Some Thoughts on NETL, 15 Years Later

Scott E. Fahlman
School of Computer Science
Carnegie Mellon University

1. Overview

NETL [Fahlman 79] is a knowledge-representation system combining two key ideas: first, the use of a semantic network representation, with built-in capabilities for inheritance, inference, and simple search; second, the use of massively parallel hardware to perform most of these built-in operations in near-constant time, without the need for complex, hand-crafted indexes or application-specific search procedures.

NETL was developed as my Ph.D. thesis project in the period from 1974 to 1977, and was published in book form in 1979. In the years since then, the NETL ideas have been influential in a number of ways, though NETL has never really become a part of the mainstream in AI. In this short paper I will present my own personal view of the evolution of NETL, its strengths and weaknesses, and the lessons it might still hold for AI.

2. The Development of NETL

Sometime around 1971, Marvin Minsky and Seymour Papert posed the following question as a mind-expanding exercise for the new crop of grad students in the MIT AI lab: Is AI lacking ideas or just technology? If money were no object and you could have all the hardware you want (it must be clearly specified hardware -- no oracles or magic boxes), what would you ask for? And how much of AI could you solve with this hypothetical pile of hardware?

I was one of those students. My answer was to imagine a large set of hardware pattern-matching boxes. You would have one for each external object that you might want to recognize from sensory inputs. Alternatively, you could have one for every object in a data base that you might want to locate in response to an imprecise query. It seemed to me that what we would now call "fuzzy match" is an important component of intelligence, and that what we would now call "data parallelism" is a good way to implement fuzzy match. This idea generated some lively discussion, and then we all went on to other topics.

In 1973, while trying to find a good topic for my Ph.D. thesis, I began to think about a fundamental limitation of AI systems (then and now): while we can stuff lots of raw real-world knowledge into an AI system, it is very hard to make all that knowledge *effective* in whatever problem-solving process the system is engaged in. A fact just sits there, waiting for the processor to look at it.

We can build very general systems for inference and search in our knowledge base, but these scale up poorly as the size of the knowledge base increases. The more the system knows, the slower it becomes. On the other hand, we can build clever application-specific data structures and procedures to access the knowledge more efficiently. This is an important technique in mainstream AI, but the resulting systems are narrow, brittle, and very difficult to build. It seemed to me that there must be a way to build a very general inference system that would operate quickly and scale up well to huge sizes. We humans seem to have such a facility in our brain, and our broad, flexible intelligence depends on this.

NETL was born when I began to apply my old ideas for massive parallelism to this difficult problem, mixing in some ideas about semantic networks and marker passing from M. Ross Quillian [Quillian 68]. NETL is a semantic network memory, with nodes representing "things" and classes and links representing relationships and statements. In NETL, each node and link is represented by a simple processing element, capable of storing a few distinct single-bit *markers* and passing them to their neighbors in the network. The physical network is wired up to match the topology of the semantic network. The marker bits flow through the net in parallel, under overall control of a central processing unit.

The parallel hardware scheme used in NETL fell into place very quickly, but it took me three years to understand its properties and to develop a knowledge-representation language to go with it.

A NETL-style parallel memory offers several advantages for AI:

- Certain kinds of inference and search, such as property inheritance and set intersection, occur in constant time.
- Because simple inference is fast, the knowledge can be put into the system in raw form, without any application-specific programming. Each item can be used in a variety of ways, some of which were not anticipated when the item was added.
- NETL can make use of implicit knowledge almost as easily as explicit knowledge. If the knowledge base knows that all elephants are gray and (by some chain of reasoning) that Clyde is an elephant, there is no need to store the assertion that Clyde is gray.
- In recognition, all the possible matches can be evaluated at once.
- Because one marker can gate the flow of another, it is possible to implement exceptions and context-dependent statements in NETL.
- The abilities and weaknesses of NETL are in many ways (but not all) more like those of a human than like those of an AI program on a serial computer.

