is `((%F SIMP) %A %B)`, that is a tree in Lisp, in a format that is hard to visualize. The result of `lisptreem(expr)` is the internal representation of the result of `expr`, drawn as a 2D tree, as a '`verbatimmatrix`'.

The result of `lisptree2(expr)` includes the result of `expr` and `lisptreem(expr)`.

The last input/output pair below, in `(%i11)` and `(%o11)`, shows how we can draw a series of equalities using a matrix.

```
(%i1)     matrix([10,20],[30,40]);
(%o1)
```

$$\begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix}$$

```
(%i2)    bmatrix([10,20],[30,40]);
(%o2)
```

$$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

```
(%i3) barematrix([10,20],[30,40]);
(%o3)
```

$$\begin{matrix} 10 & 20 \\ 30 & 40 \end{matrix}$$

```
(%i4) verbatimmatrix(["a"], ["bcd"]);
(%o4)
 a
bcd
(%i5) lisptreem(f(a,b));
(%o5)
 f__.
 |  |
 a  b
(%i6) lisptree2(f(a,b));
(%o6)
```

$$f\,(a,b) = \begin{matrix} \texttt{f\_\_.} \\ \texttt{\textbar\ \ \textbar} \\ \texttt{a\ \ b} \end{matrix}$$

```
(%i7) lisptree2(    matrix([10,20],[30,40]));
(%o7)
```

$$\begin{pmatrix} 10 & 20 \\ 30 & 40 \end{pmatrix} = \begin{matrix} \texttt{matrix\_\_.} \\ \texttt{\textbar\ \ \ \ \ \ \textbar} \\ \texttt{\textbar\_\_\_.\ \ \ .\_\_\_\textbar} \\ \texttt{\textbar\ \ \ \textbar\ \ \ \textbar} \\ \texttt{10\ \ 20\ \ 30\ \ 40} \end{matrix}$$

```
(%i8) lisptree2(    bmatrix([10,20],[30,40]));
(%o8)
```

$$\begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{matrix} \texttt{bmatrix\_\_.} \\ \texttt{\textbar\ \ \ \ \ \ \textbar} \\ \texttt{\textbar\_\_\_.\ \ \ .\_\_\_\textbar} \\ \texttt{\textbar\ \ \ \textbar\ \ \ \textbar} \\ \texttt{10\ \ 20\ \ 30\ \ 40} \end{matrix}$$

```
(%i9) lisptree2(barematrix([10,20],[30,40]));
(%o9)
```

$$\begin{matrix} 10 & 20 \\ 30 & 40 \end{matrix} = \begin{matrix} \texttt{barematrix\_\_.} \\ \texttt{\textbar\ \ \ \ \ \ \ \ \textbar} \\ \texttt{\textbar\_\_\_.\ \ \ \ .\_\_\_\textbar} \\ \texttt{\textbar\ \ \ \textbar\ \ \ \ \textbar\ \ \ \textbar} \\ \texttt{10\ \ 20\ \ \ \ 30\ \ 40} \end{matrix}$$

```
(%i10) lisptree2(verbatimmatrix(["a"], ["bcd"]));
(%o10)
```

$$\begin{matrix} \texttt{a} \\ \texttt{bcd} \end{matrix} = \begin{matrix} \texttt{verbatimmatrix\_\_.} \\ \texttt{\textbar\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \textbar} \\ \texttt{\textbar\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \textbar} \\ \texttt{\textbar\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \textbar} \\ \texttt{a\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ bcd} \end{matrix}$$

```
(%i11) lisptree2(matrix([a,"=",b],["","=",c]));
(%o11)
```

$$\begin{pmatrix} a & = & b \\ & = & c \end{pmatrix} = \begin{matrix} \texttt{matrix\_\_\_.} \\ \texttt{\textbar\ \ \ \ \ \ \ \ \ \ \textbar} \\ \texttt{\textbar\_\_.\_\_.\_\_\textbar\ \ \textbar\_\_.\_\_.\_\_\textbar} \\ \texttt{\textbar\ \ \textbar\ \ \textbar\ \ \textbar\ \ \ \ \textbar\ \ \textbar\ \ \textbar} \\ \texttt{a\ \ =\ \ b\ \ \ \ \ \ =\ \ c} \end{matrix}$$

