(3)

SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

COMPUTING DIVISION

DISTRIBUTED INTERACTIVE COMPUTING NOTE 736

ML, LCF, and HOPE
Meeting at RAL on 17 November 1982

Issued by
C P Wadsworth

30 Nov 82

DISTRIBUTION:
R W Witty
C P Wadsworth
Those present + apologies
Tech Meetings/Workshop/LCF file

(new page)

(3)

SCIENCE AND ENGINEERING RESEARCH COUNCIL
RUTHERFORD APPLETON LABORATORY

COMPUTING DIVISION

SOFTWARE TECHNOLOGY INITIATIVE

ML, LCF, and HOPE
Meeting at RAL on 17 November 1982

Present: R W Witty, RAL (Chairman)
C P Wadsworth, RAL (Notes)
R Milner, Edinburgh
R M Burstall, Edinburgh
J Scott, Edinburgh
M Hennessey, Edinburgh
K Mitchell, Edinburgh
D Schmidt, Edinburgh
A Mycroft, Edinburgh
B Sufrin, Oxford
M Raskovsky, Oxford
J Hughes, Oxford
J Darlington, Imperial College
J Cunningham, Imperial College
S Zappacosta, Imperial College
C Jones, Manchester
A Wills, Manchester
J Welsh, UMIST
R Gallimore, UMIST
D Coleman, UMIST

Apologies for absence were received from

M Gordon, Cambridge
L Paulson, Cambridge
C A R Hoare, Oxford
K Hanna, Kent
J Cramer, Imperial College
I Pyle, York
R Boot, NCC

Gordon and Paulson had sent written comments for the meeting, a copy of which is appended to these notes.

1. INTRODUCTION

RWW welcomed participants. The meeting was the first of many he hoped to organise under the SERC's Software Technology Initiative (STI) in which researchers in particular areas are brought together to identify common needs and goals and to provide guidance to the STI on how best to exploit their research achievements.

CPW explained the background to this meeting. The SERC's Software Technology Panel has already reviewed favourably several proposals for the further exploitation of ML, LCF, and HOPE, and is keen to see that the right decisions are taken now both to meet the needs of particular research projects and to propagate knowledge and availability of these systems to the wider community. There is particular interest in mounting versions on the ICL PERQ since PERQs running Unix will be widely available in the SERC community shortly. The Panel also felt there is a specific need for a tutorial on ML, LCF, and HOPE.

2. REVIEW OF ML, LCF, AND HOPE

2.1 Background

CPW characterised ML, LCF, and HOPE for discussion purposes as being at the centre of an area of research activity that can be distinguished as typed functional programming, derived from Landin's ISWIM (not itself typed).

ML is an ISWIM-style higher order functional programming language whose distinctive features are

(a) a polymorphic type discipline, with compile-time type checking, and

(b) a facility for abstract type modules.

LCF is an application and extension of ML as a programming metalanguage for conducting proofs in a particular object logic (called PPLAMBDA). The LCF proof style relies heavily on the use of higher order functions, for goal-directed proof via tactics. The logic is embedded in ML as particular data types term, form, and thm with relevant primitives (abstract syntax constructors and destructors on terms and formulae, axioms and inference rules as operations producing theorems), with quotations providing a concrete syntax for input. A hiearchical filing system for theories of the logic is an important practical component of the system, to store useful theorems in theory files for later retrieval.

HOPE may be viewed as ML without assignment (a deliberate design choice) with additional features:

(a) pattern matching with respect to data constructors,

(b) overloading of variable meanings (different meanings for the same symbol at different types),

(c) dynamic data structures (possibly "infinite") through lazy evaluation.

Currently, HOPE lacks a mechanism for trapping exceptions/failures. Pattern matching and the use of equations to define functions give HOPE a quite distinct character compared to ML that makes it more suitable for program specification and transformation.

2.2 Implementations

2.2.1 LCF/ML

(a) DEC10/TOPS10 in Stanford Lisp, since ported to Rutgers Lisp (Edinburgh).

(b) Multics system in MacLisp family (INRIA).

(c) VAX Unix in Franz Lisp (Cambridge), with an extended logic including disjunction and existential quantification.

(d) VAX Unix in Franz Lisp (Goteborg, Sweden), with a completely new object logic - Martin Lof's Intuitionistic Type Theory.