Because of these features, NETL generated a good deal of interest when the book first appeared. At that time, there was little interest in parallel processing among AI researchers. The early flowering of "neural network" approaches in the 1960's had stalled (temporarily, as it turned out), and parallel machines were still costly and hard to program. NETL was *one* of the elements that led AI people to take a new look at parallel approaches, or at least not to reject them out of hand. This helped to pave the way for the resurgence of neural nets in the 1980's.

3. What ever became of NETL?

NETL had some serious disadvantages as well. The most obvious was that, in the late 1970's, NETL was just a theoretical exercise using imaginary parallel hardware. The system was simple enough that one could predict pretty reliably how a large NETL system of, say, a million processing elements would behave, but it would have been extremely expensive to build such a system. Even in those days, it was not unthinkable to create 10^6 simple processors, each capable of storing 16 marker bits, but the reconfigurable network of inter-unit connections, all operating in parallel, was another matter.

After receiving my doctorate and moving to Carnegie Mellon, I began exploring a number of switching network designs, and I finally developed a plausible design for a million-element NETL machine [Fahlman 80]. This machine was never built for two reasons: first, by this time, my own interest was turning away from NETL and toward more fuzzy models of knowledge representation and inference (see below); second, Danny Hillis and his colleagues at MIT, inspired in part by the data-parallel style of NETL, had begun a large project to produce what they called a "connection machine" [Hillis 85], the precursor to the line of machines developed by Thinking Machines Corporation (and their imitators). The Connection Machine architecture incorporated many features that went beyond the specific needs of NETL, but it was clear that if this machine succeeded it would be a good engine for studying NETL-style parallel AI. And so it is, though the existing Connection Machines are primarily used for tasks other than "thinking".

A second limitation of NETL was in the nature of marker-passing parallelism itself (see [Fahlman 82]). It took me a couple of years during the thesis research to really understand this issue. Basically, a NETL machine that passes single-bit markers can perform many useful functions in parallel -- transitive closure, set intersection, and so on -- but it cannot do arbitrary theorem proving or pattern-matching in parallel. A NETL machine can mark all dogs with marker M1, all dog-owners with marker M2, and all dogs hated by *some* dog-owner with marker M3. But NETL cannot, in parallel, mark every dog who is hated by *his own* owner. This would require the "I own you" and "I hate you" markers to carry some sort of signature field, plus machinery in each DOG node to see if these signatures match. Such matching requires much more expensive hardware than the simple marker-passing of NETL, and it introduces the possibility of contention and delay if too many of these complex messages are sent to the same recipient. (Colliding marker bits, on the other hand, can simply be OR'ed together into a single message.)

A number of researchers, including Eugene Charniak and Jim Hendler, have explored what can be done with these more complex massively parallel message-passing systems, but this approach has never held much interest for me. I personally believe that the additional operations provided by message-passing parallelism are not worth the added cost and that these operations are not really needed for human-like intelligence.

A third limitation of NETL is that the knowledge representation I developed for it was informal and not rooted in any existing logic. This was a deliberate choice: I have always been more interested in what a piece of knowledge *does* than what it *means*. In NETL I was more concerned with links, marker flows, and queries than with how to express the equivalent knowledge in some formal system of non-monotonic logic. However, this informal approach did permit some paradoxical situations to arise when exceptions (and thus non-monotonic inference) were present in the network [Fahlman 81]. My student, Dave Touretzky, explored the question of how to formalize non-monotonic inheritance in his Ph.D. thesis [Touretzky 86], and this line of inquiry lives on in subsequent work by Touretzky, Thomason, Horty, and others.

A final limitation of NETL, in my view, is that it is an implementation of standard, symbolic, non-fuzzy AI in parallel hardware. In NETL, either Clyde is an elephant or he is not; NETL has no good way of dealing with degrees of uncertainty, or with elephants that are very poor specimens of their class. While this "crisp", symbolic orientation is an attraction for some researchers, my own interest has turned increasingly to the question of how we represent and reason about the fuzzy, imprecise, uncertain knowledge that plays so large a role in the real world. (I share Lotfi Zadeh's belief that these issues are very important, though I have no particular attachment to his particular brand of fuzzy logic.)