Remember that in our Section 1 'ST'eve knew very few meanings for the equality sign. We can use the code below to show him more meanings – that are written in the same way in mathematical notation but differently in Maxima:

```
(%i1) P : [1,2]$
(%i2) Q : [3,-4]$
(%i3) f(x) := a*x + b$
(%i4) eq1 : f(P[1]) = P[2];
(%o4)                          b + a = 2

(%i5) eq2 : f(Q[1]) = Q[2];
(%o5)                         b + 3 a = -4

(%i6) ab  : solve([eq1,eq2], [a,b]);
(%o6)                      [[a = -3, b = 5]]

(%i7) subst(ab, f(x));
(%o7)                              5 - 3 x

(%i8) define(g(x), %);
(%o8)                           g(x) := 5 - 3 x

(%i9) g(1);
(%o9)                                  2

(%i10) g(3);
(%o10)                                -4
```

Typing '`2+3;`' and then `RET` means "calculate this and show the result"; '`$`' means "calculate this and don't show the result"; '`:`' is assignment; '`:=`' defines a function; '`eq1 : f(P[1]) = P[2]`' stores an equality, as an expression, in `eq1`; '`solve([eq1,eq2], [a,b])`' treats the equalities in `eq1` as two equations to be solved and returns the solution as two other equalities.

4.1 Substitution

This, that is an expanded version of the `(%i7)` above,

```
(%i11) subst([a=-3,b=5], a*x+b);
(%o11)                             5 - 3 x
```

is similar to the operation that we were using to obtain particular cases of equations, but here we are "obtaing a particular case" of an expression, '`a*x+b`', that is not an equation. In the notation that we used in section 2 this would be:

$$\underbrace{(ax+b)\begin{bmatrix} a := -3 \\ b := 5 \end{bmatrix}}_{-3x+6}$$

or:

$$(ax+b)\begin{bmatrix} a := -3 \\ b := 5 \end{bmatrix} = -3x+6$$

Juxtaposition. In some cases, like $2(3+4)$, the juxtaposion means an operation that was elided, but whose name we know, and we can write it explicitly: $2 \cdot (3+4)$. In other cases the operation doesn't have a standard notation, and we have to improvise. Let's use the symbol '$\boxed{\text{ap}}$' for application, and '$\boxed{\text{s}}$' for substitute:

$$\begin{array}{rcl} 2(y+z) & \Rightarrow & 2 \cdot (y+z) \\ f(y+z) & \Rightarrow & f \ \boxed{\text{ap}} \ (y+z) \\ (a+b)[a:=42] & \Rightarrow & (a+b) \ \boxed{\text{s}} \ [a:=42] \end{array}$$

The operation '$\boxed{\text{s}}$' is common in Logic and Computer Science – see for example [Harper, p.6] and [HindleySeldin2008, p.7] – but it is mostly treated as "obvious" in other places; some exceptions are [Kindt, p.146, p.149, p.167] and [FreudenthalDPh, p.482 and p.488].

Process and object. We can see the substitution as a process, drawn as an arrow below,

$$a+b \xrightarrow[a \text{ by } 42]{\text{substitute}} 42+b$$

that was reified, as in [Sfard, p.44], and written as a suffix:

$$(a+b) \ \underbrace{[\ \underbrace{a:=42}_{a \text{ by } 42}\ ]}_{\text{substitute } a \text{ by } 42} = 42+b$$

Infix. In Maxima we can define new operators, and we can make them infix, prefix, postfix, or n-ary. So we can do this, and implement '$\boxed{\text{s}}$' as '`_s_`':

```
(%i1) "_s_"(expr,substs) := subst(substs,expr)$
(%i2) infix("_s_")$
(%i3) (a+b) _s_ [a=42];
(%o3)
```

$$b+42$$

Simplification and lazyness. The result in '`(%o3)`' above was $b+42$, not $42+b$, because Maxima (almost) always simplifies expressions, and its algorithm for simplifying sums can reorder and collapse terms, and it can transform, for example, `2+3+4x+5x` into `9x+5`.

