How algorithmic checkability determines the syntax of formal proofs

Alain Prouté*

June 11, 2023

Abstract

I present a method for discovering (or rediscovering) the structure of formal mathematical proofs, whithout assuming any *a priori* syntactic form for proofs. The key point is that the algorithmic checkability of proofs essentially determines their very syntax.

Contents

1 Introduction

When Isaac Newton gave his famous formula $F = Gmm'/d^2$, he did not *define* gravitation. He only proposed a *model* for gravitation. The reason for this is that gravitation is a natural phenomenon that we don't have to *define* but rather to *discover*, to *describe* or to *modelize*. On the contrary, when mathematicians first introduced the notion of group for example, they *defined* it because the structure of group is nothing other than an abstract game that we can define as being whatever we like. The game may be interesting or not interesting, but it can in no way be *false*, whichever its definition is.

By contrast, a *model* of a natural phenomenon can be false, or at least inaccurate. Newton's model of gravitation has been proved slightly wrong by Einstein. Einstein's general

*Retired from Université Paris-Diderot - France

relativity theory, despite it's considerable success and accuracy (as was also the case for Newton's theory) could very well be proved slightly wrong in the future as well.

Now, mathematical proofs also are a natural phenomenon, because mathematicians are proving mathematical statements since millenaries, and during all that time, they did not try to define the notion of proof.([1]) Despite the growing need in rigour in mathematical proofs, especially since the XIX^{th} century, their structure did not fundamentally change during at least the last two millenaries or so. For example, Archimedes' proofs for the computation of the surface and volume of a sphere are very acceptable today after some cosmetic cleanup. Actually, G. Gentzen, one of the pioneers of proof theory, called "natural deduction" one of his models for the notion of proof, emphasizing the fact that indeed mathematical proofs belong to the domain of natural phenomenons.

Therefore, we *cannot define* the notion of mathematical proof. What we have to do is to *modelize* it, and as what happens in physics, such a modelization can be slightly (or deeply) wrong. How a model such as Newton's formula or Einstein's theory is confirmed "good" is by its adequacy with experiments. The same thing happened with Gentzen's models, which have been confirmed "good" by the fact that writing formal proofs in these models was possible. However, the trust in Gentzen's models has been a bit exagerated to my opinion. In 1980, W.A. Howard [1] wrote an article containing a theorem establishing an isomorphism between one of Gentzen's models and a lambda-calculus with dependent types. This is known as the "Curry-Howard isomorphism". This is indeed a theorem, and actually so trivial that Howard does not even need to prove it. Most proof theorist have thereafter admitted the fact that there is an isomorphism between *mathematical proofs* and this lambda-calculus. However, doing so, they implicitely (and maybe sometimes unconsciously) admit that Gentzen's model is perfectly accurate.

The history of the last four decades or so has however proved that Gentzen's model is not perfectly accurate, as shown for example by what happened to P. Martin-Löf who explained in 2006 in [3] why his 1984 type theory [2], based on the Curry-Howard isomorphism, was actually incorrect. The conceptors of the proof assistant Coq also had to amend their system after they discovered that an elementary theorem of Diaconescu was not provable in this system.

These misadventures have of course led to corrections, but the simple fact of beginning with a concept we are not sure is correct and being obliged to correct it, is unsatisfactory, in particular because we don't know if further corrections will be necessary in the future. This is the reason why I tried to find a method for avoiding such misadventures and directly find the right language for proofs, without assuming the perfect correctness of any model. This project can seem desperate at first glance, but actually, it is feasible as shown in this article.

There are facts which are *indisputable* in mathematics. I call "indisputable" something that mathematicians, and especially *non logician* mathematicians, will admit without discussion. For example, nobody will say that the equality $1 + 1 = 2$, which is taught in schools everywhere in the world, is false. For this reason, I will freely use such equalities. Those you will find below could very well be given as exercices in elementary class rooms. Therefore, they are indisputable.

Here are some comments about this notion of "indisputable". Because, the notion of proof

[1] Aristotle tried to defined the notion of *truth* of a statement, which is a different approach, far from the notion of proof.

and more generally all mecanisms thanks to which we are able to do mathematics are natural phenomenons which manifests themselves only through the activity of mathematicians, we have to find informations about them in the behavior of mathematicians. In some sens, mathematicians are the *custodians* of these mecanisms. Therefore, if all mathematicians agree that $1 + 1 = 2$, it must be "true", whatever the formal meaning of "true" is.

I will also use the fact that mathematical proofs (whichever form they have) are *mechanically checkable*. This is something that mathematicians experience every day. If we have to read an article, and in particular to check that all proofs it contains are correct, we know that we will go through the end of this process, and even that this will require a time somehow proportionnal to the length of the article.([2]) In other words, checking a mathematical proof is an *algorithmic process*. I rewrite this principle below in a more precise form, which will lead to discover the syntactic structure of formal mathematical proofs, and we will see that, modulo cosmetic variations, and with Occam's razor in hand, this problem has only one solution. Hence, what matters is not that proofs are algorithms (a fact that many see as a consequence of the Curry-Howard isomorphism), but the fact that the verification of proofs itself can be performed by an algorithm. By contrast of course, the *search* for a proof of a given statement is not an algorithmic process.

Actually, that mathematical proofs are algorithmically checkable is confirmed by the following facts. If we ask a mathematician to give a definition of the notion of mathematical proof, the mathematician will most often renounce to answer the question. On the contrary, if the question is "what are mathematical proofs good for ?", the mathematican will almost always answer that "they are used for ensuring that a given statement is true, and this in an indisputable way". This is a pragmatic answer whose accuracy is confirmed by our everyday practice of mathematics. Actually, this is clearly another indisputable fact.

