1990

Expert systems in typography

David B. Fisher

Expert Systems in Typography

By

David B. Fisher

A thesis

Submitted to the Faculty of the
Computer Science Department

at

Rochester Institute of Technology

In partial fulfillment of the
Requirement for the
Degree of

Master of Science in Computer Science

July 1990

Expert Systems in Typography

By

David B. Fisher

A thesis

Submitted to the Faculty of the
Computer Science Department

at

Rochester Institute of Technology

In partial fulfillment of the
Requirement for the
Degree of

Master of Science in Computer Science

Approved By:

Professor, John A. Biles

Professor, Frank J. Cost

Professor, Peter G. Anderson

July 1990

Title of Thesis: EXPERT SYSTEMS IN TYPOGRAPHY

I David B. Fisher prefer to be contacted each time a request for reproduction is made. I can be reached at the following address:

15 NORTON ST
HONEOYE FALLS NY 14472

Date: Aug 2, 1990

Acknowledgements

I would like to acknowledge the help of the following people, Al Biles, Frank Cost, and Lisa Fisher. Without their help, encouragement and patience this thesis never would have been completed.

Contents

List of Figures

Abstract

Electronic publishing systems are providing authors with a tremendous amount of power formerly only available to printers: the choice of typographic design and layout. However, many authors use this power without the proper background, with potentially disastrous results. We have developed a prototype expert system that provides users of electronic publishing systems with the printer's knowledge in computer form in one narrow area of the publishing field to aid them in preparing their publications.

1.0 Introduction

This thesis was inspired by an article titled "Using Expert Systems in Typographic Design" [BILE87]. Its opening paragraph, perhaps, best describes the problem:

> "Beatrice Warde, who wrote extensively about the subtle art of typographic design in the early part of this century, described type as having a similar obligation to words that a crystal goblet has to wine[WARD56]. Warde believed that type, like crystal, must always be transparent, and never stand between the writer and the reader. If type is "seen" by the reader, it fails to perform its proper function. Typography that announces itself is bad typography."[BILE87]

Typographic design, and other formerly printer's decisions, are now being made by authors and users of desk top publishing and electronic publishing systems as part of their routine writing. Many of these users have never been instructed on how to design a document, and many have never really noticed how a typeface or page layout can affect the look of a page of text. Users get caught-up in all the fancy features provided by their system, even when they are totally irrelevant to the

task at hand, generating perfect examples of "bad typography announcing itself."

We explored one way of alleviating this problem, an expert system. This system provides users with a way to consult with an expert in publishing, via a computer program. By incorporating the knowledge of an expert the user can create his own design, but with proven technique as a constraint.

There are many aspects of a printer's job that require complex decision making. Each page, line, even each character, must be carefully placed so as to properly blend with the others and present the subject matter in an appropriate manner. From the first character on the title page to the last number in the index (and beyond) hundreds of possibilities exist.

The decisions made are based on years of tradition, education and in some areas an "artistic" talent. Printers are experts in every sense of the word, and their knowledge is unique to their field.

Enter computers. Computerization is usually introduced wherever a manual method is costly or time consuming. Printing is most definitely that. From its origin with hand picked, and set, type through various mechanization steps (ie. linotypes), to computerized type creation, setting, printer configuration

and pagination, the industry has continued to stream-line the more costly parts of the printing process. However, this computerization has been concentrated in the final product phase and has ignored much of the design process (except for design layout aids). The design is still done by human experts, which is fine for most printing tasks. However, some design is now being done by non-expert authors on word processors and desk-top publishing systems.

More sophisticated hardware is becoming readily available to authors to allow more of the design to be done as the manuscript is being created (written). Word processing systems using "What You See Is What You Get" (WYSIWYG) displays and high resolution laser printers provide the author with a finished product that often looks as good as a professionally printed document. Many of these systems create "camera-ready" output, which can be presented to a printer for direct mass production. Some printers may dispute this claim, but the fact is, some materials (such as articles in trade journals and conference proceedings, society and business newsletters, theses, etc.) are being "published" without ever being presented to a designer. The camera-ready output is simply mass-produced.

For this thesis we have chosen to implement an expert system to solve this type of problem in one narrow area. We have

chosen bibliographies. Our expert system collects the data that describes the bibliography entries, from a user, and then prepares the data for inclusion in an EPS document.

In the sections which follow we will discuss the design and development of this system. Section Two contains some background information on publishing and expert systems. Section Three discusses the expert system itself, from the core expert system, through the user interface and knowledge base, and on to the EPS interface. Section Four describes the development process with discussions about our expert system, the knowledge acquisition process, and the expert system's inference engine and internal structures. Section Five leads the reader through some sample sessions and demonstrates the expert systems results given various inputs. Section Six presents some conclusions and recommendations for how this system could be expanded and enhanced. In the Appendicies that follow are complete code listings and further examples.

2.0 Background on Publishing, How it was

Printing and publishing have been around for over four hundred years. In that time a great deal of change has taken place, from manual type setting and hand presses to computerized type setting and on-the-fly type creation. Until recently, though, one thing has remained constant: an author wrote the words and

a printing professional determined how it would look when printed. The designer or printer would determine what typeface to use, what column style was appropriate, etc., based on the intention of the published product.

For example, a children's book would be printed in a large type size using a legible typeface. A technical text book might use a typeface like Times Roman. Books are generally set in one column and newspapers and magazines are generally set in multiple columns. These decisions and many more are made at publication time by the printer/publisher.

2.1 Effects of Electronic and Desk Top Publishing Systems

DTP (Desk Top Publishing) and EPS (Electronic Publishing Systems) are replacing typewriters as the author's writing medium. Suddenly the author has more control over the look of the words on the page. Not only can he control the look, he can see the results as he enters his text and produce final, often referred to as "camera ready", output. This eliminates the middleman, but it also eliminates the expert's input into design decisions.

Authors may have a general feel for how they want their work to look, and some may prefer DTP systems because of the control they get, but many, or perhaps most, have no formal

training in the printing disciplines. Authors, users of DTP systems, can easily get caught up in the "nifty" features available and lose track of the goal of readability.

A good example of how typography can make a difference can be seen in figure 2.1 (from "Using Expert Systems in Typographic Design" [BILE87]). Here a sample page, complete with text and wood-cut illustrations is set in Helvetica (a stark, modern typeface), and Caledonia, a more traditional, classic, typeface. Although both examples convey the same information, the second seems more appropriate to the eye. These examples also deal with typeface selection and placement of captions and headings.

essential in this experiment to train the animal to perform isolated distal and proximal forelimb movements, extensive EMG recordings were made to examine whether selective activation of proximal and distal muscle activity was indeed obtained.

Cortical Field Determination

Saccadic eye movements in the vertical and horizontal directions were observed during the performance of the task, but they were randomly distributed in time and not regularly timelocked with the occurence of any sensory signals or limb movements. Additionally, in the rare instances when the monkey emitted key-press movements 'spontaneously,' i.e., not in response to any sensory signal, the neuron was active, as shown in the fourth row of Figure 3.

Figure 3. Detail from the Apocalypse.

Since no distinct boundary between cytoarchitectural characteristics can be drawn, a boundary to divide the frontal cortex was drawn on the basis of a density analysis of giant pyramidal cells in layer V. The number of giant pyramidal cells having a diameter exceeding 29 units was counted in consecutive 50 unit sections. Saccadic eye movements in the vertical and horizontal directions were observed during the performance of the task, but they were randomly distributed in time and not regularly timelocked with the occurence of any sensory signals or limb movements.

essential in this experiment to train the animal to perform isolated distal and proximal forelimb movements, extensive EMG recordings were made to examine whether selective activation of proximal and distal muscle activity was indeed obtained.

Cortical Field Determination

Saccadic eye movements in the vertical and horizontal directions were observed during the performance of the task, but they were randomly distributed in time and not regularly timelocked with the occurence of any sensory signals or limb movements.

Figure 3. Detail from the Apocalypse.

Since no distinct boundary between cytoarchitectural characteristics can be drawn, a boundary to divide the frontal cortex was drawn on the basis of a density analysis of giant pyramidal cells in layer V. The number of giant pyramidal cells having a diameter exceeding 29 units was counted in consecutive 50 unit sections.

Figure 2.1 - Wood Cut Examples

2.2 The Need

The question now is: "How can we take years of training, and centuries of experience and tradition available in the printing professional, and make it available to the novice DTP user?". The answer, of course, is with an expert system. Such a system would encode the knowledge of the printing professional into a sophisticated computer program, which the user could query when making design decisions.

2.3 Existing Expert Systems in the Publishing Industry

In the publishing arena expert systems have been used for printer configuration and troubleshooting of printing equipment. (Diagnosis of machine failure is a common application of expert systems in many fields.)

Press Lineup Advisor is an expert system developed at the Rockwell International Graphics Division to perform printing press configuration for newspaper printing[LAN87]. It uses rule-based backward reasoning and generate-and-test problem-solving to determine the press configuration. By using a knowledge base of printer data containing information about the printer's configuration procedure, and the physical and operational constraints of the device and the surrounding environment the expert system determines which plates go on

which printer, and in what order to produce the newspaper pages in the proper sequence. The Press Lineup Advisor uses a problem-solving strategy which closely resembles that used by human experts. It follows an overall process of sequential refinement. A problem is first analyzed at a coarse level, then details are added in stages until a conflict is found or the line-up is completed. Two problem-solving methods are used in the system; rule-based backward reasoning and generate-and-test verification.

PAGE-1 is a printer problem diagnostic tool for non-impact Page Printing Systems developed by Honeywell [STRA85]. It diagnoses problems in a variety of areas including software, hardware, HVAC, paper handling, hydraulics, and electrical power supplies. This diversity of potential problem areas leads to the use of different AI paradigms within the PAGE-1 system in order to "achieve effective system performance".

These two expert systems demonstrate a typical use of expert systems in the printing industry but do not demonstrate how an expert system may be useful in typographic design. Another expert system, CRITAC, developed at Carnegie Mellon, uses artificial intelligence to proofread Japanese text [TAKE88]. It uses accumulated knowledge to detect typographical errors, Kana to Kanja conversion errors, and style errors in Japanese text. It better shows how expert systems can aid authors in

their creative process. CRITAC (CRITiquing using ACcumulated Knowledge) is based on a conceptual representation of text (called structured text), a Prolog-coded knowledge base for proofreading, and text processing to generate a physical representation of the text. It lexically parses the text and creates an internal representation (which resembles sentence diagramming). The structured text can then be managed with a relation data base. Once the text is in this form it can be checked for spelling error, style errors and language conversion errors.

2.4 How They Would Help Here

In our problem the expert has been taken out of the production loop and, therefore, doesn't get an opportunity to lend advice and expertise. As a result the user/author has the opportunity to create horrendous looking documents. Actually, an expert system wouldn't be able to stop a user from creating horrendous looking documents. It would, however, provide a way for users to create documents that they could be reasonably confident were not horrendous looking, with little or no printing experience.

When a user first creates the specifications for his document (called a style sheet or publication specification) he defines where columns of text will appear on the page, what typeface

to use for text, headers, captions on illustrations, footnotes, etc., and what different pages will look like (i.e. title page, text body pages right and left, etc.). Once these are defined, the content of the book can be added, either by typing directly into the publication specification in a WYSIWYG type system or by creating a "manuscript" separately on a compatible word processing system. The text would be "marked-up" to contain hidden codes that would identify the start and end of headings, section heads, paragraphs, footnotes, etc., along with information about the location of illustrations in relation to the text. The manuscript is then merged into the publication's specification to create a final document. This publication, usually, can be edited further without returning to the manuscript.

With either method, the key to a quality look is the publication's initial specification. Users could start with a generic specification and then use trial and error to achieve the look they desire, but it would be better to create something correct from the beginning.

Some electronic publishing systems use a WYSIWYG page specification subsystem, where blocks of text are represented by boxes in a window that is the same size as the final page. This is of some help, but doesn't aid the user in proper typographic design.

Enter the expert system. When a user indicates interest in designing a new publication specification, he would interact with the expert system as it asks questions about the intended final product. The expert system might ask about the intended reader: what age group and sex, what type of use the document will get, e.g. reference, novel, how-to, text book, etc., the kind and number of illustrations expected, etc. These questions and others would emulate what an expert would ask of the author when presented with the manuscript for publication. Once the expert system has asked the necessary questions (the number and type of questions will vary based on the answers given) it would generate the publication specification, which describes to the user's publishing system the design an expert might use for that document. Now the user can fold his manuscript into a professional quality publication specification and get out a professional looking publication. The final publication specification could still be customized by the user to meet his personal preferences.

Other aspects of the work could be generated completely by the EPS or DTP systems, with the help of an expert system. It is common for EPS's to generate a complete table-of-contents, or index, automatically from the author's formatted manuscript. Chapter titles, sections, headings, etc. are known to the EPS, so it is a simple matter to collect them with the page numbers

they appear on and create a formatted table-of-contents. Indexes require the author to identify the first occurrence of a word or phrase as one to be included in the index. The EPS then can scan the document for all occurrences of these phrases and generate a formatted index. It is easy to see how similar techniques could be used to generate lists of tables, figures, etc. With some further extensions, references to these figures can be resolved automatically (eg. "see Figure 4.1 on page 122").

All of these require little expert knowledge on the part of the EPS. It is usually sufficient to provide a few styles for the author to choose from. There are other parts of published materials that do require, or at least can be aided by, expert knowledge. These areas also tend to be the most tedious for the author to create.