Some functions in Maxima have versions that are less "active", in the sense that they perform fewer simplifications. For example:

```
(%i1)  diff(x^2, x);
(%o1)
```

$$2\,x$$

```
(%i2) 'diff(x^2, x);
(%o2)
```

$$\frac{d}{d\,x}\,x^2$$

In the terminology of Maxima `diff` is a "verb" and `'diff` is a "noun".

Another way to obtain operations that are simplified less, or that are not simplified at all, is to create new operations and copy most of the properties from the original versions to the copies, but omit the properties that say how they are simplified. In the example below '`+.`' is a variant of '`+`' that was created in that way:

```
(%i1) a+b=b+a;
(%o1)
```

$$b+a=b+a$$

```
(%i2) a+.b=b+.a;
(%o2)
```

$$a+b=b+a$$

```
(%i3) (a+.b=b+.a) _s_ [a=2,b=3];
(%o3)
```

$$2+3=3+2$$

I call these less active operations "lazy operations".

Prettifying. The mathematical notation for parallel substitution uses ':='s instead of '='s, puts each ':=' on a different line, and uses square brackets. We can implement that with an operation '`V`' that pretty-prints the kinds of substitutions that `subst` expects, like `[a=2,b=3]`, as '`bmatrix`'es. The example below also shows '`_ss_`', '`_sss_`', and '`_ssu_`', that use '`V`' to print substitutions in nice ways:

```
(%i1) o : a+.b=b+.a$
(%i2) S : [a=2,b=3]$
(%i3) V(S);
(%o3)
```

$$\begin{bmatrix} a := 2 \\ b := 3 \end{bmatrix}$$

```
(%i4) lisptreem(V(S));
(%o4)
```

```
bmatrix_____.
|           |
|           |
|           |
becomes__.  becomes__.
|        |  |        |
a        2  b        3
```

```
(%i5) o _ss_ S;
(%o5)
```

$$(a+b=b+a)\begin{bmatrix} a := 2 \\ b := 3 \end{bmatrix}$$

```
(%i6) o _sss_ S;
(%o6)
```

$$(a+b=b+a)\begin{bmatrix} a := 2 \\ b := 3 \end{bmatrix} = (2+3=3+2)$$

```
(%i7) o _ssu_ S;
(%o7)
```

$$\underbrace{(a+b=b+a)\begin{bmatrix} a := 2 \\ b := 3 \end{bmatrix}}_{2+3=3+2}$$

Substituting functions. Substitutions like $\left[\begin{smallmatrix} f(x):=x^2 \\ f'(x):=2x \end{smallmatrix}\right]$ are used a lot in Calculus, but they are tricky to implement – let's see why. We want to have this:

$$f(g(t))\ \begin{bmatrix} f(x) := g(g(x)) \\ g(x) := f(f(x)) \end{bmatrix}\ =\ g(g(f(f(t))))$$

