Christopher Strachey

and the

Programming Research Group

The Germ of the PRG

Fox (Computer Journal, 1961):

It is certainly true that mathematicians ... should know what computing is about, and something about the nature of programming of machines. On the other hand it would be a degenerate step to replace, say, the theory of convergence, or even a more abstract topic in the mathematics syllabus, by a course designed to produce a breed of professional programmers.
…

This might seem obvious, but there seems to be a great need to labour it. I have the impression that some people think that programming ... is not only more important than mathematics but can actually replace mathematics, that, in a sense it *is* mathematics.

... the idea must be resisted strenuously that an ability to code a machine is synonymous with the mathematical ability necessary to make the best use of this remarkable equipment.

Response

Gill and Strachey:

Does not practical work on such a scale inevitably call for theoretical work on studies of common principles -- in this case the theories of programming languages, algorithms, compiling processes, list structures, recursive functions etc. ...? Are these not mathematics?

[Computers] are destined to play a part so basic and revolutionary that a correct appraisal of them is essential to our survival as an important nation. If this is so then the professional mathematicians must play a leading part ... as true mathematicians. They must do for computer programs what the famous mathematicians of the past have done for real and complex numbers.

Comeback

Fox:

Any undergraduate teaching of topics in “programming languages, compiling processes, list structures, recursive functions, etc.” must be associated, in black and white, with a syllabus. The Faculty is disinclined to alter a syllabus, and no amount of Churchillian rhetoric about ... “our survival as an important nation”, “a leading part”, “true mathematicians”, ... “famous mathematicians of the past” and so on, will bring about this change.

But:

[Gill and Strachey] could also perform a valuable service by entering temporarily the teaching world. ... I would be very happy to organize in 1963 a Summer School on "Non-numerical algorithms", "Artificial Intelligence", or whatever title is thought desirable ... . A concrete approach of this kind is, I suggest, the only way of getting their ideas across.

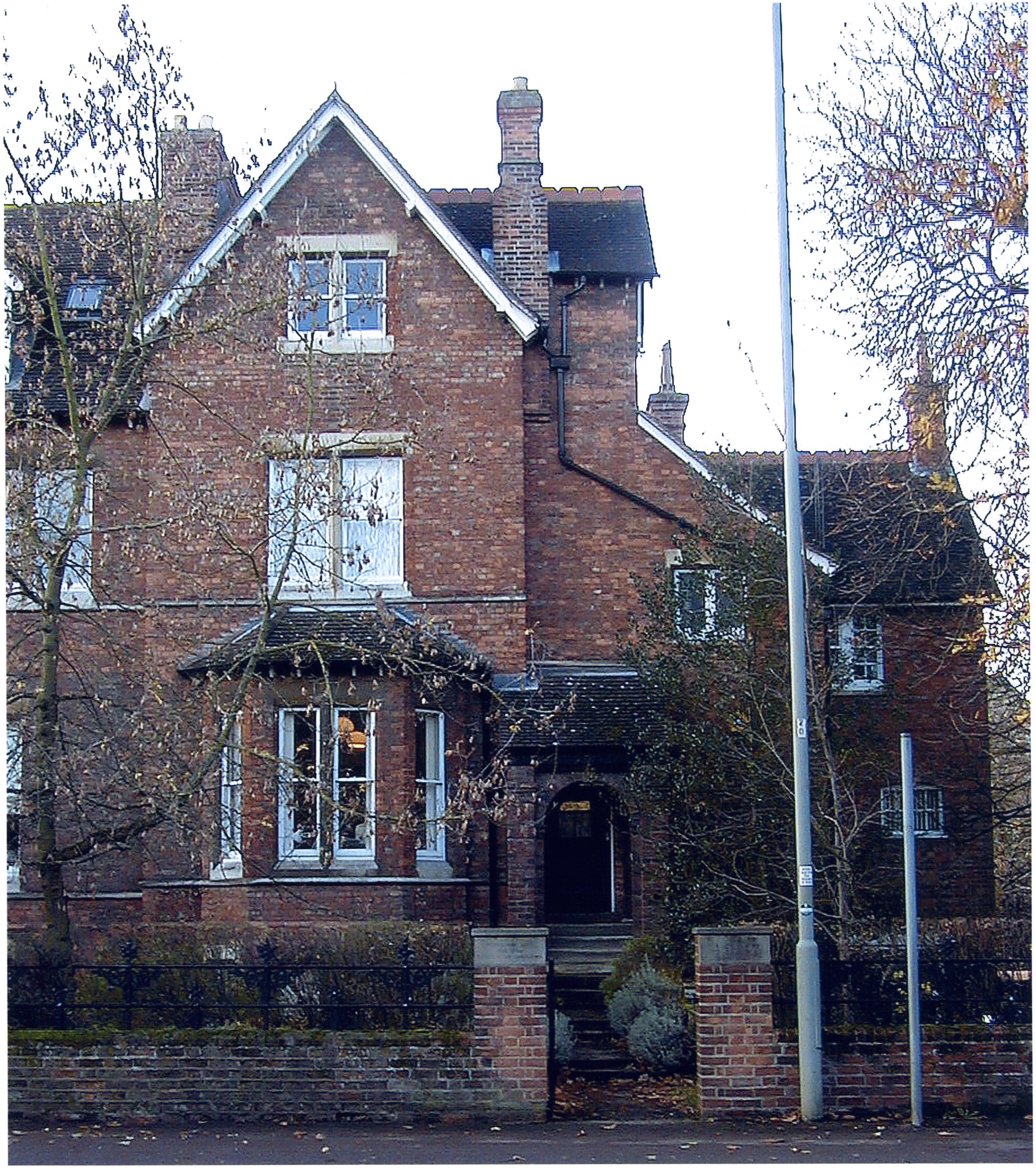
The Result

- Summer School happens in 1963
- Fox, L (ed): *Advances in Programming and Non-Numerical Computation* (1966)
- Fox obtains SRC Research Grant to set up a research group at Oxford
- PRG starts in April 1966 with Strachey as leader
  - Strachey, initially paid from the SRC grant, is “taken over” by the University as an *ad hominem* Reader in August 1967 (at which time the grant is extended till 1972).