Most publications have some form of title page, copyright page, dedication page, and other pages that make up the "front-matter" of a book, as well as "back-matter" material like bibliographies, indexes, appendices, and glossaries. These pages contain a lot of rules to follow with regards to spacing, punctuation, capitalization, etc.

Here an expert system would be helpful. By prompting the author for data and asking questions about the nature of the

work, an expert system could create the various pages with all their data properly formatted, and, by interacting with the EPS, prepare the pages for direct inclusion with the rest of the work.

3 Expert System Project Overview

We have created such an expert system. Ours is what Waterman [WATE85] refers to as a "demonstration prototype," containing 50-100 rules. It focuses on one narrow aspect of publishing, in our case bibliography creation, and produces a files for use on an electronic publishing system.

The expert system was developed using an IBM-PC clone running MS-DOS. This was chosen primarily for convenience to the developer. It is intended that the code generated will also run on the UNIX based machines at the Rochester Institute of Technology (RIT). The generated code was written in standard Prolog and tested using Arity Prolog, a Prolog interpreter that runs on the PC. It is intended that the code generated will also run in RIT's "C-Prolog" interpreter.

The output from the expert system is four files. These will be described in greater detail in sections 4 and 5. Two of these files are designed to be input files for publishing systems. The first is a simple ASCII file which can be

incorporated into any system. The second contains more information for use specifically with WordPerfect. It is "marked-up" with WordPerfect's "Reveal Codes" for proper formatting once the file is read into a target WordPerfect document.

We chose bibliographies because of their well defined rules and their general complexity. Even though these rules are well defined, most writers, at the non-professional level, have a difficult time properly formatting a bibliography without some external aid. The usual choice is some sort of sytle manual. By incorporation these facts into a expert system, which would presumably be incorporated into an EPS, the user has the facts at hand, but is a more useful form.

Figure 3.1 is a module diagram of the expert system, showing how the different section of the system relate.

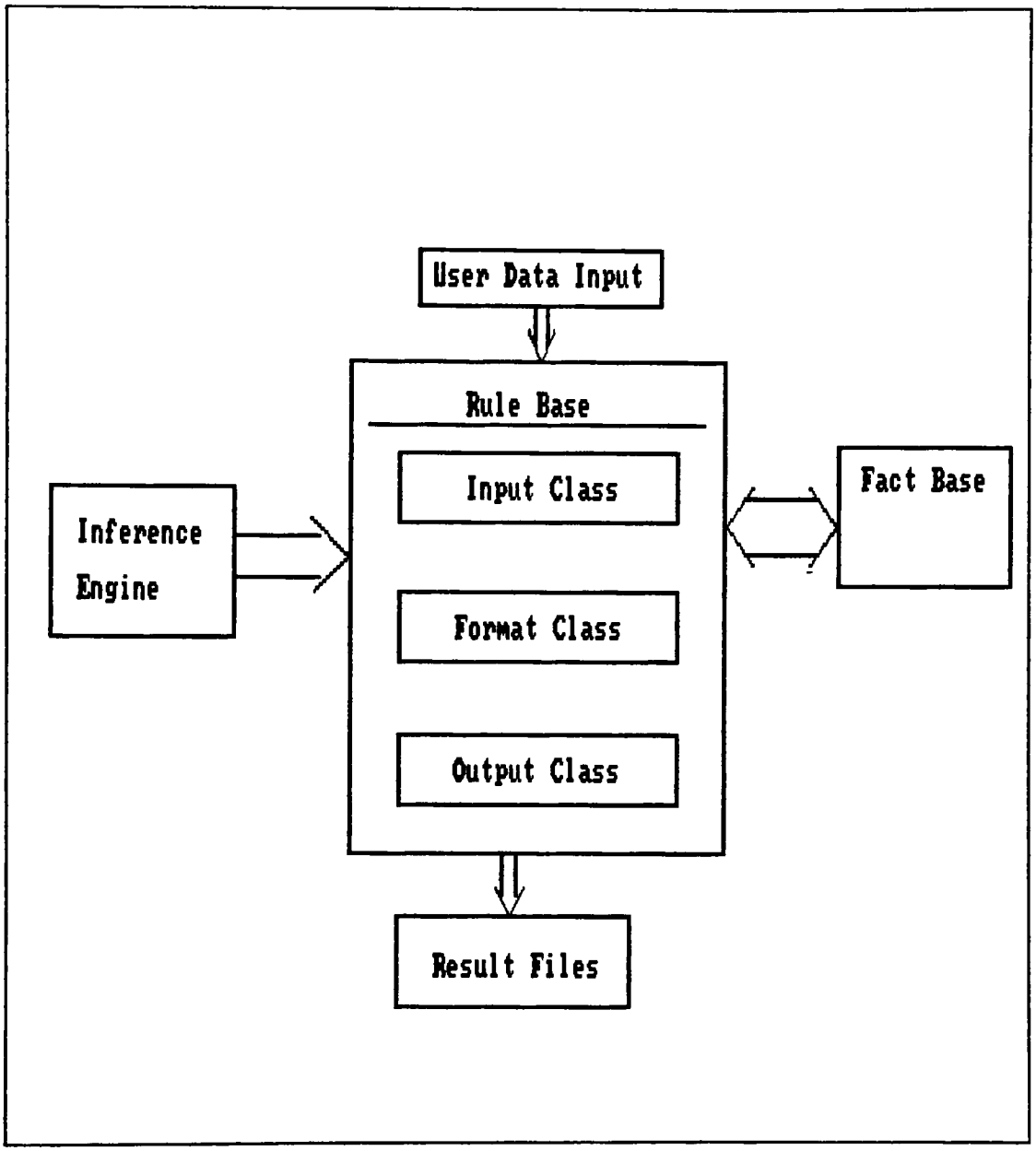

Figure 3.1 - Module Diagram

3.1 User Interface

The processing falls into two categories. The first part is a dialogue with the user to extract the data that make up the bibliography entry (i.e. author's name, book title, etc.), and to determine the type of reference each entry is and the topic matter of the work for which the bibliography is intended. Whenever possible the questions are formed to allow for yes/no answers or a choice from a list. With each question there is available the response "?", which tells the expert system the question is not understood by the user. When the system reads this character it will print out more information about the data being requested and then re-prompt the user.

The second part of the expert system's processing is batch oriented. During this phase the entries are formatted, alphabetized and prepared for output. The user still may be asked for information during this phase to clarify data entered or determine new information whose necessity wasn't known during the input phase.

The expert system then builds the entry, using earlier answers to determine what further questions need to be asked, and saving the final entry in an internal data base. Some of the data obtained from the user will cause a marked difference in

the look of the bibliography. By altering these types of responses the user also can see how different styles might look.

3.2 The Knowledge Base

The questions asked of the user are actually filling-in holes in the knowledge base. Initially the knowledge base contains all the information that can be extracted, and represented, from the expert. As mentioned, we have established a specific area of expertise, i.e. bibliographies, and ignored other possible areas, i.e. title pages. The knowledge base has been initialized with facts about bibliographies, including, but not limited to, their possible styles, the data that make up each entry, how to use punctuation, capitalization, font variants (i.e. italics), and other aspects relating to the data and the feel of the work.

The knowledge base is represented in Prolog. Access is through Prolog routines written specifically for this application and uses backward chaining. Specific examples of the rules are shown in Section Four and a complete listing of the rule base is contained in Appendicies C, E and G.

3.3 Interface to the Electronic Publishing System

The output from the expert system is in four forms, in four files. Each contains the same information, but in different formats. The first is the internal data base, which contains the data extracted from the user. This allows the user to re-format the bibliography using the same data, without typing in the data again. The second file is the formatted data, complete with style markup codings, which provide formatting information to the host EPS. The third file is the same as the second, but without the markup coding, for use in systems that don't support these codings. The fourth file is an explanation file, which provides the user with a list of rules applied during each step in the processing of each entry.

The marked-up manuscript page is generated in an ASCII format that is "human readable" and can be used to generate a publication page within an EPS. Once generated, the output can be uploaded into the EPS for inclusion with the document. The EPS will, no doubt, do further formatting to match font style, page size, etc. with the rest of the publication.

4 The Development Process

4.2 The Expert

Throughout the research for this thesis we have found references to style manuals or style guides as materials that can be used to aid in publication design. These books contain thousands of rules about all the different aspects of publication design. We also encountered as many warnings about their use in "excellent" designs:

> "The application of stock solutions to unexamined problems may produce superficially adequate results at times, due to coincidence, but over a long period there will be many more failures then successes. . . . Rules can be found in stylebook, but they are primarily to help beginners produce usable work or at least avoid the worst mistakes - they cannot produce excellent design."[LEE65]

Although, at first, such warnings would seem to indicate that stylebooks should be avoided, we must look at what we are trying to accomplish, before making a decision.

We will concede that, for the most part, publication design is as much an artistic effort as it is a mechanical effort. It could be equated to architecture, which mixes knowledge of structure with a sense of artistic design. We also will concede that it would be very difficult, if not impossible, to build an expert system that can work artistically. So, does that mean we're stuck? Not at all. The area we intend to work

in, bibliographies, deals with rules of proper form, rather than artistic appearance. The artistic part of the publication's overall design will be inherited when the bibliography is merged into the publication.

So, let us now re-examine the style guides. If you're looking for expert advice for a publication that doesn't need to be "award winning," style guides would seem quite adequate. They are, in fact, manual expert systems, providing all the rules, but leaving it up to the user to know when to apply them. It may be more accurate to say they are the extracted expert knowledge, without the inference engine to link them.

C. J. Petrie carries this even further by suggesting expert texts, like how-to books, could be scanned by natural language reading programs and expert systems built automatically. Natural language parsers still have severe limitations, but if the technical writer were restricted in style and vocabulary such extractions might be possible. [PETR85]

We have chosen a style guide for our expert. Specifically "The Chicago Manual of Style", pages 420-483, dealing with bibliographies, and other referenced sections. This book will provide all the rules we need along with explanation and help facilities, which will give section reference numbers back to the original text.

4.3 Knowledge Acquisition

Traditional knowledge acquisition involves a dialogue between the expert and the knowledge engineer (expert system developer), with repeated questioning, coding, testing, and verification of results. We used the same basic cycle in our development, with the exception that the questioning part actually involved reading (studying) the expert source text.

Rules were developed by reading the source text. These were converted into a pseudo-code we used to simplify the final coding process. For example, the following section in the expert text:

16.11 - Authors' names should be spelled in a bibliography as they appear on the title pages of their books, except that first names may be given in full in place of initials. Degrees or affiliations following names should normally be omitted (except "M.D." for an author of a medical work).
16.13 - Conversely, especially in Style B, initials may be used for all given names, regardless of how they appear on a title page or at the head of an article.

Figure 4.1 - Expert Text Example

---

was converted into this pseudo code:

```
rule : 16.11
    class : format
    subclass : author
    restriction : style = a
    restriction : author_type = single_author
    format_string : ["%W, %W %C.",author(last),
                          author(First), author(middle)]

rule : 16.13
    class : format
    subclass : author
    restriction : style = b
    restriction : author_type = single_author
    format_string : ["%W, %C. %C.",author(last),
                          author(First), author(middle)]
```

Figure 4.2 - Pseudo-code Example

The pseudo code reads like this: a rule is defined for **CMS**, "Chicago Manual of Style", Section **16.11** which is of **class format** and **subclass author**. This rule is restricted to processing records for **style 'a'** and with **author_type single**. If all these conditions are met, execute the utility **format_string** with the control string shown in quotes and the author's **last**, **first** and **middle** names as data. The second rule read similarly. The concept of class and subclass is discussed in Sections 4.4.2 and 4.4.3.

It can be easily seen how this code resembles Prolog. The pseudo-code could then be automatically processed by a knowledge-base compiler which would "write" Prolog code for the expert system. We chose not to implement this solution. It, in itself, would make a good thesis project. We also

don't anticipate a large number of these rules, so hand processing is not an unreasonable task. If this system were to be "productized," such an effort would be desirable, especially since it takes the knowledge representation effort out of the hands of the Prolog coder and places it into the hands of someone with more publishing and less computer knowledge.

The pseudo-code above was translated into the following Prolog code for the expert system:

```
rule(format,author,[16.11,16.25,16.29], format_author).
format_author(C,S,CMS) :-
        register(current_entry,E),
        determine(style,_,a),
        entry(E,author_type,single_author),
        entry(E,author,Author),
        parse_author(Author,First, Middle, Last),
        format_string(Formated_Author,'%W, %W %C.',
                      [Last,First,Middle])),
        establish(formated,[E,author,Formated_Author]),!.

format_author(C,S,CMS) :-
        register(current_entry,E),
        determine(style,_,b),
        entry(E,author_type,single_author),
        entry(E,author,Author),
        parse_author(Author,First, Middle, Last),
        format_string(Formated_Author,'%W, %C. %C.',
                      [Last,First,Middle])),
        establish(formated,[E,author,Formated_Author]),!.
```

Figure 4.3 - Rule Code Example

It is presented here for completeness. More discussion on how it is implemented in the expert system follows in section 4.4

"The Expert System."

With any expert system knowledge acquisition is the most critical and the most difficult phase. Using a book as the expert presented some problems, but also offered some advantages. In the case of this particular thesis the biggest advantage was availability. In the academic world, where experts are offering time above and beyond their regular class load and knowledge engineers are often part-time students with full-time jobs, the scheduling of meetings is difficult and the time spent in extraction of knowledge can be scarce. When the book is the expert, it is always available for long or short sessions. You can also be sure that the data derived by the expert is the same every time. We don't mean to imply that experts are inconsistent, but, with any human source, the same question can get different answers at different times. (A computer will always return the same answer when given the same set of input data.) Because the expert used here is printed in a hard format the only possibility of variablilty from one extraction session to another is in the interpretation of the engineer. This variable will always exist.

