

January 13, 1987

Annual Fellow Report for 1986

John Backus

This year my report will cover the progress of the Functional Programming project. It will be a brief report since virtually all of our work in 1986 comprised the design and description of the function level language we now call "FL" (formerly we referred to it as "FP*"). This work is described in the Research Report FL Language Manual (Preliminary Version) RJ 5339, a copy of which accompanies this report.

1. Progress report. The functional programming project

1.1 Personnel. John H Williams, Edward L. Wimmers, John Backus (Mgr).

1.2 The FL language. After many years of strenuous effort we have completed the design of the functional language FL. It is an experimental language with which we plan to test the hypothesis that the clearest and most powerful way of building programs is to build them at the function level, that is, by combining existing programs using higher level functions called combining forms. This emphasis on programming at the function level results in programs which have a rich mathematical structure that may be transformed and optimized according to an underlying algebra of programs.

The principal goals for the FL language are the following:

- 1) Scope.** We intend FL to be a practical language for writing most of the programs that a systems or applications programmer writes, eg, operating systems, graphics, text processing, interactive, number-crunching, and combinations of these kinds of programs. We also intend it to be very useful to relatively inexperienced users; we expect they will construct their own applications programs by combining and modifying relatively complex programs written by experts. Finally, we hope to virtually eliminate the need for a separate operating system with its own special language for system control; we hope to do this by making the necessary operations part of FL, so that a running FL system (with tasks, which are not yet included in the language) would incorporate the functions of an operating system.
- 2) Accessibility.** FL is at present a two level hierarchy of languages. At the lowest level is the principal and simplest one, the one that ties everything together: this is the function level FP language with a number of enhancements. This is the basic language for describing transformations of data. At the next level is the language of FL programs; these are virtually identical to FP functions except that they can refer to and alter an object called the state. The state contains the permanent file system and the input and output streams associated with the I/O devices of the system.

We expect the concepts of an FP function and of an FL program to be very simple and accessible to the ordinary user (although these concepts differ radically from current concepts of programs).

IBM 06

In the future we expect that a third, and highest level of the language will be tasks, programs that can run subtasks concurrently; these will be used primarily by experts, although we hope to make the concepts required a good deal simpler to understand than they are at present. Tasks are not now part of the FL language.

- 3) **Programming efficiency.** One of our principal goals is to make it possible to write most programs in much less time than is presently required using current languages (we are striving for an order of magnitude improvement). Our hopes for achieving such large improvements are based on a number of factors:

- a) The ability to compose off-the-shelf programs easily.
- b) Many combining forms for putting programs together and the non-recursive programming style they make possible.
- c) A powerful, hierarchical, non-intrusive system of data types that relieves the user of concerns about data representation; facilities for introducing user defined abstract data types; facilities for making assertions about types and other properties of programs in the form of functional equations; facilities for automatic type checking with or without help from the user.
- d) Handling of many concerns about efficiency by the system rather than by the user.
- e) Availability of many operations on large data structures.
- f) Representation of housekeeping operations as reusable data-rearranging operations rather than as non-reusable control structures that control a variable scattered throughout some program text.
- g) Powerful tools to help verify certain properties of programs or fragments, tools that are based on the existence of a strong equational mathematics for function level programs.
- h) Many linguistic features (static scoping, infix notation, precedence rules) designed to make programs clearer and more readable than those in many other languages.
- i) Ability to define new combining forms and higher-order functions.

- 4) **Clear and precise semantics.** We hope to show that the two precise, equivalent, strict semantics for FL are useful to the user of the language as well as to the implementer. The operational semantics is given by rewrite rules. The equivalent denotational semantics is unusually accessible and understandable by the user.
- 5) **Operating efficiency.** Our goal is to write a compiler for FL that will produce programs that are as efficient as those produced by the best Fortran compilers. Our hopes for being able to do this are again founded on the availability of a powerful algebra of programs that allows us to mechanically transform a clear, concise, easily-written, but very inef-

ficient program into a very different, complex, but very efficient program that is exactly equivalent to the original.

For more information about FL, see the attached manual for the language.

1.3 Other work. In addition to the work on FL Williams and Wimmers worked with Joe Halpern on improving their results on the completeness of a non-strict version of FP called FP84. They characterized various safe rewrite strategies with respect to different notions of completeness. These results were published as RJ5099.

1.4 Other activities.

Publications:

Halpern, J.Y., Williams, J.H., and Wimmers, E.L., "Completeness of Rewrite Rules and Rewrite Strategies for FP", IBM Research Report RJ 5099, June 6, 1986.

Halpern, J.Y., Williams, J.H., and Wimmers, E.L., "Good Rewrite Strategies for FP", Symposium on Logic in Computer Science, Cambridge, Massachusetts (June, 1986) 149-162.

Backus, J., Williams, J.H., and Wimmers, E.L., "FL Language Manual (Preliminary Version)", IBM Research Report RJ 5339, October, 1986.

University & professional interactions:

Williams was Visiting Industrial Professor, Stanford University, 1985-86, teaching 309B (Functional Programming), Winter Quarter. Williams organized, developed, and taught this graduate level course. (He was ranked by his students as one of the three best instructors in the Computer Science Department.)

Williams represented ARC at the Stanford University CS Department Computer Forum, February 5, 1986.

Williams represented ARC at the Cornell University CS Department Industrial Affiliates Meeting, October 6, 1986.

Wimmers presented talk "Good Rewrite Strategies for FP" at Logic in Computer Science Symposium on June 17, 1986.

Wimmers presented talk "Good Rewrite Strategies for FP" at Almaden Research Center for visiting French professors on September 9, 1986.

Backus participated in a panel discussion at University of California, Berkeley, "On Being a Researcher". February 21, 1986.

Professional activities:

Backus, Williams continued as members of IFIP Working Group 2.2.

Williams served as Chairman of program committee, ACM Conference on Lisp and Functional Programming, Boston, MA, August, 1986.

Williams is the Organizer and General Chairman, of the ACM and IFIP sponsored Conference on Functional Programming Languages and Computer Architecture, to be held in Portland , Oregon, in September, 1987.

Williams organized and proposed to IFIP Technical Committee 2 the formation of a new international working group on Functional Programming. (This was approved on 4/4/86 and will become Working Group 2.8 upon formal approval by the IFIP General Assembly.)

Miscellaneous activities:

Williams was Ph.D. recruiter at Cornell University.

Williams was Ph.D. recruiter at University of Wisconsin.

Williams was a member of the ITL on Programming Languages.