Two Seminal Books

- Fox, L (ed): *Advances in Programming and Non-Numerical Computation* (Pergamon 1966)

- Steele, TB (ed): *Formal Language Description Languages for Computer Programming* (North-Holland 1966)
  - Proceedings of conference in Vienna (1964)
    - Semantics by symbol manipulation (→Algol 68)
    - Syntax, Chomsky languages etc.
    - The CUCH (Bohm)
    - λ-calculus (Landin)
    - Semantics by functions (Strachey)



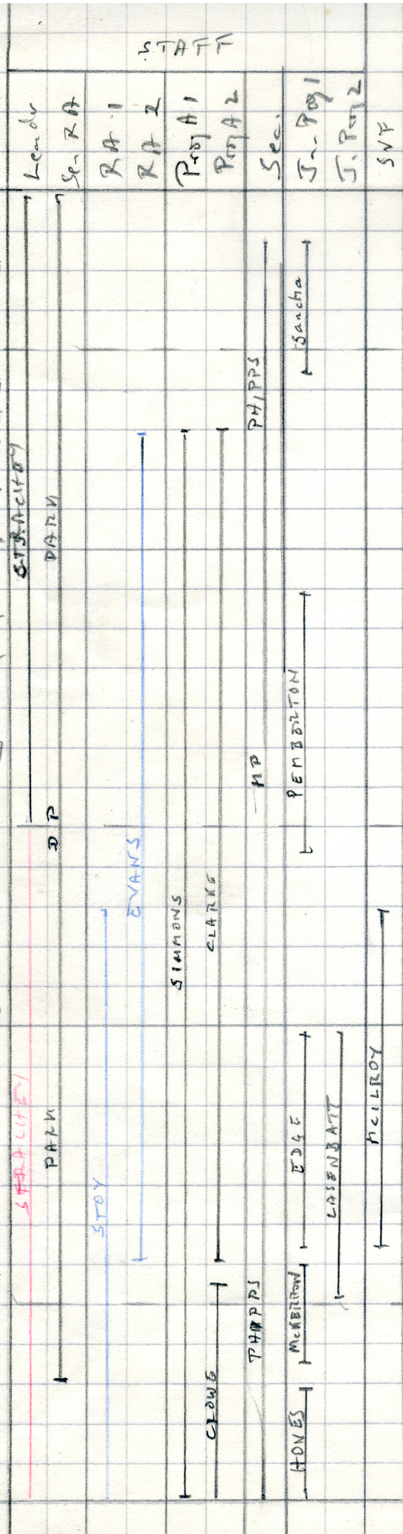
Strachey’s Objective:

“It has long been my personal view that the separation of practical and theoretical work is artificial and injurious.

Much of the practical work done in computing, both in software and in hardware design, is unsound and clumsy because the people who do it have not any clear understanding of the fundamental design principles of their work.

Most of the abstract mathematical and theoretical work is sterile because it has no point of contact with real computing.

One of the central aims of the Programming Research Group as a teaching and research group has been to set up an atmosphere in which this separation cannot happen.”



1966	Apr	Publications; refs 1,2
	May	
	Jn	Program Schemata (DP)
	Jly	Streams on KDF9 (CS)
	Aug	
	Sept	Publication refs 3, 4
	Oct	Start Transferring CPL from Titan to Atlas
	Nov	
	Dec	Work on Compound Data Structures
1967	Jan	
	Feb	Serious difficulties with Atlas Operating system
	Mar	
	Apr	Transfer GPM to KDF9
	May	I/O Streams on KDF9
	J	
	J	SRC grant amended to a total of £106,350 [illegible] to 31/7/72
	A	
	S	Fundamental Concepts in Prog. Lang. Lectures (ref 5)
	O	First Diploma (MSc) Students (4)
	N	Pub. ref. 5.
	D	
1968	J	
	F	Publication; ref 6
	M	Transfer of BCPL from MIT to KDF9
	A	
	M	Transfer of BCPL from Oxford KDF9 to Cambridge Titan
	J	
	J	Publication Ref. 7
	A	Application to SRC for Modular One Computer
	S	
	O	Student Numbers: Diplomas 3, 1st Yr DPhil 3
	N	
	D	Computer Application granted.

Initial Work

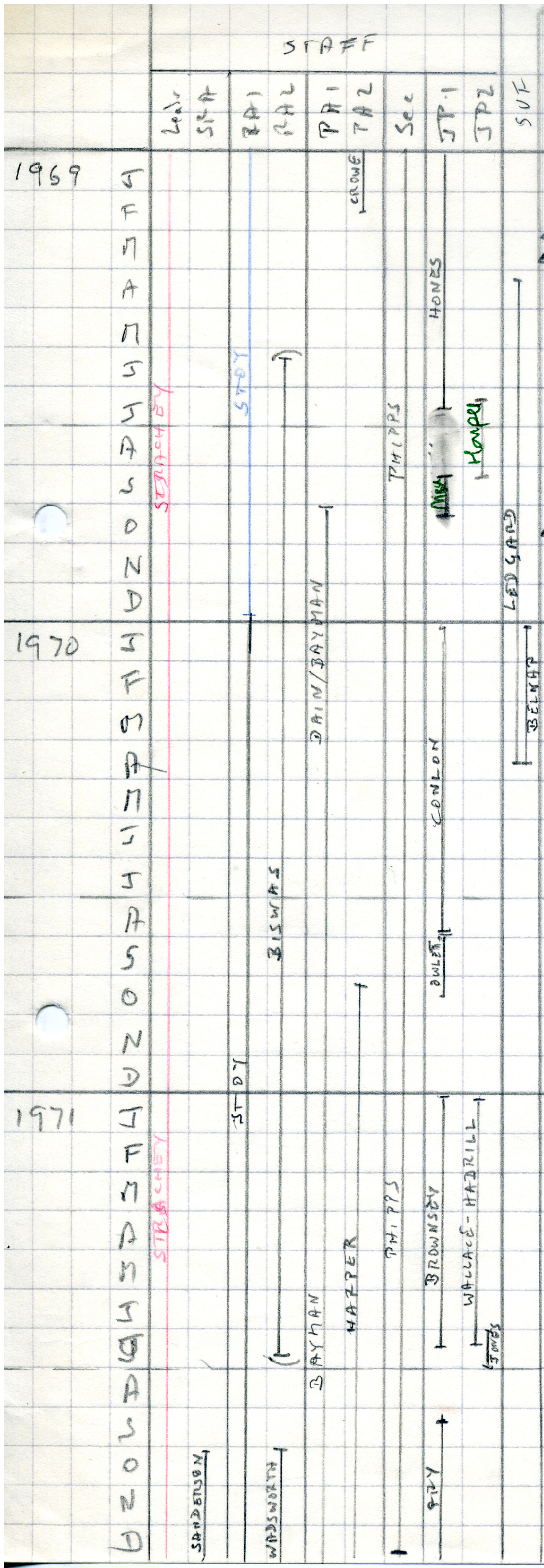
- “Theory”
  - Tidying up CPL Papers (several CPL meetings)
  - *Fundamental Concepts in Programming Languages*
  - Compound Data Structures (Park)
  - Continuing work on program schemata (Park)
  - “Mathematical Semantics” – for first Diploma students

- “Practice”
  - (Attempts to move CPL from Titan to Atlas)
  - Experiments on KDF9
    - Streams
  - Transfer of other work to KDF9
    - GPM
    - BCPL

- Visitors
  - Doug McIlroy works on coroutines (invents pipes)

McIlroy: “I went to Oxford for a year, solely so I could imbibe denotational semantics from the source."

