Lecture notes on the structure of programs and proofs.

"I have also included a short chapter on equipment, from the angle of how little you can get away with, rather than how much you need."
Jocasta Innes "The Pauper's Cookbook"
(Penguin Books, 1971)

"The need to prove a program correct is primarily an aesthetic one (see A.P. Ershov, Computer Bull., 1972, p. 352) and thus only the best will do; by the same token, however, the standards of proof vary with the aesthetic awareness of the programmer. It may be noted that a very similar observation can be applied to mathematics."
W.M. Turski "Computer Programming Methodology"
(Heyden, 1978)

Admittedly, this lecture series is very tentative, but even when it raises more questions than I can give answers, I shall be content when the questions are sensible. Let me tell the main considerations that led to this undertaking.

To begin with, there is a more than superficial analogy between computer programs and mathematical proofs. Both are discrete artefacts. (Note that also in the case that a mathematical theorem deals with the continuum, its proof is discrete in the sense that it

must be carried out in a finite number of steps.) Furthermore – probably as a consequence of their being discrete – programs and proofs share the property that, essentially, we know of only one way of getting them right, viz. by going very carefully through them. Trying to establish the correctness of a program by testing it is as pointless as trying to establish the correctness of a proof by observing a number of instances in which the theorem holds. A leading theme should therefore be the investigation to what extent and to what benefit this analogy can be exploited.

To the observed analogy I would like to add the strong belief that the benefit can act both ways and can be considerable. (This was not a very good sentence. Sorry!) That computing science can benefit from 25 centuries of mathematical experience goes, I think, without saying. The observation that the automatic computer embodies some sort of revolution is a trite remark; yet it is true. Thanks to its speed and its capacity it is dramatically different from anything we had before and the challenge to program it well is truly without precedent in our culture. Considered as a formula, a program is usually several orders of magnitude longer than the formulae to the manipulation of which traditional mathematical practice is geared. As a result, it is far from unthinkable that the computing scientist has a greater awareness than the average mathematician of the economies in-

volved.

Rules of our game.

We are not prescribing the law to anybody, we are not even attempting to propose something for standardization, but have rules for ourselves. In particular, we shall try to apply the techniques of scientific thought to the best of our ability (and, quite likely, not far beyond our awareness of them).

A major technique is now known as "separation of concerns". It refers to the way in which we can do justice to the fact that our heads are so small. When faced with something still entangled, we try to isolate different aspects in the hope of dealing with them in turn and in isolation. How fortunate one's choice of "aspects" to be dealt with in isolation is depends on many circumstances: the "aspect" to be studied should, in one way or another, be sufficiently small to fit in our heads, yet it should encompass enough to deserve, at least for some time, our exclusive attention. One thing, however, is certain: the fortunate choice of aspect, the fortunate isolation of some concern, is, as a rule, immediately recognized as such. (A nice example is the use of BNF to capture the context-free aspect of programming language grammars. Suggested by John Backus in

in the summer of 1959, it made half a year later, as used by Peter Naur, the famous Report on the Algorithmic Language ALGOL 60 a monument in its time.)

We shall immediately apply this technique by separating the concern about general acceptance from all the others. There is nothing wrong with paying great care to the presentation of your material in an effort to reach your audience, but the care should be justified by the quality of the material. I have seen too many programming language design efforts paralyzed by the tacitly accepted constraint not to frighten the "average programmer" by something unusual to be still willing to confuse the scientist with the salesman. Recommendations that something is "natural", "intuitively appealing" or "easy-to-read" should be viewed with the gravest suspicion: such recommendations fail to distinguish between convenient and conventional. (When, recently, a professor of the Department of Electrical Engineering & Computer Science of MIT defended a clumsy technique with "But our people like to work with pictures.", he failed to convince me.)

Occasionally it may seem that I am carrying the principle of separating my concerns too far. I, too, have a past that I remember, and that past includes experience with designs so large that every

attempt at disentanglement had to be tried. The examples I can show in this lecture series are, however, of necessity small.

My insistence on distinguishing between convenient and conventional implies that I have adopted another yardstick for quality than familiarity. Comparing different arguments leading to the same conclusion, I have come to the opinion that some arguments are objectively simpler than others, and this with the same objectivity in which arithmetic with Arabic numerals is simpler than with Roman numerals. You may call the quality I am after "mathematical elegance"; that name is perfectly adequate provided we dissociate "elegance" from fashion and personal taste: among a great number of mathematical colleagues I found a much greater consensus about what was really elegant than they had suspected. For the time being it suffices to capture mathematical elegance by the slogan "short is beautiful".

Because the reader may find me using the first-order predicate calculus more readily than he is used to, a word of explanation is in order. The historical explanation is easy: over the last ten years the predicate calculus became an indispensable tool in our daily reasoning. It derives its superiority over verbal arguments from more than its precision and

brevity. It allows the development of a fairly uniform style with such simple rules of formula manipulation that one can let the symbols do the work. (The objection that such formalisms hamper the development of "mathematical insight" or "understanding" should be countered by the question "How do you understand that 112 is a multiple of 7?")

To avoid any misunderstanding, let me state quite explicitly that it is not my purpose to erect some formal system; I gladly leave that to the logicians, the philosophers, and the mathematicians interested in the so-called foundations of their field. I fully respect how formalists constrain themselves, but in my more sober moments I am more pragmatic and see no specific virtue in being a formalist: I shall use formal methods when I feel they help. If you need a next slogan by way of summary, "rather rigorous than formal" will do.

The relation to heuristics.