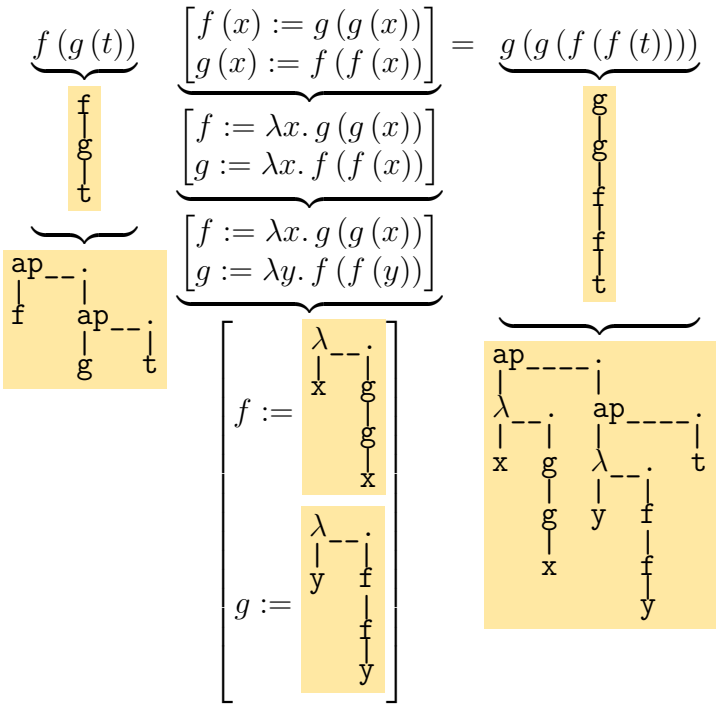
but look at this code:

```
(%i1) o  : f(g(x))$
(%i2) S1 : [f(x)=g(g(x)),
            g(x)=f(f(x))]$
(%i3) S2 : [f=lambda([x],g(g(x))),
            g=lambda([x],f(f(x)))]$
(%i4)  subst(S1,o);          /* wrong: */
(%o4)
                f(f(f(x)))

(%i5) psubst(S1,o);          /* wrong: */
(%o5)
                f(f(f(x)))

(%i6) psubst(S2,o);             /* ok: */
(%o6)
              g(g(f(f(x))))

(%i7) o _sss_ S2;               /* ok: */
(%o7)
f(g(x)) [f := lambda([x],g(g(x)))] = g(g(f(f(x))))
        [g := lambda([x],f(f(x)))]
(%i8) o _sss_ S1;               /* ok: */
(%o8)
 f(g(x)) [f(x) := g(g(x))] = g(g(f(f(x))))
         [g(x) := f(f(x))]
```

both '`subst`' and '`psubst`', that are Maxima built-ins, don't do what we expect when we use '`S1`', but they work well with '`S2`', in which we translated the functions to lambdas.

A good solution is to make '`_s_`' translate functions to lambdas when needed. Note that the lines `(%i7)` and `(%i8)` above both yield $g(g(f(f(x))))$ – that's because '`_sss_`' calls this more complex '`_s_`', that calls a function called '`__s_fstolambdas`' to perform this translation.

Here's a way to visualize how this substitution works:



The path is $\left(\begin{smallmatrix}\bullet & \bullet\\ \downarrow & \uparrow\\ \bullet & \bullet\end{smallmatrix}\right)$, and in the last part, the '↑', Maxima handles the β-reductions in the standard way, in which Church-Rosser theorem guarantees that the order of the reductions doesn't matter. See [HindleySeldin2008, pages 7, 11, and 14] for the details, and note that the clause (g) in p.7 "*chang[es] bound variables to avoid clashes*".

A few years ago I thought that the right thing to do would be to implement this '[:=]' in Lean "to show how natural it is". That turned to be much harder than I expected, and to make things worse Lean doesn't have a REPL and doesn't have a built-in LaTeX generator... while in Maxima '`__s_fstolambdas`' is just 9 lines and the rest of '`_s_`' is just 6 lines. I wanted to convince people that this '[:=]' is a natural operation, *even though I couldn't find it explained in details in the literature*, and showing how it can be implemented in 15 lines in Maxima seemed to be a good way.

I will explain more reasons for not using Lean in the next section.