Aho: “Doug McIlroy, though, I think is probably the author of translation...of pipes. That he had written, I think, this unpublished paper when he [was] at Oxford back in the ‘60s....You should read this paper because it's UNIX pipes. One of the interesting things about Doug is that he has had these great, seminal ideas which not everyone knows about. And whether his standards are so high that he doesn't publish them...or what? But it's remarkable...”



First system for Mod. One written in BCPL and compiled (by KDF9) into virtual machine code.

Mod One arrives. System working within 48 hours.

Vienna meeting of WG2.2. CS meets DS

Formal amendment of grant to £168,080 to include computer

Grant amended to £170,830 to allow for extra lineprinter cost.

Dana Scott at Oxford

Development of Theory of Reflexive Domains and start of Scott-Strachey Formal Semantics

Student Nos: MSc 0, DPhil: 1st 4 2nd Yr 3

Publication. Technical Monograph PRG 1

Pub. ref. 8.

Work on Virtual Machines and measurements starts

Students Nos: MSc: 5, DPhil 1st Yr 0 2nd Yr 3 3rd Yr 3

Publication PRG 2,3

Publication PRG 7 ([illegible])

Report on PRG by Miss Bowell (of SRC)

Grant amended to £222,300 and extended to 31/7/74

Student Nos: MSc 2, DPhil: 1st Yr 4, 2nd Yr 1, 3rd Yr. 3

1969

- (1968) JH Morris’s thesis (MIT): proves minimality of *Y* (*Y* is “worst possible” operator)

- PRG gets its own computer
  - (32K store; paper tape; no disc)
  - Prepare system on KDF9 in Jan, Feb
  - One visit to Hemel Hempstead
  - Machine arrives in March; system working within 48hrs.

- Dana Scott visits in Michaelmas Term
  - Strachey meets Scott at WG2.2 in Vienna (April)
  - Long weekly seminars in Michaelmas Term
    1) “Type-theoretic alternative to CUCH, ISWIM, OWHY”
    2) More explicit representation and derepresentation
    3) Inverse limit construction for reflexive domain

- PRG plans to publish Technical Monographs

Morris's Thesis Proof

- **Definition:** $A = B$: A $\mathbf{cnv}_{\beta}$ B.

- **Definition:** $A \supset B$: For any E, whenever $E[B]$ has a normal form,

  $E[A] = E[B]$.

- **Theorem:** If $A \supset FA$, then $A \supset YF$.

- **Corollary:** If $A = FA$, then $A \supset YF$.

1969

- (1968) JH Morris’s thesis(MIT): proves minimality of *Y* (*Y* is “worst possible” operator)

- PRG gets its own computer
  - (32K store; paper tape; no disc)
  - Prepare system on KDF9 in Jan, Feb
  - One visit to Hemel Hempstead
  - Machine arrives in March; system working within 48hrs.

- Dana Scott visits in Michaelmas Term
  - Strachey meets Scott at WG2.2 in Vienna (April)
  - Long weekly seminars in Michaelmas Term
    1) “Type-theoretic alternative to CUCH, ISWIM, OWHY”
    2) More explicit representation and derepresentation
    3) Inverse limit construction for reflexive domains

- PRG plans to publish Technical Monographs

The Modular One and OS1...

- ***Always*** used with IC interpreter, ***never*** the raw machine

- OS

  “The most important single feature, however, is the hierarchical nature of its control structure, which avoids the need for a special job-control language.”

  - “Job Control Language” should be (subset of) the programming language
  - Job invocation same as function call
    - Difference: in case of error abandon job

- Streams (and later files)

  “The input/output system uses a very general form of stream; the filing system is designed to have a clear and logical structure.”

  - a stream characterized by operations available on it
  - file structure very similar to Unix’s
    - inodes, hard and symbolic links, etc
    - (eventually) predecessor directories (form of version control)

A Stream

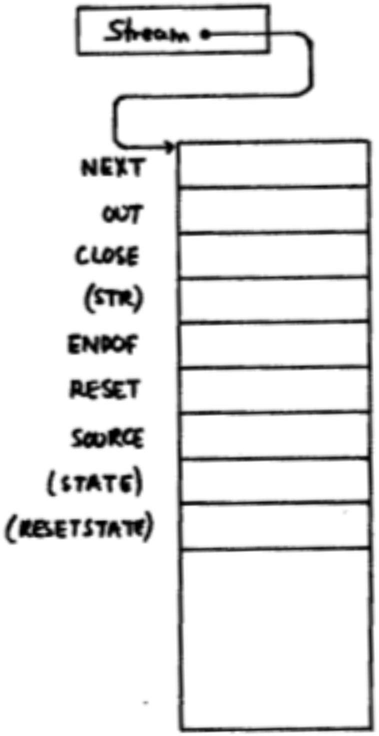


Fig. 14 Structure of a slow stream

let Next[S] = (S↓NEXT)[S]

and Out[S,x] **be** (S↓OUT)[S,x]

. . .

The Modular One and OS1...

- ***Always*** used with IC interpreter, ***never*** the raw machine

- OS

  “The most important single feature, however, is the hierarchical nature of its control structure, which avoids the need for a special job-control language.”

  - “Job Control Language” should be (subset of) the programming language
  - Job invocation same as function call
    - Difference: in case of error abandon job

- Streams (and later files)

  “The input/output system uses a very general form of stream; the filing system is designed to have a clear and logical structure.”