Using a book does offer some problems. The knowledge engineer does need to have more of a working knowledge of the application. In our case, publishing, as with most

specialized areas, there exists a set of terms that may not be known outside the field. In the section of the expert text used for this expert system we encountered terms like three-em dash which is not something found outside of publishing. If a question arises that the text doesn't make clear, the knowledge engineer doesn't have a source that can be asked for explanations. In these situations it helps to have a back-up human source for expert advice.

4.4 The Expert System

The expert system developed here can be divided into several parts. The basic unit is the inference engine, or shell. This controls the processing of the rules, which then control the flow of the program. The rules fall into three classes; input, format and output. As we will see in the next section these classes are an important concept of the overall expert system. The remaining code is a collection of utilities available throughout the code.

4.4.1 The Inference Engine

Controlling the flow of the program is a shell called the inference engine. It is actually a very simple piece of code, and can be incorporated into any expert system that is batch processing oriented, as is the case here.

It is more accurate to say that the rules control the flow of the program (data-oriented flow control is the nature of Prolog and other artificial intelligence languages). Each rule is identified with a class and a subclass. The expert system simply executes each rule for each class and subclass combination in order. The rules are designed to quickly determine if they should be applied to the current task and fail when appropriate.

The rules are applied to entries. Each entry is one reference in the bibliography. The entries are numbered internally as they are entered by the user.

The flow of the program looks like this:

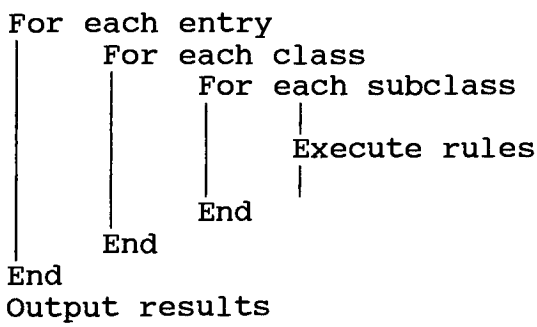

Figure 4.4 - Inference Engine Flow diagram

A simplified version of the inference engine is shown in figure 4.5. Familiarity with Prolog will show the use of "cut-fail" to loop through a set of terms. (The actual code

is in Appendix B.)

```
bib :-
     initialize,
     class(C),                  /* For Each Class... */
     process_class(C),
     fail.

bib.

process_class(C) :-
     initialize(C),
     subclass(S),               /* For Each Subclass... */
     process_subclass(C,S),
     class_done(C),!.           /* Until all entries processed */

process_subclass(C,S) :-
     rule(C,S,CMS,Functor),
     Arg_List = [C,S,CMS],
     Goal =.. [Functor | Arg_List],
     call(Goal).              /* ... Execute the rules */
```

Figure 4.5 - Inference Engine Code

4.4.2 Classes

Classes define the major loop of the process control. In this implementation of the expert system, three classes are defined; input, format, and output. Each class performs a function on each subclass of each entry.

Input: The input class prompts the user for the data that make up the entries; author's name, title, etc. It makes extensive use of utilities that accept text from the keyboard and arrange it into lists for internal storage.

Format: The format class takes the raw data from the user and formats it into the proper form for the bibliography. It determines the style to be used (bibliographies fall into two basic styles), and then establishes formatted values for each subclass of each entry. Again, utilities are used to perform generic functions like capitalization of the first letter of each word, parse a name or date into it's component parts, etc.

Output: The output class orders these entries alphabetically and arranges the parts of each entry according to the style selected and the rules in the knowledge base. It establishes output data records in the internal database, which are then dumped to output files for the user to include in the final document. The actual writing to the file is done as part of the system exit and clean-up.

4.4.3 Subclasses

The subclasses are the component parts of the entries in the bibliography. They consist of the authors's name, book or article title, sub-title, series name and volume number, edition, city, publisher, periodical name, date of publication and pages. Of course not all subclasses apply to all entries. Books would not require the pages or periodical name

subclasses, while the periodical would not have city, edition, or series data. The rules that prompt the user for entry data know the type of entry being processed, through a subclass called type, and will "fail" before asking the user for superfluous facts.

4.4.4 Rules

The rules used throughout the code are formatted into two types of terms. First, a rule term serves as a link between the class/subclass processing and the code for the rule. It has the form:

```
rule(CLASS, SUBCLASS, CMS_SECTION, ROUTINE_NAME).
```

The objects for the rule term are the CLASS and SUBCLASS as we've already seen, the CMS_SECTION, which is the section in the Chicago Manual of Style (our expert text), which provided the rule, and the routine to be called to process this rule (ROUTINE_NAME).

The second term is the routine which is executed to process the rule. For ease of development they are named "class_subclass," but any name can be used. Sample routines are shown in figure 4.6.

```
rule(format, title, [16.31], format_title).

format_title(C,S,CMS) :-
     register(current_entry,E),
     entry(E,title, Title),
     capitalize_first(Title, Formated_Title),
     establish(formated, [E, title, Formated_Title]),!.

rule(output, title, [16.31,16.124], output_title).

output_title(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,book),
     formated(E, title, Title),
     output(italic, on),
     output(Title), output('.'),
     output(italic,off),output(' '),!.

output_title(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,periodical),
     formated(E, title, Title),
     output('"'),
     output(Title), output('.'),
     output('"'),output(' '),!.
```

Figure 4.6 - Example Rule Routines

Each routine has the same basic structure. The data base is accessed to determine the number of the entry being processed through a register structure. Other "short-circuiting" terms then determine if the rule is valid for the entry being processed. In the example above, the term "entry(E,type,book)" would short-circuit for a periodical entry and the term "entry(E,type,periodical)" would short-circuit for a book entry. The remaining terms process the data and assert new facts into the internal data base for later use. All rule routines end with a cut, "!", which

prevents backtracking into the routine. The engine, as seen above, loops through the classes and subclasses by failing, which causes backtracking. The rules are written such that if the execution proceeds all the way through the rule, it's work is done. Backtracking through a rule in search of another solution would be a waste of time.

4.4.5 Tools and Utilities

Throughout the code a variety of tools and utilities were written to help modularize the code. The utilities fall into two classes; those that make the expert systems run, and those that are specific to the application. We will discuss some of the more interesting ones here.

We make use of a number of utilities that will be found in almost any Prolog program in one form or another, such as: list membership utilities, a read sentence routine and a register handler to allow for global variables. We've added a utility to store facts in the internal database called "establish," utilities to initialize by class and the expert system as a whole, and utilities to determine when a class is done.

A set of "ask" utilities were developed to allow the user to be questioned. These accept as object parameters a prompt

message, explanation facility code and a list of valid answers, and they return a verified answer to the expert system. They come in four versions: ask_YN, which prompts for a yes or no answer to a question; ask_text, which accepts any string of words and returns a list of these words with blanks removed and punctuation separated; ask_list, which prompts for a list of text strings; and ask_number, which accepts only a number as input. Each of these routines accept a "?" as input and will print the explanation facility message for the user and re-ask the question.

For the format class a set of utilities were developed, which modifies the appearance of words. The capitalize_first routine scans a list of words and capitalizes the first letter of each. (Words such as 'the' and 'a' are or are not capitalized as appropriate to their location in the string.) A string formatting utility uses a formatting code (similar to the printf function in 'C', but processing words in a list) to rearrange and modify text in preparation for output. Two parsing routines were developed to parse names into first, middle and last, and dates into month, day and year.

For the output class, utilities were developed to handle where blanks are printed, to handle the output of the mark-up codes (in this case Word Perfect "Reveal Codes," but, by using the utility, other codes could be added easily), and write the

results for each of the four result files created.

4.4.6 The Explanation Facility

Enclosed in each rule is a code for the section in our expert text, the Chicago Manual of Style. Every time the user is prompted for data, this section number is made available to the "ask" code. The user can request help at any prompt, and the section from the manual is displayed. Whenever a complete rule is applied, a note is made in the database. One of the output files from the expert system contains a list of rules applied during each phase of the batch processing. Its format is illustrated in figure 4.7.

For entry 1:

When inputting the type rules 16.2, and 16.3 where applied.
When inputting the author rules 16.2, and 16.3 where applied. When inputting the title rules 16.2, and 16.3 where applied.
When inputting the city rule 16.2 was applied.
When inputting the publisher rule 16.2 was applied.
When inputting the date rules 16.2, and 16.3 where applied.
When formatting the author rules 16.11, 16.25, and 16.29 where applied.
When formatting the title rule 16.31 was applied.
When formatting the city rule 16.62 was applied.
When formatting the publisher rule 16.62 was applied.
When formatting the date rule 16.62 was applied.
When outputting the author rules 16.15, 16.25, and 16.29 where applied.
When outputting the title rules 16.31, and 16.124 where applied.
When outputting the city rule 16.63 was applied.

When outputting the publisher rule 16.63 was applied.
When outputting the date rules 16.62 was applied.

Figure 4.7 - Example Explanation File

If the user has any question about why an entry was created the way it was, he can refer to the explanation file and use the information contained in it as a pointer into the expert text. A more "on-line" approach would be nice, and would be a necessity if this were other than a prototype system.

5.0 Sample User Sessions

In order to better demonstrate the use of the expert system, we will present a full sample user session in which two entries are made and then show the results. (In the figure below the user's responses will be shown in bold face type for clarity.) The examples used throughout this section were taken from the expert text, section 16.

?- **bib.**

Is this entry for a book or a Periodical?

1. book
2. periodical
3. done

1

Which best describes the writer of the book?

1. single_author

2. multiple_author
3. editor
4. institution

1

Enter the author's name as it appears on the title page.

John R. Woodrush

Enter the full title, with any sub-title, of the book

Songs my father taught me

Is this book part of a series? [Y / N]

n

Is this a first edition? [Y / N]

y

Enter the city of publication.

New Haven

Enter the name of the publisher.

Birdwatchers press

Enter the year of Publication.

1985

Is this entry for a book or a Periodical?

1. book
2. periodical
3. done

1

Which best describes the writer of the book?

1. single_author
2. multiple_author
3. editor
4. institution

1

Enter the author's name as it appears on the title page.

Ian Barbour

Enter the full title , with any sub-title, of the book

Myths, models, and paradigms: A comparative study in science and religion

Is this book part of a series? [Y / N]

n

Is this a first edition? [Y / N]

y

Enter the city of publication.

New York

Enter the name of the publisher.

Harper & Row

Enter the year of Publication.

1974

Is this entry for a book or a Periodical?

1. book
2. periodical
3. done

3

Bibliographies come in two basic styles. The best style depends on the topic of the publication.
    Style A: literature, history, and art.
    Style B: natural and social sciences.

Which style best fits your publication?

1. a
2. b

a

Enter the name of the file for final output.

fig51

Figure 5.1 - Sample User Session

Four output files from this session will contain the results of the expert system's processing. They are shown below:

Internal Database File FIG51.ARI:

```
entry(1,type,book).
entry(1,author_type,single_author).
entry(1,author,['John','R',.,'Woodrush',@]).
entry(1,title,['Songs',my,father,taught,me,@]).
entry(1,city,['New','Haven',@]).
entry(1,publisher,['Birdwatchers',press,@]).
entry(1,date,1985).
entry(2,type,book).
entry(2,author_type,single_author).
entry(2,author,['Ian','Barbour',@]).
entry(2,title,['Myths',',',models,',',and,paradigms,:,'A',
comparative,study,in,science,and,religion,@]).
entry(2,city,['New','York',@]).
entry(2,publisher,['Harper','Row',@]).
entry(2,date,1974).
no_of_entries(3).
class_done(input).
```

Explanation file FIG51.EXP:

For entry 1:

When inputting the type rules 16.2, and 16.3 where applied.
When inputting the author rules 16.2, and 16.3 where applied.
When inputting the title rules 16.2, and 16.3 where applied.
When inputting the edition rule 16.2 was applied.
When inputting the city rule 16.2 was applied.
When inputting the publisher rule 16.2 was applied.
When inputting the date rules 16.2, and 16.3 where applied.
When formatting the author rules 16.11, 16.25, and 16.29 where applied.
When formatting the title rules and 16.31 where applied.
When formatting the city rules and 16.62 where applied.

When formatting the publisher rules and 16.62 where applied.
When formatting the date rules and 16.62 where applied.
When outputting the author rules 16.15, 16.25, and 16.29 where applied.
When outputting the title rules 16.31, and 16.124 where applied.
When outputting the city rules and 16.63 where applied.
When outputting the publisher rules and 16.63 where applied.
When outputting the date rules and 16.62 where applied.

For entry 2:

When inputting the type rules 16.2, and 16.3 where applied.
When inputting the author rules 16.2, and 16.3 where applied.
When inputting the title rules 16.2, and 16.3 where applied.
When inputting the edition rule 16.2 was applied.
When inputting the city rule 16.2 was applied.
When inputting the publisher rule 16.2 was applied.
When inputting the date rules 16.2, and 16.3 where applied.
When formatting the author rules 16.11, 16.25, and 16.29 where applied.
When formatting the title rules and 16.31 where applied.
When formatting the city rules and 16.62 where applied.
When formatting the publisher rules and 16.62 where applied.
When formatting the date rules and 16.62 where applied.
When outputting the author rules 16.15, 16.25, and 16.29 where applied.
When outputting the title rules 16.31, and 16.124 where applied.
When outputting the city rules and 16.63 where applied.
When outputting the publisher rules and 16.63 where applied.
When outputting the date rules and 16.62 where applied.