The completed artefact, be it program or proof, is one thing. How do you make it — the topic of heuristics — is a completely different matter.

Remark. The (on the average more Platonic) mathematician tends to "discover" a proof; the computing

scientist, who feels more akin to the engineering profession, tends to "design" his programs. This difference in terminology does not invalidate our analogy. It does reveal a difference in attitude, perhaps even a difference in methodological awareness. (End of Remark.)

In the majority of the mathematics curricula today, heuristics is a neglected topic. Mathematical results are taught quite openly and explicitly, but on how these results have been achieved and, more generally, on how to <u>do</u> mathematics the curriculum is usually remarkably silent (a silence I have heard defended by the dogma "mathematicians are born, not made"). In this lecture series I expect to ignore heuristics too, though it may become my target in the years to come.

Here, rightly or wrongly, I am relying on the analogy between programs and proofs, an analogy which is the inspiration behind my effort to transfer what programming methodology has taught us to mathematical methodology in general. Let me therefore give a (very) short summary of how programming methodology was born, and what was the result.

Programming methodology became a topic of explicit academic concern after the so-called "software crisis"

had been openly admitted at the NATO Conference on Software Engineering at Garmisch-Partenkirchen in 1968. People starting to figure out how to program, quickly discovered that this question was empty until we chose what kind of program one was heading for.

As a result, the first years of programming methodology have been characterized as "a morbid pre-occupation with style". In retrospect I think this was a wise thing to do: it has saved us the labour of formalizing all sorts of programming habits that were already rejected on aesthetic grounds. And once the formal discipline had emerged, it became quickly accepted as an objective yardstick: programming language features, the semantics of which were hard to formalize elegantly, became ipso facto suspect. The above summary of recent history has been given just in case one of my readers feels that, in my discussions of proofs, I display a similar "morbid pre-occupation with style".

In its later years, programming methodology caused an inversion of the way in which the problem of program correctness was approached. Instead of trying to prove the correctness of an already given program, we learned to choose the structure of the correctness proof first and to derive from that the program so as to suit the proof's needs. This inversion made the whole exercise of great heuristic

value. Let me give you, by way of illustration, an example from another area.

Problem. A two-person game consists of placing on an, initially empty, rectangular table in turn a guilder with its face flat on the table. (Guilders, already present, may not be moved.) The game ends when no next guilder can be placed; the last player to place a guilder has won the game. Question: does there exist a winning strategy for one of the players? (End of Problem.)

The reader is free to stop here for a while in order to think about the problem himself.

Answer. Because with a very small table - e.g. a table that admits only one guilder - the first player wins, there doesn't exist a winning strategy for the second player, but we may look for one for the first player. Instead of looking for such a strategy directly, we ask ourselves: what would we have to prove about a proposed winning strategy for the first player? Well, that he can terminate the game with an odd number of coins on the table. He can answer each move of his opponent when all coins, except the first one to be placed, are paired. The first player places the first coin in the centre of the table, and in each subsequent move he mimicks his opponent's behaviour, but rotated around the centre over 180°. (End of Answer.)

I know from experience that among intelligent, knowledgeable, and otherwise well-trained people the time needed for finding this answer can vary from several minutes to several evenings. Clearly, someone had acquired thinking habits that are hard to recommend.

On one inference rule.

Let me state explicitly – to forestall fruitless discussions – that, in the following, I shall confine myself to two-valued logic and shall accept for any proposition P that I shall write down the Law of the Excluded Middle, i.e. $P \vee \neg P$. By definition, the constants T and F satisfy T and $\neg F$ respectively. I shall freely make use of well-known equalities such as

$A \vee B = B \vee A$
$A \vee F = A$
$A \vee A = A$
$A \vee T = T$
$A \vee (B \wedge A) = A$ (Rule of Absorption)
$\neg(A \vee B) = \neg A \wedge \neg B$ (de Morgan's Law)
$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$ etc.

and the equivalent set of formulae obtained by replacing in the above the ordered quadruple

$(\vee, \wedge, T, F)$ by $(\wedge, \vee, F, T)$. An appeal to the above equalities will be made in what we shall denote by "boolean simplification".

Essentially we shall use only one inference rule

(0)
$$\frac{\begin{array}{c} A \vee B \\ C \vee D \end{array}}{A \vee C \vee (B \wedge D)}$$

Note. With boolean simplification, this inference rule subsumes the time-honoured "modus ponens", usually written as

$$\frac{\begin{array}{c} A \Rightarrow B \\ B \Rightarrow C \end{array}}{A \Rightarrow C} \quad .$$

I prefer to regard $A \Rightarrow B$ as an asymmetric notation for the disjunction $\neg A \vee B$ or $B \vee \neg A$, and would rewrite the modus ponens

$$\frac{\begin{array}{c} \neg A \vee B \\ C \vee \neg B \end{array}}{\neg A \vee C \vee (B \wedge \neg B) = \neg A \vee C \vee F = \neg A \vee C} \quad .$$

(End of Note.)

There are various reasons for avoiding the use of the implication sign in the formulation of theorems

proved or to be proved. A well-known pattern of proving a theorem A is by proving two (weaker) lemmata first:

lemma 0: $A \vee C$

lemma 1: $A \vee \neg C$

$$A \vee A \vee (C \wedge \neg C) = A \quad .$$

Formulated in terms of implications, this becomes

$$\frac{\neg A \Rightarrow C \qquad \neg A \Rightarrow \neg C}{\neg A \Rightarrow F}$$

, hence A

or

$$\frac{\neg C \Rightarrow A \qquad C \Rightarrow A}{T \Rightarrow A}$$

, hence A

In the left-hand formulation it is called a "proof by reductio ad absurdum", in the right-hand formulation it is called a "proof by case analysis". Since the one formulation can be translated mechanically into the other, the whole supposed distinction between the two is no more than an artefact of an asymmetric notation of a symmetric situation. A notation that induces many a mathematician to believe that a meaningless distinction is meaningful should inspire the gravest suspicion indeed.