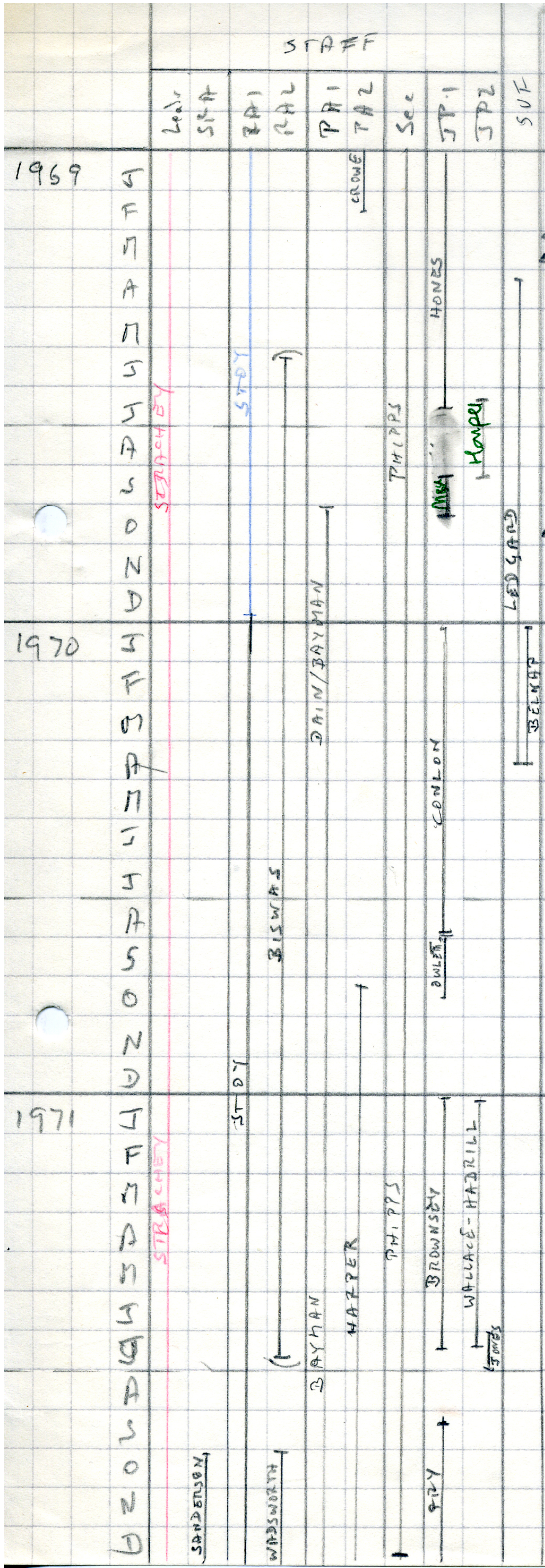
  - a stream characterized by operations available on it
  - file structure very similar to Unix’s
    - inodes, hard and symbolic links, etc
    - (eventually) predecessor directories (form of version control)

1969

- PRG gets its own computer
  - (32K store; paper tape; no disc)
  - Prepare system on KDF9 in Jan, Feb
  - One visit to Hemel Hempstead
  - Machine arrives in March; system working within 48hrs.

- Dana Scott visits in Michaelmas Term
  - Strachey meets Scott at WG2.2 in Vienna (April)
  - Long weekly seminars in Michaelmas Term
    1) “Type-theoretic alternative to CUCH, ISWIM, OWHY”
    2) More explicit representation and derepresentation
    3) Inverse limit construction for reflexive domains

- Plan PRG Technical Monographs



First system for Mod. One written in BCPL and compiled (by KDF9) into virtual machine code.

Mod One arrives. System working within 48 hours.

Vienna meeting of WG2.2. CS meets DS

Formal amendment of grant to £168,080 to include computer

Grant amended to £170,830 to allow for extra lineprinter cost.

Dana Scott at Oxford

Development of Theory of Reflexive Domains and start of Scott-Strachey Formal Semantics

Student Nos: MSc 0, DPhil: 1st 4 2nd Yr 3

Publication. Technical Monograph PRG 1

Pub. ref. 8.

Work on Virtual Machines and measurement starts

Students Nos: MSc: 5, DPhil 1st Yr 0 2nd Yr 3 3rd Yr 3

Publication PRG 2,3

Publication PRG 7

Report on PRG by Miss Bowell (of SRC)

Grant amended to £222,300 and extended to 31/7/74

Student Nos: MSc 2, DPhil: 1st Yr 4, 2nd Yr 1, 3rd Yr. 3

STAFF

		Lead	SRA	RA1	RA2	PA1	PA2	Sec	GP1	GP2	SVT	Others	Grants	
1972	J													Work on Continuation starts.
	F	STRACHEY												
	M							CAMPBELL	FRY					
	A		SANDERSON	STOY										
	M					BAYNAM	HARPER							Publication PRG8
	J													
	J	ON LEAVE												Publication PRG9
	A													
	S													
	O				WADSWORTH									Student Nos: MSc 2 ; DPhil 1st Yr 3, 2nd Yr 4 3rd Yr. 1
	N													
	D								CLIFFE					Grant amended to £224,900 to allow for cost of GPO approved modems
1973	J			STOY										
	F													
	M													Publication PRG10/
	A	CS												
	M													
	J		SANDERSON											
	J													Grant B/RG/41961 awarded for £4,925 from 1/10/73 to 3/9/75 for R E Milne to continue work on formal semantics.
	A													
	S													
	O													Student Nos: MSc 1, DPhil. 1st Yr 5, 2nd Yr 2 3rd Yr 4
	N					BAYNAM	HARPER							
	D	STRACHEY												Application for grant for 2 yrs from 1/10/74 for P D Mosses to continue compiler generator.
1974	J											MILNE		Publications PRG 11,12/
	F			ON LEAVE										
	M													Publication (on microfiche) PRG 13
	A													
	M													
	J													
	J													End of grant
	A													
	S													
	O													
	N			STOY									MOSSES	
	D													

