SLAC-PUB-3334
May 1984
(T/E)

ON THE LOGICAL FORM OF PRIMITIVE RECURSIVE FUNCTIONS*

Christoffer Gefwert

Academy of Finland, Helsinki

and

Stanford Linear Accelerator Center

Stanford University, Stanford, California 94305

Submitted to Journal of Philosophical Logic

* Work supported in part by the Department of Energy, contract DE-AC03-76SF00515 and by the Academy of Finland.

...the difference between man and the
presently available computers: We exercise
judgment, and the computers do not.

Errett Bishop

1. Introduction

As an example of a philosophical investigation we shall engage in a *semantical* explanation of the syntax and semantics of mathematical Language based on notes made of a lecture (unpublished) given by Per Martin–Löf in Oxford 1975 being an extension of a, likewise, unpublished paper with Peter Hancock as co-author.[1] The author expresses his gratitude to professor Martin–Löf and professor Georg Kreisel for making some clarifying comments. Without these papers and the helpful comments this paper would never have been written.

In order to make the investigation more comprehensive we shall use as an example the terminated program $0 + 2 = 2$ which, as Wittgenstein says, is "a rule of syntax".[2] We are going to show that when primitive recursive functions are understood, this is the case in virtue of a *strict finitist* conception of philosophy. Such a view has been attributed to Wittgenstein.[3] To semantically evaluate the program $0 + 2$ we have to recover the logical depth grammar for primitive recursive functions, which,when codified, evaluate this *person program* and uniquely determines its use as an *arithmetical sentence.* This is more or less *verbatim* in accordance with Errett Bishop's insight that, since a "computer is lacking in judgment, the theorems of constructive mathematics do not in general represent computer programs. They represent person programs, which in some instances can be transformed into computer programs and in other instances cannot".[4]

This is, perhaps, the right place to spell out how we regard Brouwer's position vis-a-vis the person program idea being used here. This question has been investigated by Göran Sundholm. The view adopted in his exposition is essentially the same as adopted here. It is therefore useful to quote Sundholm at length: "Indeed, when the theories of constructions are given their meaning-explanatory role the separation will immediately arise between the mathematical activity reported in the constructivist language and the meaning-explanations in the secondary reflective activity. Now for Brouwer, as witnessed by his philosophical writings (1929) and (1948), mathematical reflection (reflection on the mathematical activity) is itself a part of mathematics: the secondary activity is indeed in the first. A nice example of this is given in the remarkable self-reflexivity of his proof of the Bar-theorem. An immediate consequence of such a view on the primary and secondary activities would be that meaning-theoretical considerations are ruled out; as soon as we have a language there arises questions of meaning over that language".[5] Therefore Brouwer's *first act of intuitionism* seems inevitable within this framework:[6]

> *Completely separating mathematics from mathematical language and hence from the phenomena described by theoretical logic, recognizing that intuitionistic mathematics is an essentially langueless activity of the mind...*

In the person program idea the use of language forces the separation of activity and reflection over that activity, whereas for Brouwer activity and reflection are one. So the activity must be essentially *languageless*. For Wittgenstein, on the other hand, such an attitude expressed a profound misunderstanding concerning the nature of human thought. This is well documented in his argument against the possibility of a *private language*, being one of the corner-stones of his philosophical doctrines, and which is adopted in the person program idea.

It might be useful to compare a person program with the output of an abstract machine of a certain kind presented by Alan Turing.[7] A Turing machine can be understood as an "ideal" machine that provides a finite output of instructions.[8] By analogy, a person program can be understood as the output of a generative "Turing machine" producing a finite string of rules by canonical steps determining *how* a task is to be terminated. Furthermore, according to Crispin Wright, these rules, being "grammatical propositions—what we ordinarily take for necessary truths—must all be in some sense both independent of each other and incapable of conflict with fact".[9] In short: a person program is constituted by canonical expressions and sentences, the informal reading of which provides *complementary* propositions expressing knowledge of meaning. These grammatical propositions, provides instructions on *what* to do, in virtue of Wittgenstein's principle.[10]

For pedagogical reasons we reverse the actual order a person program is executed. If we were to engage in the philosophical investigation of primitive recursive functions by canonical steps as such, it would be almost impossible for the reader to follow what is going on. We have therefore chosen to tinker a bit on the actual order of introducing a person program. The result is that the *Begriffschrift* (translation manual), *i.e.*, the logical depth grammar, is introduced in the later part of this paper, whereas it obviously should be formulated *first* if it were to be faithful to Wittgenstein's principle: one can only judge (assert) the sense of a proposition *once*.[11]

2. On Explicit and Implicit Knowledge

The reason for our engagement in a person program in order to provide the logical depth grammar for primitive recursive functions is that "primitive recursive functions are so absolutely basic and foundationally unproblematic (or rather, just as problematic as the natural number sequence), that they are generally accepted as a starting point for meta-mathematical research".[12] It is precisely this insight that the *use* of primitive recursive functions are taken for granted *vis-à-vis* meaning that we want to correct and supplement. We want to show *how* one is to understand (use) natural numbers and primitive recursive functions. The use of primitive recursive functions cannot, when we contemplate on this question more seriously, be taken for granted. If this were the case one would never be able to use them in a *correct* way. And still: this is exactly *what* we do. And *this* is the philosophical problem: to understand (know) *what* we do with them. One must be able to explicitly show how they are to be used if they are to be used *correctly:* this is the task of showing the logical form of primitive recursive functions. When we engage in providing a philosophical investigation of the logical depth grammar of natural numbers and primitive recursive functions we *dissolve* problems concerning their meaning (use).

In general a depth grammar, showing the logical form of a Language, is constituted by a *formal* and a *non-formal* part.[13] The investigation we are to engage in, however, only concerns formal rules. Recall that the expression $0 + 2$ essentially is a program that provides information of its own evaluation.[14] It provides information of its own logical depth grammar. What we know when we have proved a statement is its truth, and what we know when we have understood a statement is its meaning. That a person knows the truth of a proposition manifests itself in his ability to prove (compute) it. In the terminology used by Michael Dummett, knowledge of truth of a statement is *verbalizable* knowledge, *i.e.*, "when the speaker is able to *state*, in other words, what the condition is for the truth of a sentence".[15] We *use* the depth grammatical rules in a deductive argument, and we verbalize them in the activity of proving (computing).

Thus the task in a philosophical investigation is to recover (codify and explain) these rules which we, implicitly, use in practice. We use the notion "implicit" as standing for *implicit knowledge* in the way Dummett uses the term when he states that "...knowledge which, in general, constitutes the understanding of the language of mathematics must be implicit knowledge. Implicit knowledge cannot, however, meaningfully be ascribed to someone unless it is possible to say in what the manifestation of that knowledge consists: there must be an observable difference between the behaviour or capacities of someone who is said to have that knowledge and someone who is said to lack it".[16] If we want to be more precise, we can formulate this insight like Dag Prawitz, and say that "we have an *adequacy condition* on a sentence A where B_A is a kind of behavior counted as a sign of grasping the meaning of A, *i.e.*,[17]

$$\text{if P knows the meaning of A, then P shows behavior } \mathrm{B_A}\text{"} .$$

Knowledge of the meaning of a linguistic expression can only manifest itself in the ability to use the expression correctly. Thus there is no one linguistic act, like that of proving (computing) a theorem, which conclusively shows that we know the meaning of a linguistic expression. In particular, it is not in our ability to explain its meaning that our understanding of a linguistic expression manifests itself. If that were the case, almost no mathematician could be said to understand the primitive notions of mathematics. As Dummett says:"...the understanding of an expression cannot, in general, be taken to consist in the ability to give a verbal explanation of it, and hence must constitute implicit knowledge of its contribution to determining the condition for the truth of a sentence in which it occurs; and an ascription of implicit knowledge must always be explainable in terms of what counts as a manifestation of that knowledge, namely the possession of some practical capacity".[18] As far as we are concerned the manifestation of implicit knowledge takes the form of an ability to successfully terminate a *task*. The task in this case is to be able to successfully compute the program $0 + 2$.