Example. About nine mathematicians visiting an international congress we are invited to prove

$$A \vee B \vee C$$

with

A: there is a triple of mathematicians that is incommunicado (i.e. such that no two of them have a language in common)

B: there exists a mathematician mastering more than three languages

C: there exists a language mastered by at least three mathematicians.

With the auxiliary propositions

D: there exists a mathematician that can communicate with more others than he masters languages

E: there exists a mathematician that can communicate with more than three others,

we have

Lemma 0: $C \vee \neg D$

Lemma 1: $B \vee D \vee \neg E$

Lemma 2: $\dfrac{A \vee E}{A \vee B \vee C}$ by applying (0) twice.

We would like the reader to convince himself that Lemma 0 is "obvious" in the sense that one can start as well by observing

$C \vee$ "each mathematician communicates in different languages with those others he can communicate with", etc.

as by observing

$\neg D \vee$ "there exists a mathematician that shares a language with at least two others" etc.

and that Lemma 1 is equally "obvious" in the same sense. Lemma 2 is slightly less obvious. My simplest argument establishes it with

H: there exists a pair MM of two mathematicians that cannot communicate with each other

Lemma 2.0 : $E \vee H$

Lemma 2.1 : $A \vee E \vee \neg H$;

the last lemma is easily established by observing the communication facilities between the mathematicians of the pair MM on the one hand and the remaining 7 on the other.

For <u>none</u> of the lemmata anything is gained by suggesting a specific form of case analysis by formulating the lemma as an implication. This, regrettably, is often the tacitly assumed association with the formulation of a theorem as an implication. I think this convention regrettable.

The formulation of a theorem should stand as a logical firewall between its usage and its proof, and neither its (expected) usage, nor its (proposed) proof should be allowed to bias its formulation, because that neutrality is exactly what being a logical firewall is about.

The above example has been posed in the Mathematical Olympics, not because it is sufficiently difficult or requires an unusual argument, but because -thanks

to this unfortunate convention — it could be formulated in a way that suggests a case analysis that is confusing:

"Nine mathematicians meet at an international congress. When it is given that of any three of them, at least two can communicate in a common language, an no mathematician masters more than three languages, show that there exists a language common to at least three of them."

In the equivalent formulation

"Nine mathematicians meet at an international congress. When it is given that each mathematician masters at most three languages and each language is mastered by at most two mathematicians, show that there exists a triple of mathematicians, no two of which have a language in common."

the problem would have been regarded as trivial. (End of Example.)

Note. In mechanical theorem proving, a theorem to be proved of the form $(A \wedge B) \Rightarrow C$ is usually reformulated as $\neg(A \wedge B \wedge \neg C)$ or $(A \wedge B \wedge \neg C) \Rightarrow F$. Thanks to de Morgan's Law, this is equivalent to $\neg A \vee \neg B \vee C$, and thus the same homogeneity is achieved. It is not much more than the question whether we prefer the postfix "$\Rightarrow F$" over the prefix "$\neg$". Amazingly enough, people with that background in mechanization tend to maintain the implication, with the result that — e.g. by distinguishing between "assertions" and "goals" — most of their rules

are stated 2, 4, or 8 times. (End of Note.)

Remark. In electrical engineering it is not unusual to denote T by 1 and F by 0. The conjunction $\wedge$ is then multiplication and is often denoted by just juxtaposition; the disjunction $\vee$ is then often denoted by +. There is no doubt in my mind that this notation explains why — as one of my spokesmen has assured me — the average electrical engineer, while being certain that the conjunction distributes over the disjunction, hesitates whether the disjunction distributes over the conjunction: $A(B+C) = AB + AC$ certainly looks more familiar than the corresponding $A + BC = (A+B)(A+C)$!

The poor electrical engineer has paid the price for two methodological mistakes:

0) the false analogy between $(\vee, \wedge, T, F)$ and $(+, *, 1, 0)$ — of electrical origin —
1) the default convention that, when a binary operator is missing, "multiplication" is meant — of mathematical origin — .

(End of Remark.)

Correction. The time-honoured "modus ponens" is not the inference rule I wrote it was, but the simpler

$$\frac{A \qquad A \Rightarrow B}{B}$$

Showing that also the above is a special instance of our inference rule is left as an exercise for the reader. (End of Correction.)

* * *

Before proceeding I would like to explain why I prefer to make a clear distinction between the (one and only) inference rule on the one hand and the equalities on the other.

Firstly, it is in the application of the inference rule, when we derive an assertion that is weaker than the conjunction of what we have got, that we exercise most of our freedom: how the inference rule is applied is much more indicative for the structure of the proof than the boolean simplifications. (They do destroy information, but the destruction is as uninteresting as when we reduce 3+4 to 7.) That is why I don't like inference rules and identities all lumped together.

Secondly, I cannot escape the feeling that equality of propositions is a curiously neglected concept. One explanation offered is that our Western languages are

not very well suited for its verbal expression: quickly we feel compelled to use quotation marks and to insert noise nouns like "statement" as in:

> The statement "John's two children are of different sex" is equivalent to the statement "John has one daughter and one son"

(which, to make matters worse, is also applicable to a John with three sons and no daughter).