Raw Output Bibliography File FIG51.OUT:

Bibliography

Barbour, Ian. Myths, Models, and Paradigms: A Comparative Study In Science and Religion. New York: Harper Row, 1974.
Woodrush, John R. Songs My Father Taught Me. New Haven: Birdwatchers Press, 1985.

Marked-Up Bibliography File FIG51.MRK:

```
[HRt][HPg][LARGE][CENTR]Bibliography[C/A/Flrt][large]
[HRt][>INDENT][<Mar Rel]Barbour, Ian [ITALC]Myths, Models, and
Paradigms: A Comparative Study In Science and Religion.[italc]
New York: Harper Row, 1974.
[HRt]Woodrush, John R. [ITALC]Songs My Father Taught
Me.[italc] New Haven: Birdwatchers Press, 1985.
[HRt]
```

The formated entry would appear like this:

Bibliography

Barbour, Ian. *Myths, Models, and Paradigms: A Comparative Study In Science and Religion*. New York: Harper Row, 1974.

Woodrush, John R. *Songs My Father Taught Me*. New Haven: Birdwatchers Press, 1985.

Figure 5.2 - User Session Output Files

Notice how capitalization in the data entered by the user doesn't matter. The expert system made all capitalization decisions.

In addition to the facts stored in the '.ari' file the following facts were established internally. These were not stored because they can be regenerated from the internal database file.

```
formated(1,author,'Woodrush, John R.').
formated(1,title,['Songs','My','Father','Taught','Me',@]).
formated(1,city,['New','Haven',@]).
formated(1,publisher,['Birdwatchers','Press',@]).
formated(1,date,1985).
formated(2,author,'Barbour, Ian.').
formated(2,title,['Myths',',','Models',',',and,'Paradigms',:
,'A',
```

```
'Comparative','Study','In','Science',and,'Religion',@]).
formated(2,city,['New','York',@]).
formated(2,publisher,['Harper','Row',@]).
formated(2,date,1974).

outputted(0,'[HRt]').
outputted(0,'[HPg]').
outputted(0,'[LARGE]').
outputted(0,'[CENTR]').
outputted(0,'Bibliography').
outputted(0,'[C/A/Flrt]').
outputted(0,'[large]').
outputted(0,'[HRt]').
outputted(0,'[>INDENT][<Mar Rel]').
outputted(2,'Barbour, Ian.').
outputted(2,' ').
outputted(2,'[ITALC]').
outputted(2,'Myths').
outputted(2,',').
outputted(2,' ').
outputted(2,'Models').
outputted(2,',').
outputted(2,' ').
outputted(2,and).
outputted(2,' ').
outputted(2,'Paradigms').
outputted(2,:).
outputted(2,' ').
outputted(2,'A').
outputted(2,' ').
outputted(2,'Comparative').
outputted(2,' ').
outputted(2,'Study').
outputted(2,' ').
outputted(2,'In').
outputted(2,' ').
outputted(2,'Science').
outputted(2,' ').
outputted(2,and).
outputted(2,' ').
outputted(2,'Religion').
outputted(2,.).
outputted(2,'[italc]').
outputted(2,' ').
outputted(2,'New').
outputted(2,' ').
outputted(2,'York').
outputted(2,': ').
outputted(2,'Harper').
outputted(2,' ').
outputted(2,'Row').
```

```
outputted(2,', ').
outputted(2,1974).
outputted(2,.).
outputted(2,'[HRt]').
outputted(1,'Woodrush, John R.').
outputted(1,' ').
outputted(1,'[ITALC]').
outputted(1,'Songs').
outputted(1,' ').
outputted(1,'My').
outputted(1,' ').
outputted(1,'Father').
outputted(1,' ').
outputted(1,'Taught').
outputted(1,' ').
outputted(1,'Me').
outputted(1,.).
outputted(1,'[italc]').
outputted(1,' ').
outputted(1,'New').
outputted(1,' ').
outputted(1,'Haven').
outputted(1,': ').
outputted(1,'Birdwatchers').
outputted(1,' ').
outputted(1,'Press').
outputted(1,', ').
outputted(1,1985).
outputted(1,.).
outputted(1,'[HRt]').
```

Figure 5.3 - Other Facts Established

We can also show how the same entries would be stored if style 'B' where chosen. The internal database file would be the same, and the explanation file very similar, but the bibliography files would look like this:

.OUT File:

Bibliography

Barbour, I. 1974. Myths, Models, and Paradigms: A Comparative Study In Science and Religion. New York: Harper Row.
Woodrush, J. R. 1985. Songs My Father Taught Me. New Haven: Birdwatchers Press.

.MRK File:

```
[HRt][HPg][LARGE][CENTR]Bibliography[C/A/Flrt][large]
[HRt][>INDENT][<Mar Rel]Barbour, I. 1974. [ITALC]Myths, Models, and Paradigms: A Comparative Study In Science and Religion.[italc] New York: Harper Row.
[HRt]Woodrush, J. R. 1985. [ITALC]Songs My Father Taught Me.[italc] New Haven: Birdwatchers Press.
[HRt]
```

Formated output:

Bibliography

Barbour, I. 1974. *Myths, Models, and Paradigms: A Comparative Study In Science and Religion.* New York: Harper Row.
Woodrush, J. R. 1985. *Songs My Father Taught Me.* New Haven: Birdwatchers Press.

Figure 5.4 - Session Output in Style B

If at any time during the user's input session he entered a question mark, "?", the expert system would have provided explanations about why the question was being asked. This is demonstrated below:

```
Is this entry for a book or a Periodical?

1. book
2. periodical
3. done

1

Which best describes the writer of the book?

1. single_author
2. multiple_author
```

```
3. editor
4. institution

1

Enter the author's name as it appears on the title page.

?
Chicago Manual of Sytle, section 16.2:

In the main, individual entries in all scholarly reference
lists and bibliographies include similar information about a
published work.  For a book, these facts are
     Name of the author, or authors, the editors, or the
         institution responsible for the writing of the book
     Full title of the book, including the subtitle, if any
     Title of the series, if any, and the volume or number in
     the
         series
     Volume number or total number of volumes of a multivolume
     work
     Edition, if not the original
     City of publication
     Publisher's Name
     Date of publication

Chicago Manual of Sytle, section 16.3:

For an article in a periodical, the facts given are
    Name of the author
    Title of the article
    Name of the periodical
    Volume Number (sometimes the issue number)
    Date
    Pages occupied by the article


Enter the author's name as it appears on the title page.

Barbara Stanwick
...
```

Figure 5.5 - Example of Explanation Facility

These examples are provided to give the reader an idea of the input processing environment. A great deal of processing happens "under-the-hood." We will now show various inputs

with specific problems to be solved and the results provided by the expert system.

It is common for books to have multiple authors. Here is such a case formatted in both styles:

Style A:

Unwin, L. P., and Galloway, Joesph. *Peace In Ireland.* Boston: No Such Press, 1984.

Style B:

Unwin, L. P., and Galloway, J. 1984. *Peace In Ireland.* Boston: No Such Press.

Figure 5.6 - Two Author Example in Both Styles

The challenge to the expert system is mostly in the parsing of the name and the placement of commas and the word 'and'. Three or more authors would be formatted similarly:

Merk, J. S., Fogg, I. J., and Snowe, C. Q. *Meteorologist's Handbook.* Chicago: Alwether and Clere, 1983.

Figure 5.7 - Multiple Author Example in Style A

If the author is actually an editor the entry would be:

Herbert, Sandra ed. *The Red Notebook of Charles Darwin.* Ithaca N. Y.: Cornell University Press, 1980.

Figure 5.8 - Editor as Author Example

If the book is written by an organization the entry would be:

International Monetary Fund. *Surveys of African Economies*. Vol. 7, Algeria, Mali, Morocco, and Tunisia. Washington, D.C.: International Monetary Fund, 1977.

Figure 5.9 - Organization as Author

This example also shows the existence of a series name and volume number reference in the bibliography entry.

A "parse author" routine was developed to accept names in most any normal form and extract the component parts for later formatting. Some examples:

Names entered as:	Medical	Parsed into: First	Mid	Last
Hellen Caldwell		Hellen		Caldwell
M. M. Bober		M.	M	Bober
David B. Fisher		David	B	Fisher
david b. fisher		david	b	fisher
David Byrne Fisher		David	Byrne	Fisher
J. M. Smyth, M.D.	Yes	J.	M	Smyth, M. D.
J. M. Smyth, M.D.	No	J.	M	Smyth
John L. Jones, Jr.		John	L	Jones, Jr.

Figure 5.10 - Parsing Author's Names Example

If an affiliation is encountered (ie M.D., CPA, etc.) it is usually ignored, with the exception of "M.D." This affiliation is appropriate if the text referenced is medical in orientation. This will be determined by prompting the user:

```
Is the text titled:
        [Dr,.,Spock's,Baby,and,Child,Care,@]
a Medical text?
 [ Y / N ]

Y
...
```

Figure 5.11 - Affiliation Prompting Example

The results for a name with affiliation would be formatted like this:

Spock, M. D., Benjamin., and Rothenberg, M. D., M. B. *Dr. Spock's Baby and Child Care*. 40th ed. New York: Pocket Books, 1985.

Figure 5.12 - Affiliation Usage Example

This example also demonstrates the inclusion of an edition number for book beyond their first edition.

The difference between books and periodicals in primarily in the way the title is set. A book's title is set in italics:

Eliot, T. S. *Four Quartets*. London: Faber and Faber, 1944.

Figure 5.13 - Book Title Example

A periodical title is the article's name, which is set in quotes, and the name of the periodical is set in italics:

Scott, Spencer. "Childhood's End." *Harper's* , January 1979, 16-19.

Figure 5.14 - Periodical Title Example

Dates appear in two different places in a bibliography entry based on the style in use. The user can enter the date in most any form commonly found in a publication, and the expert system will parse out the component parts (month, day and

year), and format or discard them as necessary:

Entered as:	Date Parser returns:		
	Month	Day	Year
10/11/90	October	11	1990
10/90	October		1990
Jan. 1990	January		1990
January 12, 1990	January	12	1990
may 12, 1880	May	12	1880
1968			1968
90			1990

Figure 5.15 - Date Parsing Example

As an additional example, the bibliography for this thesis can be examined. It is included at the end of this document in style "A" and in style "B" in Appendix L.

6.0 Conclusions and Recommendations

In conclusion, we believe we have accomplished the task defined in our proposal. We have developed a working prototype expert system which provides expert knowledge in one narrow area of the field of publishing. This system is, admittedly, a long way from a production piece of software, but it does provide some proof that such a system is feasible.

Working with a book as an expert was a very interesting part of this project. This type of expert wouldn't work for all expert systems, but many expert systems are "how-to" oriented,

and for every topic which falls into this category there exists a book of this type. C. J. Petrie talks about natural language parsers which would read how-to books and extract expert system rules. This would be the next logical step, and is believed to be feasible (especially if the technical writer is restricted to a vocabulary known to the parser)[PETR85].

The inference engine written for this project is simple, but very powerful. It is very general in its design, and could be used in a variety of applications. Only the rules would need to be written. It is, however, batch processing oriented and would not work well for a dialogue type problem solving system. In cases where a large amount of data is being processed, in background, with some expert knowledge controlling the flow, this inference engine would work well. It has an interesting ability to work on the input data, establish new facts based on it, and then pass this new knowledge on to the next level of processing intelligence. (This is the class structure in our engine). It can also separate data into different categories (the subclasses) which can interrelate with each other, and repeatedly process these levels for each instance of a collection of data (the entries).

Probably the weakest part of the prototype is the input

section. Although some useful tools were developed to aid in extraction of data from the user, they fall short of today's trend to window oriented displays, and dialogue boxes. An interesting extension of this input processing does present itself though. If this expert system was linked into a library's computerized card catalog system the user could get their bibliography listing by providing the book's call number (ie. Library of Congress or Dewey Decimal System numbers). The computerized card catalog entry for a book provides all the necessary information, and most of the information for a periodical. Any additional information would have to be provided by the user (ie. which article in a periodical, or which pages were references).

The section of our expert text which we chose to implement, bibliographies, is considerably larger and more complex than the prototype we have developed. Other, more specialized or unique, cases exist and are well documented in the manual (ie. film strips, computer programs, slides, unknown author works, etc.), but implementing all of these would have brought us well out of the realm of prototype. We implemented those cases that were most common and useful.

The goal of any thesis, or any academic project, is, of course, to provide education for the student. That was definitely accomplished. As the author of this thesis I've

learned about expert systems through all stages and explored the knowledge extraction process in its traditional manner and with a different and exciting twist. I have also gained a better understanding of artificial intelligence and become a much better Prolog programmer.

