BIT 34 (1994), 148–164.

PROOF VERSUS FORMALIZATION

PETER NAUR

DIKU Dept. of Computer Science, University of Copenhagen, Universitetsparken 1, DK 2100 Copenhagen Ø Denmark

Dedicated to Carl-Erik Fröberg on the occasion of his 75th birthday.

Abstract.

The use of informal and formal modes of expression in some proofs in mathematics and computing are examined. Gauss's 5th proof of the Law of Quadratic Reciprocity is shown to depend equally on informal and formal modes, with no dominance of either. A proof of an algorithm by Warshall uses informal and formal modes side by side. A flaw in the argument occurs at a highly formalized part of it. The proof is considered unconvincing by its use of indirect argument. For comparison, proofs by induction of the same algorithm are given, both in a more formal and in an informal form. The apparent advantage of Warshall's use of non-sequential algorithm description is shown to be deceptive. A transformation of a program in algorithmic form is shown to be readily provable. It is concluded that the important choice of the most advantageous description form cannot be reduced to strict criteria.

1. Introduction.

With a view to the often repeated claim that the development of effective computer programs depends on proving them correct, and the attendant claim that proof depends on formalization of the relevant description, the present paper will examine some actual proofs taken from mathematics and computing, so as to obtain some empirical basis for judging the claims.

2. Formalization in a proof by Gauss.

As an example of a proof at the highest level of mathematical accomplishment, a proof by Gauss will here be examined for its use of and reliance on formal modes of expressions. The proof to be considered is one of the so-called Law of Quadratic Reciprocity in the theory of numbers (quoted here from Smith [1959]). The primary reason for selecting this particular proof is that it is one on which Gauss, one of the greatest mathematicians of all time, has lavished quite extraordinary attention. In

Received November 1993.

fact, as reported by Gauss himself, the theorem under consideration occupied his thoughts for many years, and in his constant search for what he calls a natural proof he established no less than eight different proofs of it. Before Gauss, Legendre, the first one to express the theorem, had published a proof of it, which was, however, invalid, owing to its dependence on unproven assumptions. The proof that shall here be considered in some detail was Gauss's fifth proof, and one that he felt was not only rigorous, as were also his other proofs, but natural. It will be taken here to represent the notion of a mathematically sound proof.

As an indication of the subtlety of the issue of Gauss's proof, here is a brief statement of its content. The Law of Quadratic Reciprocity is a matter of congruences of the form

$$x^2 = p(\mathrm{mod}\, q),$$

expressing that x^2 and p are equal modulo q. Here p and q are relatively prime integers and x is an integer. The main question is whether for given p and q the congruence has any solution x, in other words, whether a square, x^2, may be found anywhere in the infinite sequence

$$p \bmod q,\ p \bmod q + q,\ p \bmod q + 2q,\ p \bmod q + 3q, \ldots$$

if for example $p = 7$ and $q = 3$ the sequence starts

$$\mathbf{1}, \mathbf{4}, 7, 10, 13, \mathbf{16}, 19, 22, \mathbf{25}, \ldots$$

having many squares as shown in bold type, while with $p = 3$ and $q = 7$ the sequence starts

$$3, 10, 17, 24, 31, 38, 45, 52, 59, 66, 73, 80, \ldots$$

having no square so far. The property of a pair of primes (p, q) to have or not have a solution x of their congruence $x^2 = p(\mathrm{mod}\, q)$ is called their quadratic character. The Law of Quadratic Reciprocity says that if both p and q are of the form $4n - 1$ then (p, q) and (q, p) have opposite quadratic character, otherwise, i.e. if p or q or both are of the form $4n - 3$, the character of (p, q) is the same as that of (q, p). Thus by the theorem, since both 3 and 7 are of the form $4n - 1$, the absence of squares in the second sequence shown above will persist no matter how long the sequence is continued.

Gauss's fifth proof of the Law of Quadratic Reciprocity has seven paragraphs. The first two contain general observations on proofs in the field of number theory. Paragraphs 3 to 6 establish some auxiliary notation and theorems, and paragraph 7 proves the main theorem. In the present analysis this overall structure is less important than the style of the finest details of expression. In terms of these details the total proof can be considered as a sequence of altogether 60 statements. These statements fall into three categories: (1) Let-statements, in which temporarily valid definitions are expressed; (2) Assertions, expressing currently valid states of affairs; and (3) Hypotheses, expressing states of affairs that are proved true in the text that

follows. In order to illustrate these categories, their use in relation to a short section of Gauss's text is as follows:

Gauss's text	Corresponding statement
Let the symbol (k, p) represent the number of products among	21. Let (k, p) ...
$k, 2k, 3k, \ldots k(p-1)/2$	
whose smallest positive residues modulo p exceed $p/2$. Further if x is a non-integral quantity we will express by the symbol $[x]$ the greatest integer less than x so that $x - [x]$ is always a positive quantity between 0 and 1. We can readily establish the following relations	22. Let $[x]$ ...
	23. Assert.
1. $[x] + [-x] = -1$	24. Assert.

