Heuristic Presentations: the Role of Structuring

URI LERON

1. Introduction

Good expositors often supplement the formal parts of their exposition with informal mathematical ideas that, they feel, are neither captured by the formalism, nor easily reconstructible from it. It appears, therefore, that the formal code used to communicate mathematical knowledge is not sufficiently self-explanatory. The first part of the paper (Sections 2 and 3) discusses this insufficiency, as well as some informal practices ("heuristics") used by expositors in trying to alleviate it. The issues are well-known but an elaboration was felt to be needed in preparation for the second part. The main contribution of the paper is to extend the context in which these practices have normally been used, and to attempt to *standardize* their use by building them right into the formalism. Consequently, another formalism emerges for presenting mathematical procedures—proofs, definitions, constructions, algorithms and examples (Sections 4 and 5). This formalism, called *the structural method,* turns out to be a well-known method of contemporary computer science, used to organize complex programs so as to make them easier to read, debug and modify. It was first introduced in connection with mathematical proofs in [1]. Here I take a more "evolutionary" approach: The structural method is shown to be a natural consequence of embedding accepted heuristic practices into the existing formalism. Also, the fundamental role of the *pivot* is further articulated (see especially 4.2.2). Section 4 is a detailed case study of one example—the Cantor-Berstein theorem of set theory.

The structural method, born of the marriage of formal and informal methods, should hopefully benefit both. On the one hand, the formalism will become more self-explanatory, more *communicative* of the ideas behind it, without sacrificing rigor. On the other hand, heuristic methods of presentation will be used by more people more often: rather than being merely optional, they will have become standard.

However, as much as I believe the structural method to be superior to the linear method, I am well aware of the fundamental limitations of *any* universal method of formal presentation. Firstly, being merely a method of *presentation,* it addresses only one part of the teaching/learning process. Most of the route to learning and understanding will have to be taken by the student anyway. Secondly, being a *formal* method, it clearly needs to be supplemented by the enormously valuable informal methods of discussion, investigations, discovery, etc. Many students are intimidated by any kind of formalism, and if the students are not listening—it matters little what we are saying. Thirdly, being a *universal* method, it does not address differences in learning styles of individual students. It remains to be seen where and when it should be beneficial to use the structural method, the linear method, bits and pieces of each, or no formal presentation whatever.

2. The insufficiency of the linear method

There is a lot more to a proof than the formal, step-by-step, "linear code" of it, as it often appears in lectures, textbooks or research articles. When I transmit a proof through any of these channels, I am aware of many ideas and connections which are essential to any real understanding of the proof, but are not part of the formal code. Not only are these ideas not clear from the code—they are often actually *obscured* by it. (This phenomenon is well-illustrated in the example of Section 4.) Yet these ideas are usually suppressed in the presentation since the prevalent paradigm of mathematical communication does not encourage, perhaps even discourages, their inclusion. Meanwhile, at the other end of the channel, you are trying to make sense of my proof. Not satisfied with just following my step-by-step, linear account, you are hard at work decoding the ideas and connections buried beneath the formal code—*the very ideas I suppressed* in coding the proof.

The linear formalism can be described as a *minimal* code for transmitting mathematical knowledge. Minimal, that is, with respect to a mature mathematician's ability to decode it, to make sense of it, to gain working knowledge of it. On closer examination, however, it appears in several respects to be a *sub*-minimal code, resulting in irretrievable loss of important information. Firstly, even under the most favorable conditions—reading a proof in my own subspeciality—a lot of the effort, time and frustration that go into recreating the writer's ideas are wasteful and unjustified. Secondly, I am discouraged from reading research articles in other mathematical fields where the decoding becomes unbearably hard. (That this is indeed a problem of *decoding* is evidenced by the observation that I am more than willing to read popular expositions in these fields, where the mode of communication is typically reversed: ideas are stressed and formal code is suppressed.) Third—and worst—many students in our mathematics classes are simply unable to decode. They are reduced to meaningless manipulation of the formal code itself, with no awareness of the ideas and concepts it codes for.

This shallow manipulation of code is evidenced by examples like the following, in which students write absolute nonsense in a perfect mathematical style. Asked to prove that the sets N and E (the natural and even numbers, respectively) are equivalent, one student wrote:

> Define a function f: $N \rightarrow E$ by $f(n) \in E$ for all $n \in N$. Then f is one-to-one since $f(m) = f(n)$ implies $m = n$. Also, f is *onto,* for given any $e \in E$ let $n \in N$ satisfy $f(n) = e$; then n is a source for e. (1)

It is therefore important to modify our formalism so that it incorporates more of the ideas we consider important for understanding. In other words, we should try to make the formalism more self-explanatory. (2) Of course, this does not *guarantee* understanding; it only makes the route to it (which students will have to take anyway mostly on their own) more accessible and less tortuous.