Appendicies

Appendix A - Pseudo-Code

```
rule : 16.11
     class : format
     subclass : author
     restriction : style = a
     restriction : author_type = single_author
     format_string : ["%W, %C%C.", author(last),
author(first), author(middle)]!

rule : 16.11
     class : format
     subclass : author
     restriction : style = b
     restriction : author_type = single_author
     format_string : ["%W, %C. %C.", author(last),
author(first), author(middle)]!

rule : 16.15
     class : format
     subclass : author
     restriction : style = a
     restriction : author_type = multiple_authors
     for_each (author)
          {
          format_string : ["%W, %C. %C.",author(last),
author(first), author(middle)]
          }!

rule : 16.15
     class : format
     subclass : author
     restriction : style = b
     restriction : author_type = multiple_authors
     for_each (author)
          {
          format_string : ["%W, %C. %C.",author(last),
author(first),author(middle)]
          }!


rule : 16.15
     class : output
     subclass : author
     restriction : author_type = multiple_authors
     for_each_except_last (author)
          {
          output(formated(author))
          output(", ")
          }
     for_last
          {
          output("and ")
```

```
          output(formated(author))
          }!


rule : 16.24
     class : output
     subclass : author
     restriction : author_type = editor
     output(formated(author))
     output(" ed. ")!

rule : 16.29
     class : output
     subclass : author
     restriction : author_type = organization
     capitalize_first : author!

/*********************************************************/
/*   title                                               */
/*********************************************************/

rule : 16.31
     class : format
     subclass : title
     capitalize_first : title!

rule : 16.31
     class : output
     subclass : title
     output(title)
     output(". ")!


/*********************************************************/
/*   Editions                                            */
/*********************************************************/

rule : 16.54
     class : output
     subclass : edition
     restriction : edition = 2
     output("2nd edition")!


rule : 16.54
     class : output
     subclass : edition
     restriction : edition = 3
     output("3rd edition")!

rule : 16.54
     class : output
```

```
     subclass : edition
     output("%Nth edition",edition)!

/*********************************************************/
/*   City                                                */
/*********************************************************/
rule : 16.62
     class : format
     subclass : city
     capitalize_first(city)!

rule : 16.62
     class : output
     subclass : city
     restriction : style = a
     output(city)
     output(": ")!

/*********************************************************/
/*   Publisher                                           */
/*********************************************************/
rule : 16.62
     class : format
     subclass : publisher
     capitalize_first(publisher)!

rule : 16.62
     class : output
     subclass : publisher
     restriction : style = a
     output(publisher)
     output(", ")!

rule : 16.62
     class : output
     subclass : publisher
     restriction : style = b
     output(publisher)
     output(".")!



/*********************************************************/
/*   Date                                                */
/*********************************************************/
rule : 16.62
     class : output
     subclass : date
     restriction : style = a
     output(date)
     output(".")!
```

```
rule : 16.62
     class : output
     subclass : date
     restriction : style = b
     output(date)
     output(". ")!
```

Appendix B - Inference Engine Code

```
/*********************************************************/
/*            Expert System in Typography                */
/*             by David B. Fisher                        */
/*                                                       */
/*        Module:        Engine                          */
/*        Function: Main Inference Engine                */
/*                                                       */
/*********************************************************/

/*******
        Entry Point to Bibliography generator
*******/

bib :-
     initialize,
     class(C),
     process_class(C),
     fail.

/*******
        Exit Point from Bibliography generator
*******/
bib :-
     ask_text(filename,[Filename|_],[0.0]),
     database_to_file(Filename),
     output_to_file(Filename),
     markup_to_file(Filename),
     explain_to_file(Filename),
     turn_capture(off).

ask_text_data(filename,$Enter the name of the file for final
output.$).

/*******
     Process for each entry, each class...

          initialize entry loop registers
          perform initialization for the class
          While the class hasn't indicated completion...
               process the Subclasses...
*******/
process_class(C) :-
     set_register(current_entry,0),
     set_register(relative_entry,0),
     initialize(C),
     not(class_done(C)),
     repeat,
     subclass(S),
     process_subclass(C,S),
     class_done(C),!.

/******
```

```
     Process for each subclass

          fetch the next rule
          build the term to process rule
          call the rule...
*******/
process_subclass(C,S) :-
     rule(C,S,CMS,Functor),
     Arg_List = [C,S,CMS],
     Goal =.. [Functor | Arg_List],
     call(Goal).

/******
     Classes...
******/
class(input).
class(format).
class(output).

/******
     Subclasses...
******/
subclass(type).
subclass(author).
subclass(title).
subclass(sub_title).
subclass(series_vol_number).
subclass(edition).
subclass(city).
subclass(publisher).
subclass(periodical).
subclass(date).
subclass(pages).
```

Appendix C - Input Class Code

```
/*************************************************************/
/*            Expert System in Typography                    */
/*            by David B. Fisher                             */
/*                                                           */
/*        Module:        Input                               */
/*        Function: Rules for Input Class                    */
/*                                                           */
/*************************************************************/

rule(input,type,[16.2,16.3],ask_type).

ask_type(C,S,CMS) :-
     inc_register(current_entry),
     register(current_entry,E),
     ask_multiple(type,TYPE,CMS),
     (ask_type_done(TYPE);
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,type,TYPE])),!.
ask_type_done(done) :-   /* No more entries from failure
above */
     establish(class_done,[input]),
     register(current_entry,E),
     establish(no_of_entries,[E]).


ask_multiple_data(type,$Is this entry for a book or a
Periodical?$,
     [book,periodical,done]).


/***
     AUTHOR
***/

rule(input,author,[16.2,16.3],ask_author).

ask_author(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,book),      /* is the entry for a book? */
     ask_multiple(author,AUTHOR_TYPE,CMS),
     ask_author_2(C,S,CMS,AUTHOR_TYPE,Authors),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,author_type,AUTHOR_TYPE]),
     establish(entry,[E,author,Authors]),!.
ask_author(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,periodical),     /* is the entry for a
periodical? */
     ask_author_2(C,S,CMS,single_author,Author),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,author_type,single_author]),
     establish(entry,[E,author,Author]),!.
```

```
ask_author_2(C,S,CMS,single_author,ANS) :-
     ask_text(author,ANS,CMS).
ask_author_2(C,S,CMS,editor,ANS) :-
     ask_text(editor,ANS,CMS).
ask_author_2(C,S,CMS,institution,ANS) :-
     ask_text(institution,ANS,CMS).
ask_author_2(C,S,CMS,multiple_author,ANS) :-
     ask_list(authors,[],ANS,CMS).

ask_text_data(author,$Enter the author's name as it appears
on the title page.$).
ask_text_data(editor,$Enter the editor's name as it appears
on the title page.$).
ask_text_data(institution,$Enter the institution's name as
it appears on the title page.$).
ask_text_data(authors,$Enter the names of the authors, one
at a time.  Return with no entry will terminate list$).
ask_multiple_data(author,$Which best describes the writer of
the book?$,

[single_author,multiple_author,editor,institution]).


/***
     TITLE
***/

rule(input,title,[16.2,16.3],ask_title).

ask_title(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,TYPE),
     ask_title_2(C,S,CMS,TYPE,Title),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,title,Title]),!.

ask_title_2(C,S,CMS,book,Title) :-
     ask_text(book_title,Title,CMS).
ask_title_2(C,S,CMS,periodical,Title) :-
     ask_text(periodical_title,Title,CMS).

ask_text_data(book_title,$Enter the full title , with any
sub-title, of the book$).
ask_text_data(periodical_title,$Enter the title of the
article.$).

/***
     SERIES
***/

rule(input,series_vol_number,[16.2],ask_series).
```

```
ask_series(C,S,CMS) :-
     register(current_entry,E),
     register(current_entry,E),
     entry(E,type,book),
     ask_YN(series,Ans,CMS),!,
     Ans==yes,
     ask_text(series,Series_name,CMS),
     ask_text(volume,Volume,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,series,Series_name]),
     establish(entry,[E,volume,Volume]),!.

ask_YN_data(series,$Is this book part of a series?$).
ask_text_data(series,$Enter the series title.$).
ask_text_data(volume,$Enter the volume number cited.$).

/***
     EDITION
***/

rule(input,edition,[16.2],ask_edition).

ask_edition(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,book),
     ask_YN(edition,Ans,CMS),!,
     Ans == no,
     ask_number(edition,Number,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,edition,Number]),!.

ask_YN_data(edition,$Is this a first edition?$).
ask_number_data(edition,$Enter the edition number.$).

/***
     CITY
***/

rule(input,city,[16.2],ask_city).

ask_city(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,book),
     ask_text(city,City,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,city,City]),!.

ask_text_data(city,$Enter the city of publication.$).


/***
```

```
     PUBLISHER
***/

rule(input,publisher,[16.2],ask_publisher).

ask_publisher(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,book),
     ask_text(publisher,Publisher,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,publisher,Publisher]),!.


ask_text_data(publisher,$Enter the name of the publisher.$).

/***
     DATE
***/

rule(input,date,[16.2,16.3],ask_date).

ask_date(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,book),
     ask_number(year,Year,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,date,Year]),!.
ask_date(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,periodical),
     ask_text(date,Date,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,date,Date]),!.

ask_number_data(year,$Enter the year of Publication.$).
ask_text_data(date,$Enter the date of the issue of the
periodical.$).


/***
     PERIODICAL
***/

rule(input,periodical,[16.3],ask_periodical).

ask_periodical(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,periodical),
     ask_text(periodical,Periodical,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,periodical,Periodical]),!.
```

```
ask_text_data(periodical,$Enter the name of the
periodical.$).


/***
     PAGES
***/

rule(input,pages,[16.3],ask_pages).

ask_pages(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,periodical),
     ask_number(first_page,Start,CMS),
     ask_number(last_page,End,CMS),
     establish(rule_applied,[E,C,S,CMS]),
     establish(entry,[E,pages,[Start,End]]),!.


ask_number_data(first_page,$Enter the number of the page the
article starts on.$).
ask_number_data(last_page,$Enter the number of the page the
article ends on.$).
```

Appendix D - Input Utilities

```
/*********************************************************/
/*            Expert System in Typography                */
/*             by David B. Fisher                        */
/*                                                       */
/*        Module:        ask_util                        */
/*        Function: Utilities for Input Class            */
/*                                                       */
/*********************************************************/
ask_list(Text_key,List_i n,List_out,CMS):-
     ask_text(Text_key,Item,CMS),
     build_list(List_in,List_out,Item),
     (One == done ;
          ask_list(Text_key,List_out,A,CMS)).

build_list(I,I,done).
build_list(In,Out,Val) :-
     append_list(In,[Val],Out).


ask_text(Text_key,Ans,CMS) :-
     ask_text_data(Text_key,Message),
     repeat,
     nl,write(Message),nl,nl,
     getsentence(Response),
     need_help(Response,CMS),
     Ans = Response.

ask_YN(Text_key,Ans,CMS) :-
     ask_YN_data(Text_key,Message),
     repeat,
     nl,write(Message),sp,
     write('[ Y / N ]'),nl,nl,
     get0(Response),
     need_help(Response,CMS),
     verify(Response,Ans).
verify(89,yes).          /* Y */
verify(78,no).      /* N */
verify(121,yes).    /* y */
verify(110,no).          /* n */

ask_number(Text_key,Ans,CMS) :-
     ask_number_data(Text_key,Message),
     repeat,
     nl,write(Message),nl,nl,
     getsentence(Number),
     need_help(Number,CMS),
     integer_list(Number),
     [Ans,@] = Number.

integer_list([I|_]) :-
     integer(I).
```

```
ask_multiple(Text_key,Ans,CMS) :-
     ask_multiple_data(Text_key,Message,List),
     repeat,
     nl,write(Message),nl,nl,
     write_items(1,List),nl,
     getsentence(Answer),
     need_help(Answer,CMS),
     get_list_entry(List,Answer,Ans).

get_list_entry(List,[In|_],Out) :-
     (member_return(In,List,Out) ;
      integer(In),member_number(In,List,Out)).

need_help([?,@],[CMS|_]) :-
     cms(CMS,Text),
     write(Text),nl,nl,!,
     fail.
need_help(63,[CMS|_]) :-
     cms(CMS,Text),
     write(Text),nl,nl,!,
     fail.
need_help(C,_) :-
     C \== [?,@].


cms(16.2,'CMS 16.2 message text').
cms(_,'No help available').
```

