THE HISTORY OF STANDARD ML
IDEAS, PRINCIPLES, CULTURE

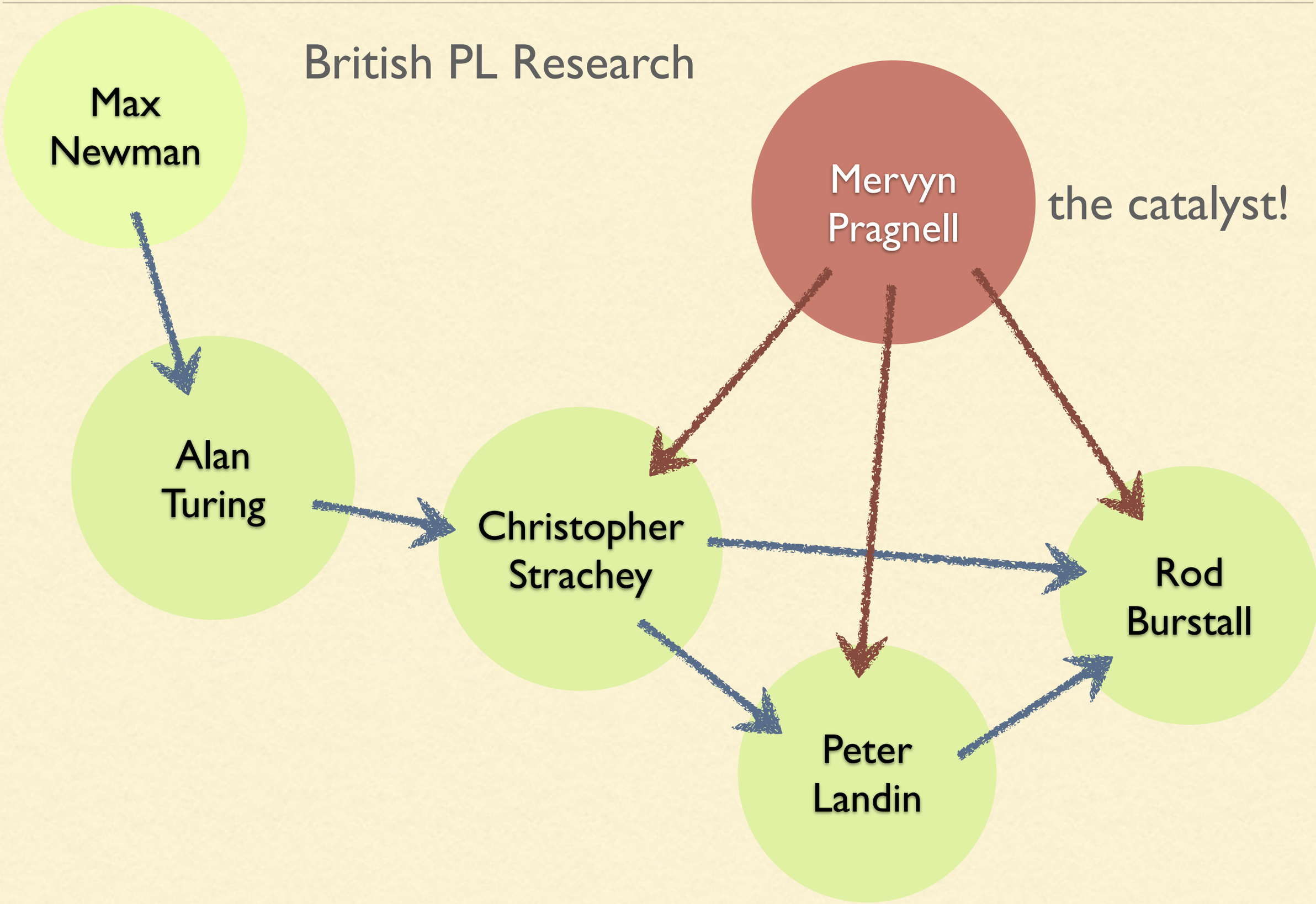
David MacQueen
University of Chicago (Emeritus)

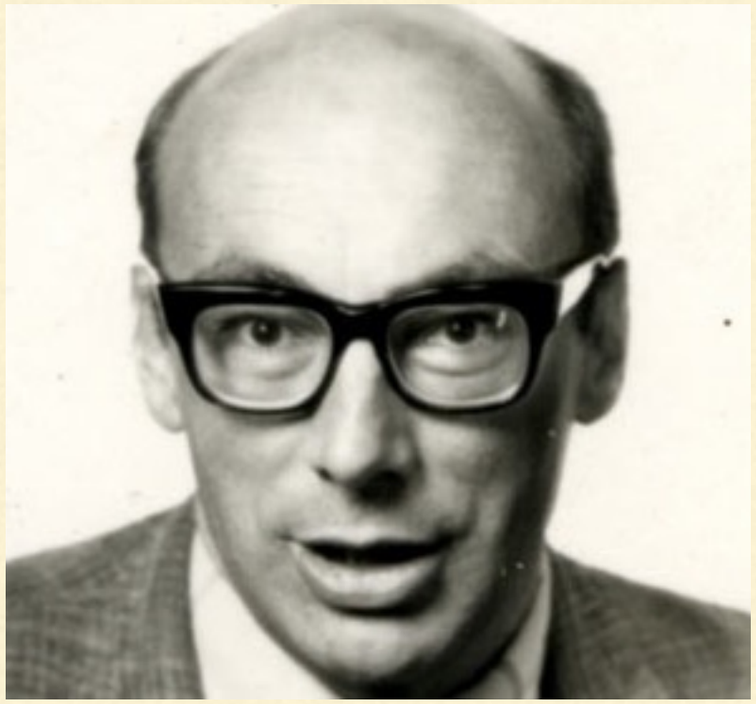
ML Family Workshop
September 3, 2015

Let us start by looking back a bit further at some of the people who founded the British community of programming language research.

For instance, Turing, Strachey, Landin, etc.

British PL Research

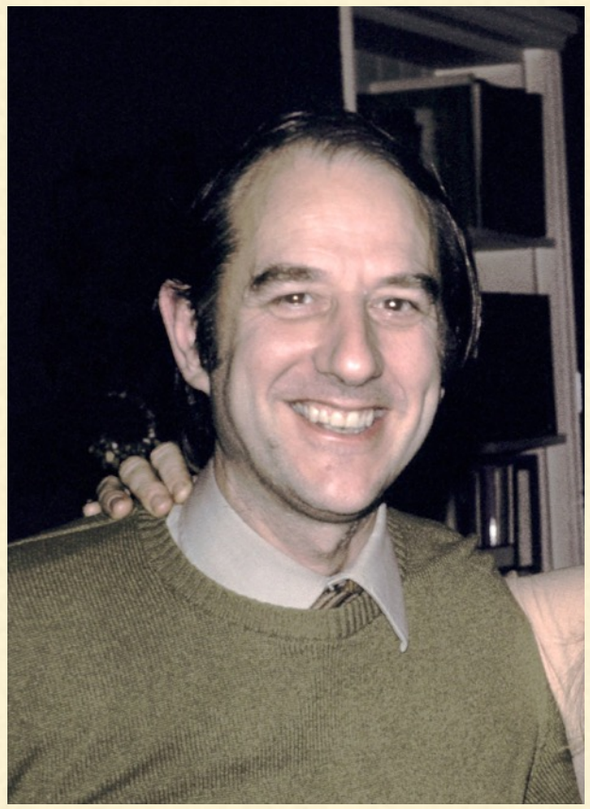




Peter Landin

I used to go out to a cafe just around the corner from this reference library … and one day I was having my coffee in Fields cafe, and a voice came booming across the crosswise tables, and this voice said "I say didn't I see you reading *Principia Mathematica* in the reference library this morning?" And that's how I got to know the legendary Mervyn Pragnell who immediately tried to recruit me to his reading group.

Peter Landin talk at the Science Museum. 5 June 2001, available on Vimeo



Rod Burstall

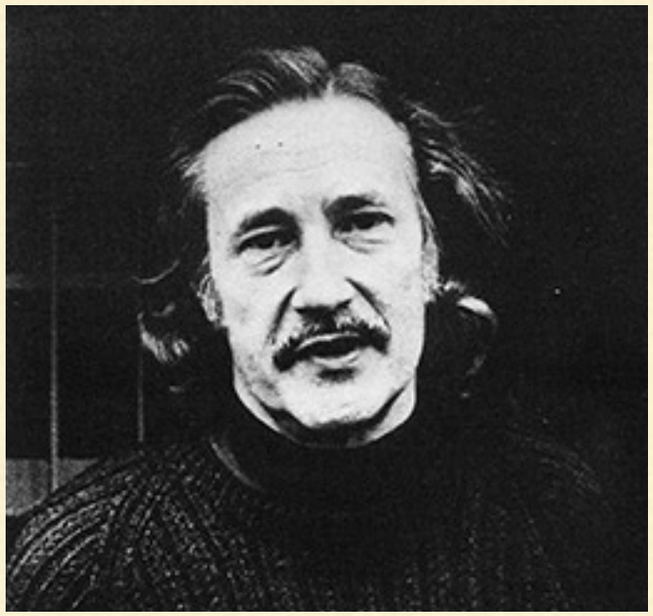
’Rod Burstall … recalls that, while looking for a logic text in a London bookshop, he asked a man whether the shop had a copy. "I'm not a shop assistant," the man responded, and "stalked away," only to return to invite him to join the informal seminar where he would meet Peter Landin and, subsequently, Christopher Strachey.’

Mervyn Pragnell's Underground Study Group

"The sessions were held illicitly after-hours at Birkbeck College, University of London, without the knowledge or permission of the college authorities.[8] Pragnell knew a lab technician with a key that would let them in, and it was during these late night sessions that many famous computer scientists cut their theoretical teeth. This also appears to be the place Landin would first meet Strachey, and it marks the beginning of an important intellectual relationship between these two men."

Along with Strachey, Landin, and Burstall, Robin Milner admitted attending "once or twice".

Christopher Strachey (1916 - 1975)



Friend of Turing (at Cambridge & Manchester)

First checker playing program

Playing songs on Manchester Mark I

Combined Programming Language (CPL)

"functions as first-class citizens"

Coined "currying"

L-values

CPL => BCPL => B => C => C++

"Fundamental Concepts in Programming Languages" (1967)

Parametric polymorphism

Denotational Semantics

with Dana Scott from 1969

Continuations

with Wadsworth

Time-sharing (1958)

Employed Landin, 1960-64

Wadsworth on Strachey

Strachey had an acute sense of when something was "right" — generally when it was simple enough and elegant enough that it could be seen intuitively to be right — and he abhorred overelaboration or contrived methods that "sort of worked". A favorite motto of his … was "You can push a pea up a mountain with your nose if you really want to, but that does not mean that it is a good way of getting it there". For me, this was a kind of "Strachey test".

Burstall on Strachey

His elegance of manor was accompanied by an elegance of thought and language which was a continual inspiration.

Peter Landin

The mechanical evaluation of expressions, 1964 — SECD

A correspondence between ALGOL 60 and Church's Lambda-notation: Part I; Part II, 1965 — *streams*

A generalization of Jumps and Labels, 1965 — *continuation precursor*

The next 700 programming languages, 1966 — ISWIM

Programs and their Proofs: An Algebraic Approach (with Burstall), 1969 — *foreshadows algebraic data types*

PAL: an implementation of ISWIM at MIT (Evans)

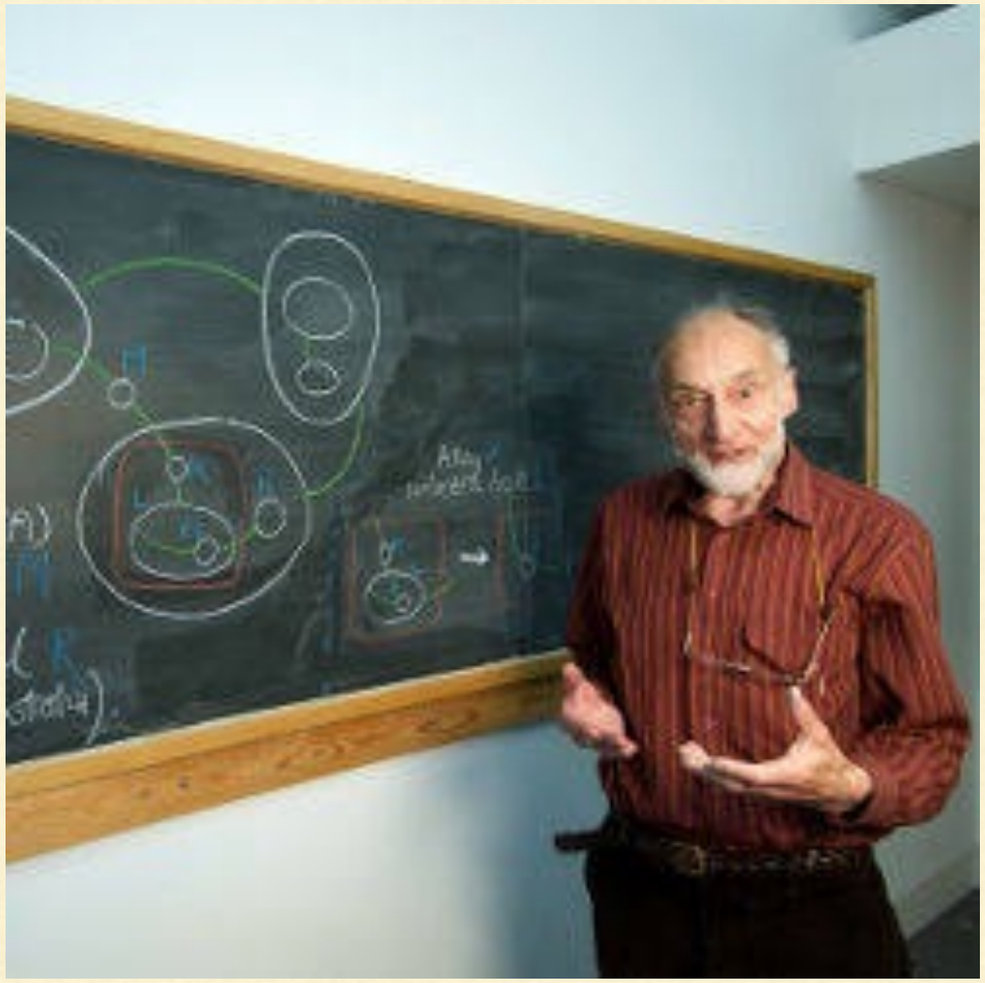
More Background for Strachey and Landin

Special Issue on Strachey
Higher-Order and Symbolic Computation
Volume 13, Issues 1-2, April 2000

Special Issue on Landin
Higher-Order and Symbolic Computation
Volume 22, Issue 4, December 2009

Landin's Jumps and Labels paper
Higher-Order and Symbolic Computation
Volume 11, Issue 2, December 1998

Robin Milner



King's College Cambridge, 1957
School teacher
Ferranti - programmer
City University, London
Swansea
Stanford 1971 - 72
Edinburgh 1973 - 95
Cambridge 1995 - 2010
Turing Award 1991

The idea of a machine proving theorems in logic, and the idea of using logic to understand what a machine was doing ... this double relationship began to inspire me because it was clearly not very simple.

Principles they lived by

Strachey, Landin, Burstall, Milner (and others like Tony Hoare) established a British tradition of programming language research characterized by:

1. Realizing the importance of foundations and semantics in the study of computation and programming.

2. Seeking clarity, rigor and elegance through the use of mathematical ideas and techniques, particularly from logic and algebra.

These principles were strongly embedded in the Edinburgh community.

The Situation (Edinburgh, Late 70s)

Edinburgh LCF completed 1978-9, with ML as its metalanguage

Luca Cardelli arrives in Edinburgh as a grad student, Fall 1978

Rod Burstall and Dave MacQueen are working on HOPE in 1978

Milner and Burstall sub-communities unified at King's Buildings in 1979

Community that will form the nucleus of the Laboratory for Foundations of Computer Science (LFCS) comes together, though LFCS will not be formally created until Jan. 1986

Edinburgh Connections

LCF/ML aka DEC10 ML

- Embedded within the LCF system as its MetaLanguage
- Supported PPLAMBDA object language (Scott's LCF logic)
  - terms, formulas, theorems
  - theorems an abstract type whose values can only be produced through inference rules of the LCF logic
- Quotation/antiquotation of object language syntax
- Proof tactics and higher-order tacticals for combining tactics

Main Features of LCF/ML

- Based on Landin's ISWIM
- Type inference -- Milner's let-polymorphism, principal types
- Abstract types (`abstype` declarations)
- Simple binary product and sum types: `t1 # t2`, `t1 + t2`
- Mutable variables declared with "`letref`"
- Nested tuple and list binding patterns ("varstructs")
- Looping conditionals

  `{if … then|loop … }* else|loop …`
- Failures and failure trapping passing strings (tokens)

DEC10 ML Implementation

- Implemented in Stanford (later Rutgers) Lisp
- ML translated to Lisp code
- Lisp code interpreted (hence slow!)
- Parser based on Vaughan Pratt's precedence parser (POPL 1973) — hence , ; ;; as separators

VAX ML (aka Cardelli ML)

- 1980: Luca starts work on his own dialect of ML and a compiler implementing it.

- Working compiler (including garbage collection) being distributed by the end of 1981.

- Early version, 1980 — 1982
  described briefly in mlchanges.doc

VAX ML: Language Innovations

- Labeled records and variants — *structural!*
  partly inspired by Plotkin's lectures on domain theory
- Declaration combinators (next slide)
- **ref** type operator with interface: ref, !, :=
- Stream I/O, with bidirectional streams
- Basic modules (with separate compilation, serialization)

Declaration Combinators

- and — simultaneous
- enc — sequential (enclosing) ==> `d1; d2`
- ins — local (inside) ==> `local d1 in d2 end`
- rec — recursive
- with — special for forming abstract types
  (with `t <=> ty` type declarations)

VAX ML compiler (Edinburgh, 1980 - 1982)

- Runs under VAX/VMS
- Written entirely in Pascal, including runtime system
- Functional Abstract Machine (FAM) as intermediate language
- Generates native VAX machine code from FAM code
- Serialization/pickling of modules for export/import

Distributed to users starting in 1981

Role of VAX ML

- Demonstrates viability of ML as general purpose language with an efficient implementation
- Creates incentive to control proliferation of dialects (B. Sufrin) leading to Milner's proposal for "Standard" ML (April 83)
- An immediate precursor of Standard ML
- A testbed for early experiments with Standard ML design

Standard ML Design

- Design Meetings: April1983, June 1984, May 1985
  - Proposal drafts (Core, Modules, I/O)
  - Comments, correspondence, meeting records
- Formal definition, 1986-89
  - The Definition of Standard ML (Milner, Tofte, Harper) "SML 90"
  - Some formal foundations provided by Tofte's thesis

April 1983 - First Meeting

Prompted by Bernard Sufrin, Robin writes a first draft of a new language proposal incorporating ideas from LCF/ML, VAX ML, and Hope.

A group fortuitously assembles in Edinburgh in early April to discuss Robin's proposal, meeting in Robin's living room.

Rod Burstall	Alan Mycroft	
Luca Cardelli	Larry Paulson	
Guy Cousineau	David Rydeheard	Gerard Huet
Mike Gordon	Don Sannella	Peter Mosses
David MacQueen	John Scott	David Schmidt
Robin Milner	Brian Monahan	
Kevin Mitchell	Stefan Sokolowski	
physical participants		*virtual participants*

A PROPOSAL FOR STANDARD ML (TENTATIVE)

Robin Milner, April '83.

1. Introduction

The language proposed here — called here "Standard ML" but a better name may be found — is not supposed to be novel. Its aims are

(i) to remove some redundancies and bad choices in the original design of ML;

(ii) to "round out" ML in one particular respect — namely the use of patterns in parameter passing not only for the standard data constructions (pairing, lists) but also for constructions which are user defined;

(iii) to make sure that just enough input/output is standardised to allow serious work to be done (the user should be able to define enough higher-level i/o functions within the language that he feels no pressing need to extend it);

(iv) thereby to determine as clearly as possible — for the benefit of the user community — what is guaranteed in ML,

First draft code example

```
local
    rec data 'a seq ← nil | cons of 'a # 'a seq
in
abstype 'a monoid <=> 'a seq
with
    local rec var ap ← fun nil, m . m
                         | cons(x,l), m . cons(x, ap(l, m))
    in
var empty ← absmonoid nil
and singleton(x) ← absmonoid (cons(x, nil))
and concat (absmonoid l, absmonoid m) ← absmonoid (ap(l, m))
;
```

First Draft features

a form of data type declaration; data constructors in patterns

no records or variants (from VAX ML)

clausal function expressions: `fun v1. e1 | … | vn. en`

monomorphic references and equality

"local" declaration instead of Cardelli's "ins" operator

escape with token and a single trap form

```
e1 trap v1. e1 | … | vn. en
```

Further drafts (for Core SML)

4/83: Changes to proposal for Standard ML, Milner

6/83: A Proposal for Standard ML (second draft), Milner (49 pages)

11/83: A Proposal for Standard ML, Milner (27 pages) ["final"]

6/84: Record of the Standard ML Meeting, Edinburgh, 6-8 June 1984
MacQueen and Milner

7/84: Standard ML - The Core Language, Milner [changes summary]

7/84: The Standard ML Core Language, Milner [LFP 84 draft?]

10/84: The Standard ML Core Language, Milner

6/85: Report on the Standard ML Meeting, Edinburgh, May 23-25, 1985, Harper

9/85: The Standard ML Core Language (Revised), Robin Milner

Other Design Drafts - I/O and Modules

Stream I/O:

12/83: Stream Input/Output, Cardelli [Polymorphism 3,1]

2/85: Proposal for I/O in Standard ML, K. Mitchell and Milner

6/85: Standard ML Input/Output, Harper [ML Workshop 85]

Modules:

8/83: Modules for Standard ML, MacQueen [preliminary, incomplete draft]

8/84: Modules for Standard ML, MacQueen [LFP 84, Polymorphism]

10/85: Modules for Standard ML, MacQueen [final draft before Definition]

The Definition of Standard ML (SML '90)

Work on the formal definition started sometime in 1986. Three drafts of the formal definition of the entire language appeared as Edinburgh LFCS Tech Reports written by Milner, Harper, and Mads Tofte (Robin's student).

8/87: The Semantics of Standard ML, Version 1

8/88: The Definition of Standard ML, Version 2

5/89: The Definition of Standard ML, Version 3

The Definition was eventually published in 1990 by MIT Press.

The exn type and exception constructors

While work on the Definition was proceeding, there was one more significant change to the Core language.

The **exn** type was introduced with declarations for exception constructors and pattern matching over exception patterns in exception handlers.

7/87: Exceptions as Constructors, Appel and MacQueen

5/88: Unifying Exceptions With Constructors in Standard ML, Appel, MacQueen, Milner, Tofte

Three Early Implementations

- Cardelli's VAX ML => "subStandard ML"

  Early Standard ML features added in 1983-84

- Edinburgh ML => Edinburgh SML

  Kevin Mitchell, Alan Mycroft, John Scott and Bob Harper

- PolyML by Dave Matthews at Cambridge

  Standard ML front end built for his Poly compiler

Later Implementations

- Standard ML of New Jersey
- MLKit (with Regions)
- Moscow ML
- MLton

Big Ideas in Standard ML '90

Let-polymorphism, type inference, principal types
Newman (1943!), Curry (1969), Hindley (1969), Milner

Algebraic data types, with clausal functions, case analysis
via pattern-matching [from Hope]

Modules with signatures, functors, sharing specifications,
and generative structures ("strong structure sharing")

Exceptions as an extensible data type

Support for ref types using "imperative type variables"

The Evolution of Algebraic Data Types

The history design of algebraic datatypes goes back to Landin.

1. The informal data descriptions used with ISWIM.

2. The formal development in Landin and Burstall's paper "Programs and Their Proofs: An Algebraic Approach".

3. Burstall's toy language NPL from 1977.

4. The Hope language (1980).

An AE is either

an *identifier*,

or a *λ-expression* (*λexp*) and has a *bound variable* (*by*)
which is an identifier or
identifier-list,
and a *λ-body* (*body*)
which is an AE,

or a *combination* and has an *operator* (*rator*)
which is an AE,
and an *operand* (*rand*)
which is an AE.

meta-ISWIM

```
datatype AE = ID of identifier
            | LAMBDA of {bv: identifier,
                         body : AE}
            | COMB of {rator : AE, rand : AE}
```

Mistakes in the Design Process

1. Freezing the formal definition in the form of a published book: If a programming language is implemented and used, its definition will need to be "maintained", and even allowed to evolve (with extreme care). The definition should have been an **open** but carefully **managed** document. [sml-family.org is finally doing something about this.]

2. The SML '90 "Basis" environment specified in Appendix C of the Definition was totally inadequate (only 43 items), leading to incompatible basic libraries for different implementation. This wasn't fully corrected until the publication of the "Standard ML Basis Library" (Gansner and Reppy) several years after SML '97.

Further Developments: The 1990s

- SML '97: The Definition of Standard ML (Revised)
- The ML2000 program

SML ‘97

In 1995, the Newton Institute program on Semantics of Computation brought Milner, Harper, Tofte and MacQueen together in Cambridge, where we started working on a revision of the Definition of Standard ML. The notable changes are:

Type abbreviations in signatures (SML/NJ 0.93; Harper, Leroy POPL 94).

Opaque signature matching.

Weak structure sharing (structure sharing implies only type sharing).

Value polymorphism (elimination of imperative type variables).

Replication of datatypes.

ML 2000

A series of meetings from 1993 through about 2000 devoted to the effort to define a "next generation" of ML. Consensus was not achieved, mainly because of disagreement over the idea of adding object-oriented features to the language.

The Moby language of Fisher and Reppy could be considered one byproduct of the program, demonstrating a possible combination of ML and objects. OCaml may be another example of a hybrid language.

A summary paper:

Principles and Preliminary Design for ML2000

An Advertisement

A new web-site: sml-family.org

Online copies of the SML ’90 and SML’97 Definitions

Alternate type-theoretic definition from CMU

Successor ML definition, a work in progress

Revision, extension of the SML Basis libraries

A history site providing documentation of the history of Standard ML.

Final Thoughts

Trying to recover the history of a 30 year-old design can be difficult, but it is also fascinating.

There are hundreds of documents, but there are also gaps and fading memories, and in some cases memories that are lost forever.

But is worth trying to understand where ideas came from, and how they developed over time, and and why various alternatives were eliminated, if only to avoid remaking old mistakes!