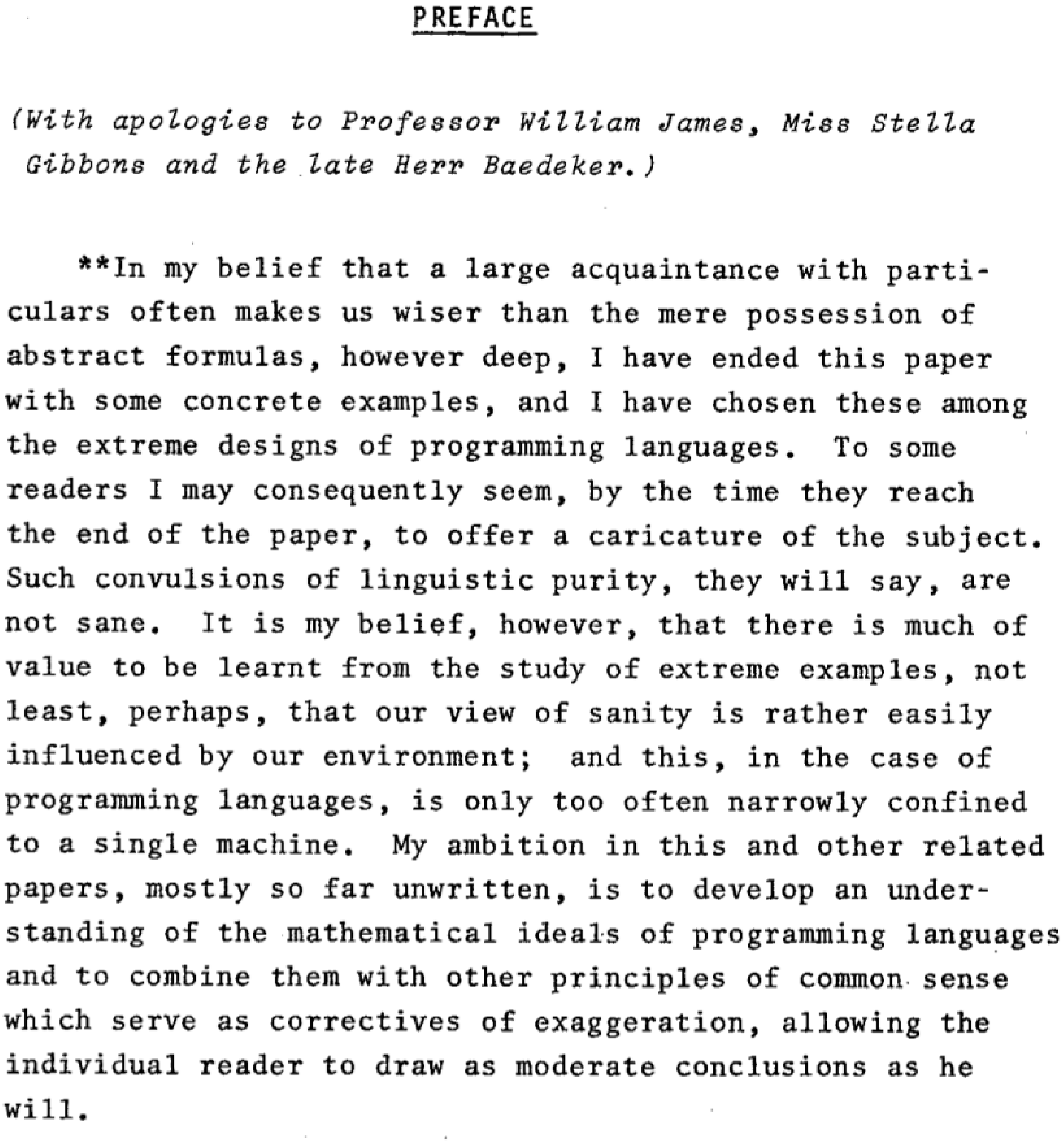
Later Work

- Theoretical
  - Continuations (starts 1972) (*Wadsworth*)
  - Semantics of Algol 60 (*Mosses*)
  - Monographs
  - Lecture courses
  - Proof Rules and Math. Semantics (*Ligler*)
  - Adams Essay (*Strachey and Milne*)

- Practical
  - OS6 / OSPub
  - Design of Instruction Sets (*McGregor*)
  - Semantics & Pragmatics of λ-calculus (graph-reduction) (*Wadsworth*)
  - Compiler Generator (*Mosses*)
  - (Implementation of PAL) (*Turner*)

    →SASL→KRC→Miranda (→Haskell)
  - Computing Mechanisms (*Derret*)

PRG-10: The Varieties of Programming Language

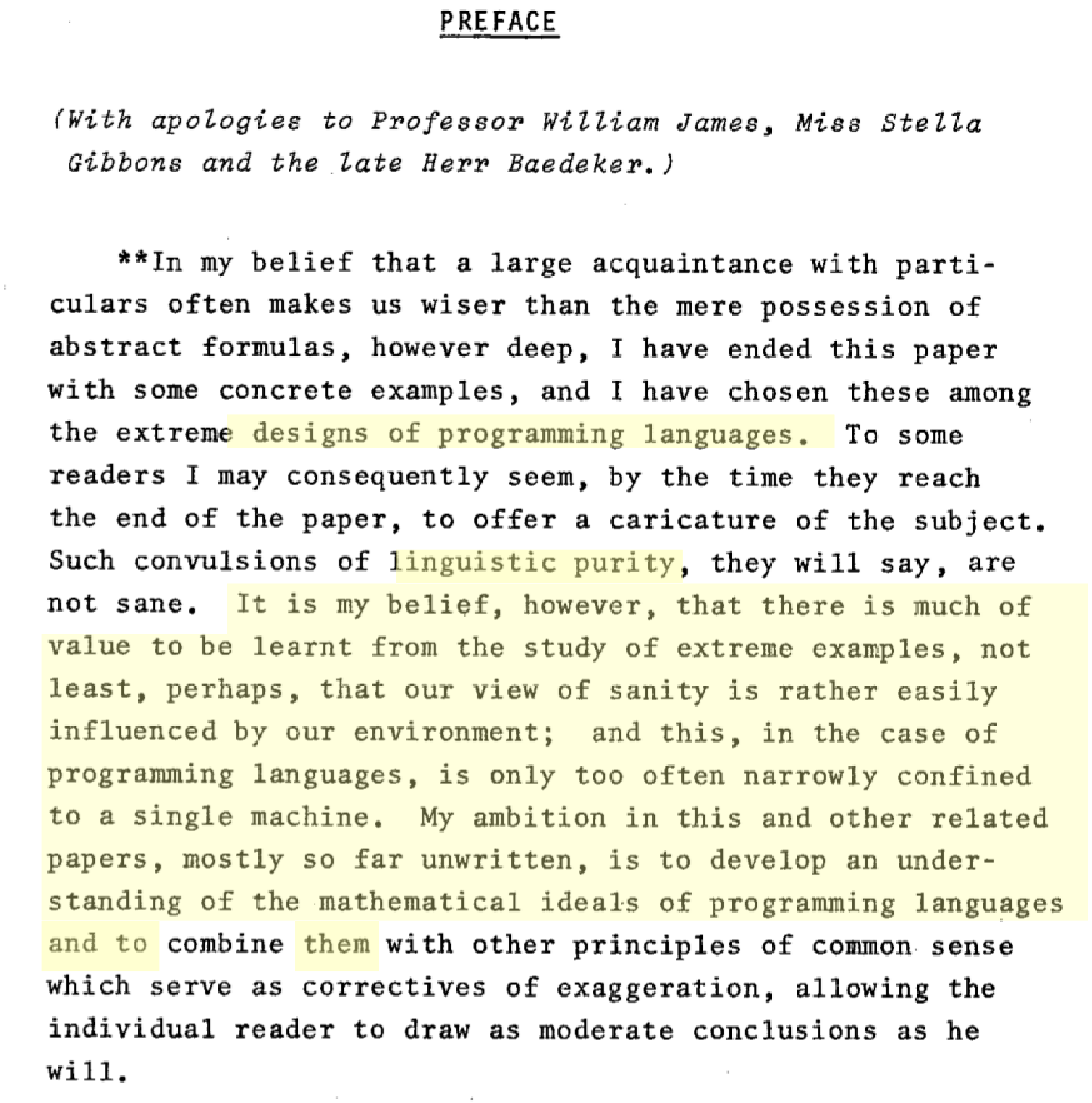


PREFACE

(With apologies to Professor William James, Miss Stella Gibbons and the late Herr Baedeker.)