Cambridge(c) and INRIA(b) are maintaining common source files by using only the MacLisp subset of Franz Lisp.

2.2.2 Cardelli ML

A second version of ML only (called Cardelli ML for these notes) without the LCF object logic has been developed by L Cardelli (Bell Labs). This has named records and sums (variants), a reference operator for arbitrary types, and more flexible declarations and data abstraction constructs (sufficient to encode those of LCF ML).

Cardelli ML is implemented in Pascal under VAX VMS, since ported to VAX Unix. This produces code for an Abstract Machine (called Cardelli AM for these notes). Cardelli AM is implemented in VAX assembler.

2.2.3 HOPE

(a) DEC10/TOPS10 in POP2 and in PROLOG (Edinburgh).

(b) VAX Unix in Franz Lisp, compiling HOPE to Cardelli AM, with Lisp interpreter for Cardelli AM (from D MacQueen, Bell Labs).

(c) Part of ALICE project at Imperial College (see section 4 below).

3. PORTING TO THE PERQ

Users would increasingly find the restriction to linear text for I/O a limiting factor in the existing implementations of ML, LCF, and HOPE. The more sophisticated forms of interaction possible with the PERQ (pointing, menu selection, windows, etc) make it a natural vehicle for the further development of all these systems.

It was noted that steps are already in hand to mount Franz Lisp under PERQ Unix, through the Lisp support team of R Rae in Edinburgh AI. It was decided that the best way to mount LCF and HOPE on the PERQ is therefore to port their existing Franz Lisp implementations from VAX Unix. This should not require more than a few days work once Franz Lisp is running under PERQ Unix. For LCF it will be essential that Franz Lisp on the PERQ includes the DUMPLISP operation for saving core images.

Several research projects (Hanna, Hennessey/Mitchell, Sufrin) had a separate and more immediate need for ML on the PERQ without the LCF logic. Ideally this should be as efficient as possible, though R Milner suggested, with general agreement, that the initial requirement is for an ML workbench that is pleasant to use so that people can get into it quickly. It was felt that such an ML system would also be more suitable for dissemination of ML to a wider community.

Cardelli ML could be used as a starting point. It was easier to use than LCF ML and considerably faster in its compiled code, though currently it lacked a supporting environment. Porting to the PERQ would require

(a) an interpreter or code generator for Cardelli AM,

(b) a garbage collector, and

(c) possible changes due to differences in Pascal dialect.

It was estimated that this would involve two man-months of work.

B Sufrin suggested that the medium term requirement for a fast implementation be met by microcoding Cardelli's AM when the 16K writeable control store is available for PERQ. (This would also be useful as a target machine for HOPE - see section 4 below.) RWW commented that where there was a clearly felt need SERC funds can be generated through the STI to pay ICL to do the work. A complete specification of Cardelli's AM to commercial standards would be needed to put the proposal to ICL.

RWW would pursue the question of porting Cardelli ML to the PERQ with the two research projects that had indicated a willingness to do some of the work (Hennessey/Mitchell, Sufrin).

Extending Cardelli ML to full LCF was not considered worthwhile. It was felt better to keep Franz Lisp as a common base for LCF on VAX and on PERQ.

4. DEVELOPMENT OF HOPE

R Burstall described briefly current work with HOPE at Edinburgh. Examples from the category theory approach to program specification are being worked on - approximately 100 pages had now been programmed in HOPE. Extensions to HOPE for better modularity, and a mechanism for handling exceptions, are being considered. Current implementations were slow and would need to be improved. Edinburgh were considering either proceeding via Spice Lisp or seeking a microcoded version of Cardelli's AM. It was noted, however, that Spice Lisp would not be a viable route

for some years yet.

J Darlington outlined work at Imperial and Westfield Colleges. The ALICE project at Imperial is building a programming environment centred around HOPE, including a transformation component based on the ideas of the LCF metalanguage (tactics and simplification). ALICE includes a Compiler Target Language, CTL. Compilers written in HOPE producing ALICE CTL code have been implemented for both HOPE and PROLOG. ALICE CTL is quite different to Cardelli's AM and uses the technique of graph reduction in its implementation. In the medium term a fast interpreter for CTL would be needed, possibly microcoded. All the work at Imperial is being done on the Edinburgh DEC10. S Eisenbach (Westfield College) is producing HOPE compilers and run-time systems targetted at 16-bit micros. There is close liaison between Edinburgh and Imperial to maintain compatability of HOPE languages.

