The streamlining of the mathematical argument

I plan to extend the work done so far in collaboration with drs. A.J.M. van Gasteren at the Eindhoven University of Technology. This work was the first phase of a more general project, viz. the streamlining of the mathematical argument, which was inspired by our observation that many --if not most-- mathematical arguments we encountered in the literature were unnecessarily complex and lacking in rigour by being incomplete. A few explorations sufficed to convince us that the potential for improvement was dramatic indeed, and we had to conclude that the mathematical community hardly pays any attention to the conscious pursuit of mathematical elegance (in the sense of the Concise Oxford Dictionary "elegant: simple and surprisingly effective"). In passing we note that high technology has now reached a stage of ambition in which mathematical elegance is no longer a dispensable luxury but often decides between success and failure.

Support from the SP Venture Research Unit was most welcome --and, we felt, appropriate-- as soon as we discovered that, in the mathematical community, methodological concerns were markedly unpopular and that support from traditional sources was therefore unlikely. [Mathematical results are taught quite openly and quite explicitly, as in the best tradition of the University. How to do mathematics, however, is taught only implicitly, by osmosis, so to speak, as in the tradition of the guilds, which guard the secret of their craft by avoiding the explicitness that would bring it out into the public domain. The transition from craft to science always evokes opposition from the guild members and the Mathematical Guild is no exception.]

Our explorations in the first phase have largely been confined to what had to be done first, viz. finding out what we wanted our streamlined arguments to look like, finding the relevant criteria for elegance (as technical concept), etc. Considerations of heuristics --i.e. how to design the effective argument settling a given question-- has been consciously left to a later phase: there is no point in trying to develop a design methodology without a sufficiently clear picture of the key structural characteristics of the programs or proofs to be designed --with "structural characteristis" we refer to the ways of avoiding monolithic designs, to the

degree of separation of concerns, to the choice which relationships are to be captured by syntax, the patterns according to which things are left anonymous or are named, the levels of abstraction that can be distinguished, etc.— . Similarly, when we learned that formal calculi of all sorts would have an important role to play, we postponed the problem of axiomatic foundation of such calculi: the intellectual investment involved would be premature without the "moral certainty" that one is founding the calculi one needs. [These postponements directly reflect the separation of concerns that enabled the development of programming methodology that took place during the seventies.]

The main results so far have been —with due credit to W.H.J.Feijen, C.S.Scholten, and the other members of the (Eindhoven) Tuesday Afternoon Club—

(i) We identified shortcomings such as the Sins of Omission, of Repetition, and of Needless Distinction, and designed notational techniques catering for their avoidance and thus admitting the concise presentation of effective arguments. The resulting style has now been adopted in a book, in two completed Ph.D.Theses, in several articles and in numerous technical notes, and has been a precondition for our other results. (By way of verbal illustration, compare the following definition from Webster's New Collegiate Dictionary

> "continued fraction: a fraction whose numerator is an integer and whose denominator is an integer plus a fraction whose numerator is an integer and whose denomiator is an integer plus a fraction and so on"

with

> "continued fraction: a fraction whose numerator is an integer and whose denominator is an integer plus a continued fraction" .)

(ii) We developed a theory of predicate transformer semantics, using an extended predicate calculus dealing with boolean functions on some domain. [The relational calculus is a particularization for the special case that the underlying domain is a Cartesean square.] The crispness of the theory is a convincing demonstration of how much can be achieved with a calculus that is notationally geared to the manipulations at hand.

(iii) We recently discovered how our methodology for the formal design of programs could be transferred to the design of proofs. This discovery is in any case of great promise, and probably of profound significance (see (II), below).

(iv) We found several instances of how (more or less) special-purpose calculi can greatly facilitate design and correctness proof of programs that would otherwise have been hard to conceive and to justify. In program design, the use of the predicate calculus is now well-established. Last year the calculus of regular expressions was successfully applied in a new context --viz. that of distributed computations-- and more recently A.J.M. van Gasteren developed with W.H.J.Feijen a special "ring calculus" for the design of a class of permutation algorithms. We sense a trend (see (I), below).

The above results are not independent and clearly belong to the same "school of thought" that grew while I was at the Eindhoven University of Technology. At the University of Texas at Austin I found a very different culture, of which a strong tradition in mechanical theorem proving --established by Robert S.Boyer and J Strother Moore-- was one of the main attractions. I intend to exploit to the fullest my opportunities of acting as liaison officer between the two intellectually different cultures and to search for a fruitful blend of Austin's inclination towards mechanization and formal foundation and Eindhoven's stress on mathematical elegance. I see the following opportunities.

(I) The (highly) effective use of elegant calculi seems a very promising area, the exploration of which has only begun. Tailoring a calculus to one's manipulative needs is clearly one of the main rules of the game, a game which Eindhoven, however, still plays "in Euler's style", i.e. without much formal foundation and without much basic insight in the limits of such calculi. (Uniqueness and completeness proofs, for instance, have traditionally bothered us.) The professional design of elegant calculi would make a well-established tradition of pure logic lead to "vehicles of formal reasoning" of great practical value.

(II) The links connecting proof design and program design are getting stronger and stronger, both from practical experience (see (iii), above) and from a theoretical

point of view --the logician Per Martin-Löf sees an exact analogy between proofs and programs-- . Consequently, for computing science/programming methodology, "proof-driven programming" seems an appropriate catch-phrase for a now promising research area. Austin's greater readiness to mechanize symbol manipulation will almost certenly give such a project a flavour it never gets as long as it is pursued in the Eindhoven style; at the same time it will open up a new area of application for the Austin tradition. [Note that a formally derived program is an object fundamentally different from the intuitively constructed ones that pervade current industrial practice: the relation is as between a proven theorem and an unjustified conjecture.]

(III) American mechanized symbol manipulation is impressive; tied as it is to LISP --by now 25 years old-- it is also unbelievably ugly. Decades of stagnation --the normal result of premature standardization-- have been covered up by building faster, cheaper hardware. Mechanized symbol manipulation, but not in LISP's straight-jacket, is an insufficiently explored avenue, and I am eager to see the experiment taken.

(IV) Independently, we and Boyer/Moore have found cases in which a syntactic analysis of the formal statement of the demonstrandum gave strong heuristic guidance, occasionally to the point of almost dictating how the proof had to be conducted. The time seems ripe to estimate whether our joint experiences warrant a more conscious exploration in that direction.

In view of the above I would like to extend my research contract with an Austin branch. Besides availability of the necessary funds, the realization depends of course on the availability of the necessary talent. Two PH.D. students that expect to earn their title by the end of the year have shown their interest in joining me for a number of years. The one, Warren Hunt, is the first Ph.D. student of Boyer/Moore; his original background is in electronig engineering, for his thesis he has applied mechanical theorem proving to the justification of circuitry. At my last year's lectures he was the brightest spot in my audience and I would be happy to be able to attract him. The same holds for Bas Braams; in his student days he switched from mathematical engineering (in Eindhoven) to theoretical Physics (in Utrecht).

During the last four years, partly at UKAEA Culham Laboratory (UK) and partly at the Max Planck Institut für Plasmaphysik in Garching (FRG) he has become a brilliant plasma physicist with extensive computational experience, and seriously considers becoming a computing scientist as well. They have not yet committed themselves (and have, of course, plenty of other opportunities, but I am afraid that that would hold for whomever I would like to attract). I would prefer to attract two "post docs" of sufficiently different backgrounds so as to stimulate each other.

In addition I would propose to be paid part of my summer salary and a budget for travel and communication.