Appendix E - Format Class Code

```
/*************************************************************/
/*      Expert System in Typography                          */
/*            by David B. Fisher                             */
/*                                                           */
/*        Module:        Format                              */
/*        Function: Rules for Format Class                   */
/*                                                           */
/*************************************************************/

/***
     TYPE...
***/

rule(format,type,[0.0],format_type).

format_type(C,S,CMS) :-
     inc_register(current_entry),
     are_we_done(format).

/***
     AUTHOR...
***/

rule(format,author,[16.11,16.25,16.29], format_author).

format_author(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,a),
     entry(E,author_type,single_author),
     entry(E,author,Author),
     parse_author(Author,First, Middle, Last),
     (((Middle == ''),format_string(Formated_Author,'%W,
%W.',[Last,First]));
     format_string(Formated_Author,'%W, %W
%C.',[Last,First,Middle])),
     establish(rule_applied,[E,C,S,CMS]),
     establish(sort_key,[Formated_Author - E]),
     establish(formated, [E, author, Formated_Author]),!.

format_author(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,b),
     entry(E,author_type,single_author),
     entry(E,author,Author),
     parse_author(Author,First, Middle, Last),
     (((Middle == ''),format_string(Formated_Author,'%W,
%C.',[Last,First]));
     format_string(Formated_Author,'%W, %C.
%C.',[Last,First,Middle])),
     establish(rule_applied,[E,C,S,CMS]),
     establish(sort_key,[Formated_Author - E]),
     establish(formated, [E, author, Formated_Author]),!.
```

```
format_author(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,a),
     entry(E,author_type,editor),
     entry(E,author,Author),
     parse_author(Author,First, Middle, Last),
     (((Middle == ''),format_string(Formated_Author,'%W,
%W',[Last,First]));
     format_string(Formated_Author,'%W, %W
%C.',[Last,First,Middle])),
     establish(rule_applied,[E,C,S,CMS]),
     establish(sort_key,[Formated_Author - E]),
     establish(formated, [E, author, Formated_Author]),!.

format_author(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,b),
     entry(E,author_type,editor),
     entry(E,author,Author),
     parse_author(Author,First, Middle, Last),
     (((Middle == ''),format_string(Formated_Author,'%W,
%C.',[Last,First]));
     format_string(Formated_Author,'%W, %C.
%C.',[Last,First,Middle])),
     establish(rule_applied,[E,C,S,CMS]),
     establish(sort_key,[Formated_Author - E]),
     establish(formated, [E, author, Formated_Author]),!.

format_author(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,a),
     entry(E,author_type,multiple_author),
     entry(E,author,Authors),
     for_each(Authors, format_author_for_each_a,
Formated_Authors),
     establish(rule_applied,[E,C,S,CMS]),
     establish(sort_key,[Formated_Authors - E]),
     establish(formated, [E, author, Formated_Authors]),!.

     format_author_for_each_a(Author, Formated_Author) :-
          parse_author(Author,First, Middle, Last),
          (((Middle ==
'' ),format_string(Formated_Author,'%W, %W.',[Last,First]));
          format_string(Formated_Author,'%W, %C.
%C.',[Last,First,Middle])).


format_author(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,b),
     entry(E,author_type,multiple_author),
```

```
     entry(E,author,Authors),
     for_each(Authors, format_author_for_each_a,
Formated_Authors),
     establish(rule_applied,[E,C,S,CMS]),
     establish(sort_key,[Formated_Authors - E]),
     establish(formated, [E, author, Formated_Authors]),!.

     format_author_for_each_b(Author, Formated_Author) :-
          parse_author(Author,First, Middle, Last),
          (((Middle ==
''),format_string(Formated_Author,'%W, %C.',[Last,First]));
          format_string(Formated_Author,'%W, %C.
%C.',[Last,First,Middle])).

format_author(C,S,CMS) :-
     register(current_entry,E),
     entry(E, author_type,institution),
     entry(E, author,     Institution),
     capitalize_first(Institution, Formated_Institution),
     establish(sort_key,[Formated_Institution - E]),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, author,
Formated_Institution]),!.

/***
     TITLE...
***/

rule(format, title, [16.31], format_title).

format_title(C,S,CMS) :-
     register(current_entry,E),
     entry(E,title, Title),
     capitalize_first(Title, Formated_Title),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, title, Formated_Title]),!.

/***
     EDITION...
***/

     /*** No Formating rules for Edition are necessary... */

/***
     SERIES NAME and VOLUME NUMBER...
***/

rule(format, series_vol_number, [16.44],
format_series_vol_number).

format_series_vol_number(C,S,CMS) :-
     register(current_entry,E),
```

```
     entry(E,series, Series),
     capitalize_first(Series, Formated_Series),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, series, Formated_Series]),!.

/***
     PERIODICAL NAME...
***/

rule(format, periodical, [16.124], format_periodical).

format_periodical(C,S,CMS) :-
     register(current_entry,E),
     entry(E,periodical, Periodical),
     capitalize_first(Periodical, Formated_Periodical),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, periodical,
Formated_Periodical]),!.

/***
     CITY...
***/

rule(format, city, [16.62], format_city).

format_city(C,S,CMS) :-
     register(current_entry,E),
     entry(E, city, City),
     capitalize_first(City, Formated_city),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, city, Formated_city]),!.

/***
     PUBLISHER...
***/

rule(format, publisher, [16.62], format_publisher).

format_publisher(C,S,CMS) :-
     register(current_entry,E),
     entry(E, publisher, Publisher),
     capitalize_first(Publisher, Formated_publisher),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, publisher,
Formated_publisher]),!.

/***
     DATE...
***/

rule(format, date, [16.62], format_date).
```

```
format_date(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,b),
     entry(E, date, Date),
     parse_date(Date,Month,Day,Year),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, year,  Year]),
     establish(formated, [E, day,   Day]),
     establish(formated, [E, month, Month]),!.

format_date(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,a),
     entry(E, date, Date),
     parse_date(Date, Month, Day, Year),
     ((Day == '',Month ==
'',format_string(FDate,'%K',[Year]));
      (Day == '',format_string(FDate,'%W %K',[Month,Year]));
               format_string(FDate,'%W %K,
%K',[Month,Day,Year])),
     establish(rule_applied,[E,C,S,CMS]),
     establish(formated, [E, date, FDate]),!.

/***
     PAGES...
***/

     /* No format rules necessary for Pages... */
```

Appendix F - Format Utilities

```
/*************************************************************/
/*            Expert System in Typography                    */
/*             by David B. Fisher                            */
/*                                                           */
/*        Module:        FormatU                             */
/*        Function: Format Class Utilities                   */
/*                                                           */
/*************************************************************/
/***
     Capitalize the first letter in each word
***/

capitalize_first(WordList, Out) :-
     cap_each(WordList, [], Out).

cap_each([],Out,Out).
cap_each([Word|Rest], Out, Ans) :-
     cap_word(Word,CWord),
     append(Out,[CWord],NewOut),
     cap_each(Rest,NewOut, Ans).
cap_word(the,the).
cap_word(a,a).
cap_word(of,of).
cap_word(and,and).
cap_word(is,is).
cap_word(Word,CWord) :-
     name(Word,[FC|RC]),
     toupper(FC,UFC),
     name(CWord,[UFC|RC]).
toupper(C,C) :-
     C >= "A", C =< "Z".
toupper(C,U) :-
     C >= "a", C =< "z",
     U is C - "a" + "A".
toupper(C,C).

/***
     Format a string based on a control format record
***/

format_string(Ans, Format, Params) :-
     name(Format,Format_List),
     fs(Ans_List,[], Format_List,Params),
     name(Ans,Ans_List).
fs(Ans, Ans, [], _).
fs(FAns, Ans, [37|Rest],Param) :-
     fs_p(FAns, Ans, Rest, Param).
fs(FAns, Ans, [FC | Rest], Param) :-
     append(Ans,[FC],Ans_out),
     fs(FAns, Ans_out,Rest, Param).
fs_p(FAns, Ans, [_|Rest],['' |Param]) :-
```

```
     fs(FAns, Ans,Rest,Param).
fs_p(FAns, Ans, [87|Rest],[Data|Param]) :-
     name(Data,[FC|RC]),
     toupper(FC,UFC),
     append(Ans,[UFC|RC],Ans_out),
     fs(FAns, Ans_out,Rest,Param).
fs_p(FAns, Ans, [67|Rest],[Data|Param]) :-
     name(Data,[FC|RC]),
     toupper(FC,UFC),
     append(Ans,[UFC],Ans_out),
     fs(FAns, Ans_out,Rest,Param).
fs_p(FAns, Ans, [75|Rest],[Data|Param]) :-
     name(Data,AsIs),
     append(Ans,AsIs,Ans_out),
     fs(FAns, Ans_out,Rest,Param).

/****
     Parse Author into it's component parts...
****/

parse_author([First, Last, @],First,'',Last).
parse_author([First, M, ., Last, @],First,M,Last).
parse_author([F,., M, ., Last, @],First,M,Last) :-
     format_string(First,'%C.',[F]).
parse_author([First, Middle, Last,@],First,Middle,Last).
parse_author([F,., Middle, Last,@],F,Middle,Last).
parse_author([First, M1, ., M2, ., Last,@],First,M,Last) :-
     M = [M1,.,' ',M2,.].
parse_author([F,., M1, ., M2, ., Last,@],F,M,Last) :-
     M = [M1,.,' ',M2,.].
parse_author([First, M, ., L, ',' | Affl],First,M,Last):-
     use_affliation(Affl,Use_Affl),
     format_string(Last,'%W, %K',[L, Use_Affl]).
parse_author([F,., M, ., L, ',' | Affl],F,M,Last):-
     use_affliation(Affl,Use_Affl),
     format_string(Last,'%W, %K',[L, Use_Affl]).
parse_author([First, Middle, L ,',',|
Affl],First,Middle,Last):-
     use_affliation(Affl,Use_Affl),
     format_string(Last,'%W, %K',[L, Use_Affl]).
parse_author([F,., Middle, L ,',' | Affl],F,Middle,Last):-
     use_affliation(Affl,Use_Affl),
     format_string(Last,'%W, %K',[L, Use_Affl]).
parse_author([First, L, ',' | Affl],First,'',Last):-
     use_affliation(Affl,Use_Affl),
     format_string(Last,'%W, %K',[L, Use_Affl]).

use_affliation([@],'').
use_affliation([m,.,d,.,@],Affl) :-
     determine(medical_text,_,YN),
     ((YN == yes, Affl = 'M. D.') ; Affl = '').
use_affliation([jr,.,@],'Jr.').
```

```
use_affliation([sr,.,@],'Sr.').
use_affliation([ii,.,@],'II.').
use_affliation([iii,.,@],'III.').
use_affliation(['M',.,'D',.,@],Affl) :-
     determine(medical_text,_,YN),
     ((YN == yes, Affl = 'M. D.') ; Affl = '').
use_affliation(['JR',.,@],'Jr.').
use_affliation(['SR',.,@],'Sr.').
use_affliation(['Jr',.,@],'Jr.').
use_affliation(['Sr',.,@],'Sr.').
use_affliation(['II',.,@],'II.').
use_affliation(['III',.,@],'III.').

/****
     Parse Date into it's component parts...
****/

parse_date(Y,'','',Year) :-
     integer(Y),
     year(Y,Year).
parse_date([Mon,'/',D,'/',Yr,@], Month, Day, Year) :-
     day(D,Day),
     year(Yr,Year),
     month(Mon, Month).
parse_date([Mon,'/',Yr,@], Month, '', Year) :-
     year(Yr,Year),
     month(Mon, Month).
parse_date([Mon,D,',',Yr,@], Month, Day, Year) :-
     year(Yr,Year),
     day(D,Day),
     month(Mon, Month).
parse_date([Mon,.,D,',',Yr,@], Month, Day, Year) :-
     year(Yr,Year),
     day(D,Day),
     month(Mon, Month).
parse_date([Mon,Yr,@], Month, '', Year) :-
     year(Yr,Year),
     month(Mon, Month).
parse_date([Mon,.,Yr,@], Month, '', Year) :-
     year(Yr,Year),
     month(Mon, Month).
month(1,'January').
month(2,'Febrary').
month(3,'March').
month(4,'April').
month(5,'May').
month(6,'June').
month(7,'July').
month(8,'August').
month(9,'September').
month(10,'October').
month(11,'November').
```

```
month(12,'December').
month(jan,'January').
month(feb,'Febrary').
month(mar,'March').
month(apr,'April').
month(may,'May').
month(jun,'June').
month(june,'June').
month(jul,'July').
month(july,'July').
month(aug,'August').
month(sep,'September').
month(sept,'September').
month(oct,'October').
month(nov,'November').
month(dec,'December').
month(Jn,'January').
month(Feb,'Febrary').
month(Mar,'March').
month(Apr,'April').
month(May,'May').
month(Jun,'June').
month(June,'June').
month(Jul,'July').
month(July,'July').
month(Aug,'August').
month(Sep,'September').
month(Sept,'September').
month(Oct,'October').
month(Nov,'November').
month(Dec,'December').
month(Mon,Month) :-
     cap_word(Mon,Month).

year(Y,Year) :-
     Y < 100,
     Year is 1900 + Y.
year(Year,Year).
day(D,Day) :-
     format_string(Day,'%W ',[D]).
```