5 Why not Lean?

The "right way" to handle "procedural students" that don't know anything about Logic would be to teach Logic to them. There is a very good example of how to do that in [Yalep] – note in particular the "Pedagogical progression" in its Appendix B – and the authors have a comparison with other computational approaches in [YalepSurvey]... *but we don't have time for the right way.*

Let me quote a few sentences from [BottoniTFM, p.30]:

> However, Lean is not currently suitable for high school or undergraduate university teaching. High school mathematics curricula in Switzerland are already crowded, and introducing abstract topics like first-order logic or natural induction would push the limits.

The syllabus of Calculus 2 in my campus is huge, and it was hard to cover it completely even when the students came in much better prepared. I needed an approach in which a) the non-textbook material would be just a small part of each class, b) the students with little experience with computers would be able to follow almost everything, and c) the programs would be easy to install at home, and easy to use both at home and at the computer lab. So I ended up using Maxima with the interface described in [OchsEmacsConf2024]; my presentation at the EBL 2025 ([OchsEBL2025]) was about how people who installed Maxima in that way could install Lean easily and then follow a basic tutorial on Lean, but in Calculus 2 I show Lean very briefly, just as a curiosity.

6 Holes

Maxima comes with a function called '`dpath`', that expects an object and a "path", and 'd'isplays a box around subobject obtained by following that path. In the examples below the path is '`2,1`', that means "second child node, then first

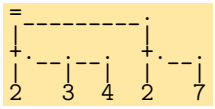
child node":

```
(%i1) o : 2 +. 3 +. 4 = 2 +. 7;
(%o1)
```

$$2+3+4=2+7$$

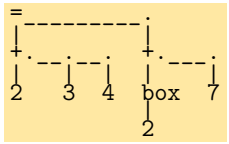
```
(%i2) lisptree2(o);
(%o2)
```

$2+3+4=2+7=$



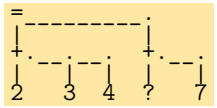

```
(%i3) lisptree2(dpart(o,2,1));
(%o3)
```

$2+3+4=\boxed{2}+7=$




```
(%i4) lisptree2(substpart("?",o,2,1));
(%o4)
```

$2+3+4=\ ?\ +7=$



In the last line we used '`substpart("?", obj, <path>)`', that does something similar to '`dpart`' but replaces the object there by a '`"?"`' instead of by a box.

Note that in the example above we started with $2+3+4=2+7$, that is *true*, and we obtained $2+3+4=?+7$, that is a *problem*, or an *exercise*. I will refer to the '?' as a *hole*; see [FreudenthalDPh, p.465].

Many exercises in Calculus books can be rewritten as true expressions with holes. For example, the exercise 11 in [ApexCalculus4, p.109] is "compute the derivative $\frac{d}{dx}(\ln x + x^2)^3$ using the chain rule", and we can rewrite it as a true

expression with holes by doing this:

```
(%i5) S : [f (x)=x^3,
           fp(x)=3*x^2,
           g (x)=log(x)+x^2,
           gp(x)=1/x+2*x]$
(%i6) o : RCV _ssu_ S;
(%o6)
```

$$\underbrace{\begin{pmatrix} \frac{d}{dx} f(g(x)) \\ = \quad f'(g(x))\, g'(x) \end{pmatrix} \begin{bmatrix} f(x) := x^3 \\ f'(x) := 3x^2 \\ g(x) := \log x + x^2 \\ g'(x) := 2x + \frac{1}{x} \end{bmatrix}}_{\begin{pmatrix} \frac{d}{dx} (\log x + x^2)^3 \\ = \quad 3(\log x + x^2)^2 \left(2x + \frac{1}{x}\right) \end{pmatrix}}$$

```
(%i7) mkholes(o, /* [1,2,1,1,2], */
                 [1,2,2,1,2],
                 [1,2,3,1,2],
                 [1,2,4,1,2],
                 [2,2,2]);
(%o7)
```

$$\underbrace{\begin{pmatrix} \frac{d}{dx} f(g(x)) \\ = \quad f'(g(x))\, g'(x) \end{pmatrix} \begin{bmatrix} f(x) := x^3 \\ f'(x) := \ ? \\ g(x) := \ ? \\ g'(x) := \ ? \end{bmatrix}}_{\begin{pmatrix} \frac{d}{dx} (\log x + x^2)^3 \\ = \quad ? \end{pmatrix}}$$

This program is a *manipulable object*: students can, for example, duplicate the 4-line command with the definition of 'S', modify the substitutions in the copy, and run the new code in the REPL to see what changes in the result...

