CONCEPTUAL UNITY VS COGNITIVE LOAD

DIVAKARAN DIVAKARAN

1. The Core Tension

Much of the appreciation for mathematics comes from seeing underlying structure and conceptual unity across seemingly different topics. As instructors, we wish to communicate this unified perspective to students. However, often students are overwhelmed, and our attempts are not as successfull. I would like to stress that my aim is not to make students comfortable, but to give them the best learning experience. The right kind and amount of struggle is key to learning - we will call it productive struggle. Thus, the two important questions are:

(1) What is the right stage for introducing an abstraction and why?
(2) How do we distinguish productive struggle and unproductive struggle? How do we ensure that our students are struggling productively?

These questions are hard to answer as we (the teachers) are often blind to the difficulties experienced by students. Either because we forgot the struggles we had as a student or because we didn't struggle as much as some of our students. As you became a mathematics teacher, it is highly likely that you were extremely passionate and/or found it easier than many others.

A struggle is deemed productive if it helps the student achieve a positive outcome. The student gains new confidence and is willing to take on new challenges. However, the outcome is not determined just by the content, but depends on several other factors. For instance, providing better support during their struggle can mitigate things to a significant extent. But, there are other factors, and not all are within our control. This makes our decisions harder.

In this talk, I want to discuss a few examples from my teaching and what I learned from them.

2. Example 1: Using Functional Programming in Discrete Mathematics

Discrete mathematics can feel like a bag of disconnected topics - as it was historically the areas of mathematics that are useful for a computer scientist. One unifying thread is the use of induction and recursion. Functional programming provides a great playground for these ideas. So, I hoped that using functional programming in discrete mathematics would help students appreciate this structure. I believe I gained further clarity about structural induction when I was learning functional programming, even though I had been using induction and structural induction for many years.

However, in both the iterations of the course, I was not successfull in conveying the beauty of functional programming that so evident to me. Some students recognised and appreciated my enthusiasm for the subject, but that still didnt translate to learning the language. I didn't understand the reason behind the failure and I turned to others for advice. My friend Prathamesh had very successfull experience interweaving functional programming and discrete mathematics. And my colleague Ramanujam said that he too have heard of many successfull stories. So, what went wrong? Of course, the simplest (and plausibly correct) answer is that I am not as good a teacher as Prathamesh and others. But, then I would ask

Date: June 20, 2026.

a follow-up questions: How can I become a better teacher? What can I do to ensure success in my teaching?

I of course spoke to many people including colleagues and friends, but I also had a chat with ChatGPT. ChatGPT told me that in the following code while I see an elegant inductive structure, students were distracted by the brackets, colon operator and the two definitions.

```
sum [] = 0
sum (x:xs) = x + sum xs
```

Also imperative programming (asking computer what to do) is perhaps more natural than declarative programming (you tell computer what it is). The above syntax, that uses pattern matching is a perfect example of the declarative style. I had not anticipated this difficulty as declarative style is more natural for mathematicians. But, students in this course need not be mathematicians. They are just beginning their journey. Of course, there will be other reasons and if I understand why students found it hard, I will be able to address them and become a better teacher.

3. Example 2: Vector Space Redundancies

Mathematicians often like to have a minimal set of axioms. Famously, many mathematicians thought that Euclid's fifth postulate was redundant, that is, it was a consequence of the other four postulates. It was proved much later that the fifth postulate is not a consequence of the other four - we can come up with "geometries" where the first four axioms are satisfied, but the fifth axiom is not satisfied. This was the birth of hyperbolic and spherical geometry.

Similarly, one may ask if axioms of vector spaces are essential. Byrant proves in [Bryant] that commutativity of vector addition can be inferred from the other axioms. In [Rigby-Wiegold] the authors show that actually you need only 6 axioms. I believe, according to their definition, the empty set would become a vector field. But, with a small modification, we can solve this issue. Basically, we can replace Axiom 3 and Axiom 4 with a single axiom: $\exists$ **a vector** $0 \in V$ **such that** $\forall v \in V$**,** $0.v = 0$. This vector is the additive identity as $v + 0 = 1.v + 0.v = (1 + 0).v = 1.v = v$. Also, $(-1)v$ is the additive inverse as $v + (-1)v = (1 + (-1)).v = 0.v = 0$. Also, we cannot just keep our current "Existence of additive identity" as it is and delete "Existence of additive inverse". Because, then although v + (-1)v = 0.v, we cannot conclude that is the zero vector. It is still unclear to me if the other axioms are redundant or not. It is a very interesting exercise to come up with examples that satisfy all but one axiom. Some interesting examples for vector spaces over a field (not necessarily $\mathbb{R}$) is provided in [user7530].