As we said above, we regard an expression like $0 + 2$ essentially as a program. When we engage in a philosophical investigation, what do we have to recover from the program in order for the investigation to terminate? We have to recover the regulative rules, so "If you want to know what $2 + 2 = 4$ means, you have to ask how we work it out".[19] We have to recover the formal (canonical) rules constituting the person program. When we engage in this we do it according to the proposition-as-rules idea. That is, we follow Wittgenstein's principle, as explained elsewhere.[20] This is to say that when we codify formal (and non-formal) rules, the codification,being a proposition, shows the sense of the expressions and sentences respectively. It shows their *logical form*. And the sense is shown by a canonical step in a canonical instantiation, *i.e.,* codification. Or, alternatively, we can say that when we engage in codifying rules we are creative (metaphysical) subjects, that is, we *participate* in the person program.[21]

3. On Wittgenstein's Imperative

The explanation of primitive recursive functions we are to engage in is typical of how substance can be given to Wittgenstein's idea that meaning is *use.*[22] To be more explicit: the meaning of an expression is determined by the rules that governs its use in the Language of which it forms a part. It is absurd to interpret Wittgenstein's idea as saying that meaning is conferred automatically on the expressions and sentences of a Language. If this were the case no philosophical problems would arise. We know they do. As Wittgenstein said: "The results of philosophy are the uncovering of one or another piece of plain nonsense and bumps that the understanding has got by running its head up against the limits of language...".[23]

This is clearly evident in the Language of primitive recursive functions, *e.g.,* in the question of what determines the meaning of the symbol 0. The first Peano axiom? Hardly. If the axiom automatically were to arrive equipped with its meaning it would be impossible to understand which of the expressions "0" or "natural number" is explanans and which is explanandum. We would only achieve a vicious circle in attempting to explain the meaning of the first Peano axiom: "0 is a natural number". For example, one can try to arrive at a realization of Wittgenstein's imperative by making the futile attempt to define a natural number as what we get to from zero by iterating the successor operation a finite number of times. However, this is circular, because the definition uses the words ***finite number*** which are synonymous to the words ***natural number*** whose meaning it attempts to explain. There is no way out of this circle but to say that a natural number is what we iterate up to: which is to explain nothing. In essence, this is what Russell's explanation tells us when it states that: "...'0' and 'number' and 'successor' cannot be defined by means of Peano's...axioms, but must be independently understood".[24]

What Russell requires to be independently understood is exactly what a person program sets out to explain. It is important to realize that the two first Peano axioms "0 is a natural number" and "the successor of any number is a successor", as far as meaning is concerned, are redundant, since they presuppose understanding of the expressions "0" and "natural number". This does not mean that 0 is not a natural number, it certainly is, but the first Peano axiom is not the rule from which the meaning of the symbol is learnt. In the semantical (philosophical) investigation of the program $0 + 2$ the expressions standing for the canonical (normal) forms are the rules $\overline{+}$, $\overline{0}$ and $\overline{S(S(0))}$. The explanations are not redundant *vis-à-vis* meaning, since the symbols, the meaning of which we are to recover, do not occur in the explanans except for the primitive function constant 0. Here, again, it is important to recall that to understand a rule, we must understand the conclusion under the assumption that the premises have been understood. The premiss, being a canonical step, is understood because the predicate is contained in the subject and we take the subject to be the object: the limit of Language.[25] It is evident that the two Peano axioms, as they are usually formulated, are not in accordance with Wittgenstein's principle. Indeed, an expression is a natural number, not in virtue of its *form*, but in virtue of its *function* as we shall see later.

4. The Alphabet

The ***symbols*** that we are going to use in the person program will have the following *form* and be divided into function ***constants:***

$0, S, +, \ldots$

and numerical ***variables:***

$x, y, z, \ldots$

Furthermore, we divide function constants into ***primitive*** and ***defined*** ones. In the investigation we are to put up only 0 and S are primitive. All the rest are defined. An ***expression*** (meaningless in general) is simply a string of symbols like:

$y + x, x + S(0), \ldots$

in conjunction with a parenthesis notation. To construct the Language, as we shall do when formulating its syntax and semantics, we also need ***syntactical variables.*** We symbolize these by:

$f, g, \ldots$

which stand for ***function constants.*** Furthermore we have:

$v, w, \ldots$

which stand for ***numerical variables,*** and:

$a, b, c, \ldots$

which stand for arbitrary object-valued functional ***expressions.*** Thus, in general,

$$a(v_1, \ldots, v_k)$$

will stand for an arbitrary object-valued functional expression whose variables are among $v_1, \ldots, v_k$ rather as in the ordinary notation for polynomials. A different placing of $v_1, \ldots, v_k$ within the parentheses reflects a difference of input place within the functional expression. A single occurrence of a variable within the parentheses may stand for many linked argument places within the functional expression. Although one writes variables over functional expressions in two parts, these being a part for the variables, and a part for the expression over and above the variables, these two parts cannot in any sense be separated in an actual functional expression. One may vary the functional expression while retaining the same structure of inputs, but not take away what is varied here, leaving the input structure, so to speak, on its own. With this the alphabet is introduced.

5. On Natural Numbers

The meaning of a statement of the form:

x is a natural number

is the *way* in which a natural number is determined as the value of a recursively defined function for x as principal and natural numbers as subordinate arguments. Thus, to understand that x is a natural number, we must know how to determine a natural number y such that,

$$f(x_1, ..., x_k, x) = y$$

under the assumption that f is a (k + 1)–place function defined by recursion and $x_1, ..., x_k$ are natural numbers. What determines the value of a recursively defined function for given arguments is the scheme of recursion, being, according to Wittgenstein "a standard for the classification of cases".[26] A recursion schema expresses the general form of a *context,* that is, it is what we acquire when we engage in explaining an expression or sentence in accordance with the constructivist imperative of Frege's contextual principle.[27] We seek a clarification of the depth grammar of the numerals and the word "number", that is, "(i)f the question: 'What are numbers?' is approached from this point of view we will no longer succumb to its suggestive spell. Instead of setting up the nature of number in a formula we will describe the uses of the word "number" and of the "numerals".[28]

Arithmetic is simply the depth grammar of number-words.[29] Hence we cannot understand a particular statement of the form that we are considering without looking back on this scheme. As is implicit in what we just have said, a natural number is an expression for which we have understood the statement above. Or, as we may say, a natural number is an expression which we have understood as a natural number. Its relation to Wittgenstein's answer: "A natural number is the exponent of an operation",[30] becomes clear if we reformulate it thus: A natural number is the principal argument of functions defined by recursion.

The meaning of a statement of the form:

$$a(v_1, ..., v_k) \text{ is a numerical function}$$

is the way in which a natural number is determined as the value (denotation) of $a(v_1, ..., v_k)$ when natural numbers are assigned as values to the variables $v_1, ..., v_k$. Thus, in order to understand that $a(v_1, ..., v_k)$ is a numerical function, we must know, given natural numbers $x_1, ..., x_k$, how to determine a natural number x such that:

$$\overline{a(v_1, ..., v_k)} = x \text{ for } \overline{v}_1 = x_1, ..., \overline{v}_k = x_k$$

What determines the value of $a(v_1, ..., v_k)$ for given arguments are the rules of denotation, and hence we cannot understand a statement of this form without looking back on these rules before we can proceed. We shall return to the meaning of numerical functions later.