We can also ask students to modify the 'S' above to obtain true expressions that solve other exercises of the book.

Very few students understand right away how they can find the derivative of $\frac{d}{dx}\sin(42x)$ by this method by solving a simpler problem first – the simpler problem is $\frac{d}{dx} f(g(x)) \begin{bmatrix} f(x):=? \\ g(x):=? \end{bmatrix} = \frac{d}{dx}\sin(42x)$. By making the students work together the weaker students will see that the stronger students found a way to decompose bigger problems into smaller problems, and will see that there is a method there that they can learn.

Note that I haven't shown (yet!) how to transform our expandable proof from section 1 into a manipulable object – I am just showing how to manipulate objects that look like its *justifications*.

7 Extremely Bad Foundations

In [FeynmanJoking] Richard Feynman tells several stories about his stays in Brazil and his experiences teaching classes here. This is from one of them, about a class that he taught in 1952:

> I taught a course at the engineering school on mathematical methods in physics, in which I tried to show how to solve problems by trial and error. It's something that people don't usually learn, so I began with some simple examples of arithmetic to illustrate the method. I was surprised that only about eight out of the eighty or so students turned in the first assignment. So I gave a strong lecture about having to actually *try* it, not just sit back and watch *me* do it.
>
> After the lecture some students came up to me in a little delegation, and told me that I didn't understand the backgrounds that they have, that they can study without doing the problems, that they have already learned arithmetic, and that this stuff was beneath them.

We use the term "Foundations of Mathematics" for the axioms and logical rules behind the maths that a person does; nowadays the most common foundations are ZFC and Type Theories. I will use the (non-standard) term "Bad Foundations" for people who don't know any logical rules and who do mathematics by just memorizing methods and applying them, and "Extremely Bad Foundations" for people who not only have bad foundations but who also believe that all methods that require trial and error – or even trial-and-improve; see [DrijversBoonReeuwijk, p.180] – are wrong, and they refuse to learn them.

Some courses are supposed to work like sieves that only let through the students whose mathematical skills are fine enough. But the system was already dysfunctional, and then we had COVID and now LLMs, and now lots of very coarse students with Extremely Bad Foundations are reaching classes like Calculus 2, and we need to do some something for them that is better than just failing them, or, even worse, letting them pass these advanced courses unchanged...

One idea – *that I am treating as a work in progress; I have only a few examples ready, and they were tested only on small classes, and without a rigorous methodology* – is that we can use Computer Algebra Systems like Maxima to treat some methods from Calculus as expandable objects, make the computer draw the expanded versions of some examples, and use that to show to students with extremely bad foundation how the other students think, and what are the steps that they don't write down. It should be possible to use "holes" to convert these expanded version into exercises, and if the students with bad and extremely bad foundations

do these exercises then it *may be possible* that they will develop certain pattern-matching skills by drill, and they *may* reach the point that John Mason describes in [MasonShift, p.4], in which there happens "*a splitting of attention, [with] one part [of the mind] involved in calculation while another part remains aloof but observant*". After learning certain new procedural skills they *may* have an "aha! moment" and start to understand how Logic works.

I will close with another example of a method as a manipulable object. [Stewart8, p.639] explains ODEs with separable variables starting by the general case, $\frac{dy}{dx} = \frac{g(x)}{h(y)}$, then solving it by a method that involves differentials and includes steps that many students find hard to believe; and then its Example 1, in the following page, is $\frac{dy}{dx} = \frac{x^2}{y^2}$, that has solutions that are very hard to draw by hand...