There was some discussion about the merits of propagating two languages (HOPE and ML). R Milner summarised the general view that HOPE and ML represent two points in a spectrum of possible (typed) functional languages. Differing aims had led to different compromises in the design of the two languages and both should continue to be developed. The ultimate functional language could not be envisaged yet - variety and experimentation, rather than standardisation, were needed at this stage.

5. THE IOTA SYSTEM

CPW reported on initial steps toward obtaining this system for the UK academic community.

IOTA is an ambitious project at Kyoto combining many ideas in program specification and verification (data abstraction, algebraic specifications, theory modules, "types of types" (sypes), assertions, invariants, Hoare-style proof rules, etc) into a working system. UK interest in IOTA had been awakened following a talk by one of the designers, R Nakajima, at a recent IBM Seminar in Newcastle.

In an exchange of letters Nakajima replied that IOTA is currently implemented on DEC20 and DEC10 in Uta Lisp (a dialect of Stanford Lisp), though with "some modifications of the source assembly code of the Lisp system". This implementation is expected to be complete in 3 to 6 months. Documentation will appear as a volume of Springer Lecture Notes early next year. Kyoto would like to port IOTA to a VAX under both VMS and Unix, using the MacLisp/Franz Lisp family, but must wait two years to install a VAX in their institute.

CPW telephoned Nakajima the day before the meeting. SERC and the UK community would be particularly interested in a VAX Unix version. If Kyoto showed interest, SERC funds could probably be generated to sponsor the mounting of IOTA on a VAX in the UK. A two-part proposal was suggested:

(a) a Visiting Fellowship for one of the IOTA project team to come to the UK for, say, two months to mount IOTA under VAX Unix in Franz Lisp,

(b) a follow-up visit, say two months later, by one of the designers to give an IOTA Workshop including "hands on" demonstration using the VAX implementation.

Nakajima indicated interest in exploring the proposal further. RWW undertook to pursue the question of generating funds for a Visiting Fellowship and Workshop and will write when a specific offer can be made. A UK host site with a VAX running Unix/Franz Lisp would be needed. For obvious reasons Kyoto would like this to be somewhere with a local Franz Lisp expert available.

6. TUTORIALS on ML, LCF, HOPE

The meeting welcomed the suggestion that a tutorial be held to propagate knowledge of ML, LCF, and HOPE, and discussed the form this should take.

It was felt that LCF per se was a specialised interest and should be separated from the general concepts embodied in ML and HOPE. It was also felt inadvisable to attempt to use two languages in the same tutorial.

Accordingly it is recommended that three separate tutorials be held, one each on ML and HOPE with a further tutorial on LCF. Each should be at least two days with approximately half the time spent on practical work. Summer 1983 is the suggested date, since it is reasonable to expect the PERQ implementations to be up and running by then. (If backup arrangements should prove necessary, access to the VAX implementations was the preferred alternative since these are the versions that will become widely available on the PERQ.)

Edinburgh and Imperial College offered to act as host sites.

7. RESEARCH PLANS

No time.

8. OTHER BUSINESS

A Mycroft raised the question of the future of the DEC10 implementation of the Boyer-Moore theorem prover when the DEC10 ceases service. He suggested porting it to Franz Lisp under Vax Unix. D Coleman commented that a feasibility study for this port had been done at Berkeley (by Bateman?) and suggested contacting them.

C O M P U T E R

L A B O R A T O R Y

University of Cambridge

3 November 1982

First, we would like to apologize that we cannot attend the meeting, and hope that this letter will convey our views. The paragraphs are numbered in correspondence to the Agenda.

2. Gerard Huet (of INRIA) has ported the DEC-10 implementation of LCF to Multics MacLisp, cleaned up the code, and made it portable over the MacLisp family. He and Larry Paulson have agreed to maintain a common source file. Some modules have been entirely rewritten. In particular:

> Gerard has rewritten the theory package for greater efficiency, flexibility, and robustness. He plans to allow complex hierarchies of theories, and theories that can be shared by several users. We hope that recent work by Sannella and Burstall will allow us to incorporate ideas from the specification language, CLEAR.
>
> Larry has installed a pretty-printer, and rewritten all the code for manipulating PPLAMBDA objects. He has extended PPLAMBDA to include disjunction, existential quantifiers, and predicates.
>
> Gerard and Larry have extended the ML-to-Lisp translator so that the Lisp code it generates can be given to the Lisp compiler. Compiling the code makes it run several times faster, and allows it to be re-loaded almost instantaneously, since the slow parser and type-checker are not invoked. Compiled code occupies much less storage when loaded.