3. The two heuristics

We now consider the two informal practices (heuristics) that in the next section we propose to extend and build into the formalism. Though widely recognized as practices of good exposition, these methods are only occasionally exercised in higher-level mathematics. Being optional rather than standard, they are little used in the classroom, less in textbooks, least in research articles and colloquia.

The first heuristic is that of prefacing a long, complex proof with a short, intuitive overview. The overview has a psychological as well as mathematical function. Psychologically, its purpose is to convince the reader (or listener) that behind the complexity there is a simple, natural idea. The essential requirement here is that it be short and simple, so that it can be "grasped at a glance." (Is this not what "overviewing," or "viewing from the top" is all about?) Mathematically, the overview should somehow capture the central idea of the proof. A good test for this requirement is that subsequently the overview could (a) be made formal and rigorous, and (b) be refined to a complete proof. (This, incidentally, excludes from the present discussion personal metaphors, real-life analogies, etc.) In this function the overview might be said to form a *skeleton* of the proof, later to be fleshed out to a complete proof (3). All in all, the overview lends coherence to the detailed exposition, creates a whole out of the parts, a forest out of the trees.

The second heuristic typically appears in "construction" problems. Suppose we wish to construct some mathematical object satisfying certain preset conditions. Such a problem is analogous to a system of equations, so I shall refer to it as *a system of constraints,* and to an object satisfying the constraints—*a solution-object.*

Examples

1. Complex numbers. Construct a (minimal) field containing the reals and a square root of -1.
2. The Gram-Schmidt orthogonalization process. Given a finite set of vectors in an inner-product space, construct an orthonormal set spanning the same subspace.
3. Proofs concerning limits. Given a positive epsilon, construct a delta such that . . .
4. Ruler and compass constructions in Euclidean geometry.
5. Solving equations—linear, quadratic, differential, etc. Given the equations construct a solution that satisfies them.

There are two ways to present the solution to construction problems. In the "linear" way, one simply *states* the definition of the solution-object, then *proves* it is indeed a solution. (It is easy to recognize this method by its opening words, usually "let," "consider," or "define," as in "consider the following function . . .") In a "heuristic presentation," in contrast, one uses the given constraints to *search* for the form of the solution-object, then uses that form to *define* it. Here one typically engages in a kind of wishful thinking: "Suppose we had already found a solution-object, what would it look like?")

> *Example*
> Contrast "Let $C = \{ (x, y) \mid x, y \in R \}$ . . . " with "Suppose we already had such a field C. Then it must contain an element whose square is -1; call it i. Since C is a field containing R, it must contain all combinations of the form $a + bi$, where a and b are reals." Similar comparisons of style could be made for the definition of addition and multiplication in C, as well as for the other examples given above.

While most people will probably agree that these heuristic principles make for a better, more communicative exposition, the temptation is still great to resort to the linear method, partly because of its conciseness and apparent elegance. It is important, therefore, to emphasize that the heuristic presentation offers more than a condescending nod to the feeble of mind: it actually contains more information. This added information is an example of the ideas, mentioned before, that are important for meaningful learning but are suppressed by the linear method.

4. An example: the Cantor-Bernstein Theorem

We now follow through our plan in the case of one theorem and its proof—the Cantor-Bernstein Theorem of set theory. The theorem is a nontrivial extension to infinite cardinals of the following property of numbers:

> If $m \leq n$ *and* $n \leq m$ then $m = n$.

When moving to infinite cardinals, equality of numbers is replaced by equivalence of sets, weak inequality with equivalence of one set with a subset of the other. (Recall that two sets are *equivalent* if there exists a one-to-one map of one onto the other.)

Theorem [*Cantor-Bernstein*] Let A and B be sets. If there exist one-to-one maps from A into B and from B into A, then there exists a one-to-one map from A *onto* B.

We introduce appropriate notation (Figure 1). Given one-to-one maps $f: A \rightarrow B$ and $g: B \rightarrow A$ we wish to construct a one-to-one "onto" map $h: A \rightarrow B$.