But here, we have a small problem which is that we need to know what "true" means for a statement. Actually, "true" (and "false" likewise) are just meaningless for statements in the absolute, because the truth of a statement depends on which *model* we use for interpreting mathematics. This fact has been brilliantly illustrated by K. Gödel and P. Cohen when they proved the relative consistency and independence of the axiom of choice and the continuum hypothesis by constructing alternative models of set theory from a given model of this theory.([3])

Now, if we propose a mathematical statement to a mathematician and just pretend that this statement is true, the mathematician will not accept this assertion if we do not provide a proof of the statement. This behavior can be considered as part of the *ethics* of mathematics, stating that every assertion must be proved before it can be accepted as "true". In other words, mathematicians do not care about truth at all. They care only about provability. This is why the fact that "true" and "false" are meaningless, when there is no reference to a particular model of mathematics, has no importance in everyday mathematics.

As said above, for a non logician mathematician, a proof of a statement is just a tool for ensuring that this statement is "true", whatever "true" means. Since "true" does actually not mean anything, the proof itself, not the truth of the statement, is the real goal of the game. Furthermore, mathematical proofs do not need to have a semantics. The algorithmic

[2] N. Bourbaki would have said : "cela ne requiert qu'une attention en quelque sorte mécanique."

[3] This was also probably what B. Russell has in mind when he said that in mathematics, we don't know what we are talking about (in the absence of a model), and we don't know if what we are saying is true (because it depends on the model).

nature of the verification of proofs implies that they are of a purely syntactic nature. The language of proofs is a language with no signification at all. Of course, this is only the point of view of non logicians. There are logicians who define semantics for proofs in various ways, and I know people (which includes myself) who are playing this game.

Before you read the sequel, you must be fully aware of the fregean dichotomy between the *signifier* (expression of a language) and the *signified* (what this expression represents). This is of primary importance. Indeed, we will essentially work with a singleton set, its unique element and its two subsets, which can look so trivial that nothing interesting can be said about them. However, this unique element and these two subsets can be represented by an infinity of distinct expressions, and this is far from trivial because, as we shall see, these expressions contain all possible statements and all possible proofs of mathematics.

To close this introduction, and because this article is not about mathematics but about metamathematics, it can be seen as not as *indisputable* as an article of actual mathematics. I recognize this fact, and therefore I invite the readers to construct their own opinion on this matter. At least, I hope that readers will agree that I do not rely on the accuracy of any model of the notion of proof, and that my assertions are either truely indisputable equalities, or facts required by the necessity of preserving phaneroalethicity (a notion defined below in order to modelize the notion of algorithmic checkability).

2 Preliminaries

In the language of mathematics, we use three main kinds of expressions. We have *terms* for representing so-called *mathematical objects*, we have *statements* for representing *truth values*, and we also have *proofs*.

An expression, should it be a term, a statement or a proof, can have *free variables*. If it is the case, this expression can be *meaningful* only within a *context* which declares all these free variables. In order to avoid unnecessary complications, I assume that the variables declared in a context have two by two distinct names.

I use the letter Γ for representing an arbitrary context. Each declaration in a context will be written $(x \in X)$ where x is a variable (a.k.a. symbol) and X is *an expression representing* a set. Hence, a sequence of declarations :

$$(x_1 \in X_1) \dots (x_k \in X_k)$$

is a context, where $(x_k \in X_k)$ is the *most recent* declaration. Notice that a declaration can *depend* on *previous* declarations in the sense that X_k can need to be interpreted in the context $(x_1 \in X_1) \dots (x_{k-1} \in X_{k-1})$. In other words, X_k can have free occurrences of the variables $x_1, \dots, x_{k-1}$. The context with no declaration at all is called the *empty context*.

In usual mathematics, a context is not made of declarations only. Indeed, if we have produced a proof p of some statement E, in a context Γ, we can record this within the context for using it later on. Hence, we allow entries of the form $(p \vdash E)$ (read "p proves E") in a context. Such entries can be called "facts". We also sometimes have to make an "assumption" (for example when we prove an implication), so that we can have entries of the form $(\zeta \vdash E)$, where ζ is a symbol instead of a proof of E, but actually plays the same role as a proof of E (because of what "assuming" means).

Quite often, I omit to mention the context, but only when the context is arbitrary and all expressions involved are meaningful in this same context. In any case, everything is always to be interpreted in some context.

I also assume that the reader is familiar with the notion of replacement. Given an expression E, possibly having free occurences of a variable x representing an arbitrary element of some set X, and given a term a representing an element of this same set X, the notation $E[a/x]$, that reads "E where a replaces x", represents the result of replacing all free occurrences of x in E by a. As is well known, this process must be performed with care. Precisely, one can have to *rename* some bound variables in E before performing the replacement in order to avoid any *capture* of a variable that appears free in a.

3 Phaneroalethicity

Consider an algorithm taking as input a statement E and a context Γ within which E is meaningful, and returning one of the following two answers:

- "the truth of E is apparent in the context Γ".
- "I cannot draw any conclusion from the examination of E".

Such an algorithm will be called a "phaneroalethicity algorithm" and denoted **PA**.([4]) A statement E whose truth is apparent according to **PA** in the context Γ is said to be "phaneroalethic" in the context Γ, an assertion that we simply denote $\mathbf{PA}(E,\Gamma)$. Of course, the algorithm is supposed to be designed in such a way that if $\mathbf{PA}(E,\Gamma)$, then E is indeed true in the context Γ (the algorithm does not lie).

The algorithm **PA** is not supposed to be a proof search algorithm. It is only supposed to draw a conclusion by a simple syntactic examination of E, using only the context Γ. In particular, the duration of this examination should be roughly linear in the size of E. A simple example of such an algorithm is the one which always answers that it cannot draw a conclusion, but of course, it cannot be very useful. Nevertheless, the algorithm can easily answer that E is true when asked if E is phaneroalethic in Γ if Γ contains an entry of the form $(p \vdash E)$, should this entry be a fact or an assumption. Notice that if the context contains the *declaration* $(x \in X)$, the *statement* $x \in X$ should also be considered as phaneroalethic in Γ.