This LCF is running on INRIA's Multics system (in MacLisp), and at Cambridge's VAX/Unix system (in Franz Lisp). We would like to bring it up on Edinburgh's VAX/VMS (Franz Lisp), and on the Edinburgh DEC-10 (MacLisp or Rutgers Lisp), if people there are interested and willing to help.

Further system work may have to wait, since our real goal is to use LCF, not to develop it.

The Goteborg group has extended the LCF system in different directions, including an improved top-level and a new logic, Martin-Lof's Intuitionistic Type Theory. Larry will be in Goteborg on the date of this SERC meeting, and hopes to find a common ground with their efforts. Though they are also using Franz Lisp under VAX/Unix, the two Lisp sources are incompatible.

3. Luca Cardelli's ML system should be made available on PERQs. His language is quite different from the ML in LCF. It is perhaps too baroque, but has convenient data structures and more flexible declarations. Its reference variables may provoke controversy, but are more powerful than those in ML. A further plus is the speed of its compiled code.

However, we feel it is too early to consider implementing LCF in Luca's ML. Luca's system is unsupported and provides no input/output. Although LCF is partly implemented in ML, this code would need extensive editing to run under Luca's system. The Lisp code may be impossible to re-write. Even if the porting succeeded, it would be yet another implementation of LCF to maintain.

We feel that the best way to mount LCF on to the PERQ is to first mount Lisp, then use the existing Lisp implentation of LCF. We hear that Carnegie-Mellon University has implemented a system called Spice Lisp on PERQ's. This is an implementation of Common Lisp, which is compatible with MacLisp and should accept our system with little change. Furthermore, Lisp is useful in its own right, and will make PERQ's more attractive to AI researchers.

4. We have not used HOPE ourselves, but are impressed with the language's design and strongly encourage more development. HOPE has the same general aim as ML, but is simpler and cleaner. We would like to see some of David Turner's ideas incorporated (not necessarily normal order evaluation), and a fast implementation using the techniques developed by Luca and others. Eventually, HOPE could become a language for building systems like LCF.

We would like to see research into the roles of failure trapping, input/output, assignable variables, and normal order evaluation in such languages.

We hope someone will work on debugging tools for HOPE and ML. Their polymorphism makes it hard to write debuggers, because there is no type information at run-time and only partial information at compile-time.

5. We are not familiar with this work but would be interested in learning more about it.

6. We favor a tutorial on HOPE and ML. It must take place where there are

many terminals running these languages, for no one can understand what polymorphism is really about without personally typing in function definitions. Recent extensions to LCF, discussed below, are bringing it closer to conventional logics and making it easier to understand. We will require a few months to discover the consequences of these extensions, and suggest that no LCF tutorial be held before then.

7. Larry Paulson is interested in performing structural induction in LCF. He has extended the logic with disjunction and existential quantifiers, in order to allow recursive structures to be axiomatized directly, rather than through domain equations. Such axiom systems are simpler and more natural than the previous constructions that used lifting and coalesced sums. He plans to automate the construction of theories of recursive structures, including (when applicable) proof that a domain is flat and construction of its theory of equality. In particular, this would facilitate compiler proofs, since it would automatically construct the theory of a language's abstract syntax. He has formalized part of Manna and Waldinger's proof of the Unification Algorithm, and plans to continue this.

Mike plans to attempt some non-trivial digital system correctness proofs in an extension of PPLAMBDA containing terms denoting sequential machines, together with inference rules based on the laws of SCCS. Some simple examples have already been done in a prototype implementation, and the approach seems promising.

Mike has a student interested in comparing and (perhaps) combining functional and logical programming. One possibility is to provide a mechanism for deducing consequences of PPLAMBDA theorems with a very efficient derived inference rule based on Prolog. The general idea is to approach the ideal of Program=Logic+Control by experimenting with PPLAMBDA as the logic and ML as control.