6. Denotation Rules

In his famous article *On Denoting* Russell differentiates between *acquaintance* and *knowledge about* as "the distinction between the things we have presentations of, and the things we only reach by means of denoting phrases".[31] Here the subject encounters ***objects*** of thought, ***i.e.,*** "(a)ll thinking has to start from acquaintance; but it succeeds in thinking ***about*** many things with which we have no acquaintance".[32] In a philosophical investigation we are not primarily interested in a subject acquainted with an object of thought (expressing knowledge of facts) since "...it is not the property of an object that is ever 'essential', but rather the mark of a concept".[33] That is, if we construct the depth grammar by canonical steps in virtue of Wittgenstein's principle, the object, as far as meaning is concerned, "drops out of consideration as irrelevant".[34] We are interested in ***what*** it is to engage in an activity of denoting (expressing knowledge of meaning). Especially we want to focus on what goes on when one denotes a form as a numerical function.

In order to understand that a statement of the form $a(v_1, ..., v_k)$ is a numerical function we must refer back to statements of the form $\overline{a(v_1, ..., v_k)} = x$. Here we have denoted the form $a(v_1, ..., v_k)$ as a numerical function which is shown by the denotation line above the expression. But to denote is to engage in an activity according to certain rules. These rules must be made explicit by canonical steps. Hence one cannot determine the value of a numerical function until the denotation rules have been given. They read:

$$\overline{v} = x$$

$$\overline{0} = 0$$

$$\frac{\overline{a} = x}{\overline{S(a)} = S(x)}$$

$$\frac{\overline{a}_1 = x_1, \ldots, \overline{a}_k = x_k \quad \overline{a} = x \quad f(x_1, \ldots, x_k; x) = y}{\overline{f(a_1, \ldots, a_k; a)} = y}$$

In the last rule, f is a (k + 1)–place function defined by recursion. We can now also explain the meaning of a statement of the form:

f is a (k + 1)–place function defined by recursion

Its meaning, like that of any statement, is what we know when we have understood it. And, to understand it, we merely have to understand that the expressions $a(v_1, \ldots, v_k)$ and $b(v_1, \ldots, v_k; v, w)$ which in a purely formal way are associated with the symbol f, are numerical functions.

7. The Meaning of Natural Numbers

Before we engage in explaining the meaning of natural numbers we must explain the notations for arbitrary computations. For arbitrary computations we use the following notation:

$$\begin{array}{c} v_1 = x_1, \ldots, v_k = x_k \\ \hdashline a(v_1, \ldots, v_k) = x \end{array}$$

Here the upper lines show the assignments $x_1, \ldots, x_k$ to the variables $v_1, \ldots, v_k$ of the functional expression and correspond to *what* inputs have been fed into what input holes in virtue of a person program. The result of a computation:

$$\begin{array}{c} v_1 = x_1, \ldots, v_k = x_k \\ \hdashline a(v_1, \ldots, v_k) = x \end{array}$$

is that $\overline{a(v_1, \ldots, v_k)} = x$ *for* $\overline{v}_1 = x_1, \ldots \overline{v}_k = x_k$. This can be translated verbally in the following way:

1. x is the *value* of $a(v_1, ..., v_k)$ for arguments $x_1, ..., x_k$ in the argument places $v_1, ..., v_k$ respectively, or when the variables $v_1, ..., v_k$ are assigned values $x_1, ..., x_k$ respectively.
2. x is the *output* of $a(v_1, ..., v_k)$ for inputs $x_1, ..., x_k$ in the input position $v_1, ..., v_k$ respectively.
3. $a(v_1, ..., v_k)$ *denotes* x when $v_1, ..., v_k$ denote $x_1, ..., x_k$ respectively.

We can now realize that "value", "output" and "denotation" have a common canonical form. As far as meaning is concerned they are, when used within different disciplines (mathematics, computer science, philosophy), used in a redundant way. This is one of the redundancies that we eliminate by person programs in virtue of Wittgenstein's principle.

One may never proceed in a computation using in different places assignments of different object expressions as values to the same variable. For occurrance of the same variable at different places show that the argument places shown by these occurrences are places for the same argument. Furthermore, a *computation (proof)* is not in the proper sense a proof of its result. It is something that produces its result, *i.e.,* it is not an essential part of the result as far as meaning is concerned. One could say that proofs proper, establishing links between grammatical propositions, themselves *belong* to grammar. Their function is to make connections between propositions created by canonical steps: "The proof is part of the *surroundings* of the proposition".[35] A computation (proof proper) is a process of which its result is the upshot, and not an argument for it, *i.e.,* "a constructively valid proof already constitutes a realization of the computational content of the result being proved. In other words, a proof *is* a computation".[36] One can imagine a notation like the arrangement of addition and multiplication sums children are taught which one has no inclination to call the successive inference of propositions.

Given these explanations, it is clear that what allows us to understand (determine the meaning of) the first Peano axiom is the first clause:

$$\begin{array}{c} v_1 = x_1, \ldots, v_k = x_k \\ \hdashline a(v_1, \ldots, v_k) = x \\ \hline f(x_1, \ldots, x_k; 0) = x \end{array}$$

of the recursion schema. Indeed, that f is a (k + 1)-place function defined by recursion means that the expressions $a(v_1, \ldots, v_k)$ and $b(v_1, \ldots, v_k; v, w)$ in terms of which it is defined, are numerical functions. Hence, given natural numbers $x_1, \ldots, x_k$, we know how to determine a natural number x such that $\overline{a(v_1, \ldots, v_k)} = x$ for $\bar{v}_1 = x_1, \ldots, \bar{v}_k = x_k$. The first clause of the recursion scheme stipulates that this number x is to be the value (output, denotation) of f for the subordinate arguments $x_1, \ldots, x_k$ and the principal argument 0. And, of course, since we deal with canonical forms, there is ***no other rule*** which allows us to derive a statement of the form $f(x_1, \ldots, x_k; 0) = x$. This is the meaning of the first Peano axiom, that is, the way in which a natural number is determined as the value (output, denotation) of a recursively defined function for the principal 0 and natural numbers as subordinate arguments. Now we can realize why "(d)isputes do not break out (among mathematicians, say) over the question whether a rule has been obeyed or not...That is part of the framework on which the working of our language is based...".[37] It is also the philosophical answer to the question:

What is zero?

because, as stated in the opening sentence of Frege's *Foundations of Arithmetic*, the question what the number zero is, comes to the same as to the question *what* the symbol 0 means.[38] It is to answer the question what zero means according to the proposition-as-rules idea. More importantly, it shows also the principle by which *every axiom can be read as a rule.* This gives substance to Wittgenstein's interest in Weyl's remark that "a formalist conceives of the axioms of mathematics as like chess-rules".[39] The idea, itself, is expressed by Wittgenstein in the following way: "It seems to me that the idea of the consistency of the axioms of mathematics, by which mathematicians are so haunted these days, rests on a misunderstanding. This is tied up with the fact that axioms of mathematics are not seen for what they are, namely, propositions of syntax".[40]

To understand the second Peano axiom, which, when formulated as a rule, reads:

$$\frac{x \text{ is a natural number}}{S(x) \text{ is a natural number}}$$

we must know how to evaluate a recursively defined function for the principal argument $S(x)$, given that we know how to evaluate it for the principal argument x. So let f be a (k + 1)–place function defined by recursion and $x_1, ..., x_k$ natural numbers. By assumption, we know how to determine a natural number y such that $f(x_1, ..., x_k; x) = y$. Since f is a (k + 1)–place function defined by recursion, the second of the expressions associated with the symbol f, call it $b(v_1, ..., v_k; v, w)$, is a numerical function. Hence, $x_1, ..., x_k$, x and y being natural numbers, we know how to determine a natural number z such that $\overline{b(v_1, ..., v_k; v, w)} = z$ for $\overline{v}_1 = x_1, ...\overline{v}_k = x_k$, $\overline{v} = x$ and $\overline{w} = y$. The second clause of the recursion scheme:

$$\frac{f(x_1,\dots,x_k;x)=y \qquad \begin{array}{c}\overline{v}_1=x_1,\dots,\overline{v}_k=x_k \quad \overline{v}=x \quad \overline{w}=y\\ \hdashline \overline{b(v_1,\dots,v_k;v,w)}=z\end{array}}{f(x_1,\dots,x_k;S(x))=z}$$

stipulates that the number z is to be the value of f for the subordinate arguments $x_1,\dots,x_k$ and the principal argument *S(x)*. This is the meaning of the second Peano axiom, or, what amounts to the same, the meaning of the symbol *S*. Our explanation of the meaning of *S(x)* in terms of the meaning of x, is at the same time our answer to the question:

What is the successor of a natural number?

because, to ask for the meaning of the expression *S(x)*, is the same as to ask *what S(x)* is. It is of extreme importance to realize that the two Peano axioms do *not* stipulate what a natural number is. That would be to say that a natural number is an expression for which we have derived (instead of understood) the statement "x is a natural number". This is to confuse natural numbers with numerals, which is the formalist misinterpretation of the concept of natural number: "Frege ridiculed the formalist conception of mathematics by saying that the formalists confused the unimportant thing, the sign, with the important, the meaning...Frege's idea could be expressed thus: the propositions of mathematics, if they were just complexes of dashes, would be dead and utterly uninteresting whereas they obviously have a kind of life... But if we had to name anything which is the life of the sign, we should have to say that it was its *use*".[41] An expression is a natural number, not in virtue of its form, but in virtue of its *function*, as stated above. This is how the two Peano axioms are to be understood. In general, axioms, by themselves, do not provide knowledge of meaning.

As Wittgenstein said: "Something is an axiom, ***not*** because we accept it as extremely probable, nay certain, but because we assign it a particular function...".[42] And this function is to serve as the principal argument of functions defined by recursion.

8. On Function Formation

We will now turn our attention to the rules of function formation. In general, one can say that a functional expression is like a machine that operates in a certain way. This may be a black box with zero or more input holes, various gadgets inside, and an output hole. Exactly how it is made up, of plastic, some metal, wood, or what not, is not essential to its operation. One could put it together with no idea of how it worked. Similarly to a functional expression there belongs something inessential, say an expression in certain variables. The places occupied in it by the variables, linked together as indicated by the identity and difference of the variables, correspond to the input holes of the machine. Beyond their serving the purpose of showing the input places of the functional expression, the variables are inessential. One can imagine many devices that would serve the same purpose.

It is not a matter for stipulation that something is a functional expression, just as it is not up to us that some metal parts we have bolted together should constitute a functioning machine. Rather it is in virtue of there being a definite rule for the evaluation of the functional expression under an assignment of things (fed into its input holes) that the machine continues with some operation on these things until a final product emerges from the output hole. What we put in are the arguments. What we get out, if anything, is the *value*, *i.e.*, the circumstance of being correct.

The working out of the value from the arguments is called the ***evaluation***, or ***calculation***, of the functional expression of those arguments. To evaluate is to work out the ***logical form*** of the arguments. The rules for the calculation of a functional expression are like the internal mechanism of a machine. To the exterior casing of the machine there corresponds the formal aspect of the functional expression, that is, something identifying its calculation rules, and its structure of inputs: the *form*. Again, it is important to realize that it is not in virtue of its exterior casing that something is a machine. It is essential to a functional expression that we know the rules for its calculation, and that these should be unambiguous. It is like a deterministic machine which contains no random element by which it could proceed differently every time it was put into operation. Thus should a functional expression produce a value, ***it will always produce the same value.*** Consequently, in order to understand a statement of the form:

$$a(v_1, ..., v_k) \text{ is a numerical function}$$

we must know, given natural numbers $x_1, ..., x_k$, how to determine a natural number x such that $\overline{a(v_1, ..., v_k)} = x$ for $\overline{v}_1 = x_1, ..., \overline{v}_k = x_k$. And, indeed, we do understand the statement:

$$v \text{ is a numerical function}$$

for a numerical variable v, by looking back on the denotation rule which says that:

$$\overline{v} = x$$

provided x is assigned as value to the variable v. It is this stipulation which allows us to understand the symbol v as a numerical function.

The statement:

$$0 \text{ is a numerical function}$$

is understood from the stipulation:

$$\overline{0} = 0$$

and the first Peano axiom. Thus we cannot understand the symbol 0 as a numerical function until *after we have understood it as a natural number.*

The rule:

$$\frac{a \text{ is a numerical function}}{S(a) \text{ is a numerical function}}$$

is understood from the denotation rule:

$$\frac{\overline{a} = x}{\overline{S(a)} = S(x)}$$

and the second Peano axiom. Suppose, namely, that a is a numerical function, that is, we know how to determine a natural number x such that $\overline{a} = x$. Then $S(x)$ is a natural number by the second Peano axiom, and the denotation rule above stipulates that $\overline{S(a)} = S(x)$. This is the meaning of the above rule of function formation, that is, the way in which a natural number—$S(x)$—is determined as the value of $S(a)$ under the assumption that a is a numerical function. Note, again, that we can understand this rule only *after* we have understood the second Peano axiom.

The final rule of function formation is:

$$\frac{a_1, \ldots, a_k \text{ are numerical functions}}{f(a_1, \ldots, a_k; a) \text{ is a numerical function}}$$

where f is a (k + 1)-place function defined by recursion. To understand it, we must know how to evaluate $f(a_1, ..., a_k; a)$ under the assumption that we know how to evaluate $a_1, ..., a_k$ and a, that is, that we know how to determine natural numbers $x_1, ..., x_k$ and x such that $\overline{a}_1 = x_1, ..., \overline{a}_k = x_k$ and $\overline{a} = x$. But this is precisely what the last denotation rule formulated above tells us. Indeed, since f is a (k + 1)-place function defined by recursion and the subordinate arguments $x_1, ..., x_k$ are natural numbers, we know that a natural number is determined as its value whenever a natural number is inserted into its principal argument place. In particular, this is so for x, that is, we know how to determine a natural number y such that $f(x_1, ..., x_k; x) = y$. The denotation rule stipulates that this number y is to be the value (denotation, output) of $f(a_1, ..., a_k; a)$. This is how the meaning of the expression $f(a_1, ..., a_k; a)$ as a numerical function is determined in terms of the meanings of the expressions $a_1, ..., a_k$ and a.

9. Composition of Functions

We are now coming to the thesis of composition of functions. Functional expressions will be either simple or composite. Composing functional expressions is like connecting two machines together so that the product of one is immediately taken as the input of the other, the two so connected forming a third less elementary machine. A simple functional expression is one which is not connected in this way. Thus, when we give the functional expressions of the Language, there will be clauses that relate to the simple functional expressions and their computation, and clauses that relate to composite functional expressions expressing how computations such things are formed from computations of their components. This thesis tells us that $a_1, ..., a_k$ and $b(v_1, ..., v_k) = y$ provided $\overline{a}_1 = x_1, ..., \overline{a}_k = x_k$ and $b(v_1, ..., v_k) = y$ for $\overline{v}_1 = x_1, ..., \overline{v}_k = x_k$. This thesis cannot be proved, it has to be understood. And what has to be understood is the way in which $b(a_1, ..., a_k)$ is evaluated, and that its value is the same as is obtained by *first* evaluating $a_1, ..., a_k$ and *then* evaluating $b(v_1, ..., v_k)$ for these values as arguments. So suppose that $a_1, ..., a_k$ are numerical functions, that is, that we know how natural numbers $x_1, ..., x_k$ are determined as their values

$$\begin{array}{ccc} \vdots & & \vdots \\ \overline{a}_1 = x_1 & \ldots & \overline{a}_k = x_k \end{array}$$

Suppose further that $b(v_1, ..., v_k)$ is a numerical function. Then, $x_1, ..., x_k$ being natural numbers, we know how to determine a natural number y as the value of $b(v_1, ..., v_k)$ for these arguments:

$$\frac{\overline{v}_1 = x_1, \ldots, \overline{v}_k = x_k}{\overline{b(v_1, ..., v_k)} = y}$$

Replacing the variables $v_1, ..., v_k$ in this evaluation by the expressions $a_1, ..., a_k$ and attaching to it the evaluation of these:

$$\frac{\begin{array}{ccc} \vdots & & \vdots \\ \overline{a}_1 = x_1 & \ldots & \overline{a}_k = x_k \end{array}}{\overline{b(v_1, ..., v_k)} = y}$$

we see that the natural number y is determined as the value of $b(a_1, ..., a_k)$. This is how we understand that $b(a_1, ..., a_k)$ is a numerical function. The last part of the statement about composition of functions states that *the value of a composite function is determined by the values of its components.* If we read Frege's *Bedeutung* for value, we recognize this as one of the theses that was set forth by him, *i.e.,* that "the reference (Bedeutung) of a sentence may always be sought, whenever the reference of its components is involved; and this is the case when and only when we are inquiring after the truth value".[43] Remember that *value* (the circumstance of being true) is not something which can be proved, but must be understood.

10. On Definitional Equality

In the *Begriffschrift* we find that Frege formulated the conception of *equality* as follows:[44]

Es bedeute nun

$$\vdash (A \equiv B):$$

das Zeichen A und das Zeichen B haben denselben begrifflichen Inhalt, sodass man überall an die Stelle von A B setzen kann und umgekehrt.

In virtue of Wittgenstein's principle we realize that the form of Language excludes a metaphysical doctrine of *identity* as far as meaning is concerned since "(i)t is necessary to distinguish carefully between, on the one hand, the relation of definitional equality which...is a relation between linguistic expressions and, on the other hand, the relation of identity between the abstract entities that they denote".[45] But we do have *definitional equality* which is "a relation between linguistic expressions and *not* between the abstract entities which they denote and which are the same. This is the view that Frege took of the relation of equality of content *(Inhaltsgleichheit)* which enters into his *Begriffschrift* but which he later abandoned".[46] According to Martin-Löf the relation of definitional equality is determined by the following three principles *and by these principles alone:*[47]

(i) A definiens is always definitionally equal to its definiendum.

(ii) Definitional equality is preserved under substitution. That is, if we substitute two definitionally equal expressions for a variable in a given expression, then the resulting expressions are also definitionally equal.

(iii) Definitional equality is an equivalence relation, that is, it is reflexive, symmetric and transitive.

What we still need are the rules for definitional equality. The meaning of definitional equality, that is, statements of the form:

$$\overline{a(v_1,...,v_k)} = \overline{b(v_1,...,v_k)}$$

for $\overline{v}_1 = x_1,...,\overline{v}_k = x_k$. Another way of expressing this is to say that two expressions are definitionally equal if they both are numerical functions, and, if they take the same value for arbitrarily given natural numbers as arguments. Consider now the first rule of definitional equality. Using the notation $\overline{a} = \overline{b}$, rather than $a \equiv b$, it reads:

$$\frac{a_1,...,a_k \text{ are numerical functions}}{\overline{f(a_1,...a_k;0)} = \overline{a(a_1,...,a_k)}}$$

To understand it, we must understand the conclusion under the assumption that the premises have been understood. So, suppose that we know how to determine natural numbers $x_1,...,x_k$ such that $\overline{a}_1 = x_1,...,\overline{a}_k = x_k$. By composition of functions $a(a_1,...,a_k)$ is a numerical function, and $\overline{a(a_1,...,a_k)} = x$, where x is the natural number such that $\overline{a(v_1,...,v_k)} = x$ for $\overline{v}_1 = x_1,...,\overline{v}_k = x_k$. On the other hand, the first clause of the recursion schema and the denotation rules yield:

$$\dfrac{\begin{array}{ccccc} \vdots & & \vdots & & \\ a_1 = x_1 & \ldots & a_k = x_k & \overline{0} = 0 & \dfrac{\dfrac{\overline{v}_1 = x_1 \ldots \overline{v}_k = x_k}{\overline{a(v_1, \ldots, v_k)} = x}}{f(x_1, \ldots, x_k; 0) = x} \end{array}}{\overline{f(a_1, \ldots, a_k; 0)} = x}$$

This is *how* we determine the natural number x such that:

$$\overline{f(a_1, ..., a_k; 0)} = x = \overline{a(a_1, ..., a_k)}$$

The second rule is:

$$\frac{a_1, ..., a_k; a \text{ are numerical functions}}{\overline{f(a_1, ..., a_k; S(a))} = \overline{b(a_1, ..., a_k; a, f(a_1, ..., a_k; a))}}$$

It is understood in a similar way as the first one. Assume that we know how to determine natural numbers $x_1, ..., x_k$ and x such that $\overline{a}_1 = x_1, ..., \overline{a}_k = x_k$ and $\overline{a} = x$. Then, since f is assumed to be a (k + 1)-place function defined by recursion, we can determine, first, a natural number y such that $f(x_1, ..., x_k; x) = y$ and, second, a natural number z such that $\overline{b(v_1, ..., v_k; v, w)} = z$ for $\overline{v}_1 = x_1, ..., \overline{v}_k = x_k$, $\overline{v} = x$ and $\overline{w} = y$. In virtue of the last denotation rule and composition of functions we get:

$$\overline{b(a_1, ..., a_k; a, f(a_1, ..., a_k; a))} = z$$

On the other hand, this number z is determined as the value of $f(a_1, ..., a_k; S(a))$ by the second clause of the recursion scheme and the denotation rules as follows:

$$\dfrac{\begin{matrix}\vdots \\ f(x_1,\ldots,x_k;x) = y\end{matrix} \qquad \dfrac{\bar{v}_1 = x_1 \ldots \bar{v}_k = x_k \quad \bar{v} = x \quad \bar{w} = x}{\overline{b(v_1,\ldots,v_k;v,w)} = z}}{f(x_1,\ldots,x_k;S(x)) = z}$$

$$\dfrac{\begin{matrix}\vdots \\ \bar{a}_1 = x_1\end{matrix} \quad \cdots \quad \begin{matrix}\vdots \\ a_k = x_k\end{matrix} \quad \dfrac{\begin{matrix}\vdots \\ \bar{a} = x\end{matrix}}{\overline{S(a)} = S(x)} \quad \overline{f(x_1,\ldots,x_k); S(x))} = z}{\overline{f(a_1,\ldots,a_k;S(a))} = z}$$

This is how we understand the second rule of definitional equality.

The third rule is the principle that ***definitional equality is preserved under substitution:***

$$\dfrac{\bar{a}_1 = \bar{c}_1, ..., \bar{a}_k = \bar{c}_k}{\overline{b(a_1, ..., a_k)} = \overline{b(c_1, ..., c_k)}}$$

Assume that its premises have been understood, that is, we know how to determine natural numbers $x_1, ..., x_k$ such that:

$$\bar{a}_1 = x_1 = \bar{c}_1, ..., \bar{a}_k = x_k = \bar{c}_k$$

Since, by assumption, $b(v_1, ..., v_k)$ is a numerical function, we know how to determine a natural number y such that $\overline{b(v_1, ..., v_k)} = y$ for $\bar{v}_1 = x_1, ..., \bar{v}_k = x_k$. By composition of functions we get:

$$\overline{b(a_1, ..., a_k)} = y = \overline{b(c_1, ..., c_k)}$$

which is precisely what we had to understand.

The remaining rules of definitional equality are ***reflexivity:***

$$\frac{a \text{ is a numerical function}}{\overline{a} = \overline{a}}$$

and *symmetry:*

$$\frac{\overline{a} = \overline{b}}{\overline{b} = \overline{a}}$$

These rules are understood immediately. The last rule stands for ***transitivity:***

$$\frac{\overline{a} = \overline{b} \; \overline{b} = \overline{c}}{\overline{a} = \overline{c}}$$

Transitivity is to be understood as follows. Suppose that we know how to determine natural numbers x and y such that:

$$\overline{a} = x = \overline{b} \text{ and } \overline{b} = y = \overline{c}$$

Then, since both x and y are ***determined*** as the value of b, they must be the same expressions. Therefore, the same natural number x is determined as the value of both a and c, which is precisely what we must know in order to understand $\overline{a} = \overline{c}$. With this the formulation and explanation of the rules needed to undertake the person program $0 + 2$ is complete. Note that each of the rules determines the *logical form* of this program.

11. Surveyability of Depth Grammar

To engage in a person program provides the function of contributing a synoptic view *(Übersichtlichkeit)* of the codified activity. The idea of a synoptic view is also embraced by formalism as understood by Hilbert and Bernays. Note the similarity with Wittgenstein's use of *überblickbar*, or, as Kreisel calls it, *überschaubar*, in comparison with *übersehbar* in Hilbert's and Bernay's definition of finitist meta-mathematics.[48] But, in comparison with meta-mathematics, it is only the *formal* part of the Language where the *formalist* view is correct. In the Language of primitive recursive functions we are dealing with the formal part of a more extensive Language: the Language of mathematics. We shall engage in an extension of the Language elsewhere.

Now, we can treat, by *conversion*, the logical depth grammar of primitive recursive functions meta-mathematically in the sense of Hilbert. To be convertible in virtue of a person program requires understanding, i.e. knowledge of meaning. Conversion is not something we prove: it is something we *understand.* When the Language is treated in this way, the expressions become meta-mathematical objects of an inductively defined type, and the five different, but necessary (complementary), forms of statements that we have been considering are turned into inductively defined meta-mathematical propositions proper. They become propositions which can be proved (computed) and combined by means of logical connectives and quantifiers. As an example, in particular, the expression

x is a natural number

is turned into a property proper, and

$$\overline{a(v_1, ..., v_k)} = x \text{ for } \overline{v}_1 = x_1, ..., \overline{v}_k = x_k$$

into a (k + 1)-place relation proper between expressions as meta-mathematical objects. Therefore we can form the meta-mathematical proposition:

$$\frac{(\forall x_1, ..., x_k)\,(x_1\mathcal{N}\& \,, ..., \;\& \; x_k\mathcal{N})}{(\exists x)\,(\mathcal{N}(x) \;\&\; \overline{a(v_1, ..., v_k)} = x \text{ for } \overline{v}_1 = x_1, ..., \overline{v}_k = x_k)}$$

Here the quantifiers range over expressions as meta-mathematical objects. It expresses that the open expression as meta-mathematical object $a(v_1, ..., v_k)$ is ***convertible*** in the sense of William Tait.[49] It is not to be confused with the statement:

$$a(v_1, ..., v_k) \quad \text{is a numerical function}$$

The former is a proper meta-mathematical proposition which we may try to prove. The latter can only be understood: it is a depth grammatical statement. And, it contributes to a synoptic view. Now we can also make sense out of Wittgenstein's statement that "(t)here is no metamathematics".[50] What we deal with, after conversion, is a depth grammar read as a meta-language, ***not*** a meta-language in the way Hilbert understood the term. Indeed, from the view of a philosophical investigation, a meta-mathematical attitude is only possible ***after*** a conversion, which, in turn, ***requires*** understanding. But, it is precisely a philosophical investigation which provides understanding in virtue of a person program. We have to realize that in both the formulation and the proof of a meta-mathematical proposition we use linguistic expressions in the ordinary sense. Furthermore, a proof must be ***understood.***[51] If it is not understood it cannot be executed. And understanding the meta-language in which a proof of convertibility is carried out, is at least as difficult as understanding the object language, in this case, the Language of primitive recursive functions.

Even a meta-language needs to be understood. And it is understood precisely in virtue of a person program. Now we can understand why Wittgenstein was dissatisfied with Russell's proposal that a meta-language is "another language dealing with the structure of the first language, and having itself a new structure".[52] In virtue of Wittgenstein's principle, there simply is no criterion of the *adequacy* concerning our dealing with the object language as André Maury points out.[53] The criterion of adequacy is not provided by induction. The clauses specifying the forms of the Language are not induction clauses. Were that so, it would be essential to the notion of expression that one could define functions on expressions by recursive clauses. This is evidently nothing a child has to grasp when it learns what are the expressions of its mother language. The sense of expressions which is prior to meta-mathematics is so fundamental that we fail to appreciate it. This point was explicitly stressed by Wittgenstein: "A mathematician is bound to be horrified by my mathematical comments, since he has always been trained to avoid indulging in thought and doubts of the kind I develop. He has learned to regard them as something contemptible and, to use an analogy from psycho-analysis (this paragraph is reminiscent of Freud), he has acquired a revulsion from them as infantile. That is to say, I trot out all the problems that a child learning arithmetic, etc., finds difficult, the problem that education represses without solving. I say to those repressed doubts: you are quite correct, go on asking, demand clarification!".[54] As an example one would do well to remember just how radically original a step was taken when meta-mathematics began with the introduction of the idea of an expression as a mathematical *object*. By this step questions of meaning and understanding became suppressed as superfluous. This is not the case in a philosophical investigation. On the contrary, it is absurd to maintain, at least in the Language of primitive recursive functions, that we do understand a meta-language but not the object language.

In fact, there is not a *genuine* need for a proof of convertibility. At least not in this case. And it certainly does not provide us with any philosophical insight.

The requirement of a synoptic view serves the function of providing understanding of numerical expressions. The grammatical rules (syntax) introduces order for expressions and sentences, in this case, in the context of primitive recursive functions. The demand by Wittgenstein of perspicuity *(Übersichtlichkeit)* is not a puzzling *extra* requirement in Wittgenstein's characterization of mathematical proof (computation). If proofs lacked this feature, they could not be executed at all: perspicuity is an essential part of proof (computation).[55] Person programs show that "proof must be a procedure it plain to view. Or again: the proof is the procedure that is *plain to view*. It is not something behind the proof, but the proof, that proves".[56] Indeed, as we saw in the execution of the person program 0 + 2 above, it is the case that "(i)f I have once grasped a rule I am bound in what I do further. But of course that only means that I am bound in my *judgment* about what is in accord with the rule and what is not".[57] One can execute a deduction in virtue of knowing *what* one is to do, as emphasized elsewhere.[58] The principal defect of Language and the source of philosophical perplexity is the lack of *Übersichtlichkeit* in the rules of depth grammar. It is to lack knowledge concerning what to do. The corrective is to engage in a person program in order to create by canonical steps an *übersichtliche Darstellung* that encapsulates an adequate view of the connections of expressions and sentences in a Language. One can say that: "A main source of our failure to understand is that we do not *command a clear view* of the use of our words. - Our grammar is lacking in this sort of perspicuity. A perspicuous representation produces just that understanding which consists in 'seeing connexions"'.[59]

In Language *Übersichtlichkeit* is of fundamental importance, but it is created, not found. Indeed, the mathematician, being a participator, "...is an inventor, not a discoverer".[60]

12. The Logical Form of Primitive Recursive Functions

When we engage in the philosophical task of formulating the logical depth grammar of primitive recursive functions we formulate it as a *translation manual.* The phrases following the syntactical statements in the translation manual only show how to read these statements in English. They must not be regarded as explanations of the meanings of the criteria, *i.e.,* the canonical expressions. If this were the case they would have to be understood *already*, and it would be absurd to assume this, since the task is precisely to explain the meanings of these statements. It would not be in accordance with Wittgenstein's principle. It is totally irrelevant whether they are expressed in mathematical notation or in English at this stage. However, since the *Begriffschrift* is to be formulated in English, it will be convenient to have the option to express the statements whose meanings we are to explain in English as well. When we engage in the philosophical task of formulating the depth grammar (logical form) of primitive recursive functions we end up with statements of five forms. In each of the five forms we have a *general* rule, which is to be explained first, and a number of applied rules.

Rules of Computation:

General rule:

$f(x_1, \ldots, x_k; x) = y$ — -y is the value of the recursively defined function f for the subordinate arguments $x_1, \ldots, x_k$ and the principal argument x

Applied rules:

$$f(x_1, \ldots, x_k; 0) = x$$

$$f(x_1, \ldots, x_k; S(x)) = z$$

Rules of Denotation:

General rule:

$\overline{a} = x$ — -a denotes x

-x is the value of a

-x is the output of a

Applied rules:

$\overline{v} = x$ — - assignment

$$\overline{0} = 0$$

$$\frac{\overline{a} = x}{\overline{S(a)} = S(x)}$$

$$\frac{\overline{a}_1 = x_1, \ldots, \overline{a}_k = x_k \; \overline{a} = x \; f(x_1, \ldots, x_k; x) = y}{\overline{f(a_1, \ldots, a_k; a)} = y}$$

First and second Peano axioms:

General rule:

x is a natural number

Applied rules:

0 is a natural number

$$\frac{x \text{ is a natural number}}{S(x) \text{ is a natural number}}$$

Function Formation:

General rule:

a is a numerical function

Applied rules:

v is a numerical function

0 is a numerical function

$$\frac{a \text{ is a numerical function}}{S(a) \text{ is a numerical function}}$$

$$\frac{a_1, \ldots, a_k; a \text{ are numerical functions}}{f(a_1, \ldots, a_k; a) \text{ is a numerical function}}$$

Definitional equality:

General rule:

$\overline{a} = \overline{b}$ or $a \equiv b$ - the functions a and b are definitionally equal, or have the same value

Applied rules:

$$\frac{a_1, \ldots, a_k \text{ are numerical functions}}{\overline{f(a_1, \ldots, a_k; S(a))} = \overline{b(a_1, \ldots, a_k; a, f(a_1, \ldots, a_k))}}$$

$$\frac{\overline{a}_1 = \overline{c}_1, \ldots, \overline{a}_k = \overline{c}_k}{\overline{b(a_1, \ldots, a_k)} = \overline{b(c_1, \ldots, c_k)}}$$

$$\frac{a \text{ is a numerical function}}{\overline{a} = \overline{a}}$$ - reflexivity

$$\frac{\overline{a} = \overline{b}}{\overline{b} = \overline{a}}$$ - symmetry

$$\frac{\overline{a} = \overline{b}\ \overline{b} = \overline{c}}{\overline{a} = \overline{c}}$$ - transitivity

13. Verification of the Program

We are now, finally, in the position to verify (compute) the person program 0 + 2. To successfully engage in this task is to show that this program is ***true***. It is of the form "p is true".[61] That is, when we compute the program 0 + S(S(0)), we treat the expressions occurring in the program as objects of knowledge. We do understand them (they are ***surveyable*** to us), which enables us to ***use*** them in the surveyable computation:

$$\frac{\overline{x} = 0}{0 + 0 = 0}$$

$$\frac{\overline{z} = 0}{\overline{S(z)} = S(0)}$$

$$\frac{0 + 0 = 0 \quad \overline{S(z)} = S(0)}{0 + S(0) = S(0)}$$

$$\frac{0 + S(0) = S(0) \quad \overline{S(z)} = S(S(0))}{0 + S(S(0)) = S(S(0))}$$

Since to understand a rule is to understand the conclusion under the assumption that the premises have been understood, this derivation (computation) is faithful to Wittgenstein's principle. We have explained the meaning and truth of the person program $0 + 2 = 2$ Q.E.D.

The person program provides the logical form of the primitive recursive functions we use in order to ***verify*** (compute) the terminated program $0 + 2 = 2$. As Poincaré says: "We have confined ourselves to bringing together one or other of two purely conventional definitions, and we have verified their identity; nothing new has been learned. ***Verification*** differs from proof precisely because it is analytical, and because it leads to nothing. It leads to nothing because the conclusion is nothing but the premisses translated into another language. A real proof, on the other hand, is fruitful, because the conclusion is in a sense more general than the premisses. The equality $2 + 2 = 4$ can be verified because it is particular".[62] Verification provides knowledge of truth of the ***definitional equality*** of the terminated program $0 + 2 = 2$ (contrary to Poincaré's identity). In fact, it is possible to verify the program $0 + 2$ ***precisely*** because it has a logical depth grammar showing ***how it is to be used.*** This is the same as to understand that $0 + 2$ essentially expresses a task providing information of its own evaluation. Since to understand a rule is to understand the conclusion under the assumption that the premisses have been understood, there is nothing to understand concerning the rules of computation and denotation. If we know how to derive the premises, we know how to derive the conclusion: by applying the rule in question. If one reads verification as demonstration one can say like Wittgenstein: "***In*** a demonstration we ***get*** agreement with someone. If we do not, then we've parted ways before ever starting to communicate in this language".[63]

In this verification (computation) we have demonstrated that $0 + S(S(0)) = S(S(0))$. It presupposes, if we are to philosophically understand the demonstration, a philosophical investigation of this person program. This distinction between a ***demonstration*** and a philosophical investigation, or a ***canonical proof,*** is made by Dummett when he says that "(w)e thus appear to be forced to acknowledge a distinction between a proof, in the strict sense of the word, and a mere demonstration, the latter being related to the former by the fact that a demonstration supplies effective means of constructing an actual proof. What appears in ordinary mathematical articles and textbooks are demonstrations, not proofs in the strict sense; and a demonstration provides an adequate ground for the unqualified assertion of its conclusion. But the primary notion is that of a proof in the strict sense, which we shall refer to as ***canonical proof:*** the notion of a demonstration is a secondary one, definable in terms of that of a canonical proof...".[64] As far as we are concerned, in this investigation, one could say that for us a canonical proof is:

person program + demonstration= canonical proof

which provides understanding of how to execute the program $0 + 2$. In other words, a canonical proof provides understanding (meaning) and knowledge of fact concerning the result (output) of a task. This is in accordance with Martin-Löf's view: "(f)or example, $10^{10} + 1$ is either prime or composite—we have a demonstration (convincing argument) of that fact, but we don't have a proof of either disjunct. However, our demonstration provides us with a method of finding a canonical proof of the disjunction, which *would* provide a proof of one disjunct...This fits nicely with Martin-Löf's philosophy, in which every set, including the set of proofs of a proposition, has canonical elements to which other elements reduce".[65]

REFERENCES

1. Martin-Löf, Per., *Syntax and Semantics of Mathematical Language*, notes written by Peter Hancock on half a lecture series given by Per Martin-Löf at Oxford University in Michaelmas Term 1975 (Unpublished). This paper is an extension of a joint paper with Peter Hancock, *Syntax and Semantics of Primitive Recursive Functions*, Preprint No. 3, 1975, Department of Mathematics, University of Stockholm, Stockholm, Sweden.

2. Wittgenstein, Ludwig., *Philosophical Remarks*, Oxford (1975), *143.*

3. Wright, Crispin., *Wittgenstein on the Foundation of Mathematics*, London (1980), *80*: "Such an anti-realism generates the *Strict Finitist* philosophy of mathematics outlined by Dummett. There is some evidence in RFM - notably, the repeated emphasis on the surveyability of proofs and the character of some of the criticisms of Russell's conception of logicist foundation for mathematics - to suggest that Wittgenstein may have favoured such a view." Cf. also Kreisel, G., Wittgenstein's Remarks on the Foundations of Mathematics, British Journal for the Philosophy of Science *8*, (1958), *135-58*, section 7.

4. Bishop, Errett., *Foundations of Constructive Analysis*, New York, (1967), *354-5.*

5. Sundholm, Göran., *Constructions, Proofs and The Meaning of Logical Constants*, Journal of Philosophical Logic *12* (1983) *163.*

6. Brouwer. L. E. K., *Brouwer's Cambridge Lectures on Intuitionism*, van Dalen (ed.), Cambridge (1981) *4.*

7. Turing, Alan., *On computable numbers, with an application to the Entschei-*

dungs - problem, Proc. London Math. Soc. *42* (1936) *230-65.*

8. van Dalen, Dirk., *Algorithms and Decision Problems: A Crash Course in Recursion Theory*, in G. Gabbay and F. Guenther (eds.), *Handbook of Philosophical Logic*, Vol. I, *417.*

9. Wright, Crispin, *Wittgenstein and the ...*, (cf.n.3), *392.*

10. Maury, André., *Wittgenstein and the Limits of Language*, Acta Philosophica Fennica, Vol. *32* (1981) *151.*

11. Wittgenstein, L., *Philosophical Remarks*, (cf.n.2) *111.*

12. van Dalen, D., *Algorithms and Decision Problems:...*, (cf.n.8) *425.*

13. Gefwert, Christoffer., *The Proposition - As - Rules Idea*, SLAC-PUB-3303 (March 1984), *24*

14. Gefwert, Christoffer., *A Participator: The Metaphyical Subject*, SLAC-PUB-3277 (December 1983), *24-5.*

15. Dummett, Michael., *The Philosophical Basis of Intuitionistic Logic*, reprinted in Paul Benacerraf & Hilary Putnam (eds.) *Philosophy of Mathematics*, sec. ed. Cambridge (1983) *106.*

16. Dummet, Michael., *The Philosophical Basis ...* , (cf.n.15) *19.*

17. Prawitz, Dag., *Proofs and the Meaning and Completeness of the Logical Constants*, in Hintikka, J., Niiniluoto, J., and Saarinen, E (eds.) *Essays in Mathematical and Philosophical Logic*, Dordrecht: Holland (1979) *27.*

18. Dummett, Michael., *Elements of Intuitionism*, Oxford (1977) *376.*

19. Wittgenstein, Ludwig., *Philosophical Grammar*, Oxford (1974) *333.*

20. Gefwert, Christoffer., *The Proposition - As - Rules Idea*, (cf.n.13) *10-11.*

21. Gefwert, Christoffer., *A Participator: ...*, (cf.n.14), *19-22.*

22. Wittgenstein, Ludwig., *Philosophical Investigations*, Oxford (1958) §43: For a *large* class of casees - though not for all in which we employ the work "meaning" it can be defined thus: the meaning of a word is its use in the language.

23. Wittgenstein, L., *Investigations*, (cf.n.22) §119.

24. Russell, Bertrand., *Introduction to Mathematical Philosophy*, London (1920), sec. ed., *9.*

25. Gefwert, Christoffer., *The Proposition - As - Rules Idea* (cf.n.13), *25.*

26. Wittgenstein, Ludwig., *Remarks on the Foundations of Mathematics*, Revised Edition, Oxford (1978) V §39: The theory of functions as a schema, into which on the one hand a host of examples fits, and which on the other hand is there as a standard for the classification of cases.

27. Frege, Gottlob., *The Foundations of Arithmatic*, sec. ed., Oxford (1980) *x*; Gefwert, C., *The Proposition - As - Rules Idea*, (cf.n.13) *26-7.*

28. Waismann, Friedrich., *Introduction to Mathematical Thinking*, New York (1951) *117.*

29. Wittgenstein, L., *Philosophical Remarks*, (cf.n.2) *130.*

30. Wittgenstein, Ludwig., *Tractatus - Logico Philosophicus*, London (1974) §6.021.

31. Russell, Bertrand., *On Denoting*, reprinted in R. C. Marsh ed., *Logic and Knowledge*, London (1956), *41.*

32. Ibid. *42.*

33. Wittgenstein, L., *Remarks on the ...*, (cf.n.26), I §73.

34. Wittgenstein, L., *Investigations*, (cf.n.22) §293: That is to say: if we construe the grammar of the expression of sensation on the model of 'object and designation' the object drops out of consideration as irrelevant.

35. Wittenstein, L., *Remarks on the ...*, (cf.n.26) VII §70.

36. Bishop, Errett., *Foundations of Constructive Analysis*, New York (1967) *354*.

37. Wittgenstein, L., *Investigations*, (cf.n.22) §240.

38. Frege, Gottlob., *The Foundations of Arthmetic*, (cf.n.27), i.

39. McGuinness, B. ed., *Wittgenstein and the Vienna Circle*, Oxford (1979) *103*.

40. Wittgenstein, L., *Philosophical Remarks*, (cf.n.2) *189*.

41. Wittgenstein, L., *The Blue and Brown Books*, Oxford (1969) *4*.

42. Wittgenstein, L., *Remarks on the ...*, (cf.n.26) IV §5.

43. Frege, Gottlob., *Philosophical Writings of Gottlob Frege*, (Black, Max and Geach, Peter eds. and Transl.) Oxford (1970) *63*.

44. Frege, Gottlob., *Begriffschrift*, Hildesheim (1964) *15*.

45. Martin-Löf, Per., *About Models for Intuitionistic Type Theories and the Notion of Definitional Equality*, in Kanger, S., (ed.), *Proc. 3rd Scand. Logic Symp.*, Amsterdam (1975) *101*.

46. Ibid. *93*.

47. Ibid. *93*.

48. Kreisel, Georg., *Essay-review of Wittgenstein (1956)*, British Journal of Philosophy of Science, Vol. XI (1958), *135-58*, section *7*.

49. Tait, William., *Constructive Reasoning*, in B. van Rootselar and J. F. Staal eds. *Logic, Methodology and Philosophy of Science III*, Amsterdam (1968), *185-200*

50. Wittgenstein, L., *Philosophical Grammar*, (cf.n.19) *236.*

51. Gefwert, Christoffer., *A Participator: The Metaphysicsl Subject*, (cf.n.14) *34-6.*

52. Russell, Bertrand., *Introduction*, in Wittgenstein, L., *Tractatus*, (cf.n.30) *xxii.*

53. Maury, André., *Wittgentein and the Limits of Language*, Acta Philosophica Fermica, Vol 32 (1981) *154-5.*

54. Wittgenstein, L., *Philosophical Grammar*, (cf.n.19) *381-2.*

55. Wittgenstein, L., *Remarks on the* ..., (cf.n.26) I §154.

56. Ibid. III §42.

57. Ibid. VI §27.

58. Gefwert, Christoffer., *The Proposition - As - Rules Idea*, (cf.n.13) *16-17.*

59. Wittgenstein, L., *Investigations* (cf.n.22) §122.

60. Wittgenstein, L., *Remarks on the* ..., (cf.n.26) I §168.

61. Gefwert, Christoffer., *The Proposition - As - Rules Idea*, (cf.n.13) *17-19.*

62. Poincaré, Henri., *On the Nature of Mathematical Reasoning*, reprinted in Paul Benarerraf & Hilary Putnam (eds.) *Philosophy of Mathematics*, sec. ed., Cambridge (1983) *395-6.*

63. Wittgenstein, L., *Remarks on the* ..., (cf.n.26) I §66.

64. Dummett, Micheal., *Elements of Intuitionism*, Oxford (1977) *391-2.*

65. Beeson, Michael J., *Problematic Principles in Constructive Mathematics*, in D. van Dalen, D. Lascar and T. J. Smiley (eds.), *Logic Colloquium '80*, Amsterdam (1982) *15-16.*