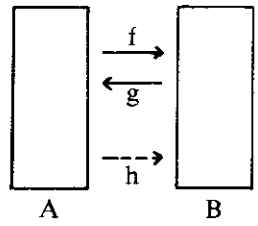


Figure 1
The Cantor-Berstein Theorem

4.1 LINEAR PROOF

Define $X_0 = g(B)$, $\varphi = g \circ f$, $Y_0 = A - X_0$ and (recursively) $Y_n = \varphi(Y_{n-1})$ for all $n > 0$. Now let $Y = \bigcup_{n=0}^{\infty} Y_n$ $X = A - Y$

Note that $Y \supseteq Y_0$, so $X \subseteq X_0 = g(B)$, hence $g^{-1}(a)$ is defined for $a \in X$ (since g is one-to-one.) Note, too, that the definition of X guarantees that $X \cup Y = A$ and $X \cap Y = \phi$. Thus the following rule gives a well-defined map $h : A \to B$:

$$h(a) = \begin{cases} f(a) & \text{if } a \in Y \\ g^{-1}(a) & \text{if } a \in X \end{cases}$$

Let $y \in Y$ be given. Then $y \in Y_n$ for some $n \geq 0$, hence $\varphi(y) \in \varphi(Y_n) = Y_{n+1} \subseteq Y$. Thus $\varphi(Y) \subseteq Y$. Suppose $h(a_1) = h(a_2)$ and distinguish three cases. If $a_1, a_2 \in Y$ then $f(a_1) = f(a_2)$ hence $a_1 = a_2$. If $a_1, a_2 \in X$ then $g^{-1}(a_1) = g^{-1}(a_2)$, hence $a_1 = a_2$. In the remaining case we have, say, $a_1 \in X$ and $a_2 \in Y$. Then $g^{-1}(a_1) = f(a_2)$, hence $a_1 = gf(a_2) = \varphi(a_2) \in \varphi(Y) \subseteq Y$. Since the result $a_1 \in Y$ contradicts $X \cap Y = \phi$, this case cannot happen. We have seen that in all cases $h(a_1) = h(a_2)$ implies $a_1 = a_2$, that is, h is one-to-one. Let $b \in B$ be given. We put $a = g(b)$ and distinguish two cases. If $a \in X$, then $h(a) = g^{-1}(a) = g^{-1}g(b) = b$. If $a \notin X$ then $a \in Y$, hence $a \in Y_n$ for some $n > 0$ ($a \in Y_0$ is impossible since $a \in g(B) = X_0$.) It follows that $a \in \varphi(Y_{n-1})$, hence $a = \varphi(y)$ for some $y \in Y$. Now $gh(y) = gf(y) = \varphi(y) = a = g(b)$ and since g is one-to-one, $h(y) = b$. We have found a source for b in all cases, so h is "onto." This completes the proof of the theorem.

4.2 STRUCTURING

Let us now shift our attention from the subject matter of the proof to its intended recipient—the student. When we realize that the presentation of the proof is primarily an act of *communication,* it becomes clear that its organization in section 4.1 is all wrong. Most of the steps, as they are being followed by the student, are intellectually indigestible at the time they appear (especially the opening 5 lines), and the little that *can* be said to help make some sense of this mass of technicalities, remains unspoken. The student is thus reduced to a step-by-step "execution" of the proof, much as a computer would execute a program. The linear formalism encourages "linear understanding," i.e., the semi-mechanical ability to follow the steps of the proof sequentially, merely checking the validity of the deduction at each step. The relatively rare case of students achieving a deeper, more mature understanding (and of instructors helping them to achieve such understanding) seems to occur not because of the linear presentation but in spite of it.

In the next few sub-sections I shall carry out the plan set up earlier, in the case of the Cantor-Berstein theorem. That is, I shall try to incorporate into the proof the heuristics discussed in Section 3. The framework for achieving this goal is supplied by the formalism and ideas of "structured programming," whose influences will be felt throughout.

The rest of 4.2 is intended to be read in conjunction with 4.3, where the finished product of the process described here is displayed. Thus it is suggested that along with each sub-section of 4.2, the corresponding part of the structured proof in 4.3 be read.