In the sequel, we assume that **PA** indeed performs such checks, among other capabilities to be discovered below. We will indeed see that we are naturally led to design an algorithm **PA** in a simple way. For the time being, we just assume that we have such an algorithm.

Of course, the notion of phaneroalethic statement is not a mathematical notion, but a metamathematical notion since it expresses a property of notations, not a property of what these notations represent.

4 Characterization of formal proofs

We need a *singleton* $\mathbf{1} = \{*\}$, i.e. a set with only one element. The main property of this set *and of all its subsets*, that we will use repeatedly below, is that "any two elements of such

[4] From the greek words φανερός (apparent) and αλήθεια (truth).

a set are equal".

Let E be a statement meaningful in some context Γ. Let x be a variable (not declared in Γ). We define the set :

$$W(E) = \{x \in \mathbf{1} \mid E\}$$

which is meaningful in Γ. Notice that because of our conventions and because E is meaningful in Γ, x cannot have any free occurrence in E.

A classical mathematician would just remark that $W(E) = \mathbf{1}$ if E is true and $W(E) = \emptyset$ if E is false, which is of course correct. However, the complexity of what we are doing here appears for example if we consider $W(R)$ where R is Riemann's conjecture, or $W(H)$, where H is the continuum hypothesis. In the first case, we just don't know (at the time I'm writing this) what $W(R)$ is, and we know that we can choose whatever we want for $W(H)$. This phenomenon is not due to the singleton itself which is a rather trivial object, but to the power of the comprehension principle, which as we know is the tool by which some paradoxes can be constructed.

The following lemma, which is essentially a consequence of the fact that x has no free occurrence in E, is trivial but is nevertheless the key for the subsequent characterization of formal proofs.

Lemma 1 *Let E be a statement and p a term representing an element of* $\mathbf{1}$*. Then E is equivalent to $p \in W(E)$.*

Remark : Of course, a context Γ is implicitely assumed, and the sentence "Let E be a statement" means that E is meaningful in Γ. Similarly, p, which can be a huge expression, is supposed to be meaningful in Γ and representing an element of $\mathbf{1}$. The conclusion is also supposed to hold in Γ.

Proof : $p \in W(E)$, i.e. $p \in \{x \in \mathbf{1} \mid E\}$, is equivalent to $E[p/x]$, which is E because x has no free occurrence in E. □

Some mathematicians could not like the above proof because of the use of the replacement. A more conventional proof could go like this :

If we have E, $W(E)$ is $\mathbf{1}$, hence contains $*$. Because $p \in \mathbf{1}$, we have $p = *$, hence $p \in W(E)$. Conversely, if $p \in W(E)$, we have $* \in W(E)$, hence E. □

Now, here is how I want to characterize mathematical proofs. This is a precise version of the main principle asserted in the introduction, namely that proofs are *mechanically checkable* :

> Given a statement E meaningful in a context Γ, any term p representing an element of $\mathbf{1}$, such that the statement $p \in W(E)$ is phaneroalethic in Γ can be considered as a proof of E in the context Γ.

This characterization is justified by the fact that if $p \in W(E)$ is phaneroalethic in Γ, then it contains all the necessary information (together with Γ) for mechanically (i.e. using the algorithm **PA**) establishing that E is true. This is why a term p satisfying the above characterization can undoubtly be called a proof of E (in the context Γ). I do not say that *all* possible proofs of E are characterized that way, but only that terms which are characterized that way *have the absolute right* to be called proofs of E, just because "they do the job".

Of course, this characterization depends on what the algorithm **PA** is able to do. At the limit, if **PA** always answers that it cannot draw a conclusion, no term can be a proof of any

statement. However, as already stated, an algorithm **PA** whose capabilities are what we expect will emerge in a natural way.

Hence, according to this characterization, any proof of any statement is just a term representing the unique element $*$ of $\mathbf{1}$. This is of course equivalent to saying that proofs have no semantics at all, and by the way it entails the proof irrelevance principle (and is actually much more radical, since all proofs of all statements appear to be "equal"). However, the only important point is the fact that $p \in W(E)$ is phaneroalethic, in other words the fact that a proof can be checked algorithmically in essentially linear time in the size of the proof, the statement and the context.

In other words, a "proof checking" algorithm **PC**, checking that a candidate proof p of a statement E in the context Γ is correct, could simply be defined by the equation $\mathbf{PC}(p, E, \Gamma) = \mathbf{PA}(p \in W(E), \Gamma)$.

5 Logical connectives and subsets of the singleton

In order to fullfil the program of discovering the structure and syntax of formal proofs without relying on any postulate, but using only ordinary mathematical reasonning, the equalities below should be considered as assertions that anyone can easily verify. This should not be a problem for mathematicians who of course already know the meaning of logical connectives and the meaning of intersection and union. Actually, proving these equalities could very well be asked to students. One could say that the exercice is somehow "exotic", but in no way incorrect or false.

Recall that logical connectives are $\bot$ (false), $\top$ (true), $\vee$ (or), $\wedge$ (and), $\Rightarrow$ (implies), $\neg$ (not), $\forall$ (for all) and $\exists$ (exists). In two of the equalities below, I wrote $E[x]$ instead of E because I want to stress the fact that E can have free occurrences of x :

$$\begin{aligned}
W(\top) &= \mathbf{1} \\
W(\bot) &= \emptyset \\
W(E \vee F) &= W(E) \cup W(F) \\
W(E \wedge F) &= \textstyle\bigcup_{\zeta \in W(E)} W(F) \\
W(E \Rightarrow F) &= \textstyle\bigcap_{\zeta \in W(E)} W(F) \\
W(\neg E) &= \textstyle\bigcap_{\zeta \in W(E)} \emptyset \\
W(\forall_{x \in X}\ E[x]) &= \textstyle\bigcap_{x \in X} W(E[x]) \\
W(\exists_{x \in X}\ E[x]) &= \textstyle\bigcup_{x \in X} W(E[x])
\end{aligned}$$

As said above, all these equalities are clearly true in classical mathematics (in any context), even if those concerning $\wedge$, $\Rightarrow$ and $\neg$ may look a bit tricky. Nevertheless, they are easy to check if we remember that $W(E)$ cannot have more than one element, that the union of the empty family of subsets of $\mathbf{1}$ is empty, and that its intersection is $\mathbf{1}$.