For me the "archetypal" ODE with separable variables is $\frac{dy}{dx} = -\frac{x}{y}$, whose solutions are circles centered on $(0,0)$, and the general method can be reconstruced from that archetypal case. The archetypal case and the general method are:

$$[\text{A}] = \begin{pmatrix} \frac{dy}{dx} & = & \frac{-(2\,x)}{2\,y} \\ 2\,y\,dy & = & -(2\,x)\,dx \\ \int 2\,y\,dy & = & \int -(2\,x)\,dx \\ || & & || \\ y^2 + C_1 & = & -x^2 + C_2 \\ y^2 & = & -x^2 + C_3 \\ \sqrt{y^2} & = & \sqrt{-x^2 + C_3} \\ || & & \\ y & & \end{pmatrix} \quad [\text{M}] = \begin{pmatrix} \frac{dy}{dx} & = & \frac{g(x)}{h(y)} \\ h(y)\,dy & = & g(x)\,dx \\ \int h(y)\,dy & = & \int g(x)\,dx \\ || & & || \\ H(y) + C_1 & = & G(x) + C_2 \\ H(y) & = & G(x) + C_3 \\ H^{-1}(H(y)) & = & H^{-1}(G(x) + C_3) \\ || & & \\ y & & \end{pmatrix}$$

and the techniques for working on the "archetypal case" and the "general case" in parallel are explained in [OchsIDARCT] and [OchsMD]. *I hope to finish the details for this example soon.*

References

[AhmadApplebyEdwards] J. Ahmad, J. Appleby, and P. Edwards. "Engineering Mathematics should be taught by Engineers!" In: *Proceedings of the Undergraduate Mathematics Teaching Conference (Birmingham, England, September 3-6, 2001)*. Ed. by P. Egerton. https://files.eric.ed.gov/fulltext/ED482954.pdf. 2001, pp. 33–40.

[ApexCalculus4] G. Hartman et al. *Apex Calculus, 4th ed.* Ed. by J. Bowen. http://www.apexcalculus.com/downloads. Hartman, 2018.

[Boaler] J. Boaler. *Experiencing School Mathematics: Traditional and Reform Approaches to Teaching and Their Impact on Student Learning*. Lawrence Erlbaum Associates, 2002.

[BottoniTFM] M. L. Bottoni, A. S. Cattaneo, and E. Saçıkara. *Teaching "Foundations of Mathematics" with the LEAN Theorem Prover*. 2025. arXiv: 2501.03352v2.

[CarlsonGulick] D. E. Carlson and D. Gulick. "Calculus for Engineering Students - First Discussion Session". In: *Calculus for a New Century: a Pump, not a Filter - a National Colloquium, October 28-29, 1987*. Ed. by L. A. Steen. https://maa.org/wp-content/uploads/2024/10/NTE8_optimized.pdf. MAA, 1987, pp. 62–63.

[DrijversBoonReeuwijk] P. Drijvers, P. Boon, and M. van Reeuwijk. "Secondary Algebra Education: Revisiting Topics and Themes and Exploring the Unknown". In: ed. by P. Drijvers. Sense, 2011. Chap. Algebra and Technology, pp. 179–202.

[EngelbrechtBergsten] J. Engelbrecht, C. Bergsten, and O. Kågesten. "Conceptual and Procedural Approaches to Mathematics in the Engineering Curriculum: Student Conceptions and Performance". In: *Journal of Engineering Education* 101.1 (2012), pp. 138–162.

[FeynmanJoking] R. Feynman. *Surely You're Joking, Mr. Feynman!* W. W. Norton, 2018.

[FilloyRojano] T. Rojano E. Filloy. "Solving Equations: the Transition from Arithmetic to Algebra". In: *For the Learning of Mathematics* 9.2 (1989), pp. 19–25.