**In my belief that a large acquaintance with particulars often makes us wiser than the mere possession of abstract formulas, however deep, I have ended this paper with some concrete examples, and I have chosen these among the extreme designs of programming languages. To some readers I may consequently seem, by the time they reach the end of the paper, to offer a caricature of the subject. Such convulsions of linguistic purity, they will say, are not sane. It is my belief, however, that there is much of value to be learnt from the study of extreme examples, not least, perhaps, that our view of sanity is rather easily influenced by our environment; and this, in the case of programming languages, is only too often narrowly confined to a single machine. My ambition in this and other related papers, mostly so far unwritten, is to develop an understanding of the mathematical ideals of programming languages and to combine them with other principles of common sense which serve as correctives of exaggeration, allowing the individual reader to draw as moderate conclusions as he will.

PRG-10: The Varieties of Programming Language



PREFACE

(With apologies to Professor William James, Miss Stella Gibbons and the late Herr Baedeker.)

**In my belief that a large acquaintance with particulars often makes us wiser than the mere possession of abstract formulas, however deep, I have ended this paper with some concrete examples, and I have chosen these among the extreme designs of programming languages. To some readers I may consequently seem, by the time they reach the end of the paper, to offer a caricature of the subject. Such convulsions of linguistic purity, they will say, are not sane. It is my belief, however, that there is much of value to be learnt from the study of extreme examples, not least, perhaps, that our view of sanity is rather easily influenced by our environment; and this, in the case of programming languages, is only too often narrowly confined to a single machine. My ambition in this and other related papers, mostly so far unwritten, is to develop an understanding of the mathematical ideals of programming languages and to combine them with other principles of common sense which serve as correctives of exaggeration, allowing the individual reader to draw as moderate conclusions as he will.

REMARKABLE RELATIONS

The Story of the Pearsall Smith Family

BARBARA STRACHEY

Later Work

- Theoretical
  - Continuations (starts 1972) (*Wadsworth*)
  - Semantics of Algol 60 (*Mosses*)
  - Monographs
  - Lecture courses
  - Proof Rules and Math. Semantics (*Ligler*)
  - Adams Essay (*Strachey and Milne*)

- Practical
  - OS6 / OSPub
  - Design of Instruction Sets (*McGregor*)
  - Semantics & Pragmatics of λ-calculus (graph-reduction) (*Wadsworth*)
  - Compiler Generator (*Mosses*)
  - (Implementation of PAL) (*Turner*)

    → SASL → KRC → Miranda (→ Haskell)
  - Computing Mechanisms (*Derret*)

Correspondence

To the Editor,
The Computer Journal.

An impossible program

Sir,

A well-known piece of folk-lore among programmers holds that it is impossible to write a program which can examine any other program and tell, in every case, if it will terminate or get into a closed loop when it is run. I have never actually seen a proof of this in print, and though Alan Turing once gave me a verbal proof (in a railway carriage on the way to a Conference at the NPL in 1953), I unfortunately and promptly forgot the details. This left me with an uneasy feeling that the proof must be long or complicated, but in fact it is so short and simple that it may be of interest to casual readers. The version below uses CPL, but not in any essential way.

Suppose $T[R]$ is a Boolean function taking a routine (or program) R with no formal or free variables as its argument and that for all R, $T[R] =$ **True** if R terminates if run and that $T[R] =$ **False** if R does not terminate. Consider the routine P defined as follows

```
rec routine $P$
    § $L$ : if $T[P]$ go to $L$
        Return §
```

If $T[P] =$ **True** the routine P will loop, and it will only terminate if $T[P] =$ **False**. In each case $T[P]$ has exactly the wrong value, and this contradiction shows that the function T cannot exist.

Yours faithfully,
C. Strachey.

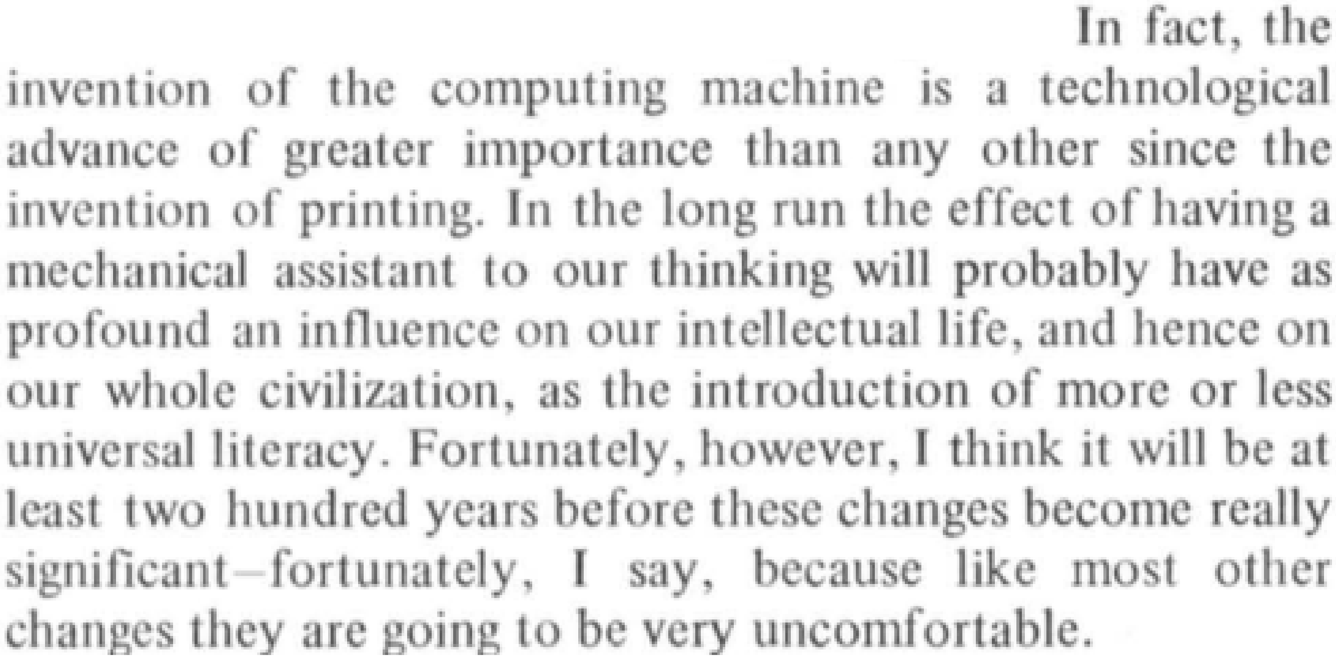
Churchill College,
Cambridge.