In the cases of $\wedge$ and $\Rightarrow$, the statement F should have no free occurrence of ζ, because in the left hand member of the equality, there is no declaration of ζ. However, the fact that F should have no free occurrence of ζ is only an impression, because we will see in another section below that F can indeed (and actually *must* in many cases) have free occurrences of ζ and that the declaration of ζ in the left hand member is just "invisible" or "implicit".

Thus, the fact that the above equalities are true in classical mathematics is just indisputable, is not a postulate, not even a kind of definition of the logical connectives.

We can of course forget about the negation since $\neg E$ is just a shortcut for $E \Rightarrow \bot$. We can also consider $\wedge$ and $\Rightarrow$ as special cases of $\exists$ and $\forall$, where the set X is replaced by $W(E)$, and $E[x]$ by F.

6 Introducing types

We are now at the point where being conscious that what we are going to do concerns *notations*, not what these notations represent, is essential. Indeed, the *correctness* of what follows does not depend only on what these notations represent, but on these notations themselves. Unlike what happens in mathematics, two notations representing the same thing are not necessarily interchangeable. One can easily check that this phenomenon is just a consequence of the need for phaneroalethicity.

In the sequel, I use the word "represents" instead of "is" in order to remind the reader that the things in question are expressions, not mathematical objects.

Let $(A_i)_{i\in I}$ represent a family of subsets of $\mathbf{1}$, where I represents an indexing arbitrary set. As usual, this is relative to a context Γ, and what the word "family" actually means is that I is meaningful in Γ, as an expression representing a set, and that A_i is meaningful in the context $\Gamma(i \in I)$ as an expression representing a subset of $\mathbf{1}$.

Hence, what matters now is how we produce phaneroalethic statements of the form $a \in \bigcap_{i\in I} A_i$ and $a \in \bigcup_{i\in I} A_i$, possibly from other phaneroalethic statements. It will be immediately visible that the expressions we introduce for representing an element in a subset of $\mathbf{1}$ are meaningful only if this subset of $\mathbf{1}$ is itself represented in a specific way. This is just a consequence of the necessity to produce phaneroalethic statements. For this reason, expressions representing subsets of $\mathbf{1}$ will not be called "subsets if $\mathbf{1}$" but "types", and a statement of the form $a \in A$, where A is a type, will be read as "a is of type A". We will also use the shortcut $p \vdash E$ (read "p proves E") for $p \in W(E)$.

We will see below that at some point we need an algorithm computing the type of a term in a given context, and that the need for this algorithm has an influence on the design of the syntax of the language of proofs. It requires that the terms we are on the point to introduce for representing proofs, must contain more informations than they would otherwise need to contain. This algorithm is denoted **TypeOf** below.

7 Intersections

Let $(A_i)_{i\in I}$ represent a family of types, as in the previous section.

Assume that the statement $a_i \in A_i$ is phaneroalethic in $\Gamma(i \in I)$ (intuitively, phaneroalethic in Γ for all i in I). Then, we can enhance the algorithm **PA** so that the statement $(a_i)_{i\in I} \in \bigcap_{i\in I} A_i$ is phaneroalethic in Γ. Indeed, from the expression $(a_i)_{i\in I} \in \bigcap_{i\in I} A_i$ and the context Γ, we can (algorithmically) reconstruct the expression $a_i \in A_i$ and the context $\Gamma(i \in I)$. Furthremore, since $a_i \in A_i$ is phaneroalethic in $\Gamma(i \in I)$, hence true, we have $* \in A_i$ for all $i \in I$. This implies $* \in \bigcap_{i\in I} A_i$, so that $(a_i)_{i\in I} \in \bigcap_{i\in I} A_i$ is true in Γ, as required for being phaneroalethic.

Notice that we could not have written $a_i \in \bigcap_{i\in I} A_i$ instead of $(a_i)_{i\in I} \in \bigcap_{i\in I} A_i$ because this expression must be meaningful in a context which does not declare i.

However, we will prefer the notation $(i \in I) \mapsto a_i$ to the notation $(a_i)_{i\in I}$ for the reason that the declaration $i \in I$ comes *before* its scope a_i. There are situations where a_i is a huge expression, and in this case the syntax $(a_i)_{i\in I}$ would become unreadable. It also happens, as we shall see below, that the sentences representing this construction in the vernacular language of mathematical proofs have the declaration at the beginning.

The fact that we use a notation generally used for representing functions, for representing the unique element of $\mathbf{1}$ may be disturbing. However, you have seen above that a notation of this kind is necessary to ensure phaneroalethicity. By the way, the choice of this notation will also make it clear that proofs can be seen as algorithms, but this is not the point for what we are doing here.

This gives us a new rule participating to the definition of the algorithm **PA**. Indeed, we have :

$$\mathbf{PA}((i \in I) \mapsto a_i \in \bigcap_{i\in I} A_i, \Gamma) \quad \text{whenever} \quad \mathbf{PA}(a_i \in A_i, \Gamma(i \in I))$$

The reader will be able to easily verify that this rule together with the other rules discussed below and with what has already been said about how **PA** can extract facts from the context, constitute a *recursive well founded computer program* defining the algorithm **PA**.

Because of the equalities in section 5 and our notational conventions, the above rule specializes into the two rules :

$$\begin{array}{rl} \mathbf{PA}((x \in X) \mapsto p_x \vdash \forall_{x\in X}\, E[x], \Gamma) & \text{whenever} \quad \mathbf{PA}(p_x \vdash E[x], \Gamma(x \in X)) \\ \mathbf{PA}((p \vdash E) \mapsto q_p \vdash E \Rightarrow F, \Gamma) & \text{whenever} \quad \mathbf{PA}(q_p \vdash F, \Gamma(p \vdash E)) \end{array}$$

which are among our familiar proof rules. Similar particularizations can be written from the other rules to come below, but this is left to the reader.

