Forth-like Languages for Artificial Intelligence Working Group Report

Chairman, James Bender
Secretary, Richard Haskell

1. Introduction

Relevant questions involving the design and use of Forth-like languages for AI are as follows:

* What advantages does Forth provide that would make it suitable for use in AI?

* Can Forth-like languages be used for AI without garbage collection? If so, then Forth could provide substantial advantages for real-time expert systems.

* What features should be included in a Forth-like language oriented towards AI?

* Is the best solution to implement existing AI languages in Forth or to add features to Forth to support symbolic processing?

* What Forth-based artificial intelligence tools are presently available?

2. Advantages of Forth or Forth-like languages for AI

Like Lisp and Prolog, Forth is an extensible language, and provides an interactive development environment. Many Lisp and Lisp machine advocates are really environmentalists: they like the interactive development environment rather than Lisp, itself. With some enhancements to give Forth a more natural capability for manipulating symbols and a good environment, Forth might be just as suitable for symbolic programming as is Lisp.

Another advantage of Forth is performance: Lisp, even on Lisp machines, performs poorly, and devours memory. On the Lisp machine, with the baroque, MIT system software, a gigaword of virtual memory is not enough. Forth is efficient and fast, in contrast with Lisp.

3. AI Without Garbage Collection

Garbage collection is a big hindrance to the use of Lisp and Lisp machines for real-time applications. Real-time is used here as having to respond to real-world events in less than a millisecond. On Lisp machines, a context switch can take 2 seconds or more. Arguably, a system which queues real-world inputs can be said to be real-time, but then answers based on these inputs may take seconds or minutes. It is not unusual for answers from Lisp machine-based systems to take 20 minutes or more (a real example from a DARPA-funded contract).

The solution to real-time AI seems to be to use dynamic memory management, rather than garbage collection. Garbage collection either is incremental and uniformly degrades performance (even in parallel, it is "stealing cycles"), or is done when the store of free list nodes is exhausted. In the latter case, in a large system, garbage collection could take 20-25 minutes. Henry Harris stated that his Prolog uses dynamic memory management rather than garbage collection. Bill Dress's RealOps appears to use an incremental form of garbage collection, the exact nature of which has not yet been revealed. Jim Bender's Fifth-Generation Forth also is guilty of employing garbage collection--one implementation uses "mark and sweep" while in another implementation, the underlying system performs incremental garbage collection. As Charles Moore reflected at the 1985 conference, garbage collection is something to be avoided. That is true for systems for which "speed" is more important than just providing reasonable answers.

4. AI Features for Forth

The principal features required to support AI are support for symbolic manipulation and object-oriented programming. Neon is one example of a very Forth-like language and implementation which supports an object-oriented representation. Charles Duff's language, Actor, is a Smalltalk-like language which is implemented as a threaded interpreter.

One might ask exactly what is Forth: a language which looks like Forth but which has a different implementation or a language which uses infix notation, but which has a Forth-like implementation. A purist might say that a language is Forth only if a user may still access a Forth-83-compatible vocabulary, interactively. Extensions to support AI can cause both types of variation to occur.

Fifth-Generation Forth (FGF) looks very Forth-like (especially postfix notation and Forth-named, stack manipulation words), but the current implementation effort is not based on a threaded interpreter like Forth. Instead, FGF is a compiler-interpreter system which has some things in common with token-threaded code, but is based on symbolic programming. With a frame-based knowledge representation language, object-oriented programming support ("Simple" system), and a production rule system, FGF is more like an expert system shell than a language. Any shell still needs a programming language interface, and it is a Forth-like language, in this case.

5. Forth or AI Languages in Forth?

Jack Park's approach to expert systems using Forth (Expert-1, etc.) is a model for doing AI directly in Forth. Henry Harris's Prolog, Louis Odette's Prolog, and Bill Dress's OPS-5 are examples of implementing AI languages in Forth. The best that can be said of AI languages in Forth is that they perform about as well as the same language written in C, while maintaining the interactive nature of Forth and Forth's closeness to the hardware. Another possibility for future exploration is

to generate threaded code as an object code for a compiler. Actor, being developed by Charles Duff, is the sole example of this approach which has been described at this year's conference.

6. Forth-based Expert System shells and AI languages

The following is a list of known Forth-based AI tools which are actually available either as commercial or public domain products, or have appeared in print:

* Jack Park's Expert-2
* Forlog (in FigAI Notes)
* Louis Odette's Prolog on Macintosh
* George Levy's Small Expert System (Silicon Valley FIG)
* Martin Tracy's List-handler in Forth Model Library
* Book: Designing and Programming a Personal Expert System, by Townsend and Feucht, published by TAB books
* FORPS, described by Charles Matheus at this conference.

There are a number of others in various stages of implementation, including Actor and Fifth-Generation Forth.

7. Conclusion

There are clearly some advantages to be gained by using Forth for AI: extensibility, interactive environment, speed (especially for real-time), and the possibility of no garbage collection (dynamic memory management under explicit program control). Forth needs extension to support symbolic manipulation. Whether this entails building new standard vocabularies, or implementing Prolog, Lisp, or OPS5 in Forth remains an open question, which will probably be hotly debated at next year's conference.