4.2.1 *What is it all about? A top-level view of the proof*

Our first step towards making the proof more communicative is to note, and to acknowledge loudly and clearly, that it is actually based on a very simple and natural idea. We are trying to construct a function h going from A to B, using the available materials. These are the given functions from A to B, i.e. f and the inverse of g (where it exists). Thus:

> Partition A as $A = X \cup Y$, and take h to be f on X and g^{-1} on Y. By choosing the partition wisely, we hope to make the map h well-defined, one-to-one and "onto." (See Figure 2a)

In a way, we can view this idea as forming a whole proof by itself, except that certain large gaps have been temporarily left open. We call this the *top level,* or Level 1 of the proof. (See Level 1 of the structured proof in 4.3) By pushing the more technical details down to lower levels, it becomes possible to present a skeleton of the proof that is short and simple enough to be grasped at a glance. It is the same idea which the classroom instructor uses in prefacing a complex proof with an "intuitive" overview, except that here this idea becomes an official, inseparable part of the formal proof. Note in comparison, that in the linear presentation, this central and simple idea is not localizable to any particular place in the proof. Not only is it not stated explicitly, but its fragments are scattered throughout the whole proof.

4.2.2 *A global connection: the pivot*

From similar analyses of many proofs (see [1]), we can state more explicitly the mechanism by which the top level creates a direct connection between the premises and the conclusion of the theorem. The top level consists of an *act of construction,* creating a new mathematical object that interpolates between the premises and conclusion. It is a remarkable fact that such constructions seem to inhabit all nontrivial proofs. (In our example, the new object is the partition $A = X \cup Y$ such that the associated map h is well-defined, one-to-one and "onto".) I call this object *the pivot* since the rest of the proof entirely hinges upon it. (4) Given the premises, the pivot can be constructed, its properties proved; given the pivot, the conclusion follows almost immediately. However, in the top level we only *postulate* the pivot and show how it leads to the conclusion of the theorem. Details of the construction (or proofs of

existence) and proofs of the postulated properties are pushed down to lower levels. More formally, at the top level a *system of constraints* is established such that the rest of the proof is reduced to solving that system (in our example, "find a partition $A = X \cup Y$ such that..."). One might say that the theorem has been proved *modulo* the solution of the system. In this formulation, the pivot of the proof appears simply as the *solution-object* of the system.

4.2.3 *How shall we go about it? The elevator*
Next we turn to the task of solving the system set up at the top level. This is a construction problem, and we approach it heuristically as outlined in Section 3. The discussion leading to discovery of the solution-object (in our case, the partition), is not part of the formal proof but in my opinion, should be included in the presentation as a matter of course. This is important if we want students to view mathematical activity as elaborate commonsense rather than sorcery (as in the above linear proof, where one apparently conjures up new mathematical objects by invoking the right magical words). I view this intermediate process of descending in levels of the proof as taking place *in the elevator*. Having discovered "in the elevator" the appropriate conditions for constructing the pivot, we enter level 2 of the proof by postulating these conditions, then showing that any object satisfying them is indeed a solution to our system. Again, we push the details of the actual construction down to the next level, in order to keep the present one simple, brief and "flat" (i.e. free of many nested sub-arguments). Refer to Levels 2.2 and 2.3 and the "elevator" discussions preceding them in 4.3.

Now while the merits of heuristic presentations are widely recognized (at least in principle) for explicitly stated construction problems, it is only rarely that they are applied to the construction of pivots within proofs. Moreover, the construction problem itself is rarely announced explicitly. Thus a typical linear presentation of the pivot has the form of an *extremely clever answer to a question that has never been asked*. This is the notorious Let-us-define-a-function Syndrome (to borrow Avital's phrase), whose adverse effects on students and other people we all witness daily. A good example of this phenomenon is the introduction of the pivot in the first few lines of the linear proof in 4.1.

4.2.4 *Divide and conquer: autonomous modules*
It is not easy to see in the linear proof how properties of the partition are individually related to properties of the map h. Instead one constantly encounters what in computer science is known as the "spaghetti effect": The various strands of the proof are hopelessly and inseparably entangled. Both in computer science and in mathematics, one traces the origin of the spaghetti effect to the linear method and to its undisciplined use of references between the various parts of the program or proof (as, for instance, in the notorious use of the GOTO command in BASIC). Here, as in computer science, one can avoid this kind of trouble by dividing the proof into short, autonomous "modules," each embodying one major idea of the proof. (See the various modules of Level 2 and in 4.3.) These modules are allowed to communicate only via inputs and outputs, whence the interference between them is reduced to a minimum. In the context of proofs, the inputs and outputs are the hypotheses and conclusions, the proof being viewed as "processing" the former into the latter. At least two major benefits follow from this strategy. One, the proof now appears as a hierarchy of short, easily grasped pieces. Rather than choking by trying to swallow it all in one piece, one can now consume it in more easily digestible chunks. Two, the modules are easier to modify and re-use in related situations, when the learning experience should become more widely applicable, more "transferable." These benefits are widely recognized among contemporary computer scientists (the analogue of "transfer" being the re-usability of subprocedure and subroutines) and it is reasonable to expect similar effects to occur in mathematics. (5)