This linguistic explanation seems quite acceptable, but if we accept it, it should be an added incentive to include equality of propositions clearly and honestly in our formalism. Equality is so nicely symmetric and transitive that it must be a useful concept and if the available means for expressing it for propositions are inadequate we had better introduce adequate means for doing so. A major purpose of formalisms is to liberate who masters them by freeing him - and his thinking - from the shackles he inherited when he learned his native tongue. (Gentzen's system of so-called "natural deduction" was aimed at formalizing how arguments are expressed in natural languages, and that is precisely why I consider it a great step backwards: it fails to liberate.)

The general uneasiness with equality of propositions

is eloquently reflected in the contorted "if and only if" and the superfluous symbol "<=>", but I am afraid that it has more serious consequences. The "if and only if" is in the same vein as the definition of equality of the sets A and B as A being a subset of B and vice versa. It leads to a proof style, in the arithmetic analogue of which $x=y$ is routinely proved by demonstrating $x \leq y$ and $y \leq x$ separately, and by modern standards that is clumsy.

Hence my desire to use equality of propositions explicitly whenever appropriate.

* * *

"If there is any virtue in talking about situations which arise in analysis as if we were back with Archimedes drawing diagrams in the dust, it has yet to be revealed. Pictures after all may be suitable only for very young children; Lagrange dispensed entirely with such infantile aids when he composed his analytical mechanics. Our propensity to "geometrize" our analysis may only be evidence that we have not yet grown up.

E.T. Bell "Men of Mathematics".

We stay for a while with finite combinatorics. Let each of the 15 edges of the complete 6-graph be either red or blue; 3 of the nodes form a "monochrome triangle" that the 3 edges connecting them are of the same colour.

Prove the existence of at least one monochrome triangle.

For the above I encountered the following proof (I don't remember the source and quote from memory).

Consider an arbitrary node, say X. Among the 5 edges meeting at X, one of the two colours, say red, dominates. Let XP, XQ, and XR be three red edges. Either triangle PQR has a red edge, PQ say, or it has no red edge. In the first case triangle XPQ is monochrome, in the latter case triangle PQR is monochrome. q.e.d.

Knowing the above proof I found in the same vein the following proof for the existence of a second monochrome triangle.

We know the existence of one monochrome triangle. Let it be red; call its vertices the A's and call the three remaining nodes the B's. Then there exists a red monochrome triangle AAB or in each B at most 1 red AB-edge meets.

In the latter case, at least 2 blue AB-edges meet at each B. Then BBB is a red monochrome triangle or BBB has a blue edge, in which case there exists a blue monochrome triangle BBA, since in both end nodes of the blue BB-edge two blue AB-edges meet, which have at least one A in common. q.e.d.

The discovery of the above proof gave me only one minor satisfaction, which was derived from the introduction of the nomenclature of the A's and the B's without subscripts. (The explicit distinction between P, Q, and R in the preceding proof is pretty superfluous, as betrayed by the "PQ say"; see also "the pair MM of mathematicians that cannot communicate with each other" — EWD803-13 — where, similarly, we don't need to distinguish between the two.) But I disliked the proof for its case analysis that distinguishes between the two monochrome triangles having 0, 1, or 2 vertices in common and for the (very weird!) distinction between the "first" and the "second" monochrome triangle. I was only satisfied when I found the following proof.

We call two edges of different colour and having one node in common a "mixed V" meeting at that node. Since at any node 5 edges meet, at most $2*3 = 6$ mixed V's meet at that node. Since there are 6 nodes, there are at most $6*6 = 36$ mixed V's. Since each mixed V occurs in one bichrome triangle and each bichrome triangle contains two mixed V's, there are at most $36/2 = 18$ bichrome triangles. The total number of distinct triangles being $(6 \cdot 5 \cdot 4)/(3 \cdot 2 \cdot 1) = 20$, there are at least $20 - 18 = 2$ monochrome triangles. q.e.d.

The last proof is so much more beautiful than its

predecessor that we should try to learn as much as possible from a comparison of the two.

The trouble already started with "Consider an arbitrary node, say X", which immediately destroys the symmetry between the nodes; in the second part we have, similarly, the A's versus the B's. Without all that nomenclature we could not have carried out the case analysis; our last proof reveals that it is precisely that nomenclature that, by destroying the symmetry, has forced that case analysis upon us! And, as betrayed by the "say red", the symmetry between the colours was lost right at the start.

The notion of the "mixed V" is, of course, the cornerstone of the last proof, but I would like to argue that its introduction is "sweetly reasonable" (J.M. Stoy) after the observation of the colour symmetry. Being of equal or of different colour are the <u>only</u> functions that are invariant under colour inversion —note that, accordingly, in the last proof the colours themselves are not mentioned at all!—; furthermore somewhere the topology of the complete 6-graph has to be taken into account, hence the notion of a V, hence the mixed V. (Originally it was called a "mixed pair".) Similarly the last proof fully maintains the symmetry between the nodes (to the extent that they too remain anonymous). Its final virtue is that it is much less of an invitation to visualization. (Hence the quotation from E.T. Bell.)

On mathematical induction.

People of my generation were introduced to mathematical induction as follows.

In order to prove for some predicate P over the natural numbers

(1) $\quad (\underline{A}\, n : n \geq 0 : P\, n)$

one proves instead

(2a) $\quad P\, 0 \quad$ ("the base")
(2b) $\quad (\underline{A}\, n : n \geq 0 : P'\, n) \quad$ with $P'n = (P\, n \Rightarrow P(n+1))$.