Appendix G – Output Class Code

```
/*************************************************************/
/*             Expert System in Typography                   */
/*              by David B. Fisher                           */
/*                                                           */
/*         Module:         Output                            */
/*         Function: Rules for Output Class                  */
/*                                                           */
/*************************************************************/

/***
     TYPE...
***/

rule(output,type,[0.0],output_type).

output_type(C,S,CMS) :-
     inc_register(relative_entry),
     register(relative_entry,E),
     are_we_done(output),
     (class_done(output) ;
               sort_list(Entries),
               member_number(E,Entries,Author - Number),
               set_register(current_entry,Number),!,
               write('Outputting:
'),write(Number),sp,write(Author),nl),
     !.

/***
     AUTHOR...
***/

rule(output,author,[16.15,16.25,16.29], output_author).

output_author(C,S,CMS) :-
     register(current_entry,E),
     entry(E,author_type,single_author),
     formated(E, author, Author),
     establish(rule_applied,[E,C,S,CMS]),
     output(Author),output(' '),!.

output_author(C,S,CMS) :-
     register(current_entry,E),
     entry(E,author_type,multiple_author),
     formated(E, author, Authors),
     for_each_except_last(Authors, output_author_most,
output_author_last, Output),
     establish(rule_applied,[E,C,S,CMS]),
     output(Output),output(' '),!.

     output_author_most(Author, Output_String) :-
          concat(Author,',',Output_String).
     output_author_last(Author, Output_String) :-
```

```
          concat('and ',Author,Output_String).

teel(Output) :-
      for_each_except_last(['First, A.','Last ,B. C.'],
output_author_most, output_author_last, Output).

output_author(C,S,CMS) :-
     register(current_entry,E),
     entry(E,author_type,editor),
     formated(E, author, Editor),
     establish(rule_applied,[E,C,S,CMS]),
     output(Editor),
     output(' ed. '),!.

output_author(C,S,CMS) :-
     register(current_entry,E),
     entry(E,author_type,institution),
     formated(E, author, Institution),
     establish(rule_applied,[E,C,S,CMS]),
     output(Institution),output(' '),!.

/***
     TITLE...
***/

rule(output, title, [16.31,16.124], output_title).

output_title(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,book),
     formated(E, title, Title),
     establish(rule_applied,[E,C,S,CMS]),
     output(italic, on),
     output(Title), output('.'),
     output(italic,off),output(' '),!.

output_title(C,S,CMS) :-
     register(current_entry,E),
     entry(E,type,periodical),
     formated(E, title, Title),
     establish(rule_applied,[E,C,S,CMS]),
     output('"'),
     output(Title), output('.'),
     output('"'),output(' '),!.

/***
     EDITION...
***/

rule(output, edition, [16.54], output_edition).

output_edition(C,S,CMS) :-
```

```
      register(current_entry,E),
      entry(E,edition,2),
      establish(rule_applied,[E,C,S,CMS]),
      output('2nd ed. '),!.


output_edition(C,S,CMS) :-
      register(current_entry,E),
      entry(E,edition,3),
      establish(rule_applied,[E,C,S,CMS]),
      output('3rd ed. '),!.

output_edition(C,S,CMS) :-
      register(current_entry,E),
      entry(E,edition,N),
      establish(rule_applied,[E,C,S,CMS]),
      output(N),
      output('th ed. '),!.

/***
      SERIES NAME and VOLUME NUMBER...
***/

rule(output, series_vol_number, [16.44],
output_series_vol_number).

output_series_vol_number(C,S,CMS) :-
      register(current_entry,E),
      formated(E, series, Series),
      entry(E,volume,Volume),
      establish(rule_applied,[E,C,S,CMS]),
      output(Series), output(', vol. '),
      output(Volume),output('. '),!.

/***
      PERIODICAL NAME...
***/

rule(output, periodical, [16.124], output_periodical).

output_periodical(C,S,CMS) :-
      register(current_entry,E),
      formated(E, periodical, Periodical),
      establish(rule_applied,[E,C,S,CMS]),
      output(italic,on),
      output(Periodical),
      output(italic,off),
      output(Month),
      ((Day == '');(output(' '),output(Day))),
      output(', '),!.

/***
```

```
     CITY...
***/

rule(output, city, [16.63], output_city).

output_city(C, S, CMS) :-
     register(current_entry,E),
     determine(style,_,a),
     formated(E, city, City),
     establish(rule_applied,[E,C,S,CMS]),
     output(City),
     output(': '),!.

output_city(C, S, CMS) :-
     register(current_entry,E),
     determine(style,_,b),
     formated(E, city, City),
     establish(rule_applied,[E,C,S,CMS]),
     output(City),
     output(': '),!.

/***
     PUBLISHER...
***/

rule(output, publisher, [16.63], output_publisher).

output_publisher(C, S, CMS) :-
     register(current_entry,E),
     determine(style,_,a),
     formated(E, publisher, Publisher),
     establish(rule_applied,[E,C,S,CMS]),
     output(Publisher),
     output(', '),!.

output_publisher(C, S, CMS) :-
     register(current_entry,E),
     determine(style,_,b),
     formated(E, publisher, Publisher),
     establish(rule_applied,[E,C,S,CMS]),
     output(Publisher),
     output('. '),output(hard,return),!.

/***
     DATE...
***/

rule(output, date, [16.62], output_date).

output_date(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,a),
```

```
     formated(E, date, Date),
     establish(rule_applied,[E,C,S,CMS]),
     output(Date),
     (entry(E,type,book),output('.'),output(hard,return) ;
      entry(E,type,periodical),output(', ')),!.

output_date(C,S,CMS) :-
     register(current_entry,E),
     determine(style,_,b),
     formated(E, year, Year),
     establish(rule_applied,[E,C,S,CMS]),
     output(Year), output('. '),!.

/***
     PAGES...
***/

rule(output, pages, [16.124], output_pages).

output_pages(C,S,CMS) :-
     register(current_entry,E),
     entry(E,pages,[Start,End]),
     establish(rule_applied,[E,C,S,CMS]),
     output(Start),output('-'),output(End),output('.'),
     output(hard,return),!.
```

Appendix H - Output Utilities

```
/*************************************************************/
/*            Expert System in Typography                    */
/*             by David B. Fisher                            */
/*                                                           */
/*        Module:        OutputU                             */
/*        Function: Output class Utilities                   */
/*                                                           */
/*************************************************************/
output(' ') :-
     queued_blank.
output(' ') :-
     assertz(queued_blank).
output(['@']).
output([First|Rest]) :-
     output(First),
     (Rest = ['@'] ; output(' ')),
     output(Rest).


output(String) :-
     output_queued_blank(String),
     register(current_entry,E),
     establish(outputted,[E,String]).

output_queued_blank(_) :-
     not(queued_blank).
output_queued_blank(':') :- retract(queued_blank).
output_queued_blank(',') :- retract(queued_blank).
output_queued_blank('.') :- retract(queued_blank).
output_queued_blank(_) :-
     register(current_entry,E),
     establish(outputted,[E,' ']),
     retract(queued_blank).

/*****
     Output control codes...
*****/

output(italic,on) :-                output('[ITALC]').
output(italic,off) :-             output('[italc]').
output(hard,page) :-              output('[HPg]').
output(header,on) :-              output('[LARGE]').
output(header,off) :-             output('[large]').
output(center,on) :-              output('[CENTR]').
output(center,off) :-             output('[C/A/Flrt]').
output(hard,return) :-            output('[HRt]').
output(paragraph,hanging_indent) :-
output('[>INDENT][<Mar Rel]').

output_to_file(File) :-
     concat(File,$.out$,Filename),
```

```
     create(Handle,Filename),
     out_to_file(no,Handle).
markup_to_file(File) :-
     concat(File,$.mrk$,Filename),
     create(Handle,Filename),
     out_to_file(yes,Handle).

out_to_file(SGML,Handle) :-
     asserta(fp(Handle)),
     output_each(Handle,SGML),
     nl(Handle),
     close(Handle),
     retract(fp(Handle)).

output_each(H,SGML) :-
     outputted(E,String),
     if_cr(H,String),
     [!(sgml_test(SGML,String);write(H,String))!],
     fail.
output_each(_,_).

if_cr(H,'[HRt]') :-
     nl(H),!.
if_cr(_,_).

sgml_test(no,String) :-
     name(String,[91|Rest]).

/********
     output explainations to file...
********/
explain_to_file(File) :-
     concat(File,$.exp$,Filename),
     create(H,Filename),
     set_register(current_entry,1),
     no_of_entries(No),
     repeat,
     register(current_entry,E),
     explain_entry(E,H),
     inc_register(current_entry),
     register(current_entry,F),
     No == F,
     close(H).

explain_entry(E,H) :-
     nl(H),write(H,'For entry
'),write(H,E),write(H,':'),nl(H),nl(H),
     class(C), subclass(S),
     [!setof(R,R^rule_applied(E,C,S,R),Rules),
     Rules \== [],
     write(H,'    When '),write(H,C),write(H,'ting the '),
     write(H,S),
```

```
     ((write_item(H,Rules),write(H,' was applied.'));
      (write(H,' rules '),write_list(H,Rules),write(H,'
where applied.')))!],
     nl(H),fail,!.
explain_entry(_,_).

write_item(H,[[One]]) :-
     atomic(One),
     write(H,' rule '),
     write(H,One).

write_list(H,[One_Left]) :-
     atomic(One_Left),
     write(H,'and '),write(H,One_Left).
write_list(H,[First|Rest]) :-
     atomic(First),
     write(H,First),write(H,', '),
     write_list(H,Rest).
write_list(H,[List]) :-
     not(atomic(List)),
     write_list(H,List).
write_list(H,[List|Rest]) :-
     write_sub_list(H,List),
     write_list(H,Rest).

write_sub_list(H,[]).
write_sub_list(H,[First|Rest]) :-
     write(H,First),write(H,', '),
     write_sub_list(H,Rest).

/**********
     Output database to file
********/

database_to_file(File) :-
     concat(File,$.ari$,Filename),
     create(H,Filename),
     out_entry(H),
     no_of_entries(No),
     writeq(H,no_of_entries(No)),write(H,'.'),nl(H),
     writeq(H,class_done(input)),write(H,'.'),nl(H),
     assertz(class_done(output)),
     close(H).

out_entry(H) :-
     entry(A,B,C),
     writeq(H,entry(A,B,C)),write(H,'.'),nl(H),
     fail.
out_entry(_).


all_to_file(File) :-
```

```
     concat(File,$.ari$,Filename),
     create(H,Filename),
     out_output(H),
     out_format(H),
     close(H).

out_format(H) :-
     formated(A,B,C),
     writeq(H,formated(A,B,C)),write(H,'.'),nl(H),
     fail.
out_format(_).

out_output(H) :-
     outputted(A,B),
     writeq(H,outputted(A,B)),write(H,'.'),nl(H),
     fail.
out_output(_).

sort_output :-
     bagof(Y,Y^sort_key(Y),List),
     keysort(List,SortedList),
     asserta(sort_list(SortedList)).
```

Appendix I - Explaination Facility

```
/*********************************************************/
/*            Expert System in Typography                */
/*             by David B. Fisher                        */
/*                                                       */
/*        Module:        Explain                         */
/*        Function: Explaaination Facility               */
/*                                                       */
/* Help is requested by entering a '?' with any question. */
/* When requested the appropriate sections from the expert*/
/* text are displayed. Only a few sections have been in-  */
/* corpated into the prototype to reduce the typing       */
/* required and avoid possible copyright infringement.    */
/* Those listed are copied from "The Chicago Manual of    */
/* Style", 13th ed., The University Press of Chicago:     */
/* Chicago, 1982.                                         */
/*********************************************************/

/***
     NEED_HELP - Did the user request help, if so print it
and fail.
***/

need_help([?,@],CMS) :-
     write_cms(CMS),
     tell_user(nl),!,
     fail.
need_help(63,CMS) :-
     write_cms(CMS),
     tell_user(nl),!,
     fail.
need_help(C,_) :-
     C \== [?,@].

write_cms([]).
write_cms([CMS|More]) :-
     tell_user($Chicago Manual of Sytle, section
$),tell_user(CMS),tell_user(':'),tell_user(nl),tell_user(nl)
,
     cms(CMS,Text),
     tell_user(Text),tell_user(nl),
     fail.
write_cms([CMS|More]) :-
     not(cms(CMS,_)),
     tell_user($No help text available...See Chicago Manual
of Sytle, section $),
     tell_user(CMS),tell_user('.'),tell_user(nl),
     fail.
write_cms([_|CMS]) :-
     write_cms(CMS).

cms(16.2,$In the main, individual entries in all scholarly
```

```
reference lists and bib-$).
cms(16.2,$liographies include similar information about a
published work.  For a$).
cms(16.2,$book, these facts are$).
cms(16.2,$    Name of the author or authors, the editors, or
the institution$).
cms(16.2,$        responsible for the writing of the book$).
cms(16.2,$    Full title of the book, including the
subtitle, if any$).
cms(16.2,$    Title of the series, if any, and the volume or
number in the series$).
cms(16.2,$    Volume number or total number of volumes of a
multivolume work$).
cms(16.2,$    Edition, if not the original$).
cms(16.2,$    City of publication$).
cms(16.2,$    Publisher's Name$).
cms(16.2,$    Date of publication$).
cms(16.3,$For an article in a periodical, the facts given
are$).
cms(16.3,$    Name of the author$).
cms(16.3,$    Title of the article$).
cms(16.3,$    Name of the periodical$).
cms(16.3,$    Volume Number (sometimes the issue number)$).
cms(16.3,$    Date$).
cms(16.3,$    Pages occupied by the article$).
```