4.2.5 *At the bottom: the actual construction*
We finally arrive at the actual, detailed construction, the one that appeared so mysteriously and intimidatingly at the *beginning* of the linear proof. (Refer to Level 3 in 4.3.) Here the mystery has all but disappeared as the conditions and investigation of the higher levels lead us gently and naturally to these details. This stage is often characterized by a "debugging" activty, as we first try out some simple and natural constructions, then modify them when we find them wanting. It is convenient and beneficial to use the elevator for the purpose of keeping record of this gradual build-up of complexity. (See the elevator discussion leading to Level 3.)

4.2.6 *Over and over: a recursive view of structuring*
As pointed out earlier, the top level reduces the proof of the theorem to solving a certain system of constraints. The solution process itself may in general involve complex arguments and subproofs. Whenever we feel that such a subprocedure is itself too involved (i.e. it is not short and simple enough to be conveniently grasped at a glance), we give it the same treatment as we gave the main proof. That is, we find a top-level description of this sub-argument and proceed as before. With just a little oversimplification we may therefore say that the whole structural method amounts to a recursive extraction of top levels of arguments. A good example is the incorporation of the various modules of Level 2 between the top level (Level 1) and the actual construction (Level 3).

4.3 THE STRUCTURED PROOF
Prelude. We have to construct a map $h : A \to B$ using the available materials—the "givens" of the theorem. What maps do we have going in the right direction? Only f and g^{-1}. So it is natural to construct *h in parts* from f and g^{-1}.

Level 1 (Refer to Fig. 2(a)). We partition A as $A = X \cup Y$ (X, Y disjoint) and define $h : A \to B$ as follows:

$$h(a) = \begin{cases} f(a) & \text{if } a \in Y \\ g^{-1}(a) & \text{if } a \in X. \end{cases}$$

We shall construct the partiton (in Level 2) so as to make h well-defined, one-to-one and "onto." This completes the proof of the theorem.

Level 2.1 (Refer to Fig. 2(b)). We make h well-defined by guaranteeing (in Level 3) that $X \subseteq g(B)$ (so that g^{-1} makes sense on X) and that $X = A - Y$ (so that the definition in parts makes sense).

In the elevator. (The elevator, as a metaphor for the intermediate process of descending in levels, offers a convenient place to discuss heuristic and other informal issues concerning the next level.) What does it take to make h one-to-one? Assume $h(a_1) = h(a_2)$. If $a_1, a_2 \in X$ or $a_1, a_2 \in Y$ there is no problem since g^{-1} and f are one-to-one. So all we have to do is bar the possibility of a relation of the form $h(x) = h(y)$ with $x \in X$ and $y \in Y$, that is, $g^{-1}(x) = f(y)$. To transform this into a condition on the sets X and Y, note that this is equivalent to $x = gf(y)$, or $x \in gf(Y)$. So to bar this, we require $gf(Y) \subseteq Y$. (Refer to Fig. 2(c)).

Note. The "elevator" material is very suitable for a classroom discussion.

Level 2.2. We make h one-to-one by guaranteeing (in Level 3) that $\varphi(Y) \subseteq Y$, where $\varphi = g \circ f$. For suppose $h(a_1) = h(a_2)$. If $a_1 \; a_2 \in Y$ then $a_1 = a_2$ since g^{-1} and f are one-to-one. On the other hand, the case in which $a_1 \in X$ and $a_2 \in Y$ (or vice versa) is ruled out since this would entail $g^{-1}(a_1) = f(a_2)$, whence $a_1 = gf(a_2) = \varphi(a_2) \in \varphi(Y) \subseteq Y$. However, the conclusion $a_1 \in Y$ contradicts the assumption $a_1 \in X$.

In the elevator. What does it take to make h "onto"? (Refer to Fig. 2(c), where we also incorporated the information obtained in Exercises (a) and (b) below.) We need $B \subseteq h(A) = g^{-1}(X) \cup f(Y)$, or equivalently, $g(B) \subseteq X \cup \varphi(Y)$.