Dave Matthews at Cambridge is developing a high performance system programming language, called Poly, which is derived from Pascal and Russell. It is beginning to look a lot like ML, and it is hoped to eventually unify the two approaches. Poly might be a source of ideas for extending the ML type discipline, it might also be a good programming language for future LCF implementations. At present it is not clear whether continued SERC support for Dave will be forthcoming.

Larry Paulson

Mike Gordon

Volume I, Number 0 November, 1982

Polymorphism

The ML/LCF/Hope Newsletter

Contents

Letter from the Editors

This is the premier issue of *Polymorphism*, a newsletter for the LCF/ML/Hope community. There are now several geographically distributed groups working on LCF, ML or Hope, and there is a need for a rapid and informal communications medium permitting coordination and sharing of research results, without the long delays involved in formal publication. We think that a newsletter can help satisfy this need.

The purpose of this newsletter is to disseminate news, information, and ideas of interest to this community. To that end we invite contributions, including

- theoretical results and applications,
- announcements of meetings and conferences and calls for papers,
- announcements of new or revised implementations,
- programming tools and library packages,
- software documentation,
- queries and suggestions,
- bug reports and fixes,
- philosophical and historical commentaries,
- technical reports which are not widely available, including old reports,
- reviews of books and articles, and
- pointers to relevant literature, e.g. abstracts and bibliographies.

The emphasis is on timeliness and informality, so contributions to the newsletter need not be in a final, polished form. Presentation of speculative ideas and interim working papers is encouraged.

This introductory issue includes a list of our initial "subscribers" and their addresses, and a list of all known sites running a version of Luca Cardelli's Pascal implementation of ML (VAX ML). Please send us any corrections or additions to these lists. It would be very useful if each group could send a short summary of its activities and an indication of which ML, LCF, or Hope systems it is using or developing.

Comments on the organization, contents, and purpose of the newletter are welcome (including opinions on the provisional title). We hope to produce an issue of the newsletter roughly every two or three months, depending on the flow of contributions. A second issue with significant technical content is in preparation and will follow shortly. We expect the first few issues to contain installments of a revised manual for VAX ML, a definition of Luca's functional abstract machine, a history of LCF by Robin Milner, a yacc grammar for ML with commentary by Ravi Sethi, and a Hope manual.

Articles will not be refereed and should be in a form ready for reproduction. There will be no charge initially, and only one copy of the newsletter will be sent to each geographical location. There are practical limits on the size of each issue, so submissions should be of short to medium length. Longer documents, such as PhD theses, should be distributed directly by their authors, with an abstract being sent to the newsletter.

Luca Cardelli
David MacQueen

Bell Laboratories
Murray Hill, NJ 07974
USA

Mailing List

Gerard Berry
Ecole des Mines
Sophia-Antipolis
06560 Valbonne
France
Tel 93 336780

Peter Buneman
The Moore School of
Electrical Engineering D2
Dept. of Computer and
Information Science
Philadelphia, PA 19104
USA
Tel 215 243 7703

Luca Cardelli
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
USA
Tel 201 582 5707

Toni Cohen
Dept. of Computer and
Information Science
University of Delaware
Newark, DE 19711
USA
Tel 302 738 2712

Bob Constable
Dept. of Computer Science
Cornell University
405 Upson Hall
Ithaca, NY 14853
USA
Tel 607 256 4052

Ian Cottam
Dept. of Computer Science
Manchester University
Manchester
England

Werner Damm
Lehrstuhl fur Informatik II
Buchel 29-31
5100 Aachen
W.Germany
Tel 0241 804564

Mike Gordon
Cambridge Univ. Computing Laboratory
Corn Exchange Street
Cambridge CB2 3QG
England
Tel 0223 352435 ex 217

Soren Holmstrom and Thomas Johnsson
Chalmers University of Technology
and University of Goteborg
Dept. of Computer Sciences
S-412 96 Goteborg
Sweden

Gilles Khan
INRIA
P.O. Box 105
Domain de Voluceau
Rocquencourt
78150 Le Chesnay
France
Tel 3-9549020

Colleen Kitchen
Trinity College Dublin
201 Pearse Street
Dublin 2
Ireland

Jacek Leszczylowski
Institute of Computer Science
Polish Academy of Sciences
P.O. Box 22
00-901 Warszawa PKiN
Poland

