Notes on new Prologs, WFC 7/5/84

- Users want *mixed language* programming, with *shared data structures.* Only POPLOG and Salford Lisp/Prolog offer this at the moment. Only POPLOG has been tested on substantial applications, however these tests have revealed limitations which further work will overcome.
- Which languages to users want? Prolog and Lisp are the first choice, with POP needed by those who have applications already in POP. However, any members of this "family" could be used. The family characteristics are: interactive, with run-time type checking (other typechecking methods optional) and dynamic allocation of data structures, interpreted or incrementally compiled (other compilation methods optional). Other members of this family are Smalltalk and Snobol.
- Users want the Lisp component of a mixed language system to conform to some recognised standard. There are several standards, with numerous dialects within the standards. POPLOG does not conform to any. Function-name compatibility is not enough: the semantics of atoms, for example, must be Lisplike, and not POPlike.
- Performance is another issue, divided into speed and memory requirements. The compromises forced by mixed language means that speed will be reduced from what is possible if you try hard (for example, Quintus Prolog).
    - speed. As fast as possible: 10KLIPS is probably what one should expect on a 68000. The important point is that speed must not degrade disproportionally as the program size increases. This means that implementation techniques that "scale up" appropriately must be used. POPLOG has a well recognised problem here, but it can be remedied. Prolog-X was designed from the start to scale up well, but it does not attempt mixed language programming.
    - memory. Contrary to popular thought, it is still important to have compact representations of clauses and terms. It is mostly paging which kills POPLOG's performance, caused by bulky representations of terms and clauses. Prolog-X was designed to have very compact clauses: code size ranges from 2 to about 100 bytes, and is usually around 30.
- Memory management. A uniform heap is probably the most convenient for implementing mixed languages. This is what POPLOG, the APM Proposal, and Salford do. Conventional implementation techniques for high-performance Prologs do not use a uniform heap, and instead use several stacks, each of which is managed in a particular way. This technique is very inconvenient for mixed language working, and this is why Prolog-X and Quintus Prolog are not good substrates for mixed language systems. Management of the uniform heap will need to optimise recognised special cases, such as long-term storage (Prolog clauses and Lisp functions) and short-term storage (Prolog global terms and Lisp s-expressions) and activation storage (Lisp and Prolog local stacks).

What is a good substrate? Cambridge Lisp is a high-performance implementation of Standard Lisp, and is based on the PSL implementation method. A Lisp compiler emits code for an abstract machine (CMACRO), which is then translated to machine code for the processor. Implementations run on the 68000, 16032, GEC 63, and IBM 3081. It regularly

runs large programs such as REDUCE. An interactive programming environment based on those favoured by AI programmers is available, and is used by AI programmers at Cambridge. Implementation details of the systems are published in various articles in *Software Practice and Experience.*

I suggest the Cambridge implementation of Standard Lisp to be an ideal substrate from which to build a mixed language system. At the moment I (with ROK and ACN) am investigating the feasibility of writing a Prolog compiler to be integrated with the Cambridge Lisp system. Only relatively minor changes in the existing substrate are required to produce a high-performance mixed system. For example, an unexpected bonus is that the current garbage collector already deals with locatives pointing to the middle of a cell. This saves implementation effort, and permits the most efficient implementation of Prolog variable bindings. We also expect to take advantage of experience gained by others at Uppsala and SRI on implementing Prolog on the LM and Symbolics Lisp machines. We expect the performance to be the best possible in the situation.

NB: the mentions of POPLOG are for comparison purposes only, and are not intended as criticism of Poplog.