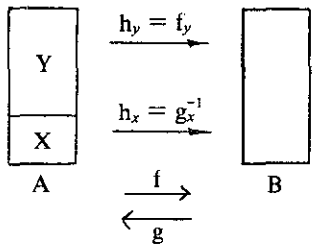
Level 2.3. We make h "onto" by guaranteeing (in Level 3) $g(B) \subseteq X \cup \varphi(Y)$. For then $B \subseteq g^{-1}(X) \cup f(Y) = h(A)$.

Exercises. Assuming the three conditions of Level 2, prove: (a) $g(B) = X \cup \varphi(Y)$, (b) $\varphi(Y) = Y \cap g(B)$. Note that of the two inclusions summarized in (b), one is equivalent to h being one-to-one, the other to h being "onto."

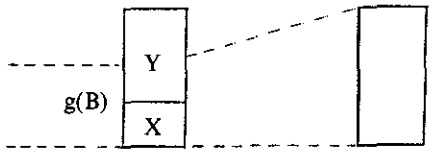
In the elevator. How to construct the partition so as to "deliver" all the promises made in the various parts of level 2? A first natural attempt (looking at 2.1) is $X_0 = g(B)$, $Y_0 = A - X_0$. Unfortunately, Y_0 need not satisfy the "closure" (or "invariance") condition $\varphi(Y) \subseteq Y$ of 2.2. However, there is a standard procedure to "close" Y_0: throw in all φ-images of elements of Y_0, and repeat this operation recursively on the the resulting sets.

Level 3. (Refer to Fig. 2(d)). We define $X_0 = g(B)$ and (recursively) a sequence of sets $Y_0, Y_1, Y_2, \ldots$ by $Y_0 = A - X_0$, $Y_n = \varphi(Y_{n-1})$ for $n > 0$. Finally, we define $Y = \bigcup_{n=0}^{\infty} Y_n$ and $X = A - Y$. Then this partition satisfies the conditions of 2.1 and 2.2 (easy to check), hence the resulting map h is well-defined and one-to-one. Also, since $\varphi(Y) = Y - Y_0$, the partition satisfies the condition of 2.3. Indeed, assume that b in B is given and that $g(b) \notin X$. Then $g(b) \in Y$, but since it cannot be in Y_0 (for $g(b) \in X_0$), it must be in $Y - Y_0 = \varphi(Y)$. Thus $g(B) \subseteq X \cup \varphi(Y)$ and h is "onto."

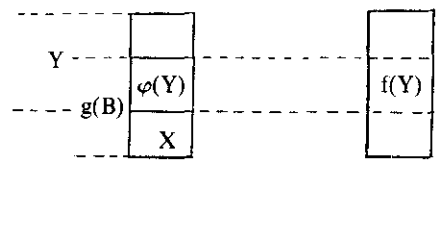
Note. Our ability to contain Level 3 within desirable length might seem to depend on the "lucky accident" that the conditions of 2.1 and 2.2 happened to be easy to check. This, however, poses no problem to the method, for we could also preserve the simplicity of Level 3 by substituting "Level 4" for "easy to check."



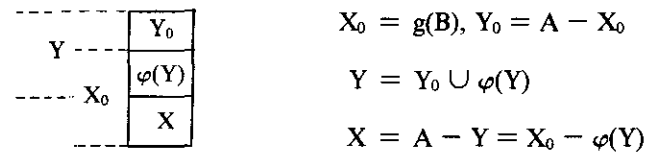
(a) Level 1: Partition of A and definition of h.



(b) Level 2.1: Conditions for h to be well-defined



(c) Levels 2.2 and 2.3: Conditions for h to be one-to-one and onto.



(d) Level 3: The construction.