I prefer the formulation: in order to prove (1) one proves

(3) $\quad (\underline{A}\, n : n \geq 0 : P'n)$ with
$P'n = (P\, n \vee (\underline{E}\, i : 0 \leq i < n : \neg P\, i))$

Remark. Proof obligations (2b) and (3) are of the same form as (1), only with P replaced by P'. One can try to meet them by means of mathematical induction, i.e. by replacing P' by the corresponding P'', but in either case one finds $P'' = P'$, i.e. the decision to carry out a proof by mathematical induction is idempotent. (End of Remark.)

Remark. Of course the usual formulation of P'n was

$$(\underline{A}\, i : 0 \leq i < n : P\, i) \Rightarrow P\, n \quad ;$$

with the equivalent formulation

$$(\neg P\, n) \Rightarrow (\underline{E}\, i : 0 \leq i < n : \neg P\, i)$$

the method is honoured with the special name "the method of infinite regress"; and here we have encountered yet another artefact of that asymmetric notation. (End of Remark.)

In general I prefer (3) over (2) for two reasons. Firstly, the separation between the base (2a) and the step (2b) induces a case analysis which is not always necessary, secondly, the P' of (3) is essentially weaker than the P' of (2b) and, hence, in general easier to prove.

It is sometimes argued that the way in which (3) subsumes the base (2a) is a "coding trick" that has later to be undone. That this is not true we shall demonstrate by redoing some very elementary number theory. Some terminology first.

The natural numbers are 0, 1, 2 and so on; for further details, see Peano. The positive integers are the natural numbers ≥ 1, i.e. 1, 2, 3 and so on. The plurals

are the natural numbers ≥ 2, i.e. 2, 3, 4 and so on. Natural numbers that are the product of two plurals are called <u>composite</u>; since the product of two plurals exceeds each of those two, we may write

(4) $\text{comp } m = (\underline{E}x,y: 2 \leq x < m, 2 \leq y < m: x \cdot y = m)$.

Plurals that are not composite are called <u>prime</u>:

(5) $\text{prime } m = m \geq 2 \wedge \neg \text{comp } m$.

From these definitions follows

(6) $(\underline{A}\, n: n \geq 1: n = 1 \vee \text{prime } n \vee \text{comp } n)$.

With PF - for Prime Factorization - defined by

PF n = n is the product of zero or more primes (the empty product being defined as 1)

we can state

<u>Theorem</u> 0. $(\underline{A}\, n: n \geq 1: \text{PF } n)$

<u>Proof</u>. We shall prove Theorem 0 by mathematical induction, i.e. we shall prove

(7) $(\underline{A}\, n: n \geq 1: \text{PF } n \vee (\underline{E}\, i: 1 \leq i < n: \neg \text{PF } i))$.

Since the product of zero primes equals 1 and the product of one prime equals that prime, we deduce

from the definition of PF

$$(\underline{A}\, n : n \geq 1 : PF\, n \;\vee\; n \neq 1)$$

and

$$(\underline{A}\, n : n \geq 1 : PF\, n \;\vee\; \neg\, \text{prime}\, n)$$

respectively. Together with (6) they yield (standard inference)

(8) $(\underline{A}\, n : n \geq 1 : PF\, n \;\vee\; \text{comp}\, n)$.

Furthermore we deduce from the definition of PF

$$(\underline{A}\, x, y : x \geq 1, y \geq 1 : \neg PF\, x \;\vee\; \neg PF\, y \;\vee\; PF\,(x \cdot y))$$

from which we deduce with (4)

(9) $(\underline{A}\, n : n \geq 1 : \neg\, \text{comp}\, n \;\vee\; (\underline{E}\, i : 1 \leq i < n : \neg PF\, i) \;\vee\; PF\, n)$.

From (8) and (9), relation (7) follows by the standard inference rule. (End of Proof of Theorem 0.)

Aside. Knowing how to prove properties of algorithms we can give a more constructive proof, of which I think that Euclid would have liked it. (I like it in any case.)

Consider the following program in which n is a positive integer constant and "bag" a variable of type

bag of positive integers:

if $n=1 \rightarrow bag := \emptyset \;[]\; n>1 \rightarrow bag := \{n\}$ fi;
do bag contains a composite plural →
    replace each occurrence of the largest composite plural c in bag by the plurals x and y, where $x \cdot y = c$
od .

The repetition leaves the product of the numbers in bag equal to n. On account of (4) – which is, of course, again needed – the lowest upper bound for composite plurals in the bag can be taken as variant function, and termination is guaranteed. (End of flside.)

Theorem 1. $(\underline{A}\ p: \text{prime } p : (\underline{E}\ q: q>p: \text{prime } q))$

(This is Euclid's famous theorem that no prime is the largest one.)

Proof Consider for arbitrary prime p the value Q, defined by

$Q = 1 +$ the product of all primes $\leq p$.

Theorem 0 allows us to conclude that Q is the product of a bag of primes and that, because $Q>1$, that bag is not empty. By virtue of Q's construction,

such a bag contains no prime $\leq p$. Hence it contains at least one prime $> p$, hence at least one prime $> p$ exists. (End of Proof of Theorem 1.)

The great invention of this proof is, of course, the construction of Q, which is contained in all proofs I have seen. The above proof is given because I have seen many proofs that distinguish between the cases prime Q and $\neg$ prime Q. (Some even used the (deeper?) theorem that the prime factorization is unique.) One or two reductiones ad absurdum are also not uncommon.