While the field of knowledge representation became increasingly formal and precise in the 1980's, I began to experiment with *Thistle*, a fuzzy version of NETL that passed around continuous values rather than single-bit markers [Fahlman 83]. I quickly realized that building Thistle networks by hand was much more difficult than building crisp, clean NETL networks. It became clear to me that automatic learning from examples was going to be extremely important in these less-symbolic systems, but I didn't know how such learning could be done. The emergence of improved learning algorithms for artificial neural networks finally provided an answer, and I have been working in that area ever since. I think that NETL, in its original form, still has some good ideas worth pursuing, but I have put this on the back burner for now.

4. NETL Today and Tomorrow

Though it has been influential in a number of ways, as described above, NETL has never really become a part of the AI mainstream. There are several reasons for this: First, NETL was perhaps a decade ahead of its time in terms of hardware. Only recently, as massively parallel machines have become more widely available and affordable, have mainstream AI researchers been able to experiment with such systems. Second, the whole field of knowledge representation has gone through a period in which everything must be formalized, even if this means throwing out useful functionality (e.g. the ability to create exceptions to general rules). Third, I and some of the other researchers who might have built large systems on top of NETL have turned instead to the exciting new field of neural networks.

Despite all this, it seems puzzling to me that people who still want to work with symbolic AI knowledge-representation systems have not paid more attention to the ideas in NETL. Hundreds of people work in knowledge representation, developing formalisms and inference rules, but very few of them have faced the question of how to scale their systems up to the size required for human-like common sense. If they think hard about this issue of scaling, I think that they will inevitably be led to something like NETL or some other massively parallel scheme.

For whatever reason, scaling issues seem not to be on the agenda. As Roger Schank once said to me, "NETL is not really AI. It's just a speedup hack -- an implementation technology. The real AI question is what knowledge should go into the system and how that knowledge can be used to solve problems." I disagree with this viewpoint, but perhaps it explains the limited role that NETL has played in the field.

Lenat's CYC project seems to be based on a view similar to Schank's: put in the right knowledge and don't worry too much about how long it will take you to find a piece of knowledge when you need it. In my view, the scaling issues are fundamental and should be dealt with *before* we start stuffing in the knowledge. Perhaps the pendulum will swing back in the near future. Perhaps researchers will look again at what we can do if given "all the hardware we want", now that hardware is so cheap.

Acknowledgment

Preparation of this paper was supported by the National Science Foundation, Grant IRI-9214873. The views expressed here are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

Fahlman, S. E. (1979) *NETL: A System for Representing and Using Real-World Knowledge*, MIT Press, Cambridge MA.

Fahlman, S. E. (1980) "Design Sketch for a Million-Element NETL Machine", Proceedings of AAAI-80 Conference, Morgan-Kaufmann, Los Altos CA.

Fahlman, S. E., D. S. Touretzky, and W. van Roggen (1981) "Cancellation in a Parallel Semantic Network" Proceedings of IJCAI-81, Vancouver, British Columbia.

Fahlman, S. E. (1982) "Three Flavors of Parallelism", Proceedings of the Fourth National Conference, Canadian Society for Studies of Intelligence, Saskatoon, Saskatchewan.

Fahlman, S. E., G. E. Hinton, and T. J. Sejnowski (1983) "Massively Parallel Architectures for AI: NETL, Thistle, and Boltzmann Machines" in Proceedings of the AAAI-83 Conference, Morgan-Kaufmann, Los Altos CA.

Hillis, W. D. (1985) *The Connection Machine*, MIT Press.

Quillian, M. R. (1968), "Semantic Memory" in *Semantic Information Processing*, M. L. Minsky (ed.), MIT Press.

Touretzky, D. S. (1986), *The Mathematics of Inheritance Systems*, Pitman & Morgan Kaufmann publishers.