Robin Milner
Dept. of Computer Science
University of Edinburgh
J.C.M.B. The King's Buildings
Edinburgh EH9 3JZ
Scotland
Tel 031 667 1081

Areski Nait-Abdallah
Computer Science Dept.
Waterloo, Ontario N21L3G1
Canada

Renzo Orsini
Istituto di Scienze dell'Informazione
Corso Italia 40
56100 Pisa
Italy
Tel 050 40862

Lasse H. Ostergaard
Dept. of Computer Science
Buildings 344 & 343
Technical University of Denmark
DK-2800 Lyngby
Denmark

Simonetta Ronchi
Istituto di Scienze dell'Informazione
Corso M.D'Azeglio 42
10125 Torino
Italy
Tel 011 655307

Dana Scott
Dept. of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213
USA
Tel 412 578 2566

Arnold G. Smith
University of Sussex
Dept of Experimental Psychology
Brighton BN1 9QY
England

Richard Snodgarss
Departmant of Computer Science
North Carolina University at Chapel Hill
Chapel Hill, North Carolina 27514
USA

Bernard Sufrin
Programming Research Group
45 Banbury Road
Oxford OX2 6PE
England
Tel 0865 58086

Rodney Topor
Dept. of Computer Science
University of Melbourne
Parkville, Vic 3052
Australia

Chris Wadsworth
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire OX11 0QX
England

Known VAX-ML System Locations

Luca Cardelli	(Unix Version 24-8-82)	Bell Labs
Bernard Sufrin	(VMS Version 12-6-81)	PRG Oxford
Simonetta Ronchi	(VMS Version 12-6-81)	Torino
Werner Damm	(VMS Version 12-6-81)	Aachen
Renzo Orsini	(VMS Version 13-10-81) (Unix Version 24-8-82)	Pisa
Mike Gordon	(VMS Version 13-10-81) (Unix Version 15-4-82)	Cambridge
Arnold G. Smith	(VMS Version 13-10-81)	Brighton
Bob Constable	(VMS Version 12-6-81)	Cornell
Jacek Leszczylowski	(VMS Version 12-6-81)	Warsaw
Lasse H. Ostergaard	(VMS Version 12-6-81)	Gotenborg
Lars Ericson	(VMS Version 13-10-81) (Unix Version 13-8-82)	CMU
Gerard Huet	(VMS Version 13-10-81) (Unix Version 11-5-82)	INRIA Paris
Gerard Berry	(VMS Version 13-10-81)	INRIA Sophia-Antipolis
Colleen Kitchen	(VMS Version 13-10-81) (Unix Version 24-8-82)	Dublin
Rodney Topor	(VMS Version 13-10-81)	Melbourn
Robin Milner	(VMS Version 13-10-81)	Edinburgh
Peter Buneman	(VMS Version 13-10-81)	PENN
Ian Cottam	(Unix Version CMU) (converting to Apollo)	Manchester
Richard Snodgarss	(Unix Version CMU)	NCU at Chapel Hill
Chris Wadsworth	(to convert to PERQ)	Rutherford Lab
Areski Nait-Abdallah	(Unix Version 13-8-82)	Waterloo Ontario
Toni Cohen	(Unix Version 13-8-82)	Delaware Univ. Newark
Hans Boehm	(Unix Version CMU)	Washington Univ. Seattle

Latest VMS Version: 13-10-81
  Version 13-10-81 differs from Version 12-6-81 because:
    - it has the "with" declaration construct
    - it accepts token quotations of arbitrary length
    - it has a working garbage collector

CMU Version
  Unix version of VMS 13-10-81.
  Some bugs were introduced in the translation.

Unix Version 13-8-82
  Differs from the CMU version because:
    - known CMU bugs have been fixed
    - other older bugs have been fixed
    - if-thenloop-elseloop construct introduced
    - arrays with constant access time predefined
    - some operators have been renamed:
        `_` --> `::`
        `::` --> `@`
        `@` --> `!`

Unix Version 24-8-82
  Differs from the 13-8-82 version because:
    - Interrupts, arithmetic exceptions and crashes are treated
      as ML failures.
    - An experimental set of file input-output primitive has been
      introduced.

Latest Unix Version: 11-5-82
  Differs from the 24-8-82 version because:
    - New typechecker for ref types.