Later Work

- Theoretical
  - Continuations (starts 1972) (*Wadsworth*)
  - Semantics of Algol 60 (*Mosses*)
  - Monographs
  - Lecture courses
  - Proof Rules and Math. Semantics (*Ligler*)
  - Adams Essay (*Strachey and Milne*)

- Practical
  - OS6 / OSPub
  - Design of Instruction Sets (*McGregor*)
  - Semantics & Pragmatics of λ-calculus (graph-reduction) (*Wadsworth*)
  - Compiler Generator (*Mosses*)
  - (Implementation of PAL) (*Turner*)
    - →SASL→KRC→Miranda (→Haskell)
  - Computing Mechanisms (*Derret*)

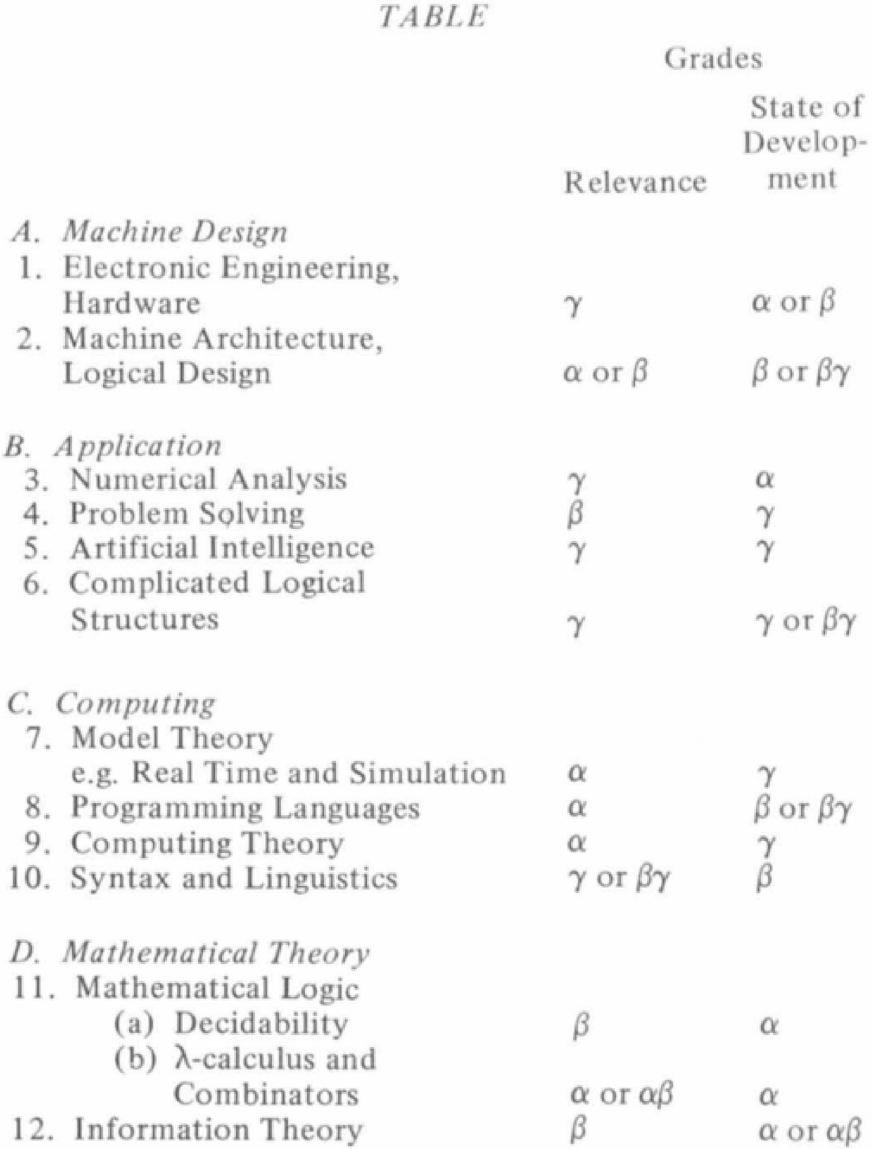
What was he thinking? (1970)



In fact, the invention of the computing machine is a technological advance of greater importance than any other since the invention of printing. In the long run the effect of having a mechanical assistant to our thinking will probably have as profound an influence on our intellectual life, and hence on our whole civilization, as the introduction of more or less universal literacy. Fortunately, however, I think it will be at least two hundred years before these changes become really significant–fortunately, I say, because like most other changes they are going to be very uncomfortable.

Is Computing Science? — Girton Centenary Symposium, 1970

What was he thinking? (1970)



TABLE

	Grades	
	Relevance	State of Development
A. Machine Design		
1. Electronic Engineering, Hardware	γ	α or β
2. Machine Architecture, Logical Design	α or β	β or $\beta\gamma$
B. Application		
3. Numerical Analysis	γ	α
4. Problem Solving	β	γ
5. Artificial Intelligence	γ	γ
6. Complicated Logical Structures	γ	γ or $\beta\gamma$
C. Computing		
7. Model Theory e.g. Real Time and Simulation	α	γ
8. Programming Languages	α	β or $\beta\gamma$
9. Computing Theory	α	γ
10. Syntax and Linguistics	γ or $\beta\gamma$	β
D. Mathematical Theory		
11. Mathematical Logic		
(a) Decidability	β	α
(b) λ-calculus and Combinators	α or $\alpha\beta$	α
12. Information Theory	β	α or $\alpha\beta$

Is Computing Science? — Girton Centenary Symposium, 1970

What was he thinking? (1973)

1. **Current State of Programming Language Theory**

    1.1 Formal Semantics *[the longest subsection]*

    1.2 Proofs of Properties of Programs *[State of Art — Hoare]*

    1.3 Programming Methodology *["In the absence of science we have to fall back on art."]*

2. **Influence of Programming Theory on Programming Practice**

    2.1 Choice of Programming Language

    2.2 Size of Programs *[Modularity; but problems of scale -- not much help from theory yet]*

3. **Possible Technical Improvements**

    3.1 Technical Problems and Myths

    (a) Unsuitable language facilities *[justified concern]*

    (b) Inefficient object code *[not justified]*

    (c) Compiling slow *[no comment]*

    (d) Inadequate debugging facilities *[justified]*

    3.2 Technical Remedies

    (a) Better languages *[not designed by amateurs]*

    (b) Better compilers and object code *[designed together]*

4. **Practical and Human Problems**

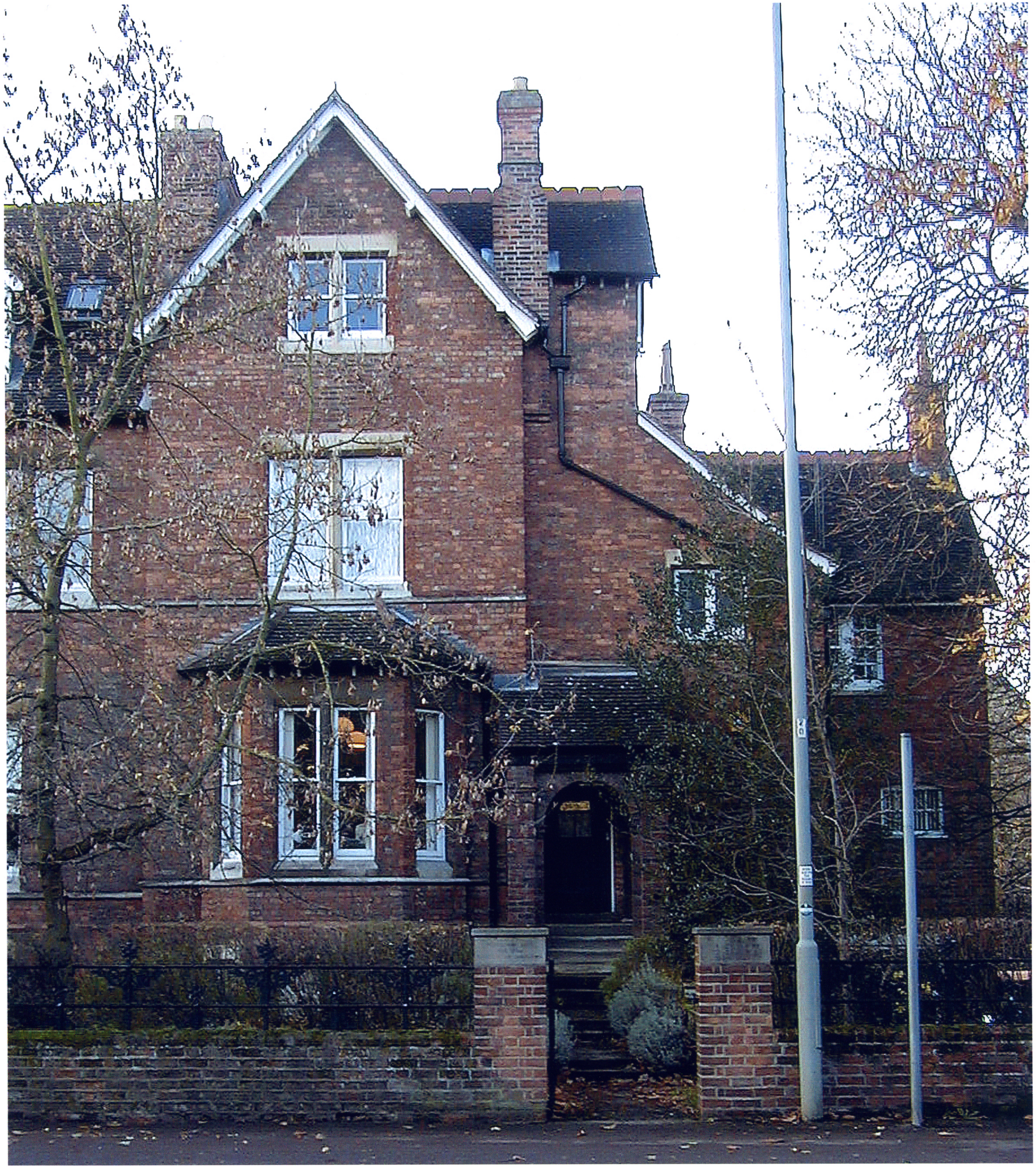
    4.1 Programming Style

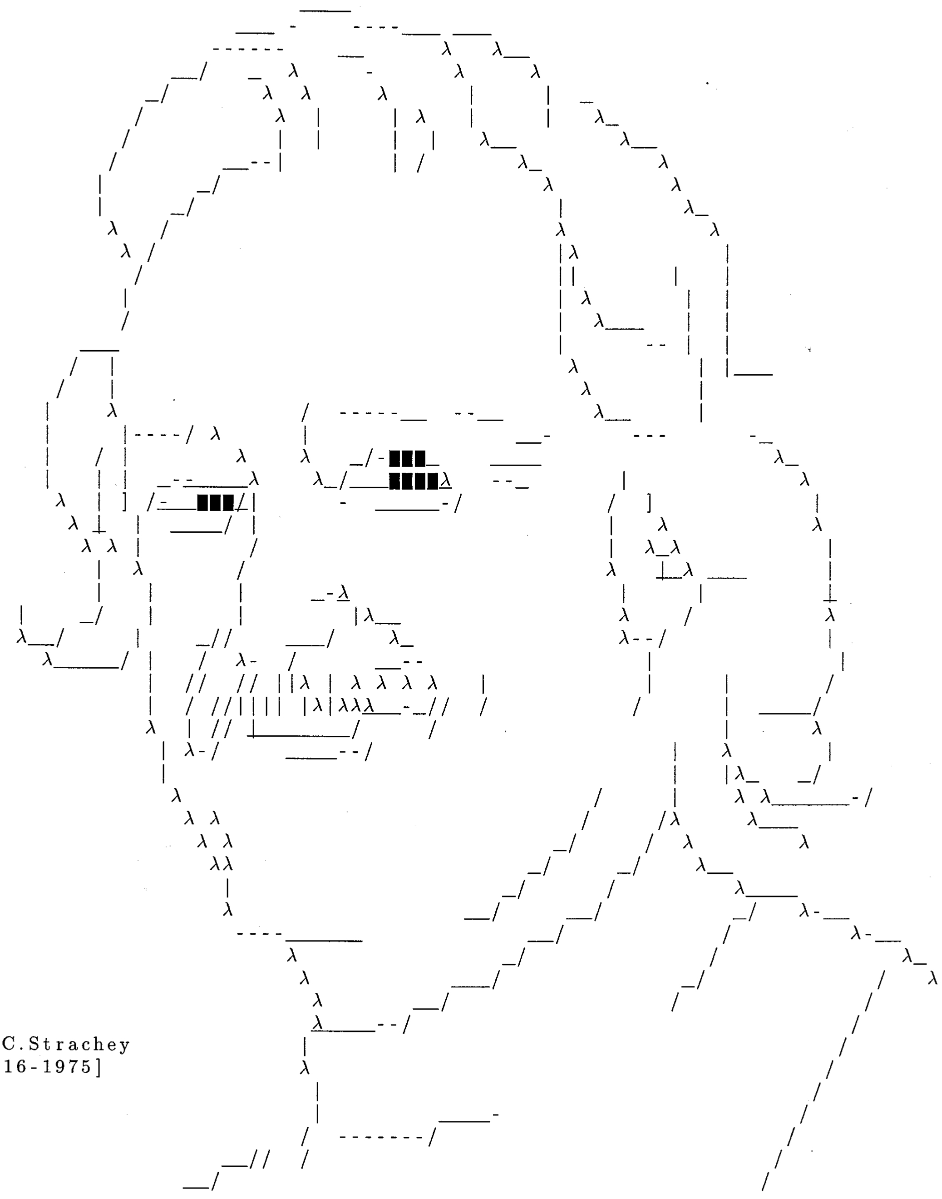
    4.2.Toughness *[Don't put up with inadequate equipment* etc.]

5. **Prospects**

    No overnight change *[e.g. jet engines co-existing with props and turbo-props]*

How Can we Put Programming Theory into Practice? — Loughborough Tech. July 1973





Prof.C.Strachey
[1916-1975]