Appendix J - Misc. System Utilities

```
/*********************************************************/
/*            Expert System in Typography                */
/*             by David B. Fisher                        */
/*                                                       */
/*        Module:        Util                            */
/*        Function: Misc. Utility Functions              */
/*                                                       */
/*********************************************************/

/***
     INITIALIZE the expert system...
***/

initialize :-
     set_register(current_entry,0),
     abolish(entry/3),
     abolish(style/2),
     abolish(formated/3),
     abolish(outputted/2),
     abolish(key_sort/1),
     abolish(class_done/1),
     abolish(no_of_entries/1),
     abolish(fp/1),
     turn_capture(on).

/***
     ESTABLISH - Add a fact to the database...
***/

establish(Functor,Arg_List) :-
     Fact =.. [Functor | Arg_List],
     assertz(Fact).

/*********************************/
/*   Register functions  */
/*********************************/

/***
     SET_REGISTER
***/
/* Retract all existing terms for this register in the data
base and
     create a new one with the new value.  Registers take
the form:

          register(name,value).

*/
set_register(R,_) :-
     register(R,N),
     retract(register(R,N)),
     fail.
```

```
set_register(R,N) :-
     asserta(register(R,N)).

/***
     INC_REGISTER
***/
inc_register(R) :-
     register(R,N),
     retract(register(R,N)),
     NN is N + 1,
     asserta(register(R,NN)).


/***
     DEC_REGISTER
***/
dec_register(R) :-
     register(R,N),
     retract(register(R,N)),
     NN is N - 1,
     asserta(register(R,NN)).


/* Write out a list of items giving each a number until the
list is empty */
     /*
     Example:
          the call: write_items(1,[children,adult,general]).
          yeilds:
                    1. children
                    2. adult
                    3. general
     */
write_items(_,[]) :-     /* Special Case of no Multiple
Choice answers available */
     tell_user(nl),tell_user(nl).
write_items(N,[X|Y]):-
     tell_user(N),tell_user('.
'),tell_user(X),tell_user(nl),
     N1 is N + 1,
     write_items(N1,Y).

/***
     List Membership utilities
***/

member(X,[X|_]).
member(X,[_|Y]) :- member(X,Y).

member_number(1,[X|_],X).
member_number(N,[_|Y],X) :-
     Next is N -1,
```

```
     member_number(Next,Y,X).

member_return(X,[X|_],X).
member_return(X,[_|Y],Z) :- member_return(X,Y,Z).

append([],L,L).
append([X|L1],L2,[X|L3]) :- append(L1,L2,L3).

break.          /* for debugging */

/* Print a Space */
sp :- tell_user(' ').

/***
     Signal the completion of a class...
***/

are_we_done(output) :-
     register(relative_entry,E),
     no_of_entries(E),
     assertz(class_done(output)).
are_we_done(Class) :-
     register(current_entry,E),
     no_of_entries(E),
     assertz(class_done(Class)).
are_we_done(_).

/***
     Capture Facility...
     Whenever comunication is made with the user it is
displayed
     on the screen, and if capture is on, it is also put in
a file.
***/

tell_user(nl):-
     nl,
     capture(nl),!.
capture(nl) :-
     capture_is(off).
capture(nl) :-
     capture_is(on),
     capture_fp(H),
     nl(H).
tell_user(Text):-
     write(Text),
     capture(Text),!.
capture(Text) :-
     capture_is(off).
capture(Text) :-
     capture_is(on),
     capture_fp(H),
```

```
     write(H,Text).

turn_capture(on) :-
     abolish(capture_is/1),
     abolish(capture_fp/1),
     asserta(capture_is(on)),
     create(H,'capture'),
     asserta(capture_fp(H)).

turn_capture(off) :-
     capture_is(on),
     capture_fp(H),
     close(H),
     abolish(capture_is/1),
     abolish(capture_fp/1),
     asserta(capture_is(off)).
turn_capture(_).

c_capture(@):-
     capture(nl).
c_capture(N):-
     name(L,[N]),
     capture(L).
```

```
/*****************************************************************/
/*            Expert System in Typography                        */
/*             by David B. Fisher                                */
/*                                                               */
/*        Module:        READSENT                                */
/*        Function: Read A Sentence                              */
/*                                                               */
/*    from "Programming in Prolog"                               */
/*                    by W.F. Clocksin and C.S.Mellish,          */
/*      page 104.  With some modifications.                      */
/*                                                               */
/*Returns as its argument a list of atoms built from strings of  */
/*non-blank characters read from standard input. Upper case letters */
/*are mapped to lower case and "words" may contain "-", "_", "'" or */
/*digits.  Punctuation to separate words includes ";", ":" and ",". */
/*The end of a sentence is a carriage return which is turned into */
/*a '@' internally.                                              */
/*                                                               */
/*****************************************************************/

     getsentence(S) :-
          read_sent(S),!.

     read_sent([First|Rest]) :-
          get00(C),           /* Priming read              */
          c_capture(C),
          read_word(C,First,NextC),     /* Read first word  */
          rest_sent(First,NextC,Rest).  /* Read rest of words*/

                        /* Given a word and the character after
                           it, read the rest of the sentence */
     rest_sent(First,_,[]) :-
          last_word(First), !.        /* Handle end of sentence*/
     rest_sent(First,C,[NextW|Rest]) :-
          read_word(C,NextW,C1),      /* Read next word     */
          rest_sent(NextW,C1,Rest).   /* Read rest of words */

     /* Given an initial character "C", read a single word "Word"
     and remember what character "NextC" came after the word. */

     read_word('@',_,'@'). /* This char. terminates a sentence */
     read_word(C,Word,NextC) :-    /* Handle punctuation    */
          single_char(C), !,
          name(Word,[C]),
          get00(NextC),
          c_capture(NextC).
     read_word(C,Word,NextC) :-         /* Handle words     */
          in_word(C,MapC), !,
          get00(C1),
          c_capture(C1),
          rest_word(C1,Rest,NextC),
```

```
     name(Word,[MapC|Rest]).
read_word(C,Word,NextC) :-          /* Eat white space  */
     get00(C1),
     c_capture(C1),
     read_word(C1,Word,NextC).

rest_word(C,[MapC|Rest],NextC) :-  /* Handle rest of chars*/
     in_word(C,MapC), !,
     get00(C1),
     c_capture(C1),
     rest_word(C1,Rest,NextC).
rest_word(C,[],C).                 /* Handle last char in word */


               /* These characters form words on their own*/
single_char(44).          /* , */
single_char(59).          /* ; */
single_char(58).          /* : */
single_char(63).          /* ? */
single_char(33).          /* ! */
single_char(46).          /* . */
single_char('@').

in_word(C,C) :-      /*These chars appear within a word */
     C>="a", C=<"z".
in_word(C,C) :-      /*These chars appear within a word */
     C>="A", C=<"Z".
in_word(C,C) :-
     C>="0", C=<"9".
in_word(39,39).                 /* ' */
in_word(45,45).                 /* - */
in_word(95,95).                 /* _ */

                    /* These words terminate a sentence */
last_word('@').

get00(A) :-
     get0(C),
     trans(C,A).
trans(13,'@').
trans(C,C).
```

Appendix K - Misc. Application Utilities

```
/*************************************************************/
/*            Expert System in Typography                    */
/*             by David B. Fisher                            */
/*                                                           */
/*        Module:        RuleUtil                            */
/*        Function: Utilities for the Rule Processes         */
/*                                                           */
/*************************************************************/

/***
     Class initializers...
***/
initialize(input) :-
     cls.
initialize(format).
initialize(output) :-
     sort_output,
     determine(style,_,Style),
     reorg_subclasses(Style),
     output(hard,return),
     output(hard,page),
     output(header,on),
     output(center,on),
     output('Bibliography'),
     output(center,off),
     output(header,off),
     output(hard,return),
     output(paragraph,hanging_indent).
reorg_subclasses(b) :-
     abolish(subclass/1),
     assertz(subclass(type)),
     assertz(subclass(author)),
     assertz(subclass(date)),
     assertz(subclass(title)),
     assertz(subclass(sub_title)),
     assertz(subclass(series_vol_number)),
     assertz(subclass(edition)),
     assertz(subclass(city)),
     assertz(subclass(publisher)),
     assertz(subclass(periodical)),
     assertz(subclass(pages)).
reorg_subclasses(_).

/****
     Generic Routine used by rulebase format code
****/
for_each(Each,Do,Ans) :-
     do_for_each(Each,Do,[],Ans).

do_for_each([[@]],_,Ans,Ans).
do_for_each([First | Rest],Do,AnsListIn,AnsList) :-
     Goal =.. [Do,First,Ans],
```

```
     call(Goal),
     append(AnsListIn,[Ans],AnsSofar),
     do_for_each(Rest,Do,AnsSofar,AnsList).

for_each_except_last(List,Do_Most,Do_Last,Ans) :-
     do_for_each_except_last(List,Do_Most,Do_Last,[],Ans).

do_for_each_except_last([],_,_,AnsIn,Ans_list) :-
     append(AnsIn,[@],Ans_list).
do_for_each_except_last([One_Left],Do_Most,Do_Last,AnsIn,Ans
_list):-
     Goal =.. [Do_Last,One_Left,Ans],
     call(Goal),
     append(AnsIn,[Ans],AnsSofar),

do_for_each_except_last([],Do_Most,Do_Last,AnsSofar,Ans_list
),!.
do_for_each_except_last([First|Rest],Do_Most,Do_Last,AnsIn,A
ns_list):-
     Goal =.. [Do_Most,First,Ans],
     call(Goal),
     append(AnsIn,[Ans],AnsSofar),

do_for_each_except_last(Rest,Do_Most,Do_Last,AnsSofar,Ans_li
st),!.


are_we_done(output) :-
     register(relative_entry,E),
     no_of_entries(E),
     assertz(class_done(output)).
are_we_done(Class) :-
     register(current_entry,E),
     no_of_entries(E),
     assertz(class_done(Class)).
are_we_done(_).
```

Appendix L - Bibliography In Style B

Bibliography

Biles, A., Cost, F. Johnson, G., and Reek, K. 1987. "Using Expert Systems in Typographic Design." *IEEE Transactions on Professional Communication*. (June): 102-111.

Farkas, D. K., Haselkorn, M., and Ramsey, J. 1987. "Desktop Publishing: The Author as Compositor." *IEEE Transactions on Professional Communication*. (June): 101.

Hiss, L. S. 1987. "A Frame Virtual Machine in C-Prolog." Master's Thesis, Rochester Institute of Technology.

Lan M.S., Pannos, R.M., and Balban, M.S. 1987. "A Knowledge-based Approach to Printing Press Configuration." *IEEE Proceedings of the Third Conference on Artificial Intelligence Applications*. (February 23-27): 38-43.

Lawson, A. 1971. *Printing Types: An Introduction*. Boston: Beacon Press.

Lee, M. 1965. *Bookmaking: The Illustrated Guide to Design and Production*. New York: R.R. Bowker.

Paterson, D.G. and Tinker, M.A. 1940. *How to Make Type Readable: A Manual for Typographers, Printers and Advertisers*. New York: Harper and Brothers Publishers.

Petrie, Jr., C. J. 1985. *Extraction of Expert System Rules from Text*. Research Paper, MCC Technical Report Number AI-017-85, Austin, Texas.

Plass, M. F. 1981. "Optimal Pagination Techniques for Automatic Type Setting Systems." Doctoral Thesis, Standford University.

Romano, F. J. 1984. *The TypEncyclopedia*. New York: R.R. Bowker Company.

Internation Standards Organization. 1986. *Standard Generalized Markup Language Manuals*. ISO-8879.

Strandgerg, C., Abramovich, I., Mitchel, D., and Prill, K. 1985. "PAGE-1: A Troubleshooting Aid for Nonimpact Page Printing Systems. *IEEE Proceedings of the Second Conference on Artificial Intelligence Applications*. (December 11-13): 68-74.

Takeda, K., Suzuki, C., Nishino, T., and Fujisaki, T. 1988. "CRITAC an experimental system for Japanese text proofreading." *IBM Journal of Research and Development*. (March): 201-210.

University of Chicago. 1982. *The Chicago Manual of Style*. 13th ed. Chicago: The University of Chicago Press.

Warde, B. 1956. "The Crystal Goblet or Printing Should be Invisible." in *The Crystal Goblet: Sixteen Essays on Typography*. New York.

Waterman, D.A. 1986. *A Guide to Expert Systems*. Reading, MA: Addison-Wesley.

Zachrisson, B. 1965. *Studies in the Legibility of Printed Text*. Stockholm: Almgvist and Wiksell.

References

Biles, A., Cost, F. Johnson, G., and Reek, K. "Using Expert Systems in Typographic Design." *IEEE Transactions on Professional Communication*. (June 1987): 102-111.

Farkas, D. K., Haselkorn, M., and Ramsey, J. "Desktop Publishing: The Author as Compositor." *IEEE Transactions on Professional Communication*. (June 1987): 101.

Hiss, L. S. "A Frame Virtual Machine in C-Prolog." Master's Thesis, Rochester Institute of Technology, 1987.

Lan M.S., Pannos, R.M., and Balban, M.S. "A Knowledge-based Approach to Printing Press Configuration." *IEEE Proceedings of the Third Conference on Artificial Intelligence Applications*. (February 23-27 1987): 38-43.

Lawson, A. *Printing Types: An Introduction*. Boston: Beacon Press, 1971.

Lee, M. *Bookmaking: The Illustrated Guide to Design and Production*. New York: R.R. Bowker, 1965.

Paterson, D.G. and Tinker, M.A. *How to Make Type Readable: A Manual for Typographers, Printers and Advertisers*. New York: Harper and Brothers Publishers, 1940.

Petrie, Jr., C. J. *Extraction of Expert System Rules from Text*. Research Paper, MCC Technical Report Number AI-017-85, Austin, Texas, 1985.

Plass, M. F. "Optimal Pagination Techniques for Automatic Type Setting Systems." Doctoral Thesis, Standford University, 1981.

Romano, F. J. *The TypEncyclopedia*. New York: R.R. Bowker Company, 1984.

Internation Standards Organization. *Standard Generalized Markup Language Manuals*. ISO-8879. 1986.

Strandgerg, C., Abramovich, I., Mitchel, D., and Prill, K. "PAGE-1: A Troubleshooting Aid for Nonimpact Page Printing Systems. *IEEE Proceedings of the Second Conference on Artificial Intelligence Applications*. (December 11-13 1985): 68-74.

Takeda, K., Suzuki, C., Nishino, T., and Fujisaki, T. "CRITAC an experimental system for Japanese text proofreading." *IBM Journal of Research and Development*. (March 1988): 201-210.

University of Chicago. *The Chicago Manual of Style*. 13th ed. Chicago: The University of Chicago Press, 1982.

Warde, B. "The Crystal Goblet or Printing Should be Invisible." in *The Crystal Goblet: Sixteen Essays on Typography*. New York, 1956.

Waterman, D.A., *A Guide to Expert Systems*. Reading, MA: Addison-Wesley, 1986.

Zachrisson, B. *Studies in the Legibility of Printed Text*. Stockholm: Almgvist and Wiksell, 1965.