Figure 2
The structured proof

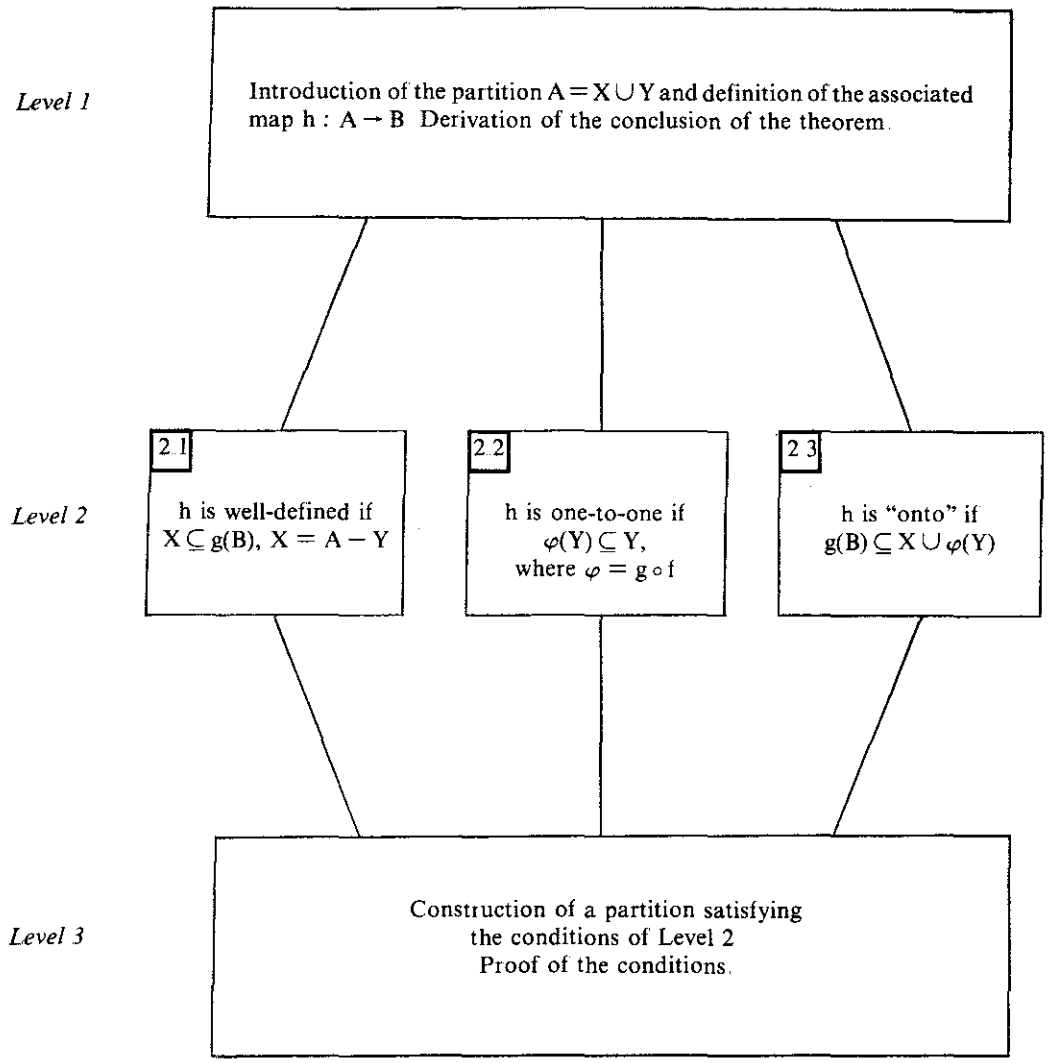


Figure 3
Structure diagram of the proof

5. Summary of the structural method

The main point of this article has been the suggestion that the informal practices discussed in Section 3—overviewing and heuristic treatment of constructions—be consistently applied to the presentation of pivots and proofs in general; in fact, that they become standardized by building them right into the formalism. More specifically, it is suggested that the presentations of proofs (in books, lectures, journal articles, etc.) might benefit from arranging them in the following format:

1. *Introduce the pivot* as a system of constraints, i.e. define it implicitly by *postulating* its properties
2. Without actually solving the system, use the pivot as introduced in step 1 to *derive the conclusion* of the theorem. (These two steps comprise the 'top level' of the proof.)
3. *Discuss* heuristically *the solution* of the system to find how the pivot might be constructed.
4. (Recursion step) *Solve* the system, repeating steps 1 — 4 if necessary. That is, *construct* (or prove the existence of) the pivot then *prove* that it satisfies the postulated properties. If some of the subproofs are themselves complicated, introduce sub-pivots and repeat the four-step procedure.

It is somewhat surprising, and certainly encouraging, to discover that when a pivot is properly identified (not an entirely trivial matter!), then the first two steps need not occupy more than a few written lines, perhaps half a page. Since they establish a direct, easily grasped connection between the premises and conclusion, these two steps offer an overview, or top-level view, of the proof. The overview reduces the proof to the task of solving the system of constraints set up in the top level. The solution may involve much detailed work, but it is all well-motivated now by the knowledge of the overall plan. Furthermore, thanks to the recursion embedded in step 4, one need never deal with

chunks too big to be easily digested—the whole proof is displayed in a hierarchy of manageable pieces. Finally, the actual definition of the pivot, often the most mystifying and intimidating part of the proof, now appears at the *end* of the proof, after its place in the proof and the reasons behind its particular form have been made clear.