I felt that grappling with the axioms in this manner, would help students understand the axioms better. I also thought it conveys the fact that often the choice of axioms is not just logical, but historical/social. Once again, the reception was not so positive. These students had not spent enought time with Euclidean geometry and redundancies of the axioms of Euclidean geometry. Thus, probably they didn't appreciate the question itself. In a previous iteration, I had tried to discuss Euclidean geometry through linear algebra and that too was not too successful. Once again, it would have been more exciting if they had answered the questions of Euclidean geometry in a more tedious way earlier. Then you see the power of abstraction in general and linear algebra in particular.

4. Example 3: Spanning trees and basis

While preparing to teach trees (as part of the discrete mathematics course), I noticed a striking similarity between the definition of a basis and a spanning tree. Both concepts are defined using three equivalent definitions. If you compare these definitions, you see that the existence of a cycle is analogous to dependence and connectedness is analogous to spanning. Interestingly the connection goes much deeper and the common abstraction is

Basis	**Spanning Tree**
A collection of vectors that span and are linearly independent	A subgraph that is connected and acyclic
A minimal spanning set of vectors	A minimal connected subgraph
A maximally linearly independent set of vectors	A maximally acyclic subgraph

called a matroid. A matroid is a lattice that is atomistic and semi modular. We won't define what atomistic and semi modular are in general, but would explain what they are for the examples we are interested in.

	Vector spaces	**Graphs**
	The relation of $\subseteq$ form a partial order on the collection of all vector subspaces of a vector space V	Let G be a connected graph. A set of edges is closed if you cannot add any new edge from the original graph without connecting previously disconnected component. The relation $\subseteq$ form a partial order on the collection of all closed sets of edges of G
	$\{0\}$ is the bottom	graph w/ no edges is the bottom
	V is the top	The full graph is the top
Lattice	Intersection acts as the meet	Intersection acts as the meet
	Sum acts as join. Recall that union of two vector subspaces is not a subspace. Sum is the smallest subspace containing the union of two subspaces	Union acts as join
Atomistic	every subspace is sum of a collection of $\langle v \rangle$	every closed set of edges is formed by taking union of individual atoms
	$\langle v \rangle$ are atoms	individual edges are atoms
Semi-modularity	$\dim(U+W) = \dim(U) + \dim(W) - \dim(U \cap W)$	Instead of dimension, we have a function r on the collection of closed set of edges. Let us define $r(X) =$ $V -$ no. of components in X. Then, $r(A \cup B) \leq r(A) + r(B) - r(A \cap B)$

I was so thrilled when I discovered this connection and was surprised why no one told about this to me earlier. As a firm believe of "Do unto others what you want others to do to you", I decided to make this as an explicit part of my course. I am happy that some students appreciated the discussion, but more than half of the students didn't understand what I was talking about.

5. Open questions and further discussion

The cost of disengagement in the examples two and three were quite minimal as it was just one lecture. However, functional programming in the first example was a substantial part of the course. We should be even more careful when we make choices for the entire course. More importantly, I would like my students (at least some) to appreciate this beautiful connection. I understand that there is a right time to introduce these things. As I am fortunately in a university where we have the freedom to design the curriculum, I want to discuss how the curriculum can be designed so that students can appreciate these connections by the end of the program (if not a course). To summarise,

(1) What is the right stage for introducing an abstraction and why?
(2) How do we distinguish productive struggle and unproductive struggle? How do we ensure that our students are struggling productively?
(3) How do we encourage students to ask for support so that bigger struggles can be productive?
(4) How can we design a curriculum so that students can appreciate these connections by the end of the program (if not a course)?

Disclosure of use of AI tools

I have used Antigravity extensively as a latex support. All compilation errors were fixed by antigravity. I also had a very productive discussion with ChatGPT about the use of functional programming in discrete mathematics. In some sense, the core idea of this talk is a result of that discussion.

References

[Bryant] Victor Bryant. *Reducing Classical Axioms*. The Mathematical Gazette, vol. 55, no. 391, 1971, pp. 38–40.

[Rigby-Wiegold] Rigby, J. F., and James Wiegold. *Independent Axioms for Vector Spaces*. The Mathematical Gazette 57, no. 399 (1973): 56–62.

[user7530] user7530 (`https://math.stackexchange.com/users/7530/user7530`). *Answer to "Is every axiom in the definition of a vector space necessary?"*, `https://math.stackexchange.com/q/1412923`