[FischbeinTacit] E. Fischbein. "Tacit Models and Mathematical Reasoning". In: *For the Learning of Mathematics* 9.2 (1989), pp. 19–25.

[FreudenthalDPh] H. Freudenthal. *Didactical Phenomenology of Mathematical Structures*. Kluwer, 1999.

[Harper] R. Harper. *Practical Foundations for Programming Languages, 2nd ed.* Cambridge, 2016.

[HindleySeldin2008] J. R. Hindley and J. P. Seldin. *Lambda-Calculus and Combinators, an Introduction*. Cambridge, 2008.

[Hut16] G. Hutton. *Programming in Haskell, 2nd ed.* Cambridge, 2016.

[Kindt] M. Kindt. "Secondary Algebra Education: Revisiting Topics and Themes and Exploring the Unknown". In: ed. by P. Drijvers. Sense, 2011. Chap. Principles of Practice, pp. 137–178.

[MasonShift] J. Mason. "Mathematical Abstraction as the Result of a Delicate Shift of Attention". In: *For the Learning of Mathematics* 9.2 (1989), pp. 2–8.

[OchsEBL2025] E. Ochs. "Adapting Lean tutorials to the Brazilian case (presentation at the EBL 2025)". https://anggtwu.net/math-b.html#2025-ebl. May 2025.

[OchsEmacsConf2024] E. Ochs. "Emacs, eev, and Maxima - now! (eev @ EmacsConf 2024)". https://anggtwu.net/emacsconf2024.html. Dec. 2024.

[OchsIDARCT] E. Ochs. "Internal Diagrams and Archetypal Reasoning in Category Theory". In: *Logica Universalis* 7.3 (2013). https://anggtwu.net/math-b.html#idarct.

[OchsMD] E. Ochs. *On the Missing Diagrams in Category Theory (First-Person Version)*. https://anggtwu.net/math-b.html#2022-md. 2022. arXiv: 2401.12345.

[OchsNSC2026] E. Ochs. "Notes on Set Comprehensions)". https://anggtwu.net/LATEX/2026-set-comprehensions.pdf. 2026.

[PuigRojano] L. Puig and T. Rojano. "The Future of the Teaching and Learning of Algebra - The 12th ICMI Study". In: ed. by K. Stacey, H. Chick, and M. Kendal. Kluwer, 2004. Chap. Symbols and Language, pp. 189–223.

[SchoenfeldWhenGood] A. H. Schoenfeld. "When Good Teaching Leads to Bad Results: The Disasters of 'Well-Taught' Mathematics Courses". In: *Educational Psychologist* 23.2 (1988), pp. 145–166.

[Sfard] A. Sfard. *Thinking as Communicating – Human Development, the Growth of Discourses, and Mathematizing*. Cambridge, 2008.

[Stewart8] J. Stewart. *Calculus, 8th ed.* Cengage Learning, 2016.

[ThomasRethinking] M. Thomas. “And the Rest is Just Algebra”. In: ed. by S. Stewart. Springer, 2017. Chap. Rethinking Algebra: A Versatile Approach Integrating Digital Technology, pp. 173–201.

[Yalep] F. T. Minh, L. Gonnord, and J. Narboux. “A Lean-based Language for Teaching Proof in High School”. In: *Intelligent Computer Mathematics*. Ed. by V. de Paiva and P. Koepke. Springer Nature Switzerland, 2026, pp. 447–467. ISBN: 978-3-032-07021-0.

[YalepSurvey] F. T. Minh, L. Gonnord, and J. Narboux. “Proof assistants for teaching: a survey”. In: *Electronic Proceedings in Theoretical Computer Science (EPTCS)*. Vol. 419. J. Narboux, W. Neuper and P. Quaresma. Nancy, France: Springer, July 2024, pp. 1–27. DOI: 10.4204/EPTCS.419.1. URL: https://hal.science/hal-04705580.