With UPF -for Unique Prime Factorization- defined by

UPF n = the bag of primes whose product equals n is unique

we can formulate

Theorem 2. $(\underline{A}\ n:\ n \geq 1:\ \text{UPF}\ n)$

Proof. By mathematical induction, i.e. by proving

$$(\underline{A}\ n:\ n \geq 1:\ \text{UPF}\ n\ \ \underline{V}\ (\underline{E}\ i:\ 1 \leq i < n:\ \neg\ \text{UPF}\ i)) \qquad (10)$$

We shall demonstrate (10) by showing how to construct for an n such that $\neg$ UPF n, an i with $1 \leq i < n$ such that $\neg$ UPF i.

From two different bags of primes we can construct two different bags of primes that have no prime in common by taking away from them the largest bag of primes contained in both of them (their "intersection"). For both bags, the product of their contents is divided by this operation by the same value ≥ 1.

From two different bags of primes, each with product = n, we construct thus two different bags of primes that have no prime in common and each with product $= n'$ with $n' < n$. Because the bags are different, they are not empty, and therefore $1 < n'$. Let the bags have p and q as their smallest primes respectively, then

$$n' = p.P = q.Q \quad \text{with } p<q \text{, i.e } P>Q$$

and Q = the product of a number of primes all $\geq q$ and therefore $> p$. Consider now i defined by

$$i = n' - p.Q$$

Clearly we have $1 \leq i < n$. Observing $i = p.(P-Q)$ we conclude from PF(P-Q) – Theorem 0 – the existence of a bag of primes with product = i that <u>does</u> contain p. Observing $i = (q-p).Q$ we conclude the existence of a bag of primes with product = i

that does not contain p : $PF(q-p)$ states that the first factor is the product of a bag of primes, which are all $\neq p$ because the prime q is not a multiple of p and the second factor, Q, is a product of primes all $> p$. From the difference of the two bags we conclude $\neg$ UPF i, hence (10). (End of Proof of Theorem 2.)

Remark. A usual phrasing of this argument to prove (10) begins with

"assume an n such that

$$\neg \text{UPF } n \wedge (\underline{A}\, i : 1 \leq i < n : \text{UPF } i)" ;$$

as above it derives from the first term $(\underline{E}\, i : 1 \leq i < n : \neg \text{UPF } i)$, a conclusion which is then confronted with the second term to conclude a contradiction, which then falsifies the assumption that such an n exists. Our phrasing avoids all this. The only thing we require is that the construction of i is applicable to any n such that $\neg$ UPF n and we did not need to assume the existence of such an n. (End of Remark.)

It is now not difficult to prove

Theorem 3. (for "$a|b$" read "a divides b")

$$(\underline{A}\, x, y, p : x \geq 0, y \geq 0, \text{prime } p : (p|x \vee p|y) = p|(x \cdot y))$$

Proof. Since

$$\frac{A \vee \neg B \qquad \neg A \vee B}{A = B}$$

Theorem 3 follows from Lemma 3.0 and Lemma 3.1, given by:

Lemma 3.0.

$$(\underline{A}\, x,y,p \,.\, x\geq 0, y\geq 0, \text{prime } p : \neg(p|x \vee p|y) \vee p|(x\cdot y))$$

Lemma 3.1

$$(\underline{A}\, x,y,p : x\geq 0, y\geq 0, \text{prime } p : (p|x \vee p|y) \vee \neg\, p|(x\cdot y))$$

Of these two, Lemma 3.0 is obvious, Lemma 3.1 follows from Theorem 2 and the obvious

Lemma 3.1.0

$$(\underline{A}\, x,y,p : x\geq 0, y\geq 0, \text{prime } p : p|x \vee p|y \vee \neg\, p|(x\cdot y) \vee \neg\, UPF(x\cdot y)) \,.$$

Note. In an earlier effort to prove (10), I have done my case analysis the other way round, and demonstrated UPF n for an n with $(\underline{A}\, i : 1\leq i<n : UPF\ i)$. For $(n=1) \vee (\text{prime } n)$ the conclusion is obvious, for (comp n) one writes

$$n = p\cdot P = q\cdot Q$$

with prime p,
$p \leq q$,
P a product of primes all $\geq p$
Q a product of primes all $\geq q$

and deduces UPF n from

$$(p=q) \vee (\text{comp } q) \qquad (11)$$

which is derived by studying $n - p\cdot Q$. The arithmetic

is very similar to the one in the proof given above, but complicated by the circumstance that one has to apply the "deep" Lemma 3.1 to products $x.y$ of which one has to show that $x.y < n$. Our above proof uses only the obvious Lemma 3.0 and the equally obvious

$$(\underline{A}\, x,y : x \geq 0, y \geq 0 : \neg C x \vee \neg C y \vee C(x.y)) \qquad (12)$$

with

$C\, n$ = there exists a bag of primes with product = n that does not contain p.

Lemma 3.0 and (12) are so obvious that we did not even bother to mention them.

For the time being we confine ourselves to the observation that in the one proof we needed the "deep" half of Lemma 3 while by carrying out the case analysis the other way round, we could get away with the "shallow" half of Lemma 3. Whether this is an instance of a more general phenomenon remains to be seen. (End of Note.)

I urge the reader to compare the above redoing of very elementary number theory with the way in which this material is presented by Richard Courant and Herbert Robbins in "What is Mathematics" (Oxford University Press paperback, 1978). In any case I should have convinced the reader that formulation (3) of mathematical induction is not a "coding trick" for

the use of which one has to pay later.

On mathematical induction and proofs of termination.

