FINAL REPORT

Project: Experts Can Finally Be Lazy: Lazy Approximation in Expert Systems
Student Researchers: Leha Blaney and Gina Castano
Advisor: Dr. Lynn Stauffer
Institution: Sonoma State University

The application of lazy evaluation to a fuzzy expert system proves to be one of considerable complexity, with some significant problem areas. In the course of our research, we isolated many of these problems. We can now offer ourselves and other researchers many directions for further research, but much is yet to be done before the real work of implementation, evaluation, and comparison with other models may begin in earnest. Our paper discusses these problem domains, the work we were able to do that was essential and preliminary to the implementation of a truly lazy system, and how we plan to continue our research from this point forward. This paper also outlines our experience with the research process, itself, and what we have learned from that experience.

Our research project began with the search for a suitable topic. We were interested in both expert systems and functional programming and wanted to see if there was a useful way to tie the two together. We discovered the Watt model of separation of control from calculation using lazy evaluation, and we found this to be an interesting concept to apply in the expert system arena. Gathering some preliminary papers and books on the two subjects, we were able to come up with several ideas for the application of the Watt model in the context of a fuzzy expert system. We made a thorough search of available information to determine that no one had already done the work we were planning. We then compiled our proposal based on this information, with the details remaining to be worked out in the course of the research. The proposal outlined several phases that were intrinsic to our plan of attack. Over the course of the year, we executed each of these phases to the best of our ability.

First, we set up an online conference for meetings, links, and discussions, and a WWW page to give the public access to our proposal and findings.

We then began a more thorough effort to gather papers and other documents on the subjects of lazy evaluation, expert systems, fuzzy logic, functional languages, logic languages, expert system shells, and various combinations of these.

Once we had gathered these materials we began the reading portion of study. The first topic to read about was functional programming. We looked at Scheme, Lazy ML, Haskell, and Miranda. Of these languages, Haskell seemed the most applicable to our goals. However, There was no fuzzy Haskell that we could find, and the task of creating a fuzzy Haskell expert system from scratch was beyond the scope of our time constraints. We do not consider this to be a bad concept, though, and hope that other researchers (or ourselves) will have the opportunity to investigate it. We are also interested at this point in creating a fuzzy expert system shell in Haskell. An expert system shell is a very high level language, written in a medium to high level language, specifically for the purposes of making the writing of expert systems as simple and streamlined as possible. A shell written in Haskell would make the use of lazy evaluation in the expert system natural and straightforward.

Our next topic of investigation was the mechanics of lazy evaluation, itself. We investigated the use of list comprehensions and lazy lists in the various functional languages we'd studied, and implemented various examples of the Watt model in these languages. We had regular brainstorming sessions wherein we expanded on our ideas for application of these tools in the expert system.

Then we read about fuzzy systems. None of us had had any prior experience working with

fuzzy logic, so this portion of our research was both intensive and fascinating. We explored various areas in which fuzzy logic has successfully been applied, with the main focus, of course, on expert systems. Here, fuzzy logic is used to get an approximate answer when no exact answer is possible, and to act as a rule builder for expert systems that actually grow in expert knowledge over the course of use.

We learned the theory and organization of traditional expert systems. Much of our course work in databases, data structures, and software engineering had prepared us for more sophisticated models, such as the expert system, so we were ripe for this part of our study and the concepts were quick and easy to grasp. Expert systems are also generally written in logic, or relational programming languages, so part of our work here was to survey a few of these languages, learning more about how they work and looking for any that might already be capable of lazy evaluation, or of using fuzzy logic. We found many were capable of fuzziness, but only one had been altered to include lazy evaluation at all, and this only a partially lazy amendment to the existing language (Prolog). At this point, we determined that not only were we going to have to use cross platform tools to make our system lazy, but if we were going to be able to write a small expert system, to do so in one of these relational languages would be a formidable and time-consuming task. Wanting to move on to our chosen area of study as quickly as possible, we chose to look at expert system shells.

We found and evaluated several expert system shells, a few of which could build fuzzy expert systems. The most promising of these was FLOPS, a shell that comes in both fuzzy and non-fuzzy flavors. We attempted to procure a copy of FLOPS from its author, William Siler, but after several emails it was beginning to seem like it was just going to take too long to achieve good communication with Dr. Siler, so, pressed for time, we went ahead and began implementing our system in CLIPS, a shell that is freely available, well documented, and quick to download. CLIPS was developed by NASA, who at one point had considered implementing the shell in LISP, a partially functional language in which it is fairly straightforward to create the functions Delay and Force, which can be used to implement lazy evaluation. For various reasons, however, NASA ultimately chose to implement their shell in C, a procedural language. At the time we began creating our expert system, we thought it would be possible to either use cross platform tools to add laziness from functional code, or use the Delay and Force functions in the C environment. As best as we can tell at this time, neither of these is possible. Perhaps this is an area for further research, but articles we read on the semantics of functional programming and its differences from the procedural paradigm make it look less than promising.

Gina had the opportunity to attend the Symposium on Applied Computing 1999 in San Antonio, Texas, where she participated in the Artificial Intelligence and Fuzzy Logic tracks. She gained insight from the lectures on contemporary work in the field.

Leha also plans to more fully develop our small bird identification expert system. The system can currently identify a few immature birds of California. Leha hopes to add the ability to identify adult birds and fledglings, and to make the list of identifiable species more complete for a larger geographical base.

We all hope to continue our work by either finding or creating an expert system shell that is written in a functional (or partially functional) language. Ideally, that language would be Haskell or Miranda. Other areas to for possible research are, as mentioned above, the implementation of laziness in C, or some other procedural language; use of lazy lists in a purely functional environment for prototyping of expert system front ends; creation of a relational language that has delayed evaluation, fuzzy logic and fuzzy sets built in; and creation of a more fully lazy version of Prolog, Datalog, Likelog, or some other existing relational language.

The opportunity to do research as undergraduates has helped us in many ways. Through gathering and reading the contemporary information in our area of study, we gained greater skill in the important area of surveying and reviewing the work of peers and of computer science professionals. By implementing lazy lists in the Watt model, choosing an expert system shell, evaluating several languages, expanding our knowledge of artificial intelligence, encountering and coping with obstacles in the course of seeking solutions, and working collaboratively, we have gained a better understanding of the research process and of the difficulties and challenges inherent in the pursuit of advanced research topics. Finally, through choosing a research topic, writing a successful proposal, and collecting and processing the results of our study, attempting to produce an honest document that will be useful to others involved in research, we have gained an important skill set that will help us in our graduate level pursuits, and in our careers.