It can be the case that we have an element x in $\bigcap_{i\in I} A_i$, and that we want to consider it as an element of A_u for some u that is known to be an element of I. We have to design a notation p that will make the statement $p \in A_u$ phaneroalethic in Γ, under the hypothesis that both $x \in \bigcap_{i\in I} A_i$ and $u \in I$ are phaneroalethic in Γ. In other words, this notation must contain enough informations so that these two statements can be reconstructed. Clearly, both x and u must be part of it.

Here, we have a problem because the expression A_i itself cannot be recovered from A_u, which is supposed to be $A_i[u/i]$. Indeed, replacement *is not* a reversible operation. The set I itself cannot be recovered from x, u and A_u, except if we are able to *compute the type* of x. This is why we have assumed above that we have the algorithm **TypeOf**.

Hence, things will go as follows. From x and Γ, we can get the type of x, which *must* have the form $\bigcap_{i\in I} A_i$. From here, we have everything needed in order to construct the statements $x \in \bigcap_{i\in I} A_i$ and $u \in I$.

Hence, we adopt the notation $x(u)$ (read "x applied to u"), and we have :

$$\mathbf{PA}(x(u) \in B, \Gamma) \quad \text{whenever} \quad \left\{ \begin{array}{ll} \mathbf{TypeOf}(x, \Gamma) & \text{has the form} \quad \bigcap_{i\in I} A_i \\ B & \text{is identical to} \quad A[u/i] \\ \mathbf{PA}(u \in I, \Gamma) & \end{array} \right.$$

Notice that the fact that $\mathbf{TypeOf}(x, \Gamma)$ has the form $\bigcap_{i\in I} A_i$ entails that the statement $x \in \bigcap_{i\in I} A_i$ is phaneroalethic in Γ, which means that $*$ belongs to all A_i, hence to A_u, so that the statement $x(u) \in A_u$ is true.

The precise meaning of the expressions "has the form" and "is identical to" is just "is alpha-equivalent to", i.e. syntactically identical up to a renaming of bound variables.

8 Unions

The case of unions is a bit more tricky than the case of intersections. Again, we consider a family $(A_i)_{i\in I}$ of types.

If $u \in I$ and $a \in A_u$ are phaneroalethic in Γ, A_u contains $*$, so that $\bigcup_{i\in I} A_i$ also contains $*$. Hence, from these, we have to design a statement of the form $p \in \bigcup_{i\in I} A_i$ phaneroalethic in Γ. Clearly, p should make use of u and a, so that we use the notation $\langle u, a\rangle$. However, this is not enough, because **TypeOf** will not be able to compute the type of $\langle u, a\rangle$, again for the reason that A_i cannot be computed from A_u (which is $A[u/i]$). Hence, we use the "enriched" notation $\langle u, a\rangle_{i\in I, A_i}$ for this metamathematical pair, that I want to call a "Heyting pair",([5]) and we can enhance the algorithm **PA** with the rule :

$$\mathbf{PA}(\langle u, a\rangle_{i\in I, A_i} \in \bigcup_{i\in I} A_i, \Gamma) \quad \text{whenever} \quad \begin{cases} \mathbf{PA}(u \in I, \Gamma) \\ \mathbf{PA}(a \in A_i[u/i], \Gamma) \end{cases}$$

Notice that the statements $u \in I$ and $a \in A_i[u/i]$ can be algorithmically reconstructed from the statement $\langle u, a\rangle_{i\in I, A_i} \in \bigcup_{i\in I} A_i$.

Despite the fact that $\langle u, a\rangle_{i\in I, A_i}$ is a notation for $*$, it is a *pair* from the syntactic viewpoint. Hence, we can possibly make use of the two canonical projections π_1 and π_2 for extracting either u or a from this pair. However, π_1 is not well defined. Indeed, consider two notations $\langle u, a\rangle_{i\in I, A_i}$ and $\langle v, b\rangle_{i\in I, A_i}$, both representing $* \in \mathbf{1}$, but such that $u \neq v$ (which can happen if both A_u and A_v contain $*$). Then we have at the same time $\pi_1(*) = u$ and $\pi_1(*) = v$ which is contradictory. This is why this operator π_1 is never used in mathematics, except in some situations described below where π_1 happens to be nevertheless well defined. Apart from these cases, its use is limited to some *metamathematical* operations, such as the one that consists in examining an existence proof in order to try to construct a term representing the thing whose existence was proved. This operator π_1 is known as the "choice operator".

The second projection π_2 is less problematic, because $\pi_2(\langle u, a\rangle_{i\in I, A_i}) = a \in \mathbf{1}$. No such contradiction as above can arise.

9 Legitimate uses of the choice operator

As said above, there are cases where the use of π_1 is legitimate. The first one is when a statement of the form $p \in W(\exists!_{x\in X}\ E[x])$ is phaneroalethic in Γ (where $\exists!$ means as usual "there exists a unique"). In this case, there is one and only one x in X such that $W(E[x])$ contains $*$, and the above contradiction cannot be carried out, so that $\pi_1(p)$ is a legitimate (well defined) expression. The operator π_1 in this case is then called the "description operator".

The second case is when π_1 is used within an expression representing the unique element of $\mathbf{1}$. In this case the problem disappears, since whatever the value produced by π_1 is,

[5] A reference to one of the rules of the so-called "Heyting semantics".

the result will always be the same one, namely $*$. For example, an expression such as $\langle \pi_1(x), \pi_2(x)(y) \rangle_{i \in I, A_i}$ is well defined because it always represents $*$, whatever the value of $\pi_1(x)$ is. This is actually the same thing as the well known fact that even if we have to *choose a representative* of some equivalence class in order to construct a function whose domain is a quotient set, what we are constructing is well defined provided that it does not depend on the choice of this representative.

However, despite the fact that a representative can be "chosen", it must still be a meaningful object, i.e. be represented by a meaningful expression. This is not what happens in the statement below:

$$\begin{array}{l}(\zeta \in W(\forall_{x \in X}\ \exists_{y \in Y}\ E[x,y])) \mapsto \\ \quad \langle (x \in X) \mapsto \pi_1(\zeta(x)), (x \in X) \mapsto \pi_2(\zeta(x)) \rangle_{f \in Y^X, W(\forall_{x \in X}\ E[x,f(x)])} \\ \quad\quad \in W((\forall_{x \in X}\ \exists_{y \in Y}\ E[x,y]) \Rightarrow (\exists_{f \in Y^X}\ \forall_{x \in X}\ E[x,f(x)]))\end{array}$$

that could appear as phaneroalethic in any context if we are not careful enough. Indeed, despite the fact that $\pi_1(\zeta(x))$ is meaningful whatever the expression $\zeta(x)$ is, the expression $(x \in X) \mapsto \pi_1(\zeta(x))$ can be meaningless because it can be the case, for some x in X, that there are two distinct elements y_1 and y_2 of Y such that both $E[x,y_1]$ and $E[x,y_2]$ are true. In this case, $\pi_1(\zeta(x))$ can be either y_1 or y_2, which makes the function $(x \in X) \mapsto \pi_1(\zeta(x))$ not well defined despite the fact that both y_1 and y_2 are well defined.

This example is the "constructive proof" that Martin-Löf gave of the axiom of choice in his type theory [2] based on the Curry-Howard correspondance, that he recognized as incorrect in [3].

There is a well known method for avoiding this trap, which is to completely forbid the use of π_1 and introduce instead an explicit description operator and an explicit syntactic formulation for using the first element of a Heyting pair for constructing a proof, i.e. for producing an expression representing $*$. It happens that both also have a very explicit formulation in the vernacular language of mathematical (informal) proofs.

As for the description operator, that we now denote δ, it can be applied to a proof of any statement of the form $\exists!_{x \in X}\ E$. In the vernacular, the operator seems to be applied to this statement instead of a proof of it, since the formulation is "the unique x in X such that E". However, this sentence will never be used if it is not clear that the statement $\exists!_{x \in X}\ E$ is true at this point of the text, so that it is indeed applied to a proof of it, even if it is a supposed (declared) proof resulting from an assumption. Hence, we can enhance the algorithm **PA** as follows :

$$\left\{\begin{array}{l}\mathbf{PA}(\delta(p) \in X, \Gamma) \\ \mathbf{PA}(E[\delta(p)/x], \Gamma)\end{array}\right. \quad \text{whenever} \quad \mathbf{PA}(p \in W(\exists!_{x \in X}\ E), \Gamma)$$

For the other problem, the formulation in the vernacular is the usual one that allows to use an existence hypothesis of the form $\exists_{y \in Y}\ E$:

$$\text{Let } y \in Y \text{ such that } E\ (\zeta),\ p$$

where p is a proof of the statement to be proved, but which can use the declaration $y \in Y$ and the assumption $\zeta \in W(E)$. In our formal language of proofs, we can use a notation such as :

$$\{p, y \in Y, \zeta \in W(E), q\}$$

as a proof of a statement C in the context Γ, where p is a proof of $\exists_{y\in Y}\ E$ in the context Γ and q a proof of C in the context $\Gamma(y \in Y)(\zeta \in W(E))$.

In order to fully justify the above formulation, we remark that y represents a meaningful object, even if this object is "arbitrarily chosen", and that q, constructed using y, does not depend on y since it belongs to $\mathbf{1}$.

Notice that in this formulation, y and ζ play the roles of $\pi_1(p)$ and $\pi_2(p)$. However, the fact that y is just a symbol makes it impossible to explicitely depend on any other variable, so that writing something like $(x \in X) \mapsto y$ cannot define anything other than a constant function, which is then correct for the reason that the two sets Y and $Y^{\mathbf{1}}$ are canonically isomorphic. It should also be noted that because the scope of the declaration of y is lexically limited to $W(E)$ and q, the first component of the Heyting pair represented by p cannot be used for "exporting" a mathematical object outside the above expression.

Now, how do we check that a statement of the form

$$\{p, y \in Y, \zeta \in W(E), q\} \in A$$

is phaneroalethic in a given context Γ ? From this statement, we can mechanically construct the two statements:

$$\begin{array}{rcl} p & \in & W(\exists_{y\in Y}\ E) \\ q & \in & A \end{array}$$

We just have to check that $p \in W(\exists_{y\in Y}\ E)$ is phaneroalethic in the context Γ, and that $q \in A$ is phaneroalethic in the context $\Gamma(y \in Y)(\zeta \in W(E))$. Notice that doing this, we use exactly once each information contained in the statement $\{p, y \in Y, \zeta \in W(E), q\} \in A$, except ζ, but ζ is not significant because it represents $*$, i.e. the only thing that matters is the fact that $W(E)$ contains $*$, but this is a consequence of the fact that $p \in W(\exists_{y\in Y}\ E)$.

It is (I think) a remarkable fact that the above two formulations of the vernacular are used since centuries. What it means, probably beyond the fact that humanity has some kind of genius, is that proving is a very natural thing. This is consistent with the fact that mathematicians are able to write read and check proofs without even wanting to define what they precisely are.

10 A simple example of a formal proof

The statement below is phaneroalethic in any context Γ :

$$\begin{array}{l} (\zeta \vdash \exists_{x\in X}\ \forall_{y\in Y}\ E(x,y)) \mapsto \\ \qquad (y \in Y) \mapsto \{\zeta, x \in X, \theta \vdash \forall_{y'\in Y}\ E(x,y'), \langle x, \theta(y)\rangle_{x\in X, W(E(x,y))}\} \\ \qquad\qquad \vdash (\exists_{x\in X}\ \forall_{y\in Y}\ E(x,y)) \Rightarrow \forall_{y\in Y}\ \exists_{x\in X}\ E(x,y) \end{array}$$

(which means that the first two lines are a proof of the last line). Notice that we have introduced a variable y' in the second line because at this point we are in the scope of the declaration of y at the beginning of this line, and we want to respect the rules stated in section 2.

The algorithm **PA** would check this statement after eliminating all logical connectives according to the equalities in section 5. The reader can simulate this computation with a little bit of patience. However, the first two lines are easily translated into the vernacular language as follows :

> Assume that $\exists_{x\in X}\ \forall_{y\in Y}\ E(x,y)$ (ζ). Let $y \in Y$. By (ζ), there is an $x \in X$ such that $\forall_{y'\in Y}\ E(x,y')$ (θ). By (θ), we have $E(x,y)$.([6]) Thus, we have proved that $\forall_{y\in Y}\ \exists_{x\in X}\ E(x,y)$.

It should be noticed that the vernacular lacks some formulations, and that some periphrasis must be used instead. In particular, there is nothing corresponding to our "applicative" notation $x(u)$ defined in section 7, and also nothing for representing Heyting pairs. Hence, the formal proof $\langle x, \theta(y)\rangle_{x\in X, W(E(x,y))}$, is something that the reader of an informal proof must synthesize mentally. The sentence "Thus, we have proved that ..." of course is an inducement for this synthesis.([7])

The variable x declared in the second line (within the curly brackets) represents $\pi_1(\zeta)$, and the hypothesis $\forall_{y'\in Y}\ E(x,y')$ is actually proved by $\pi_2(\zeta)$, here represented by θ.

The fact that $\exists_{x\in X}\ \forall_{y\in Y}\ E(x,y)$ entails $\forall_{y\in Y}\ \exists_{x\in X}\ E(x,y)$ generally does not deserve a proof in mathematical texts. It is most often considered as "obvious", and indeed, a well designed proof search algorithm can easily find proofs of implications of this kind. This example shows that a formal proof has much more details than an informal proof, and consequently that the more or less unconscious work performed by the brain of a mathematician when reading an informal proof is an important part of the work.

It should also be obvious from the above example that the language of formal proofs is a functional programming language. Proofs are essentially made of (metamathematical) pairs and functions. In other words, proofs are indeed algorithms.

The language of proofs established by the method of this article is actually the same as the lambda-calculus with dependent types refered to in the introduction, with the difference that the operators $\bigcup$ and $\bigcap$ replace the operators $\coprod$ and $\prod$. This has absolutely no impact for example on the fact that the usual rewriting rules apply to the system established here, and that the resulting rewriting system is nœtherian and confluent. The only decisive difference is that in the case of the lambda-calculus, a type of the form $\coprod_{i\in I} T_i$ can very well have several distinct elements even if all T_i have at most one element. On the contrary $\prod_{i\in I} T_i$ always has at most one element when all T_i have at most one element. Hence, a simple way of making the lambda-calculus with dependent types equivalent to the system presented here is to replace $\coprod_{i\in I} T_i$ by $(\coprod_{i\in I} T_i)/\simeq$, where $\simeq$ is the equivalence relation which identifies any two elements of $\coprod_{i\in I} T_i$. But this is precisely equivalent to introducing an axiom stating the indiscernibility of proofs. What this article proves (to my opinion) is that indiscernibility of proofs is not an option, but a mandatory characteristic of any correct formalization of *usual* mathematics.

I know that this opinion is contrary to the main stream opinion among proof theorists, which is why I have been criticized at the end of almost any of my conferences. However, such critics seem unjustified because it is of course not forbidden to propose semantics for proofs which will contradict the indiscernibility of proofs. This can be done for various very good reasons. There is simply no reason that there is one and only one possible semantics for proofs. Many different semantics can very well cohabit in mathematical logic, in the

[6] $E(x,y)$ is $\theta(y)$.

[7] The mathematician does not of course consciously synthesize the formal expression $\langle x, \theta(y)\rangle_{x\in X, W(E(x,y))}$, but his/her brain constructs something that is equivalent, otherwise proofs would not be indisputable. Actually, what happens here is the usual reasonning : "What I am doing is valid for any $y \in Y$, I have an $x \in X$ and $E(x,y)$ is true. Hence, I have proved $\forall_{y\in Y}\ \exists_{x\in X}\ E(x,y)$", which is just a *résumé* of the second line of our expression.

same way as mathematics can be interpreted in a variety of models.

The following remark has been made by several people : if all proofs (of the same statement, say) are "equal", then how can a proof assistant correctly work ? I answer by a simple comparison. Instead of a proof assistant, which is a huge and complex machinery, consider a simple calculator just able to add positive natural integers. If you enter $1+1$ into the calculator, it will replace it by 2, because it is able to undertand the difference between $1+1$ and 2, which is required for performing this computation. Nevertheless, $1+1$ is still "equal" to 2, which is actually the reason why the calculator dares to replace $1+1$ by 2. In other words, we should not confuse the signifiers (expressions) and the signified (what expressions represent). Similarly, a proof assistant based on a system assuming the indiscernibility of proofs, is of course able to perceive the difference between two proofs, and actually, the system does not care about equality of proofs, but it cares about one of its consequences which is that the choice operator π_1 must not be used, at least not for any strictly mathematical purpose. The axiom of choice can be used in mathematics, but using it *does not imply* that we use the choice operator.

11 Conjunction, implication and the description operator

I have characterized $\wedge$ and $\Rightarrow$ as follows :

$$\begin{array}{rcl} W(E \wedge F) & = & \bigcup_{\zeta \in W(E)} W(F) \\ W(E \Rightarrow F) & = & \bigcap_{\zeta \in W(E)} W(F) \end{array}$$

and we have already noticed that the notation $E \wedge F$ is not formally coherent with $\bigcup_{\zeta \in W(E)} W(F)$, because assuming that both expressions are to be interpreted in some context Γ, F appears to be interpretable in Γ in the left hand expression and in $\Gamma(\zeta \in W(E))$ in the right hand expression, hence actually *not interpretable* in Γ as soon as it has at least one free occurrence of ζ.

This is not a paradox but the consequence of the fact that the declaration and use of ζ in the expression $E \wedge F$ are not reflected by the syntax. In other words, they are just *invisible*.

This invisibility is a consequence of the fact that any two elements of $\mathbf{1}$ are equal. Indeed, because $*$ is alone in $\mathbf{1}$, it does not need to have a name. It can be unambiguously designated as "the unique element of $\mathbf{1}$", in the same way as if you are alone on a (previously) desert island, you don't need a name, since you are just "the one on the island".

The following elementary example shows that ζ can indeed be *invisibly* declared and used in an expression such as $E \wedge F$. Let A be a subset of $\mathbb{N}$, and consider the statement :

$$A \neq \emptyset \wedge \inf(A) = 0$$

where $\inf(A)$ is the greatest lower bound (infimum) of A. Since only non empty subsets of $\mathbb{N}$ have an infimum, the "hypothesis" $A \neq \emptyset$ is required for $\inf(A)$ to be meaningful. In other words, this hypothesis is used within the very expression $\inf(A)$. In order to make all these invisible things visible, the above statement could be written :

$$(\zeta \vdash A \neq \emptyset) \wedge \inf[\zeta](A) = 0$$

which, because of the equalities in section 5, is just an alternative notation for :

$$\exists_{\zeta \in W(A \neq \emptyset)} \ \inf[\zeta](A) = 0$$

stressing the fact that ζ is declared in the left hand operand of $\wedge$ and used by $\inf(A)$ in the right hand operand of $\wedge$. By the way, this also shows that in the case of this example, the conjunction is not commutative.

How $\inf(A)$ makes use of ζ is through the description operator. Indeed, $\inf(A)$ is defined in usual mathematics by a sentence beginning by "the unique element of $\mathbb{N}$ such that ...", and such a sentence is accepted only if the statement "there is a unique element of $\mathbb{N}$ such that ..." can be assumed true at this point. This is known as the "principle of description", and this is nothing other than how the vernacular language of mathematics handles the description operator. Actually, the description operator is the only way to extract a term (representing an actual mathematical object other than $* \in \mathbf{1}$) from a proof.

Here is how $\inf(A)$ is precisely constructed. We first prove, using the hypothesis $A \neq \emptyset$ that there exists a unique element i in $\mathbb{N}$ that has the property of being an infimum of A. This proof p is that of an existential statement, i.e. it represents an element of a set of the form $W(\exists!_{i \in \mathbb{N}} \ \ldots)$. Since elements of such a set are represented by Heyting pairs, we can apply π_1 $(= \delta)$ to this proof, which yields the infimum of A. Notice that this can be done even if this proof is not in the form of a Heyting pair, but in this case the expression $\pi_1(p)$ is just not reducible, which just means that the infimum cannot be computed. However, if this happens in the empty context, one can prove that the normal form of p is indeed an Heyting pair and consequently that the infimum can actually be computed.

Notice that the usual boolean truth table for the conjunction is of no help for understanding this conjunction. The above discussion also applies *mutatis mutandis* to the implication.

12 Final remarks

It should now be clear that the notion of mathematical proof has much more to do with the notion of algorithm than with the (meaningless) notion of truth.

When he proposed his natural deduction model for proofs, Gentzen was guided by how mathematicans are writing proofs. In other words, this model is the result of the *observation* of the behavior of mathematicians. The approach proposed in the present article is not based on how mathematicians write proofs, but on the single fact that proofs are algorithmically checkable. Furthermore, I have shown how this fact essentially determines the very syntax of formal proofs. This may explain why proofs are what they are, because the purpose of a proof is to *convince* people (including oneself) of the truth of a statement. Proposing an explanation that everyone can mecanically check is undoubtly one of the best methods for acheiving this goal.

Of course, the resulting syntax is also highly dependent on our set of logical connectives. If we were to use another set of connectives, the language of proofs would probably be different. However, our set of logical conectives can be justified by the fact that the operations $\bot$ and $\top$ are respectively left and right adjoint (with respect to the deduction preorder relation, whatever this relation is) to the action of forgetting a statement, that similarly $\vee$ is left adjoint to the duplication operation of a statement, and that $\exists$ and $\forall$ are respectively left and right adjoints to the action of declaring of variable. The forgetful, duplication and variable declaration actions are clearly more fundamental operations than the logical connectives, and they are very natural. This is probably the path humanity followed for "inventing" mathematics and in particular the notion of proof. It should be noticed that

the notion of adjoint functors (adjoint increasing maps in this case) is one of the most used processes for defining new concepts in mathematics, and that it was used in a natural and instinctive way even before this notion was explicitely introduced.

It may look surprising that despite the fact that we used only *classical* mathematical reasoning, accepting in particular the fact that the singleton has only two subsets, what we discover is a purely intuitionistic proof system. But this is just because all proof rules we use in classical mathematics are actually intuitionistic rules. Classical mathematics is reached by adding the axiom of choice, which is not a proof rule, to our arsenal.

The method exposed in this article also has the advantage of avoiding assuming any *a priori* syntax for proofs, and therefore avoiding the problems recalled in the introduction. Shouldn't it be how proof theory should be taught ?

References

[1] **W. A. Howard** : *The formula-as-type notion of construction.* in J. R. Hindley and J.P. Seldin, To H. B. Curry Essays on combinatory logic, lambda-calculus and formalism. Academic Press 1980.

[2] **P. Martin-Löf** : *Intuitionnistic type theory.* Studies in Proof Theory, Bibliopolis, Naples, 1984.

[3] **P. Martin-Löf** : *100 Years of Zermelo's Axiom of Choice : What was the Problem with It ?* The Computer Journal Volume 49 Issue 3, May 2006. Pages 345-350