This method of organizing and presenting proofs and other mathematical procedures has been called "the structural method." It was introduced more systematically in [1], the main purpose here being to show how it evolves from heuristic principles. The resulting proof has the following characteristics:

1. The proof is arranged in *levels of generality,* the top level being the most general, the bottom the most detailed.
2. The presentation of the levels proceeds from the *top down,* zooming from the global to the local perspective.
3. The levels themselves are further subdivided into short, *autonomous modules,* each embodying one major idea of the proof.

Thus formulated, the method is seen to be closely related to the method of "structured programming," widely recognized in contemporary computer science as an efficient way to deal with complexity of programs, making them easier to manipulate and communicate. In the analogy between proofs and programs, the proof is viewed as a "statement processing" device: given the premises as input, the conclusion emerges as output. It is not clear to me, however, whether the idea of the pivot, and the central place it occupies in the structure of proofs, has an interesting analogue in programming.

6. Conclusion: natural proofs vs. elegant proofs

The complaint has sometimes been expressed against the introduction of the structural method in mathematics, that proofs become longer and "more complicated." Thus to cover a given amount of subject matter would require more time in the classroom and more pages in a textbook or article. (6) While this complaint is factually true, I maintain that precisely therein lies the *strength* of the method. It is important to realize that the structural method does not *create* any additonal complexity—it merely brings to the surface some of the hidden complexity that was there all along, buried underneath the linear code. Since dealing with this complexity is essential for any meaningful understanding of mathematics, the successful survival of the linear formalism over such a long period can only be explained by the fact that mature mathematicians are able to recreate this hidden complexity for themselves. However, experience shows that most students do not achieve this kind of understanding under standard teaching conditions, so it is essential to make these ideas and connections (in fact, "the complexity") more explicit.

Linear proofs of theorems such as Cantor-Bernstein's (or even simple results such as $(fg)^{-1} = g^{-1}f^{-1}$ in group theory) bear an apparent and misleading simplicity that often hides a lot of complexity. Short and "slick" proofs of deep and complex results are perceived as "beautiful" or "elegant" by mature mathematicians, but are often frustrating and even paralyzing for students (7). Structured proofs, in contrast, are more or less what they seem, with no unpleasant surprises lurking behind innocent-looking appearances. It is my belief that the shift from elegant to natural proofs will benefit most students, and perhaps even mature mathematicians.

Finally, suppose that in spite of everything said so far, one must save time or paper, or cover a lot of material over a short period. Then clearly something has to give. But even then it seems better to give a structured proof and leave out (or assign as exercises) some of the low-level details. These details, mostly of a routine, technical character, are more easily reproducible than the high-level ideas and connections left out by even the most detailed linear exposition.

Notes

(1) The following question is usually effective in bringing the formalism-struck student back to common sense: "What is f(3)?"

(2) Many ingenious attempts to deal with these problems through *informal* methods of teaching and learning have become very popular in the mathematics education community. However they have influenced only marginally the mathematical community itself. The paradigm of communication in the formal-linear style still seems to dominate most written, and perhaps even lectured, college-level mathematics.

(3) Another skeleton, indeed, an alternative way of overviewing a complex proof, is the structure diagram (see 4.3). However, this seems to serve best as summary, not introduction.

(4) Sometimes the pivot takes the form of a new *relation* among existing objects, rather than a new object. Also, longer proofs may require several pivots and sub-pivots.

(5) The difference between the two disciplines stemming from the presence of the computer, becomes less significant when we realize that the structural method has been introduced in computer science entirely for the benefit of the human programmer. The computer itself still operates in a strictly linear fashion and, in fact, incurs the extra burden of linearizing the structured program.

(6) More discussion of the case for and against the structural method can be found in [1].

(7) Unable to construct any personal meaning for such a "simple" proof, the student must feel either cheated or stupid.

Acknowledgement

I am grateful to the following people for helpful discussions and comments: Hal Abelson, Andi diSessa, Tony Gardiner, Reuben Hersh, Hana Lifson, Phil Rippon, Abe Shenitzer, David Wheeler.

Reference

[1] U. Leron, Structuring Mathematical Proofs, *Amer. Math. Monthly*, *90* (1983), 174-185.