Let us start with a very classical example. On a shunting yard we start with a finite number of trains, each of finite length. As long as the shunting yard is not empty, we select an arbitrary train. If the train selected consists of a single car, we remove it from the shunting yard, if it consists of two or more cars, we make two trains out of it. Does this game terminate? It clearly does: consider t, defined by

t = the number of cars on the yard + the number of closed couplings.

By virtue of its definition, $t \geq 0$, and each move decreases t by exactly 1. Since the game terminates with $t = 0$, t equals at any moment the number of moves the game will last and the nondeterminacy of the game is of no influence on its duration. (Readers familiar with the literature about the program "tipcount" will recognize the origin of the above problem.)

Sometimes we are not interested in a sharp upper bound for the number of steps and are we satisfied

with a proof that a bound exists.

For a next example I draw the reader's attention to the program on top of page EWD803-26. Each time the occurrences of the <u>largest</u> composite plural are replaced. If the reader suspects that that choice has been made to ease the termination proof, he is correct: <u>any</u> composite plural from the bag would have done. But we would have needed another termination argument.

<u>Note</u>: $t = {}^2\log n$ - number of plurals in the bag would have done the job. (End of Note.)

Now for a somewhat harder example. Consider the simple syntax for <term>

<term> ::= <constant> | <variable> | <sum> | <product>
<constant> ::= 0|1|2|3
<variable> ::= x | y | z
<sum> ::= (<term 1> + <term 2>)
<product> ::= (<term 1> • <term 2>)

We all know how to differentiate a <term> with respect to t. The rules for differentiation are, for the above cases, respectively

$\frac{d}{dt}$ <constant> ::= 0|0|0|0
$\frac{d}{dt}$ <variable> ::= x'|y'|z'

$\frac{d}{dt}\langle sum\rangle ::= \left(\frac{d}{dt}\langle term\,1\rangle + \frac{d}{dt}\langle term\,2\rangle\right)$

$\frac{d}{dt}\langle product\rangle ::=$
$\left((\langle term\,2\rangle \cdot \frac{d}{dt}\langle term\,1\rangle) + (\langle term\,1\rangle \cdot \frac{d}{dt}\langle term\,2\rangle)\right)$

Why does the differentiation process terminate? We shall return to this later.

Finally a termination problem of a rather different nature. (How different, we shall discuss later.) A one-person game is played with a bag containing a finite number of natural numbers. As long as the bag is not empty, a move is performed, a move consisting of replacing an instance of x from the bag by an arbitrary number of natural numbers $< x$. (Only when $x=0$, we have no freedom: x is removed and, for lack of smaller natural numbers, nothing is returned into the bag.) Does this game terminate?

Before answering these questions, we shall discuss for a while the relation between termination proofs and mathematical induction in general. We have discussed so far mathematical induction on the natural numbers. The natural numbers are not only <u>totally ordered</u>, they are also <u>well-ordered</u>. In the following we shall denote our ordering relation by "$<$". We give the following definitions.

Set U is <u>totally ordered</u> means

$$(\underline{A}\, x,y: x \in U, y \in U: x=y \vee x<y \vee y<x) .$$

Set U is <u>well-ordered</u> means that every non-empty subset S of U has a least element, where y is a <u>least element</u> of S means

$$y \in S \wedge (\underline{A}\, x: x \in S: y=x \vee y<x) .$$

Set U is <u>well-founded</u> means that every non-empty subset S of U has a minimal element, where y is a <u>minimal element</u> of S means

$$y \in S \wedge (\underline{A}\, x: x<y: \neg\, x \in S) .$$

Since a least element of S is a minimal element of S, a well-ordered U is a well-founded U. A well-founded U, however, need not be well-ordered. It is the weaker requirement of well-foundedness that suffices for a proof by mathematical induction, as the following argument shows.

Since the empty subset S has no minimal element, we have for each subset S of a well-founded set U

$$(\neg \text{ empty } S) = (\underline{E}\, y: y \in U: y \in S \wedge (\underline{A}\, x: x<y: \neg\, x \in S))$$

or, by de Morgan's Law,

$$(\text{empty } S) = (\underline{A}\, y: y \in U: \neg\, y \in S \vee (\underline{E}\, x: x<y: x \in S))$$

With P some predicate defined on the elements of U and $S = \{y \mid y \in U \wedge \neg P y\}$, we have

$$(\text{empty } S) = (\underline{A}\, y : y \in U : P y) ,$$

and our last formula reduces to

$$(\underline{A}\, y : y \in U : P y) = (\underline{A}\, y : y \in U : P y \vee (\underline{E}\, x : x < y : \neg P x))$$

and this is the equality on which a proof by mathematical induction relies.

The connection between well-foundedness and termination is given by the concept of a <u>descending chain</u>. A descending chain in U is a (finite or infinite) sequence $x_0, x_1, x_2, \ldots$ of elements of U such that

$$(\underline{A}\, i,j : 0 \leq i < j : x_j < x_i)$$

(Since the ordering in U is transitive, $x_{i+1} < x_i$ suffices.) The concept of descending chains is connected to well-foundedness in the

<u>Theorem</u>: U is well-founded = all descending chains in U are finite.

<u>Proof</u>. We shall first consider the case in which U is well-founded. Since the empty chain is finite,

we have to show that all non-empty chains are finite. Since each non-empty chain has a unique element x_0, at which it starts, we have to prove

$$(\underline{A}\, x : x \in U : P\, x)$$

with $P\, x$: all descending chains starting at x are of finite length.

Under the assumption of well-foundedness of U, it suffices to prove

$$(\underline{A}\, x : x \in U : P\, x \;\vee\; (\underline{E}\, y : y < x : \neg P\, y))$$

which is obvious.