In order to clarify the use of formalization in Gauss's proof we shall classify each of the 60 statements of which it consists according to the extent to which the substance of it is expressed formally. Since a clearcut characterization is neither possible nor necessary we shall use three classes: (1) Fully formal, (2) Partly formal, and (3) Informal. Used on the section quoted above, statement 21 is partly formal, 22 is informal, 23 is partly formal, and 24 is fully formal. Using these classifications on the 60 statements of Gauss's proof, the use of formalization is distributed as shown in the following table

Numbers of statements in Gauss's proof

Kind	Fully formal	Partly formal	Informal
Let-statement	5	4	10
Assertion	18	12	8
Hypothesis	2	0	1
Total	25	16	19

In a further enquiry into Gauss's use of formalization one might attempt to determine also the extent to which the logical connections between the statements of the proof are justified by means of formal manipulations. Formal manipulation certainly is used to a large extent. Every one of the 18 fully formal assertions has the form of an equation or congruence, and one such assertion is in many cases derived from previous ones by a process of formal substitution and rearrangement. However, as follows necessarily from the large number of partly formal or informal statements of the proof, the connection between statements in many other cases

depends entirely on informal descriptions. It must be recognized that Gauss in his proof depends equally on informal and formal modes of expression, with no clear predominance of either.

We may extend these quantitative observations of Gauss's mode of expression with qualitative observations of his manner of expression. We will then find that, as far as can be seen in his actual use of the modes, the difference between the informal and the formal modes is of no concern to him. He goes back and forth between the two modes, and mixes the modes in the same sentence, without the slightest concern or difficulty. As illustration, in the above quotation statement 23 does not shun talking about $x - [x]$ in the middle of an informal sentence, and the informal definition 22 is used to justify the entirely formal assertion 24.

Gauss's attitude to formalization reveals itself explicitly in the sentence appearing immediately before the passage quoted above, in which he says: "We can shorten the following discussion considerably by introducing certain convenient notations". This sentence says explicitly that the reason for using formal notations at this point is brevity and convenience. Implicitly the sentence indicates that the proof might be carried out, with unimpaired validity, even without the use of formal notation.

As the conclusion in this analysis, on the basis of the manner of expression used by Gauss it is impossible to claim that a formal mode is necessary for presenting a valid proof, while clearly certain of the informal parts of the proof are indispensable in tying the proof to an informal context of human insight and could not conceivably be replaced by formal statements. Thus, if we insist on regarding the informal and the formal modes as alternatives between which we have to choose, then for proper statement of a proof we have to retain the informal mode. This conclusion, however, indicates the misunderstanding underlying the insistence on regarding the modes as alternatives. A more fruitful view, and one that is in perfect accordance with Gauss's manner of expression, is to regard the formalizations as merely extensions of natural language. In this view it is always possible to express any formal statements that we have formed, equivalently in natural language. To achieve this we merely have to refer back through the chain of definitions of whatever formalizations are used, until we reach their ultimate basis in natural language.

3. An algorithm that needs proving.

As an example of proof and formalization in a computing context, the present and following sections present some reflections related to a particular algorithm and its first publication by Warshall [1962], titled "A Theorem on Boolean Matrices". This algorithm is of interest here for several reasons. It is a striking case of an algorithm that in spite of its simplicity needs a somewhat subtle proof for its justification. Further, the manner of its presentation, both as given originally by Warshall and as it was expressed soon after in Algol 60 by Floyd [1962], raises several issues of interest. Moreover, it is a highly useful algorithm, generating what according to one

way of looking at the problem is known as the transitive closure of a given relation, represented as a Boolean matrix.

As the basis of the discussion, the proof of the algorithm given by Warshall [1962] is reproduced here, apart from modifying the notation so as to replace the subscripts used to identify the matrix elements by expressions within square brackets.

THEOREM. *Given a square* $(d \times d)$ *matrix* M *each of whose elements* $m[i,j]$ *is* 0 *or* 1. *Define* M' *by* $m'[i,j] = 1$ *if and only if either* $m[i,j] = 1$ *or there exist integers* $k_1, \ldots, k_n$ *such thay*

$$m[i,k_1] = m[k_1,k_2] = \ldots = m[k_{n-1},k_n] = m[k_n,j] = 1.$$

$$m'[i,j] = 0, \textit{ otherwise.}$$

Define M^* *by the following construction:*[1]

0. Set $M^* = M$.
1. Set $i = 1$.
2. $(\forall j \ni: m^*[j,i] = 1)(\forall k)$ set $m^*[j,k] = m^*[j,k] \vee m^*[i,k]$.
3. Increment i by 1.
4. If $i \leq d$, go to step 2; otherwise, stop.

We assert $M^* = M'$.

PROOF. Trivially, $m^*[i,j] = 1 \Rightarrow m'[i,j] = 1$. For, either $m^*[i,j]$ was unity initially ($m[i,j] = 1$) – in which case $m'[i,j]$ is surely unity – or $m^*[i,j]$ was set to unity in step two. That is, there were, at the L_0th application of step two, $m^*[i,L_0] = m^*[L_0,j] = 1$. Each of these, similarly, either came directly from M or from a previous application of step two. Since there are exactly d applications of step two, this procedure is finite and leads to $m^*[i,L_A] = m^*[L_A,L_{A-1}] = \ldots = m^*[L_2,L_1] = m^*[L_1,L_0] = m^*[L_0,R_1] = \ldots = m^*[R_B,j] = 1$, where all the corresponding entries in M were unity. This is exactly the sequence required in the definition of M' (to within redundant elements which may simply be struck out) to imply that $m'[i,j] = 1$.

We have yet to prove that $m'[i,j] = 1 \Rightarrow m^*[i,j] = 1$. Assume this is false. Then there is a sequence of integers $i \neq k_1 \neq k_2 \neq \ldots \neq k_n \neq j$ such that $m[i,k_1] = m[k_1,k_2] = \ldots = m[k_n,j] = 1$, but $m^*[i,j] = 0$. Let $L = \{x \mid (1 \leq x \leq n)$ and $m^*[i,k_x] = 1\}$. Let λ be the largest element of L. Surely $m^*[i,k_\lambda]$ must have been changed from zero to unity by an application of step two (for if $m[i,k_\lambda] = 1$, since $m[k_\lambda,k_{\lambda+1}] = 1$, $m^*[i,k_{\lambda+1}] = 1$ by the k_λth step 2, which would contradict the definition of λ), the γth, say. This γ must be less than k_λ; for immediately after the k_λth iteration of step two, $(\forall p)$ $m^*[p,k_\lambda] = 1 \Rightarrow m^*[p,k_{\lambda+1}] = 1$. Any p_0 such that

[1] This definition by construction is equivalent to the recursive definition: 0. $(m[i,j])_0 = m[i,j]$; 1. $(m[i,j])_{n+1} = (m[i,j])_n \vee ((m[i,n+1])_n \wedge (m[n+1,j])_n)$; 2. $m^*[i,j] = (m[i,j])_d$.

$m^*[p_0, k_\lambda]$ is set to one *after* this will result from the p_1th iteration of step two when $m^*[p_1, k_\lambda] = m^*[p_0, p_1] = 1$ leads to $m^*[p_0, k_\lambda] = 1$. But if $m^*[p_1, k_\lambda] = 1$ at this time, then either $m^*[p_1, k_\lambda] = 1$ at the time of the k_λth iteration (in which case $m^*[p_1, k_{\lambda+1}] = 1$ also), or $m^*[p_1, k_\lambda]$ is set to one at the p_2th iteration where $k_\lambda < p_2 < p_1$. We thus generate a finite ordered set $p_1 > p_2 > \ldots > p_q > k_\lambda$, where $m^*[p_q, k_\lambda] = 1$ at the time of the k_λth iteration, whence $m^*[p_q, k_{\lambda+1}] = 1$ immediately after that iteration. Then the sequence of iterations designated by the p's will surely result in $m^*[p_0, k_{\lambda+1}] = 1$ after the p_1th iteration. Since p_0 was an arbitrary element, this is true if, in particular, $p_0 = i$. Thus, if $\gamma \geq k_\lambda, m^*[i, k_{\lambda+1}] = 1$, a contradiction.

But if $\gamma < k_\lambda$, then $m^*[i, k_\lambda] = m^*[k_\lambda, k_{\lambda+1}] = 1$ before the k_λth iteration, whence $m^*[i, k_{\lambda+1}] = 1$ after that iteration of step two, a contradiction. Therefore, the assertion is true. ■

4. Proof and formalization.

The first issue that shall be studied in relation to Warshall's presentation is the claim made frequently in discussions of the need for formal specifications of programs, that formalization is a necessary prerequisite of proof (see for example Ehrig et al. [1990]; Liskov and Zilles [1977]). Warshall's presentation is of independent interest in this connection, partly by dating from a time long before the discussion of formal specifications started, partly by addressing a real problem of program proving, and not a problem chosen with a view to demonstrating a specification technique. Moreover, Warshall's presentation clearly is patterned according to established mathematical practice and thus conforms to the idea often voiced, that program development techniques should be modelled on mathematics (see e.g. Hoare [1986]).

Looking at Warshall's presentation with a view to formalization, what do we find? Taken as a whole his text is a complicated mixture of formal expressions and informal prose statements. What are the formal expressions used for? Take as an example the core of the second part of the proof, which for convenience is given below with references added:

(1) We have yet to prove that $m'[i,j] = 1 \Rightarrow m^*[i,j] = 1$.
(2) Assume this is false.
(3) Then there is a sequence of integers $i \neq k_1 \neq k_2 \neq \ldots \neq k_n \neq j$
such that $m[i, k_1] = m[k_1, k_2] = \ldots = m[k_n, j] = 1$,
but $m^*[i,j] = 0$.
(4) Let $L = \{x \mid (1 \leq x \leq n)$ and $m^*[i, k_x] = 1\}$.
(5) Let λ be the largest element of L.

What may be noted first, is that formalization is used for less than it might have been. Not only the obvious case (5), but the whole pattern of the indirect demonstra-

tion in (1) and (2), as well as the consequence (3) might have been expresseed formally, using notation that has been well established long before the paper was written. Further, the dynamic aspect of the construction, the fact that each $m^*[i,j]$ has a value that changes in the course of the step-by-step construction, could very well have been reflected in the formal notation, but is not in the main text, only in a footnote.

On the other hand, the formalization which is in fact used is very helpful, in that the corresponding information could only with difficulty have been expressed informally. This holds particularly for the crucial concept introduced in (4), which would require a considerable amount of text in order to be defined informally.

The second point that may be noted in Warshall's formulation is that while formalized expressions are used to some extent to define concepts, there is no attempt to justify the logical connections constituting the proof by formal means. Basically, every step of the argument is described informally, in prose. This means in particular that when use is being made of a concept that has been described formally, the argument rests not directly on the properties of that concept as they can be derived mechanically from its formal definition, but on the understanding the human reader will form of the concept upon reading the definition. That this must be so for part of the argument steps follows from the fact that many of the relevant notions have only been introduced informally, as already noted. But in the manner in which the argument is presented there is no indication that the author considers the basic informality of it as in any way impairing its validity. The presentation conveys the clear impression that the author, and presumably the referees of the journal in which the paper has been accepted for publication, regards the form chosen as the proper one for the purpose of definitive proof of the theorem given, a view, incidentally, that the present writer finds entirely acceptable.

If we accept Warshall's form of presentation then one of the basic arguments that is currently made in support of the use of formal specifications in program development, to wit that specifications must be formalized in order to make proofs possible, falls to the ground.

But even if formalization is not necessary for proof, it might be argued, surely formalization will help make proofs correct. Looking again at Warshall's presentation this view is not confirmed. The point is that Warshall's proof has, if not an error then at least an imperfection. This appears in the lines following those quoted above as (1) to (5). These lines repeatedly refer to $m[k_\lambda, k_{\lambda+1}]$, but if λ is $=n$, as quite possible according to (5) and (4), then $k_{\lambda+1}$ becomes k_{n+1}, which is not defined in (3). Thus the whole of the following argument fails to cover the case that $\lambda = n$.

This imperfection is not serious, in order to remove it we may add the definition $k_{n+1} = j$ to (3), extend (4) by replacing n by $n + 1$, and observe that even with these extensions $\lambda = n + 1$ is excluded as a possibility because of $m^*[i, k_{n+1}] = m^*[i,j] = 0$ according to (3). With these extensions the rest of the argument can be retained without change.

The point of this description of an imperfection in Warshall's argument is that the

flaw is connected with the most highly formalized part of the presentation. In fact, the conflict is between the upper bound of x according to (4), the definition of the ks in (3), and the use of $k_{\lambda+1}$ in the following lines.

With these examples at hand it appears that we may conclude that it is not generally true that proofs of algorithms depend on formalization of specifications, and it is not generally true that formalization is particularly helpful in avoiding flaws in program proofs.

5. Making a proof convincing.

Is Warshall's proof convincing, and if so, what makes it so? This question, by its very nature, calls for a personal answer, the power to convince being not something that can be measured objectively. My answer is that, assuming the slight imperfection that was discussed above to be repaired, his proof is quite convincing, but that in certain details it is less transparent than it might easily be, and so might gain from some revision.

This answer reflects a certain implicit understanding of what a proof is and what it should be designed to achieve. Briefly, this understanding can be characterized thus: a proof is an understandable, coherent argument, purporting to support some statement about some state of affairs. Generally a proof will consist of a series of assertions that support each other. For helping the understanding it is important that the concepts employed are clearly defined and consistently used, and that the assertions are so constituted and arranged that the overall structure of the argument, as well as the significance of each part of it, are conveyed effectively to the reader.

In many respects, Warshall's proof is satisfactory. The part of it that appears to be unnecessarily difficult to follow, and therefore is not optimally convincing, is the last part, the one following the section quoted in (1) to (5) above. One reason for the difficulty is that two very similar arguments are given twice, at a separation of eleven lines of intricate argumentation, namely first the argument that $m[i, k_\lambda] = 1$ is impossible, and later the argument that $m^*[i, k_\lambda] = 1$ before the k_λth iteration is impossible. Another reason is that the argumentation showing that "γ must be less than k_λ" appears at first to be given in the immediately following line, while a closer study will show that this particular line is only a relatively insignificant part of the argument, which extends for about ten further lines. These ten lines present an indirect argument, but this is only made clear at the very end, at the words "a contradiction". As further unnecessary complications, the letters i, j, and k, are used differently in the proof and in the construction, the quantity i is temporarily renamed p_0, and half through the proof the word "application" is replaced by "iteration".

As a concrete illustration of these remarks, here is a revised version of the second part of Warshall's proof that has been designed to avoid these difficulties.

We have yet prove that $m'[f,g] = 1 \Rightarrow m^*[f,g] = 1$. Assume this is false. Then there is a sequence of integers

$$f \neq h_1 \neq h_2 \neq \ldots \neq h_n \neq h_{n+1} = g \text{ such that}$$

$$m[f,h_1] = m[h_1,h_2] = \ldots = m[h_n,g] = 1,$$

but $m^*[f,g] = 0$.

Let $L = \{x \mid (1 \leq x \leq n+1) \text{ and } m^*[f,h_x] = 1\}$.

Let λ be the largest element of L (L cannot be empty since 1 is a member of it in virtue of $m[f,h_1] = 1$). This λ must be $< n+1$, since otherwise

$$m^*[f,h_{n+1}] = m^*[f,g] = 1, \text{ a contradiction.}$$

The value $m^*[f,h_\lambda] = 1$ must have arisen in either of four different ways:

(1) By $m[f,h_\lambda] = 1$ initially,

or by being changed from zero to one by the p_1th application of step 2, where either

(2) $p_1 < h_\lambda$, or

(3) $p_1 = h_\lambda$, or

(4) $p_1 > h_\lambda$.

But if either (1) or (2) holds, at the h_λth application of step 2, $m^*[f,h_\lambda] = m^*[h_\lambda,h_{\lambda+1}] = 1$, whence $m^*[f,h_{\lambda+1}] = 1$ after that application, a contradiction of the definition of λ. (3) is excluded because the h_λth application of step 2 leaves $m^*[f,h_\lambda]$ for any f unchanged. If (4) holds we have before the p_1th application of step 2: $m^*[f,p_1] = m^*[p_1,h_\lambda] = 1$. If $m^*[p_1,h_\lambda]$ is not one at the time of the h_λth application it must have been set to one in application p_2, where $h_\lambda < p_2 < p_1$, and just before this application we must have $m^*[p_1,p_2] = m^*[p_2,h_\lambda] = 1$. Continuing this argument we generate a finite ordered sequence $h_\lambda < p_q < \ldots < p_2 < p_1$ such that just before the time of application p_t we have $m^*[p_{t-1},p_t] = m^*[p_t,h_\lambda] = 1$, and at the time of the h_λth application $m^*[p_q,h_\lambda]$, which cannot change during that application, is one. Thus in that application $m^*[p_q,h_{\lambda+1}]$ is changed to one, and the subsequent sequence of applications $p_q,p_{q-1},\ldots,p_1$ will make $m^*[f,h_{\lambda+1}] = 1$, a contradiction. Therefore, since each of the four conceivable ways of forming $m^*[f,h_\lambda] = 1$ has been found to be impossible, the assumption of the indirect argument has been found to be false, and thus the original assertion is true. ■

When this revision has been done it must be recognized that a difficulty remains, albeit at a different level. One cannot escape the impression that Warshall's proof is difficult. Part of the difficulty is the definition of the set L and the quantity λ, which depend on a special invention, or trick. Another part is that the mode of the proof is indirect, thus paying most attention to hypothetical, impossible circumstances. This difficulty stands in strong relief when compared with the proof given for an algo-

rithm formed by generalization of Warshall's, by Aho and Ullman [1972], which uses induction over i.

Adapted to the present problem the proof by induction builds on the following *induction hypothesis*:

Immediately before the sth application of step 2 $m^*[f,g] = 1$ if and only if either $m[f,g] = 1$ or there exist integers $h_1, \ldots, h_n$, $1 \leq h_t \leq s-1$ for $1 \leq t \leq n$, such that

$$m[f,h_1] = m[h_1,h_2] = \ldots = m[h_{n-1},h_n] = m[h_n,g] = 1,$$

$$m^*[f,g] = 0, \text{ otherwise.}$$

Taking this as a hypothesis, the task of step 2 will be to change the elements of M^* so as to give the value 1 to those that depend on a chain of original elements including the index values up to s, and not only to $s-1$. In other words, the task is to set $m^*[f,g]$ to one in case there exist integers $h_1, \ldots, h_n$, $1 \leq h_t \leq s-1$ for $1 \leq t \leq n$, such that

$$(6) \qquad m[f,h_1] = m[h_1,h_2] = \ldots = m[h_a,s] = m[s,h_b] = \ldots$$
$$= m[h_c,s] = m[s,h_d] = \ldots = m[h_{n-1},h_n] = m[h_n,g] = 1.$$

In clarifying this task of step 2 it should be noted first that where the sequence of original elements in this expression includes more than one pair having s as the common index, such as the pairs involving h_a, h_b, h_c, and h_d, a sequence will always exist that involves only one pair. This will be obtained merely by removing the elements that both start and terminate with index s, such as the sequence $m[s,h_b] = \ldots = m[h_c,s]$ in (6). Thus in determining whether a sequence such as (6) exists, it is sufficient to take into account only sequences having s in one pair, such as

$$(7) \qquad m[f,h_1] = m[h_1,h_2] = \ldots = m[h_a,s] = m[s,h_d] = \ldots$$
$$= m[h_{n-1},h_n] = m[h_n,g] = 1.$$

In order to determine whether such a sequence exists for a given matrix element $m^*[f,g]$ it is clearly sufficient to determine whether the two separate sequences

$$(8) \qquad m[f,h_1] = m[h_1,h_2] = \ldots = m[h_a,s] = 1$$

$$(9) \qquad \text{and} \quad m[s,h_d] = \ldots = m[h_{n-1},h_n] = m[h_n,g] = 1$$

exist. But since these sequences involve only hs that are $< s$, by the induction hypothesis their existences are given by the elements $m^*[f,s]$ and $m^*[s,g]$. Consequently the task of step 2 will be to set to one all those elements $m^*[f,g]$ for which $m^*[f,s] = m^*[s,g] = 1$. This is precisely what is done by step 2 of Warshall's construction. Thus provided the induction hypothesis holds immediately before the sth application of step 2, it will also hold immediately before the $(s+1)$th application. To complete the induction proof of the construction it then only remains to make sure that the induction hypothesis holds before the first application of step 2, which is trivally correct.

Comparing this induction proof to Warshall's proof it appears that the induction proof again depends on invention, namely the induction hypothesis. This, however, expresses a correct, constructive insight into the process performed by the algorithm, and thus is a more useful item, and one that presumably is easier to understand than the basis of the indirect proof, the impossible quantity λ.

As conclusion of the present section, the discussion of the form chosen by Warshall for his proof may support the suggestion that what makes a proof convincing, at least to some persons, does not depend on the extent to which it is expressed formally, but on quite different matters of its organization and coherence, in particular those that help the reader to see clearly the significance of every part of it.

6. Proof without formalization.

As further support of the concluding suggestion of the previous section the present section will present a concrete case of a proof which does not depend on formalized notation. The case to be considered is the proof of Warshall's Theorem by induction.

As the basis of the construction of the proof, let a *connection chain* of the given matrix M be defined to be a finite sequence of at least two integers, each between 1 and n, both included, with the following additional property: taking each pair of immediately successive integers in the connection chain as the first and second index of an element of the given matrix M, each such element in the matrix is one. Further define the first integer of a connection chain to be the *initial member* and the last integer to be the *final member* of the chain, and the remaining integers to be the *intermediate members* of the chain. Define an *s-limited* connection chain to be one in which each intermediate member is between 1 and s, both included.

On this basis the induction hypothesis of the proof of Warshall's construction may be expressed as follows: immediately before the ith application of step 2 of the construction, an element of the matrix M^* is one if and only if there is an $(i-1)$-limited connection chain having the indices of the element as its initial and final members.

The proof of the construction proceeds by showing that by the action of step 2 the elements of M^* will be changed so as to be one if and only if there is an i-limited connection chain having the indices of the element as its initial and final members. For this to be accomplished the elements corresponding to connection chains having i as intermediate member have to be set to one if they are not one already. In the process for achieving this, connection chains in which i occurs as intermediate member more than once can be ignored, since any such connection chain implies the existence of a chain with the same initial and final member having i as intermediate member only once. For such a chain to exist, both its first part having i as its final member and its second part having i as its initial member, must exist. Each of these two parts are $(i-1)$-limited connection chains, and so their existence has already

been recorded in M^*. Consequently an element $m^*[f, g]$ has to be set to one in step 2 provided the two elements $m^*[f, i]$ and $m^*[i, g]$ that indicate chains from the first index of the element to i and from i to the second index of the element are one. When all elements of M^* have been changed according to this prescription the matrix properly records the i-limited connection chains, thus confirming the induction hypothesis.

From this demonstration of a proof without formalization it becomes clear that what may be accomplished by formalization is not mechanical inference or infallibility of argument, but rather convenience and effectiveness of description. The choice between informal and formal description cannot be decided by any strict criteria. Either form of description allows infinite variation and real descriptions are likely to included both formal and informal parts. Thus in the above description, the phrase "connection chain" has been defined to have a strictly delimited meaning in the context, and may be considered to be used formally.

7. Using set notation as a programming language.

Warshall presents a proof of an algorithm. But he uses no established programming language, and has no formal description of neither his syntax nor his semantics. How does he achieve this? Let us look at the process description of Warshall's theorem. As the first point of interest, although the article in which the theorem appears uses the term algorithm, within the theorem itself the step-by-step actions that are of interest are called a construction. One may conjecture that the author has wanted to express his theorem entirely in respected mathematical terms and so has chosen his designation to be the one used by mathematicians in stating theorems dealing, for example, with classical geometrical constructions using ruler and compass.

As the second point to note in Warshall's Theorem, the description of the construction employs five steps, numbered 0 to 4, showing clearly the sequencing of certain actions, while step 2 is expressed by means of a mixture of elementary assignment, designated with the aid of the imperative verb "set", and quantifiers of set notation. The description of step 2 raises several questions that will now be considered.

First, it must be noted that the use of the quantifiers does not conform to established notions of mathematical logic. In logic the expression $(\forall k)(p(k))$ is taken to designate the assertion: for all values of k the expression $p(k)$ will yield the value true. This is a static assertion, it does not emply an ordering of the values of k, since the evaluation of $p(k)$ does not change anything, and so an ordering could not possibly make any difference to the meaning of the expression. Indeed, the notation does not imply that $p(k)$ is evaluated any particular number of times, not even for each separate value of k, nor that it is evaluated at all.

The use of quantifiers in Warshall's step 2 is quite different. What is quantified

here is the description of an action, the assignment. Since the assignment may change something upon being done, this use of the quantifiers raises nontrivial questions about the possible ambiguity of the description. Consider for example the following expression, which is superfically similar to Warshall's step 2:

$$(\forall j)(\forall k) \text{ set } m[j,k] = \text{not } m[k,j].$$

Here the resulting change depends entirely on the order in which the index pairs are taken. Moreover, it depends on an understanding of how many times the action has to be done for each possible index pair. Without a specific understanding on this point it might be argued that the action is non-terminating because of the treatment of the elements having $j = k$.

On this background it is important to notice that with the particular action given in Warshall's step 2, the resulting change is unambiguously defined, irrespective of the order in which the index pairs are taken. This claim may be verified by the following argument. First, no element $m^*[j,k]$ having $j = i$ or $k = i$ is changed. For example, a change of $m^*[i,k]$ can only be achieved when the assignment is done with $j = i$, and thus by

$$\text{if } m^*[i,i] \text{ then } (\forall k) \text{ set } m^*[i,k] = m^*[i,k] \vee m^*[i,k]$$

which clearly will not change any element. Second, for an arbitrary element $m^*[j,k]$ the change induced by the step can only be that the value 0 is changed to 1. Third, the necessary condition that the element $m^*[j,k]$ is changed in the step is that $m^*[j,i] = m^*[i,k] = 1$ and thus depends only on elements that cannot themselves be changed in the step. It follows that the change of any element described by the step is independent of the change of any other element. Consequently the changes of elements that are prescribed in the step will be the same, no matter in which order they are made.

This demonstration does not explicitly dispose of the question whether Warshall's step 2 terminates. We have to add, as an implicit assumption, that the notation of step 2 prescribes that the assignment is performed just once for each pair of indices included by the quantifiers. This assumption is the necessary basis for the remarks on the running time behaviour of the algorithm made elsewhere in Warshall's paper.

8. Non-sequential algorithm description.

The discussion of the notation used by Warshall in his theorem suggests a further enquiry into the advantages or otherwise of certain notational forms. As an obvious issue, what is the advantage of the notation used by Warshall in his step 2? This question should be considered on the background of, on the one hand, the remaining steps of his construction, which are expressed in terms of strictly sequential, elemen-

tary actions, and on the other hand, a consistently algorithmic description, such as the Algol 60 program given by Floyd [1962].

The form of Warshall's step 2 might be considered advantageous for, at least, the following two reasons: (1) Since no ordering of the index pairs can be imposed by the notation, the description shows that the implementor is free to choose any ordering, or even simultaneous execution of the assignments. (2) Using established mathematical notation the form lends itself directly to the use of well known techniques of mathematical proof. Let us consider these two issues in turn.

Concerning issue (1) it must be noted that, as shown in the previous section, the notation by no means assures the property that it suggests, namely invariance of the results of step 2 under different orderings of the index pair. That invariance is assured only by a separate argument, such as the one given above, that depends strictly on the particular action controlled by the quantifier clauses. Thus the unambiguous use of the notation of Warshall's step 2 depends on special care, while the notation will readily permit expressions that are ambiguous in manners that reveal themselves only to a close analysis. For this reason the alleged advantage of the notation, that it shows the freedom open to the implementor in choosing the order in which the index pairs are taken, may be said to be deceptive. The cautious implementor would not rely on the form of the expressions to indicate this property, but would demand additional confirmation that the property does indeed pertain, possibly in the form of an additional statement to that effect, or a demonstration of the invariance.

As a further perspective on the notation of Warshall's step 2, let us compare it with the corresponding part of the Algol 60 version of the algorithm given by Floyd [1962]:

for $j := 1$ **step** 1 **until** n **do**
if $m[j, i]$ **then**
for $k := 1$ **step** 1 **until** n **do**
if $m[i, k]$ **then**
$m[j, k] :=$ **true**

In this form there is no suggestion that the order of treatment of j and k could be different. However, if the idea to change the order occurs to the reader, or is made in an added suggestion in a comment to the program, then the situation is essentially similar to that pertaining to Warshall's step 2 above. Again the cautious reader would demand or construct a proof of the invariance before relying on it.

The second question raised in relation to Warshall's notation, its convenience in proving properties of the algorithm, may be illustrated in analyzing a possibility for avoiding some superfluous operations. The suggestion for this possibility is close at hand in the above verification that the order in which the index pairs are taken is immaterial to the result. The first point of that verification is that no element $m^*[j, k]$ having $j = i$ or $k = i$ is changed. This statement indicates that in taking all values of j it is permissible to omit $j = i$, thus saving the superfluous handling of all

values of k for that value. The change to effect this saving can readily be incorporated in Warshall's step 2 to yield:

2. $(\forall j \ni: m^*[j,i] = 1 \wedge j \neq i)(\forall k)$
   set $m^*[j,k] = m^*[j,k] \vee m^*[i,k]$

and in Floyd's Algol 60 program to yield in the second line:

if $m[j,i] \wedge j \neq i$ **then**

We are here dealing with a program transformation, and may consider the verification that it does in fact leave the effect of the program unchanged. More particularly, in view of the claim sometimes made (e.g. by Backus [1978]), that conventional programming languages do not lend themselves to formal manipulation, such as invariance proofs, it is of interest to compare the forms the verifications take in the two notations. In the case of Warshall's notation the verification was already given above, when it was observed that for $j = i$ the assignment action cannot produce any change of variables. In Floyd's program we similarly have to determine the effect of executing his lines three and following for $j = i$:

for $k := 1$ **step** 1 **until** n **do**
if $m[i,k]$ **then**
$m[i,k] :=$ **true**

This, again, in a no less obvious manner, has no effect. Thus in this example there is no indication that a proof of the validity of the program transformation is easier or more direct when expressed in terms of set notation than when Algol 60 is used.

As the conclusion of this section, the use of a non-sequential form of process description in this case has given no particular advantage over an equivalent sequential form, neither for the purpose of indicating such actions that do not have to be executed in any definite order, nor in proving formally the validity of a transformation of the program.

9. Selection of description form.

The observations of the previous sections indicate a close interplay between argumentation and description forms. The difference between Warshall's proof and the proof of his algorithm by induction is primarily a matter of what is described. Warshall's proof describes the set L and the quantity λ, while the proof by induction describes the state of the algorithm by a general snapshot, the induction hypothesis. Through the latter description the construction of the algorithm suggests itself directly, and the proof becomes practically implied in the construction.

The choice between informal and formal description is a matter of how matters are described. In formal description the ordinary notions of the natural language are supplemented with special notions and notations given specific meanings through strict definitions. Depending on the selection of the special notions and notations

used the description of the relevant items may become intuitively more or less effective.

However, the crucial issue in this context is the judiciousness of the selection. This tends to be ignored by adherents of formal specifications of computer programs, who instead tend to claim special advantages for some particular form or style of formal description. By such an approach the effectiveness of the descriptions will become entirely dependent on the matters that are described, and for certain matters may turn out to be heavy and inconvenient.

As illustration, consider the concept of permutation. Permutations are of course well known from elementary mathematics. The notion is introduced by Knuth [1968] in these words:

A *permutation of n objects* is an arrangement of *n* distinct objects in a row.

To an author such as C. B. Jones [1986], who advocates a particular form of notation, VDM, describing permutations becomes "an interesting exercise", leading to a description such as the following truth valued function:

$$
\begin{aligned}
&\textit{is-permutation}\ (s_1, s_2)\ \underline{\Delta} \\
&\quad \text{len}\ s_1 = \text{len}\ s_2\ \wedge \\
&\quad \exists m \in \text{map}\ \mathbf{N}\ \text{to}\ \mathbf{N} \cdot \\
&\qquad \text{dom}\ m = \text{rng}\ m = \text{dom}\ s_1\ \wedge\ \forall i \in \text{dom}\ s_1 \cdot s_1(i) = s_2(m(i))
\end{aligned}
$$

The advantage of this form over the informal description by Knuth given above will most likely remain forever a question of personal style preference.

10. Conclusions.

By the evidence of the present paper several of the commonly adopted doctrines related to proof and formalization appear to be, most likely, false. This holds for the claims that formalization is required for proof and for the claim that by the use of formalization human errors may be avoided or their frequency reduced. As seen here, proofs may be formulated in terms that make more or less use of formalization, being in any case dependent on a context that may be described only informally. The use of formalization will in any case be a matter of choosing a form of description. That choice cannot be reduced to strict criteria, important concerns being the subject of the description and the preferred style of whoever has to make use of the description.

REFERENCES

Aho, A. V. and Ullman, J. D., *The theory of parsing, translation, and compiling, vol.* 1, Prentice-Hall, Englewood Cliffs, 1972.

Backus, J. W., *Can programming be liberated from the von Neumann style? A functional style and its algebra of programs*, Comm. ACM 21 (8), Aug. 1978, 613–641.

Ehrig, H., et al., *Compatibility problems in the development of algebraic module specifications*, Theor. Comput. Sci. 77, 1 & 2, 1990, 27–71.

Floyd, R. W., *Algorithm* 97, Comm. ACM vol. 5 (6), 1962 June, p. 345.

Hoare, C. A. R., *Mathematics of programming*, BYTE Magazine, 1986; also Timothy R. Colburn et al. (eds.), *Program Verification*, Kluwer, Netherlands, 1993, 135–154.

Jones, C. B., *Systematic Software Development using VDM*, Prentice-Hall, 1986, p. 183–84.

Knuth, D. E., *The Art of Computer Programming, vol.* 1, Addison-Wesley, 1968.

Liskov, B. and Zilles, S., *An introduction to formal specifications of data abstractions, in Current Trends in Programming Methodology, vol.* 1, Prentice-Hall, 1977.

Smith, D. E., *A source book in mathematics, vol.* 1, Dover, New York, 1959.

Warshall, S., *A theorem on Boolean matrices*, J. ACM 9 (1962), 11–12.