Next we consider the case that U is not well-founded, i.e. has a non-empty subset S without minimal element, i.e. such that

$$(\underline{A}\, y : y \in S : (\underline{E}\, x : x < y : x \in S))$$

from which the constructibility of a descending chain of infinite length from elements of S - and hence of U - follows. (End of Proof.)

Well-founded sets have the charming property that from two well-founded sets U and W - which may be equal - we can form the well-founded set $L(U,W)$ by <u>lexicographic pairing</u>. The elements of $L(U,W)$ are all the pairs (u,w) with $u \in U$ and $w \in W$ and the order relation is defined by

$$(u_0, w_0) < (u_1, w_1) \;=\; u_0 < u_1 \;\vee\; (u_0 = u_1 \wedge w_0 < w_1)\ .$$

Lexicographic pairing is also defined for partially ordered sets which are not well-founded. In fact we have the

Theorem $L(U,W)$ is well-founded =
U is well-founded $\wedge$ W is well-founded.

(We leave the proof to the reader; it can be done by grouping in a descending chain of $L(U,W)$ consecutive elements (u,w) with the same u together.)

Theorem Lexicographic pairing is associative, i.e.

$$L(L(U,W),X) = L(U,L(W,X))$$

(Again the proof is left to the reader; it can be done the propositional calculus.)

The associativity suggests that I should have used an infix operater to denote lexicographic pairing, say $U \mathbin{\underline{\prec}} W$. Its elements are, in fact, often denoted by juxtaposition: "$u\ w$" (and "$u\ w\ x$" etc.).

Since the natural numbers with the usual interpretation of "$<$" form by definition a well-founded set, the lexicographically ordered sequences of natural numbers form for each length the elements of a beloved well-founded set.

The finite length of descending chains in a well-founded

set forms the basis for all proofs of termination. (With s_i a possible successor state of s_j denoted by $s_i < s_j$, one has to show that $<$ is a well-founded partial order.)

With Euclid's algorithm for positive integers

```
do  x > y  →  x,y := y,x
[]  y > x  →  y := y - x
od
```

the lexicographic pair (x,y) gives the required well-founded set.

For the termination of $\frac{d}{dt}$<term> we associate with each intermediate state a bag of positive integers, i.e. for each $\frac{d}{dt}$<term> we put into the bag the number of variables or constants in <term>. Each differentiation rule replaces in this bag a number by zero or two smaller values. This association reduces the differentiation problem to a special case of the one-person game with the bag containing a finite number of natural numbers (see EWD803 - 34).

Let N be the largest value initially in the bag. Consider the lexicographically ordered sequences of length N+1:

$$f_N \quad f_{N-1} \quad \ldots \quad f_1 \quad f_0$$

where f_n = the number of instances of n in the bag. (The f-sequence is a finite tabulation of the characteristic function of the bag.) The f-sequences provide in their lexicographic order the required well-founded set.

The one-person game differs from all previous examples in that the number of moves, though finite, is a priori unbounded. In a proof of "strong termination" one shows that the number of possible moves is bounded, in a proof of "weak termination" one only shows that the number of possible moves is finite.

A program that, like our one-person game, terminates weakly but fails to terminate strongly requires a primitive of unbounded nondeterminacy. (And, by adding a counter of the number of moves, we construct from it a mechanism of unbounded nondeterminacy!) We consider such a mechanism nonexistent, firstly because it is unrealistic and seandly because the mathematical consequences of its inclusion are unattractive (one loses, for instance, continuity).

Our restriction to bounded nondeterminacy has two consequences. Firstly, the greater generality of the well-founded sets is from a theoretical point of view superfluous: just the natural numbers would

have sufficed. Secondly, well-founded sets come in handy: using well-founded sets we prove only weak termination explicitly; strong termination is then implied by the absence of unbounded nondeterminacy.

As final example of the use of the lexicographic order we consider the following game played with a bag with N integers. A move is possible when the bag contains an arbitrary value x at least twice and consists of replacing two instances of x by one instance of $x-q$ and one instance of $x+p$ for some $p>0$ and $q>0$. (For each move we may choose new positive values for p and q.) Prove that the game terminates or the minimum value in the bag sinks below any bound.

Proof. (By J. Misra.) Associate with each state of the bag the sequence of N integers obtained by arranging the values in the bag in ascending order. Each move results in a new sequence that is lexicographically smaller, but under the constraint of constant lower bound the order is well-founded. (End of Proof.)

Remark. By the same token we prove that the game terminates or the maximum value increases beyond any bound. (End of Remark.)

Exercise. Prove that the game terminates when $p+q$ is bounded. (End of Exercise.)

For the sake of completeness we mention one further context in which we encounter well-founded sets, viz. recursive definitions: when $x_j < x_i$ means that $f(x_i)$ is effectively defined in terms of $f(x_j)$, we have to show that $<$ is a well-founded ordering. (With

$$\begin{aligned} fac(n) = {} & \textbf{if } n=0 \to 1 \\ & \;[]\; n \geq 1 \to n \cdot fac(n-1) \\ & \textbf{fi} \end{aligned}$$

fac(2) is "effectively defined" in terms of fac(1), fac(0) is <u>not</u> "effectively defined" in terms of fac(-1).)

<u>Exercise</u>. Prove that

$$fib = 0 : 1 : elf\ fib \quad \underline{\text{where}}\ elf(a:b:c) = a+b : elf(b:c)$$

defines fib as infinite sequence. (End of Exercise.)

Plataanstraat 5
5671 AL NUENEN
The Netherlands

30 Nov. 1981
prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow