12-1991

Applications of Artificial Intelligence/Expert Systems to Sociological Concepts

Darryl Strode
Western Kentucky University

Strode,

Darryl A.

1991

APPLICATIONS OF ARTIFICIAL INTELLIGENCE/EXPERT SYSTEMS TO SOCIOLOGICAL CONCEPTS

A Thesis

Presented to

the Faculty of the Department of Sociology

Western Kentucky University

Bowling Green, Kentucky

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts

by

Darryl A. Strode

December 1991

AUTHORIZATION FOR USE OF THESIS

Permission is hereby

- [✓] granted to the Western Kentucky University Library to make, or allow to be made photocopies, microfilm or other copies of this thesis for appropriate research for scholarly purposes.

- [ ] reserved to the author for the making of any copies of this thesis except for brief sections for research or scholarly purposes.

Signed: Darryl A. [illegible]

Date: December 18, 1991

Please place an "X" in the appropriate box.

This form will be filed with the original of the thesis and will control future use of the thesis.

APPLICATIONS OF ARTIFICIAL INTELLIGENCE/EXPERT SYSTEMS
TO SOCIOLOGICAL CONCEPTS

Date Recommended December 5, 1991

Paul R. Wozniak
Director of Thesis

Joan Krenzin

Stephen B. Groce

Date Approved December 17, 1991

Elmer Gray
Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to thank my committee members for their guidance and helpful suggestions in this project. I would also like to thank Margaret Cline of Academic Computing and Research Services for sharing her time and knowledge of artificial intelligence.

TABLE OF CONTENTS

LIST OF FIGURES

APPLICATIONS OF ARTIFICIAL INTELLIGENCE/EXPERT SYSTEMS TO SOCIOLOGICAL CONCEPTS

Darryl A. Strode December 16, 1991 87 Pages

Directed by: Paul Wozniak, Joan Krenzin, and Steve Groce

Department of Sociology Western Kentucky University

Social scientists are using AI (Artificial Intelligence) and Expert Systems in a variety of areas. This thesis explores some applications of "intelligent" programs in the field of sociology. These applications include modeling social networks using logic programming and simulating sociological decision-making using small knowledge-based systems. PROLOG and VP-Expert are used as development tools.

In this thesis, PROLOG was used for modeling triads. In this situation, PROLOG was used to define the "actors," the relations between them, and rules governing the existence of transitive and intransitive triads. Users are able to query the knowledge-base about various aspects of this situation. In addition, VP-Expert was used to construct some small knowledge-based systems. These systems deal with determining whether a triad is transitive or intransitive, classifying adaptation to anomie (normative confusion), determining whether a type I or type II error has been made in the hypothesis testing process, determining a variable's level of measurement, and classifying individuals on the basis of socio-economic status.

CHAPTER I

INTRODUCTION

The application of artificial intelligence to sociology is a recent phenomenon. The purpose of this research is to explore applications of artificial intelligence to sociological concepts. This has been accomplished by constructing programs using logic programming techniques to model social interaction. Some small knowledge-based systems were also designed that emulate sociological decision making. The first section of this chapter deals with the history of artificial intelligence while the second examines current applications of the technology.

Artificial Intelligence (AI) is a branch of computer science that deals with the creation of intelligent machines. It attempts to capture in program form those human thought processes involved in the solution of complex problems. Areas of AI research include robotics, natural language processing (NLP), computer vision, and expert systems.

A Brief History of AI

During World War II, American and British scientists worked to develop a machine that could handle numerical computations and codebreaking operations. One of the more successful efforts was undertaken by the British to break the German "Enigma" code. This effort was called Project Ultra and utilized the talents of Alan Turing, a British mathematician (Mishkoff 1986, pp. 2-4). There was disagreement among the scientists over what type of operators such a machine should use (Harmon and King 1985, p. 2). Turing advocated the use of logical operators such as "AND," "OR," and "NOT". American scientists decided to use arithmetic operators as a less expensive option. British scientists followed suit and over time, a proliferation of complex calculating machines using numerical operators such as "+," "-," and ">" emerged. We refer to these machines today as computers.

A few scientists continued research toward developing computers that could manipulate non-numerical symbols. Scientists with this interest, along with psychologists interested in programs that simulated human behavior, met in Hanover, New Hampshire, in the summer of 1956. This gathering was known as the Dartmouth Conference, named after the local college (Mishkoff 1986). At this conference, the branch of computer science known as artificial intelligence was conceived. The name was suggested by Dr. John McCarthy, a mathematics professor and the conference's primary organizer.

The question of whether machines could possess intelligence was first addressed by Alan Turing in his work "Computing Machinery and Intelligence" (Turing 1950). He devised a situation called "The Imitation Game." There are two stages to this game. In the first stage, the game is played with a man, a woman, and an interrogator. Physically separated from the other two, the interrogator attempts to determine which is the woman. Communication between the three is conducted via written communication, preferably typed on a typewriter, Teletype or similar device. This makes it harder for the interrogator to distinguish between the two. During this time, the man tries to fool the interrogator into believing that he is the woman while the woman tries to convince the interrogator that she is who she claims to be. The second stage of Turing's Imitation Game involves replacing the man with a computer and resuming the game. If the computer fools the interrogator into thinking he is conversing with a human, the computer is presumably displaying intelligence. This has become known as the Turing Test, in honor of Turing, and has been used as a basis for determining whether or not a machine is displaying intelligence (Hofstadter 1979, pp. 594-597).

One program that has fooled many into believing that they were actually conversing with an intelligent computer is ELIZA (Mishkoff 1986). This program was created by Joseph Weizenbaum at the Massachusetts Institute of Technology. Users can converse with the program as if they were conversing with a Rogerian psychiatrist. Based on what the user types in, the program searches through the input string for a recognizable keyword. Upon finding this keyword, the program searches through its list of "canned" responses for a match. It then prints a response often interspersed with words the user previously typed in.

Applications of AI/Expert Systems

Programs created as a result of AI research have been employed in fields such as medicine, chemistry, geology, math, and the military. These programs are typically in the form of expert systems. Expert systems are a product of AI research begun in the early 1950s. They are programs that emulate the reasoning process of a human expert solving problems in a specific area. Expert systems possess knowledge relating to some problem area and rules that govern the application of that knowledge. Through the process of inference, they reach conclusions based on the rules the system possesses.

Medicine

MYCIN, an expert system for diagnosing infectious blood diseases, was developed by Edward Shortliffe at Stanford University (Firebaugh 1988). In addition to providing the physician with a diagnosis, it can also prescribe the appropriate medication. MYCIN's knowledge base contains between 400 and 500 rules. Responsibility for MYCIN's decisions is the physician's; the program acts only in an advisory capacity. MYCIN spawned the first expert system shell EMYCIN (for "Empty MYCIN"). EMYCIN retains MYCIN's inference engine but not its knowledge base.

Chemistry

In the area of chemical analysis, an expert system called DENDRAL was developed (Hayes-Roth et al. 1983). It was created by Edward Feigenbaum at Stanford University.

DENDRAL can identify the structure of an unknown parent compound by analyzing mass spectrographic, nuclear magnetic resonance, and chemical experiment data. Its performance is judged superior to that of humans at its task.

Geology

PROSPECTOR, a geological expert system, was designed for interpreting soil and geological deposit data (Hayes-Roth et al. 1983). It was created by Richard O. Duda in 1978. PROSPECTOR predicted the location of a molybdenum deposit worth $100 million. Unlike MYCIN and DENDRAL which utilize rules, PROSPECTOR's knowledge is represented in semantic nets (a method of representing objects and the relations between them). It performs at the level of an expert geologist.

Mathematics

Mathematical researchers and physicists regularly use MACSYMA, a mathematics expert system (Hayes-Roth et al. 1983). Its knowledge base contains hundreds of rules from experts in applied mathematics. MACSYMA can perform differential and integral calculus symbolically as well as simplifying symbolic expressions. In fact, it can perform over 600 distinct mathematical operations (Firebaugh 1988). MACSYMA's performance exceeds that of most human experts. It was developed at MIT in 1968 by Carl Engleman, William Martin, and Joel Moses.

Military

The Department of Defense is interested in using AI technology for military applications. The Defense Advanced

Research Projects Agency (DARPA) started the Strategic Computing Program (SCP) in 1983 for this task (McGraw and McGraw 1985). Texas Instruments Corporation was contracted to design an expert system for the SCP's Battle Management Program. It is called FRESH (Force Requirements Expert System). This system is used in identifying force requirements and as an aid for the Navy in monitoring fleet readiness. Texas Instruments has also created an expert system for F-16 pilots called EPES (Emergency Procedures Expert System). It was developed as part of SCP's Pilot's Associate Program. EPES was designed to furnish pilots with advice during in-flight emergency situations.

The previous examples show how expert systems are used in the physical sciences and the military. Social scientists are also starting to use expert systems and AI technology in fields such as anthropology (e.g., Benfer 1989; Furbee 1989; Guillet 1989; Kippen and Bel 1989; Read and Behrens 1989), political science (e.g., Alker and Christensen 1972; Alker et al. 1980; Banerjee 1986; Schrodt 1988; Thorson and Sylvan 1982), and social work (e.g., Winfield et al. 1986).

Benfer (1989) addresses the problem of individual differences among experts in creating knowledge-based systems. He states that these differences can be critical in understanding variations in observed behavior. Individual differences may assist in determining a system's outcome. Knowledge-based systems for land management and selection among health care choices are presented.

Guillet (1989) discusses a rule-based knowledge system used for native soil management. Information for the system was obtained from Lari farmers in the Colca Valley of Peru by means of a survey. The survey included questions about soil management practices, soil type, altitude, area, and the availability of water. Using open-ended questions, further information was obtained about the farmers' rationales for their management practices. The combined knowledge was then formulated into a series of rules for determining what crops should be planted.

Furbee (1989), studying the same region, describes an expert system for classifying soils called FAI/SOILS. The expertise for this system was gleaned from ten Lari farmers in the Colca Valley of Peru. FAI/SOILS also makes use of rules for determining soil type.

Kippen and Bel describe an expert system called the Bol Processor. This system analyzes the tabla (two piece drum set) drumming of North Indian musicians. Tabla music is represented by bols. Bols are a system of syllables which when strung together form phrases that are "spoken in the rhythm of the piece they represent" (1989, p. 132). Hence the name "Bol Processor." The purpose of Kippen and Bel's research was to determine whether expert systems can help separate folk models from analytical models (the problem of ethnographic description). They concluded that expert systems cannot help in the separation of folk from analytical statements due to shortcomings in the knowledge acquisition process and suggest a need for automated learning procedures.

Read and Behrens (1989) created an expert system called BATES (Bisayan Address Term Expert System). This expert system was created by translating and representing the cultural knowledge of Philippine Bisayan speakers. BATES is a rule-based expert system consisting of 43 rules. Read and Behrens' goal was to explore ways in which expert systems could be used to express the knowledge of folk systems.

One of the first articles on the application of artificial intelligence to the social sciences was by Alker and Christensen (1972). They created a computer model for simulating United Nations peace-making activities. This model, called PRECEDENT, was designed to choose among precedent logics. Artificial intelligence was seen as a progression from causal modeling. Alker et al. (1980) also used precedent logics for the resolution of insecurity dilemmas.

Banerjee (1986) used artificial intelligence for modeling social structures. His program illustrates how social structures reproduce using a model of social cognition and action based on Jean Piaget's cognitive development theory in psychology. Using PROLOG, Banerjee modeled Theda Skocpol's analysis of China's sociopolitical structure in the 1930s and O'Donnell's version of Latin American bureaucratic-authoritarian structure during the 1960s.

Schrodt (1988) discusses the application of AI to international relations (IR), specifically in modeling social phenomena. He lists three major research focuses in IR/AI: rule-based systems, historical precedent-based systems, and natural language systems. Schrodt states that AI has stimulated the development of "new, testable" theories involving international behavior. IR/AI models (also called formal cognitive models) developed in part because of the limitations of statistical models. Schrodt believes that since IR/AI models share a common basis in human psychology and human decision making under adverse conditions, they approximate traditional IR theories more so than do statistical models.

Thorson and Sylvan (1982) present an AI program called JFK/CBA that simulates the decisions made by President John F. Kennedy during the Cuban Missile Crisis. The program contains knowledge of the crisis as well as a set of rules depicting Kennedy's beliefs about the Soviet Union, diplomacy, and the military. JFK/CBA uses production rules of the form CONDITION --> ACTION to represent Kennedy's beliefs. This means if a certain condition is met some action will be taken. The program makes use of "counterfactuals," meaning the user can change historical facts to determine alternative outcomes.

Another social science field where AI/Expert Systems applications are being explored is social work. Winfield et al. (1986) discuss a prototype expert system for enuresis (bedwetting). They believe that expert systems such as this one can assist social workers by training them in correct procedures and assisting them in complex decision making.

The previous examples show how AI/Expert Systems are used in areas such as anthropology, political science, and social work. The next chapter explores some applications of AI/Expert systems in sociology.

CHAPTER II

ARTIFICIAL INTELLIGENCE AND SOCIOLOGY: A REVIEW OF THE LITERATURE

Social scientists have traditionally used computers for quantitative analysis utilizing software such as SPSS (Statistical Package for the Social Sciences), student exercises, and for word processing tasks. Some are now exploring applications of AI to sociology. These applications include theory construction, qualitative data analysis, and hypothesis formulation.

A leading advocate of using AI in sociology, Dr. Edward Brent of the University of Missouri at Columbia, has written extensively on qualitative computing methods and applying AI programming techniques to the field of sociology (See Brent 1984, 1985, 1986, 1988a, 1988b, 1989a, 1989b; Brent and Anderson 1990; Brent et al. 1989). Brent (1986) examines the application of knowledge-based systems to the social sciences, particularly in modeling Erving Goffman's dramaturgical perspective in an AI programming language. He presents knowledge-based systems as an alternative method or "qualitative formalism" for solving problems that mathematical techniques cannot handle. Brent constructed a program called N-ACT written in the PROLOG programming language. This program consists of passages from Goffman's _The Presentation of Self in Everyday Life_ (Goffman 1959) translated into equivalent PROLOG statements. The program illustrates how artificial intelligence programming methods can assist in the construction of sociological theory. The user of N-ACT alters elements of the program's knowledge base to determine what effect that has on the program's conclusions. Brent (1989a) has dramatically revised this program and renamed it ERVING (after Erving Goffman). The revised program teaches students to reason sociologically from Goffman's dramaturgical perspective. Brent et al. (1989) have also designed a program to assist in determining sample size called EX-SAMPLE, and one that aids in the selection of statistical analyses for research applications called Statistical Navigator (1989b). Both of these programs also utilize AI programming techniques and are recognized as expert systems.

Like Brent, Gilbert and Heath (1985) are also interested in AI's contribution to sociological theory construction. Gilbert and Heath suggest that artificial intelligence possibly has both methodological and substantive contributions to make to sociology. They support this claim by pointing out the fact that when a computer program is written, ideas must be fully developed and difficulties must be completely resolved (a lesson for sociological theorists). If these aspects of construction are not addressed, neither the computer program nor the sociological theory will work correctly. Using a term popular with computer scientists, they would both "crash." Gilbert and Heath suggest that sociologists build and execute computer programs using AI techniques that can express the ideas of a number of actors in a social situation to determine if the program performs up to the theoretical expectations of the researcher. If so, the program would provide support for the researcher's theory.

Sylvan and Glassner (1985) argue that rationalism is a better methodology for understanding the social world than empiricism, which they claim "seeks to manipulate empirical phenomena through experimental and other methods." Rationalist theories, however, explain phenomena through logical possibilities instead of statistical probabilities, the focus being on combination instead of causality. Sylvan and Glassner developed a formal model of a rationalist theory based on the work of Georg Simmel. LOGLISP, a logic programming language, was used to create this model. LOGLISP is an acronym for LOGic and LISP. LISP stands for LISt Processing and is the dominant language used in the United States for artificial intelligence programming. The goal of Sylvan and Glassner's project was to use this program to generate hypotheses through its logical implications, primarily the deduction of Simmel's (1955) conflict-cohesion hypothesis itself. Although the program was unable to generate the general form of the hypothesis, it did produce more specific versions of it.

Like Sylvan and Glassner, Hinze (1987) argues that qualitative computing approaches such as artificial intelligence are more applicable to the study of human behavior, norms in this case, than of formal statistics. He believes that AI provides a means for studying organizational structures, both formal and informal, as well as the concepts of role strain and role ambiguity.

The marriage of AI and sociological theory is not all roses. According to Wood (1986), designing expert systems for areas such as medicine and geology is relatively straightforward. Creating expert systems for social situations, however, is inherently difficult. Medicine and geology are physical science fields based on very precise knowledge. For the most part, sociological theories lack the predictive power of physical science theories. This is so, Wood states, because knowledge of why things happen in social situations as well as what will occur in a given social situation is less precise than that of other scientific theories. As a result, sociological knowledge is difficult to formalize into a system of rules.

Gerson (1985) also holds a pessimistic view of artificial intelligence. She believes that there is a lot of hype surrounding AI's possible contribution to qualitative sociological research. Gerson says that expert systems can perform limited tasks but cannot meaningfully interpret things. She attributes this shortcoming to the current state of the art in AI technology. Gerson believes it will be some time before AI evolves beyond the stage of "laboratory curiosity" for sociological researchers.

Another area of application for AI technology is qualitative data analysis. Drass (1980) describes a program that can be used as an aid in the analysis of qualitative field data. The program is called LISPQUAL. The program consists of LISP (an AI programming language) functions to assist in the qualitative research process. Drass divides analysis procedures into an interpretive phase and a mechanical phase. The interpretive phase involves deriving meaning from research data. The mechanical phase deals with all procedures that are noninterpretive in nature (e.g. scanning data for the presence or absence of a symbol or code). LISPQUAL was designed to handle operations in the mechanical phase. It does not perform any interpretive operations. These are left for the researcher.

Drass' program shows how AI can assist the data analysis process without affecting the nature of the process itself. It does this by handling the routine, lengthy, mechanical operations involved in a faster, more accurate manner. This allows the researcher to spend more time on interpreting the data instead of coding it.

Along the same lines, Carley (1988) created an expert system called SKI for use in coding verbal protocols. There are two steps to this procedure. Verbal data are first coded using a computer procedure called CODEF. The resulting knowledge base is then processed by the SKI expert system. SKI contains knowledge about the sociocultural environment related to a particular protocol. Using its inference engine ADDSOC, SKI can make implicit information in the protocol explicit. In doing this, the system can diagnose and correct errors made by a novice coder. The use of this expert system increases the reliability and replicability of coded data.

Like Edward Brent, Garson (1986) is also interested in applying expert systems to the social science research process. He explored the role of inductive expert systems generators in the area of political science. An expert system generator can induce a set of rules from a matrix of data. Garson constructed a system using the 1st Class expert system generator, that could explain variance in legislative satisfaction. His data consisted of factors compiled by Wayne L. Francis. These factors were "thought to determine the variance in the extent to which state legislators are satisfied with legislative outcomes" (Garson 1986, p. 13). Based on the data, Garson's expert system made the following conclusions about legislative outcomes:

1. It is appropriate to have a long time to consider bills and reject many of them.
2. It is inappropriate to have a short time to consider bills and then accept most.
3. If time is short, it is still inappropriate if one is being inundated with bills from the other chamber.

Garson advocates the use of expert systems in formulating exploratory hypotheses. Social scientists could then confirm the hypotheses using traditional methods of analysis.

Taking an entirely different approach, Woolgar (1985) is concerned with the prospects of an association between sociology and AI, but advocates viewing AI itself from a sociological perspective instead of contemplating potential applications of AI to sociology. He notes that in the abundant literature on AI, disciplines such as psychology and philosophy are universally acknowledged while sociology is rarely mentioned. When it is mentioned, Woolgar says it is veiled in terms of the "social" realm as opposed to the "scientific" world. The emphasis is more on the impact of AI on society than on the contribution of sociology to the development of AI.

Woolgar's suggestions include developing a sociology of AI researchers, looking at the products of AI as social constructions, and viewing intelligent machines as the subjects of sociological study. He states that the latter suggestion could utilize standard sociological methods in the analysis. Woolgar asks "Why not a sociology of machines?" given that machines with artificial intelligence resemble humans enough "to be treated as the subjects of sociological inquiry" (1985, p. 568).

The information presented in this chapter shows various applications of artificial intelligence to sociology. The authors presented here used a variety of AI techniques to represent sociological knowledge. One of these techniques, expert systems, will be examined in detail in the next chapter.

CHAPTER III

EXPERT SYSTEMS

An expert system is a program designed to solve problems in a specific domain utilizing artificial intelligence programming techniques. Knowledge for the expert system is obtained from persons who are considered expert in a particular field (Levine et al. 1986, p. 21).

Unlike standard programs that use algorithms to solve problems, expert systems use heuristics (shortcuts or "rules of thumb") in deriving solutions. Using the algorithmic method, a user is guaranteed a correct solution. But heuristics, like human experts, produce satisfactory results in most cases but not all (Waterman 1986, p. 17). According to Waterman, expert systems are good for solving problems that are "typically difficult," "poorly understood," and unsuitable for mathematical or algorithmic solutions. Hayes-Roth et al. (1983, p. 5) go further by identifying categories of problems that expert systems are ideal at solving. These categories are "interpretation, prediction, diagnosis, debugging, design, planning, monitoring, repair, instruction, and control."

Waterman (1986, pp. 12-15) discusses a number of advantages of expert systems over human experts. First of all, knowledge contained in expert systems is permanent. When a human expert dies, his/her knowledge dies also. If the human expert goes for a long period of time without practicing problem solving ability, his/her proficiency will decrease. An expert system's knowledge base remains current regardless of use levels. Another advantage of expert systems is transferability. It is much easier to transfer knowledge by copying a program than to educate a human being to a high level of expertise. Unlike human experts, expert systems make consistent conclusions in identical situations. Human experts are susceptible to emotions, fatigue, and illness which may cause them to make differing or incorrect decisions under similar circumstances. Perhaps the biggest advantage of expert systems over human experts is cost. The expert system is relatively inexpensive in comparison with human experts who often command high salaries for their expertise.

Although there are a number of advantages of expert systems over human experts, Waterman also points out some disadvantages as well. Perhaps the most serious is that these systems lack common sense. As ethnomethodologists have stated, common sense reasoning is something that most of us take for granted but it is essential for our everyday operation in society (see Garfinkel 1967; Schutz 1962). Due to the sheer vastness of this knowledge, its replication in an expert system would be an incredible feat. Expert systems are also unable to formulate new rules or modify their knowledge. In other words, they lack creativity. Unlike human experts, artificial experts cannot experience sensory input. The senses, touch, sight, smell, taste, and sound have an impact on many decisions that human beings make. Finally, human experts can examine everything pertinent to the situation at hand, but computerized experts focus exclusively on the problem itself. They are unable to explore issues that are relevant to, but not a part of, the problem under consideration.

Architecture of an Expert System

Currently, there is no industry standard for the development of expert systems. Some are created using high level AI languages such as LISP or PROLOG. Others are developed using expert system shells. An expert system shell is a program with the inference mechanism included. According to Mishkoff (1986), however, all expert systems share these three essential components: a knowledge base, an inference engine, and a user interface.

Knowledge Base

The heart of an expert system is its knowledge base. The knowledge base is the part of the system that contains facts and rules necessary for solving a particular problem. This knowledge can be categorized into two types: declarative, and procedural. Declarative knowledge is information pertaining to objects, events, and situations. It consists of descriptions of and relations between these items. Procedural knowledge assists the system in determining an appropriate course of action to take toward solving a problem.

Inference Engine

The second component of an expert system is the inference engine. It contains the general problem-solving capabilities of the system. The inference engine is in charge of running the expert system. In fact, it is sometimes referred to as the control system. There are two parts to the inference engine: control, and logic. The inference engine controls the order in which rules are fired and resolves conflicts when multiple rules are applicable. During this process, it continually applies rules to the knowledge base until it reaches a goal state. As it reaches each goal, it records all of the rules it uses or "fires" in solving the problem for future reference.

User Interface

The final component in an expert system is the user interface. The knowledge contained in the expert system is virtually useless unless there is some means of gaining access to it. The user interface provides this access. The user interface is the communications link between the user and the system. It also answers user inquiries concerning the reasoning process of the system.

Knowledge Engineering

Creating a computer system that can produce intelligent results requires examination of the thought processes or mental steps that people use when trying to make decisions or solve problems. The thought processes must be broken down into a series of steps that can be implemented in a computer program. The computer can then use these same steps in solving a particular problem (Levine 1986, p. 3). This process of developing an expert system is known as knowledge engineering.

The person who develops the system is called a knowledge engineer. Knowledge engineers attempt to replicate the behavior of an expert engaged in the solution of a narrowly defined problem. He/she must be able to identify the knowledge that the expert uses in solving a particular problem. In addition to particular facts, the knowledge engineer must also determine what rules of thumb (heuristics) the expert uses in solving a problem as well as the particular inference strategy involved (Harmon and King 1985, p. 5). A prototype system is then constructed that utilizes this knowledge toward solving a problem, much as the human expert would.

The knowledge engineer does not work in a vacuum. The knowledge engineer may or may not be an expert in the subject area that the system is developed for. In most cases, the knowledge engineer works in close consultation with a person known as a domain expert. The domain expert is generally considered an expert in his/her field. This person has the ability to articulate his/her knowledge clearly and is able to provide efficient solutions to problems within his/her area of expertise. The knowledge engineer constructs the expert system by obtaining this knowledge and expertise from the domain expert and entering it into the expert system.

Although this expertise is usually gleaned from experts, it can also be taken from books and other sources on the subject area (Waterman 1986, p. 9). After the necessary knowledge has been accumulated, it must then be organized. In summary, knowledge engineering involves "extracting, articulating, and computerizing the expert's knowledge" (Hayes-Roth et al. 1983, p. 12). The process of knowledge engineering assumes that the performance of an expert lends itself to computerization. The next task the knowledge engineer must perform is deciding how the knowledge should be represented in the expert system.

Knowledge Representation

Before construction of the expert system can take place, the knowledge engineer must decide how the knowledge should be structured in the system. This is known as knowledge representation (Waterman 1986, p. 20). There are a variety of knowledge representation techniques available for the knowledge engineer to choose from. These methods are rules, frames, scripts, and semantic networks.

Rule-Based Systems

Rule-based expert systems are the most popular types of expert systems in use. In these systems, knowledge is represented as a series of "production rules" (Mishkoff 1986, pp. 3-4). The knowledge engineer formalizes the knowledge of the domain expert into a set of "IF-THEN" rules. These rules have the following format:

IF "condition" THEN "action"

If the condition or premise is satisfied by information contained within the knowledge base, the rule becomes subject to being "fired" by the control system. An actual rule looks something like this:

RULE 1

```
If it is raining outside,
Then the ground is wet.
```

This rule seeks to establish whether the ground is wet. If the premise of this rule, that it is raining outside, is satisfied, then the system would conclude that the ground is wet. The system determines whether a particular fact is true either by searching its database for the information or by querying the user of the system.

Rules can also contain multiple conditions joined by either logical "ANDs" or logical "ORs." Consider the following example:

RULE 2

```
If it is raining outside, AND
   going outside is necessary,
Then an umbrella should be taken.
```

In Rule 2, the logical AND means that both conditions of the premise must be true for the conclusion to be true. If the word "OR" were substituted for "AND" in this case, either of the conditions, but not necessarily both, would have to be true for the conclusion to be true.

Unlike traditional programming methods, portions of an expert system can be easily modified. This can be done by adding, deleting, or substituting a rule. Extensive modifications are required to modify a standard program, but changes to an expert system need not affect the structure of the whole program (Levine et al. 1986, p. 4).

Frame-Based Methods

Another method of knowledge representation is the theory of Frames. This theory was conceived by Marvin Minsky in 1975. Minsky defined a frame as "a data structure for representing a stereotypical situation" (1975, p. 212). Knowledge in frames is represented as a network of nodes and relations between the nodes. A frame is composed of levels. The top level of the frame contains information that is always true about the situation. Lower levels of the frame are composed of slots which are filled with specific instances or data. Related frames can be linked together to form a system of frames. Figure 3.1 illustrates a frame of the object "Bicycle." When one thinks of this object, a mental picture composed of attributes listed in Figure 3.1 enters the mind. Each of these attributes occupies its own slot in the frame.

Scripts

Scripts, like Frames, can also represent stereotypical situations. Scripts are a product of research conducted by Roger Schank and Robert Abelson at Yale University. According to Firebaugh (1988, p. 294), scripts are similar to frames but contain additional information such as the expected sequence of events in a given situation and the goals and plans of actors involved. Such stereotypical situations as going to the doctor's office or to a restaurant

Figure 3.1. Frame for the Object "Bicycle" *

Frame:	Bicycle
Parts:	frame, wheels, tires, pedals, seat, handlebars, gears
Composition of frame:	aluminum, steel, composite
Color:	any
Size:	24", 26", 28", 30"
Speed:	single, three, five, ten, twelve, twenty-one
Type:	dirt, racing, mountain, touring
Function:	transportation
Brand:	Cannondale, Trek, Schwinn

---

* Note: This is not an exhaustive list.

can be represented as scripts. In both cases, we expect a certain series of events to transpire. For example, when one goes to a restaurant, one expects certain props to be present, such as tables, chairs, tableware, dishes, menus, and, of course, food. In addition, one would expect to encounter people in roles that lead us to believe we are in a restaurant. These people are the cooks, cashier, waiter/waitress, busboy, and even other customers. We assume that a person who goes to a restaurant is hungry and that the hunger can be relieved by consuming food. We also assume that upon completion of the meal, one will pay for the food. All of these events can be represented in the computer as a series of scripts. Provided with this knowledge of expectations, a computer can make sense out of incomplete information and draw conclusions about this scenario. Figure 3.2 highlights a portion of the famous Restaurant Script by Schank and Abelson (1977, pp. 42-46).

Semantic Networks

Semantic networks are a method of knowledge representation used to illustrate complex relations between objects, concepts, or situations (Barr and Feigenbaum 1981, pp. 180-189). Semantic networks were designed in 1968 by Ross Quillian as psychological models of human associative memory. These networks are composed of two components: nodes and arcs. Nodes are represented graphically as either boxes or circles. The nodes represent objects, concepts, or situations. Arcs (also known as links) represent relations

Figure 3.2. Script of "Restaurant" by Schank & Abelson *

Script: Restaurant (Coffee Shop)

Props: Tables, Chairs, Menus, Food, Money

Roles: Customer, Waiter, Cook, Cashier, Owner

Conditions for Entry: Customer is hungry.
Customer has money.

Results of Entry: Customer has less money.
Owner has more money.
Customer is no longer hungry.
Customer is pleased. (possibly)

Scene 1: Entering the Restaurant

Customer enters restaurant.
Customer looks at tables.
Customer decides where to sit (no "Wait to be seated" sign is present).
Customer walks to chosen table. Customer assumes sitting position.

---

* This is not a complete set of possibilities.

between the nodes. They are depicted graphically as arrows that link the nodes together. Suppose we wanted to represent the simple fact: "All tuna are fish." This can be accomplished by using two nodes -- one labeled "TUNA," the other "FISH." The nodes are then connected by an arc that is labeled "isa" to indicate the relation between them.

$$\begin{tikzcd}
\boxed{\text{TUNA}} \arrow[r, "\text{isa}"] & \boxed{\text{FISH}}
\end{tikzcd}$$

Suppose also that this particular tuna is named Charlie. A third node is added with the label "CHARLIE" as follows:

$$\begin{tikzcd}
\boxed{\text{CHARLIE}} \arrow[r, "\text{isa}"] & \boxed{\text{TUNA}} \arrow[r, "\text{isa}"] & \boxed{\text{FISH}}
\end{tikzcd}$$

At this point we have established two facts:

1. A tuna is a fish.
2. Charlie is a tuna.

From these two facts, a third can be deduced by following the "isa" arcs:

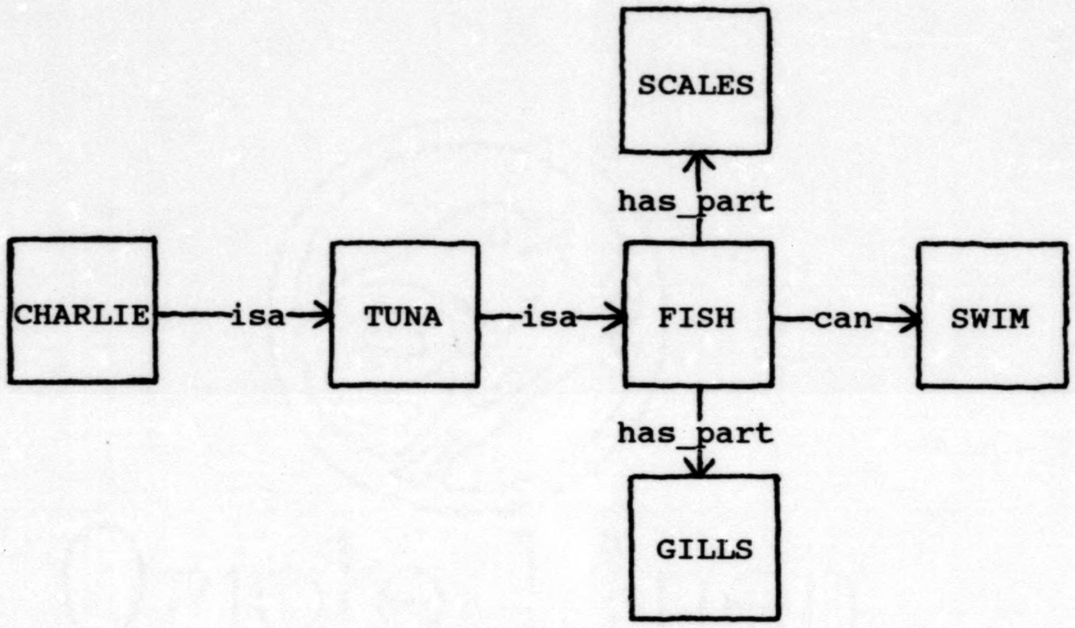
3. Charlie is a fish.

This illustrates a feature of semantic networks known as inheritance. One can go further with this model by adding properties to the object FISH such as:

1. Fish have scales.
2. Fish have gills.
3. Fish can swim.

This is done by using additional nodes to represent the properties and links to specify the relations. A complete model is presented in Figure 3.3. By inheritance, we can

Figure 3.3. Semantic Network Representation



ultimately infer from this model that Charlie is a tuna fish with scales and gills and can swim.

Automated Reasoning

In order for an expert system to reason intelligently, it must be able to infer new facts from information it already possesses. Artificial Intelligence has employed the use of logic in solving problems. By utilizing logic or rules of inference, AI researchers have been able to automate the process of symbolic reasoning in the computer. Logic provides both a means of representing knowledge and a "formalism for extracting the implications of that knowledge" (Firebaugh 1988, p. 131).

Firebaugh (1988, pp. 132-135) discusses three types of logical inference procedures that are used in drawing conclusions. These are deduction, induction, and abduction.

Deduction

Deduction is the process of reasoning from something general to something specific. Consider the following statements:

All sociologists are human.
Emile Durkheim is a sociologist.

From these two statements, a third can be deduced:

Emile Durkheim is a human.

A basic rule of inference utilizing deductive logic is called modus ponens. The syntax of this rule is:

```
P --> Q
P
-------
.·.   Q
```

This states that P implies Q and if P is true, then Q is true. Say P represents the fact that it is raining outside and that Q means one should carry an umbrella. In this example, the fact that its raining outside (P) implies that one should carry an umbrella. Given P, one would, therefore, conclude Q, that one should carry an umbrella. Rule-based expert systems use this type of reasoning to derive new facts that become part of the system's knowledge base.

Induction

The process of induction is the opposite of deduction. Inductive reasoning involves using a number of facts to induce a general conclusion. For example, if one observes that all sociologists who have attended the last three ASA (American Sociological Association) meetings brought tape recorders, one could logically infer that all sociologists who attend ASA meetings bring tape recorders. This may or may not be the case, however.

Abduction

Abduction is a form of deductive logic that provides "plausible inferences." Consider the following example:

All successful sociologists have tenure.
Achmed Abu has tenure.

From the available information, one could make the plausible inference that Achmed Abu is a successful sociologist. Again this may or may not be true. Achmed Abu might be a psychologist or a computer scientist, for example. In addition, Achmed could be one who was awarded tenure but had not been successful.

<u>Constructing Expert Systems</u>

Hayes-Roth et al. (1983, pp. 23-25) identify five major stages in the construction of expert systems. These stages are identification, conceptualization, formalization, implementation, and testing.

<u>Identification</u>

The first step in constructing an expert system is defining a goal or problem to be solved. After this is accomplished, concepts and characteristics of the problem are determined. This stage is known as identification. During this stage, the knowledge engineer consults books or other materials to gain familiarity with the subject. Specific resources necessary for project completion such as time and computing facilities must be determined. In addition, all others who will contribute to the development of the system are identified at this stage.

<u>Conceptualization</u>

In the conceptualization stage the knowledge engineer and the domain expert work together to formulate concepts and strategies necessary in solving the problem. Sometimes, the problem is diagrammed to illustrate graphically these concepts and how they work together toward solving the problem. Subgoals are dealt with similarly at this stage.

<u>Formalization</u>

The third stage of expert systems construction is the formalization stage. In this stage the knowledge engineer must concentrate on how to represent formally the concepts and strategies formulated in the previous stage. This involves the selection of an appropriate AI programming language or expert system development tool. The central task of this stage is the design of structures to organize the knowledge.

Implementation

During the implementation stage knowledge formalized in the previous stage is implemented using one of the knowledge representation methods. The knowledge engineer can choose from rules, frames, semantic nets, or scripts. This reformulated knowledge along with the program's control structure make up a prototype program that can be executed and checked for errors.

Testing

The final stage in constructing an expert system is testing. During this stage the prototype program is subjected to rigorous testing. This is done to reveal possible weaknesses or errors in the system. It is a rare occurrence for a program to execute correctly on the first attempt. Continuous revision of the rules is performed until the system performs up to the expectations of the domain expert. The emphasis in this stage is on validating the rules that make up the system.

Tools for Expert System Construction

There are a variety of methods for constructing expert systems. Expert systems may be constructed using AI programming languages such as LISP or PROLOG. They may also be created by using expert system shells. Unlike programming languages, expert system shells already have the inference engine built-in. This decreases development time and allows the researcher to concentrate more on the problem domain than on programming. Some commercial expert system shells in use today include 1st Class Fusion, M.1, Knowledgepro, EXSYS, and VP-Expert. The advantage of using an AI programming language over an expert system shell is flexibility. It gives the researcher control over project development that a preprogrammed shell cannot deliver.

PROLOG

One of the high-level languages used for expert system development is PROLOG. PROLOG is an acronym for PROgramming in LOGic. This language was developed between 1970 and 1972 in France by a team of researchers headed by Alain Colmerauer (Clocksin and Mellish 1987, p. 221).

Unlike other programming languages which are procedure-oriented, PROLOG is a descriptive language which is symbolically oriented. It is used for solving problems that involve objects and the relations between those objects (Clocksin and Mellish 1987, p. 1). The following is an example of a simple PROLOG knowledge-base:

```
1. student (daryl,wku).
2. student (jane, osu).
3. student (scott,uk).
4. majors_in (daryl,sociology).
5. majors_in (jane,computer_sci).
6. majors_in (scott,medicine).
7. is_healthy(X):-student(X,Y),majors_in(X,medicine).
8. is_wealthy(X):-student(X,Y),majors_in(X,computer_sci).
9. is_wise(X):-student(X,Y),majors_in(X,sociology).
```

The first line of this database asserts the fact that "Daryl is a student at WKU." The second line asserts that "Jane is a student at OSU" while the third asserts that "Scott is a student at UK." Note that the word "student" describes the relation between the two objects in parentheses. This is known as predicate calculus. A predicate is defined as assertions about individuals in relation to themselves and to other individuals. The predicate is generally expressed as:

```
predicate(arguments).
```

The computer accepts these assertions as facts although they do not necessarily have to be true in the real world. Take the following example:

```
president(bozo,usa).
```

This translates into "Bozo is the president of the U.S.A," which everyone knows is false; but the computer accepts it as fact. Lines four through six of the above database tell what each student is majoring in. Line four asserts that "Daryl majors in sociology" while lines five and six assert the respective majors of Jane and Scott. Lines seven, eight, and nine differ from the first six in that they state rules instead of facts. Line seven translates into "A person X," where X is a variable that can be instantiated with the name of any of the students provided that he/she meets the criteria of the rule, "is healthy if he/she is a student and he/she majors in medicine." The symbol ":-" means "if" in PROLOG while the "," represents the word "AND." It should be noted at this point that PROLOG rules can also contain "ORs" in their logic. An "OR" is represented by the ";" symbol. If we wish to determine which of the three students in our database meets the criteria for being healthy, we ask the computer in the following manner:

```
?- is healthy(X).
```

which translates into "Who is healthy?" Note that the "?-" symbol represents a system prompt and is not a part of the command. After searching its database, the computer will respond:

```
X=scott
```

since Scott is a student and he is majoring in medicine. Note that both conditions must be satisfied since the logical operator "AND" was used. At this point, the computer will continue searching its database to see if anyone else meets the criteria of the rule. If it finds someone else it will display his/her name, otherwise it will stop the search and await the next question. Line eight states that a person is wealthy if he/she is a student majoring in computer science while the last states that a person is wise if he/she is a student majoring in sociology. A user may query the computer to find out who is wealthy and who is wise by issuing the commands:

```
?- is_wealthy(X).
?- is_wise(X).
```

After initiating another search of its database, the computer should respond X=jane, and X=daryl respectively. Note that each PROLOG command must end with a terminator ".". Also note that if the computer is unsuccesful in its search for an answer to a question, or if a user asks it something that is not true, it will simply respond "no."

The next chapter shows how these logic programming techniques can be used for constructing social networks.

CHAPTER IV

APPLICATIONS TO SOCIOLOGY

This chapter is divided into two sections. The first section illustrates, in tutorial fashion, how social situations, particularly dyads and triads, can be represented and examined using the AI programming language PROLOG. The second section presents some expert systems designed for sociological decision making.

Representing Dyads and Triads using PROLOG

Since PROLOG is designed to solve problems involving objects and the relations between those objects, sociologists can use people or groups as the objects of analysis when constructing a PROLOG program. An obvious application of PROLOG in sociology is examining social networks composed of dyads and triads. Instead of representing data structures as computer scientists do, sociologists can represent complex social structures with PROLOG.

A dyad is composed of two individuals engaged in social relations. According to Stark (1985, p. 4) it is the smallest group of sociological interest. If one member leaves the relationship, the dyad ceases to exist.

Representing one type of dyad using PROLOG is a straightforward procedure. The first step that should be taken is to tell the computer what actors are involved in the situation. For example, say our initial actors are Alan and Betty. The PROLOG commands to tell the computer this information are:

```
     person (alan).
     person (betty).
```

These commands tell the computer that Alan and Betty are persons. Now that the actors have been defined, the relations between them must be defined. This is done with the following statements:

```
     likes (alan, betty).
     likes (betty, alan).
```

The first statement asserts the fact that "Alan likes Betty." Alan's feelings for Betty are unrequited at this point. The second statement "Betty likes Alan" makes the feeling mutual. Thus, a dyadic relationship has been formed in the computer's memory. To tell the computer that the relationship between Betty and Alan is a dyad, the following PROLOG rule must be entered:

```
     dyad (A,B) :- person (A), person (B),
                   likes (A,B), likes (B,A).
```

This rule says that a dyad exists if A and B are persons and person A likes person B and person B likes person A. This rule can be tested by entering the following command substituting "Alan" for A and "Betty" for B:

```
     ?- dyad (alan, betty).
```

This command is equivalent to the question "Do Alan and Betty comprise a dyad?" Provided that the previous statements had been entered, the computer should respond either "yes" or "true" depending on what version of PROLOG you are using. Versions available include PROLOG-86, PD (Public Domain) PROLOG, and Turbo Prolog. If we wanted to know "Who is involved in this situation?", we could enter the command:

```
     ?- person (X).
```

The computer would respond:

```
     X = alan
     X = betty.
```

If we wanted to know "Who likes whom?", we would enter the command:

```
     ?- likes (X,Y).
```

The computer should respond:

```
     X = alan
     Y = betty

     X = betty
     Y = alan.
```

When a third member is added, the dyad becomes a triad. A triad is composed of three persons engaged in social relations (Caplow 1968; Stark 1985). Suppose we wanted to add a third member named "Carol" to our current duo. The first step would be to tell the computer that this third person exists. This can be done by adding the following statement to our current knowledge base:

```
     person (carol).
```

Carol's relations with Alan and Betty must now be established. The following lines will accomplish this task:

```
     likes (carol, alan).
     likes (carol, betty).
     likes (alan, carol).
     likes (betty, carol).
```

It isn't necessary for a triad's members to all like each other as our trio does. The only requirement for a triad to exist is that its three members engage in social relations.

This requirement can be met by adding the following statement:

```
engage_in_social_rel (alan, betty, carol).
```

The final task is telling the computer what constitutes a triad. A PROLOG rule representing a triad appears as follows:

```
triad (A,B,C) :- person (A), person (B), person (C),
                 engage_in_social_rel (A,B,C).
```

This rule states that A, B, and C, defined as persons, comprise a triad if they engage in social relations. To determine if Alan, Betty, and Carol make up a triad, we would enter the command:

```
?- triad (alan, betty, carol).
```

If all previous statements had been entered correctly, the computer would respond either "yes" or "true" since Alan, Betty, and Carol have all been defined as persons and engage in social relations.

As can be seen, Carol shares a mutual liking with both Alan and Betty. This is known as a transitive or balanced triad (Stark 1985, p. 6). It isn't necessary for all members of a triad to like each other in order for a triad to exist. This occurs when there are no two members who like each other and have opposite relations with the third. In other words "Your friends are my friends" and "Your enemies are my enemies." For example, if both Betty and Alan, who like each other, held a mutual dislike for Carol, the triad would still be balanced. In all, there are four different combinations of balanced triads. Each of these combinations can be represented as a series of PROLOG statements. They are presented in Figure 4.1 along with corresponding PROLOG statements. If any of the four combinations in Figure 4.1 are true, the triad is transitive or balanced. A PROLOG rule can be created to determine whether a triad is transitive by combining all possible conditions into one statement.

```
transitive (A,B,C) :-   likes (A,B), likes (B,A),
                        likes (A,C), likes (C,A),
                        likes (B,C), likes (C,B);

                        likes (A,B), likes (B,A),
                  dislikes (A,C), dislikes (C,A),
                  dislikes (B,C), dislikes (C,B);

                  dislikes (A,B), dislikes (B,A),
                     likes (A,C),    likes (C,A),
                  dislikes (B,C), dislikes (C,B);

                  dislikes (A,B), dislikes (B,A),
                  dislikes (A,C), dislikes (C,A),
                     likes (B,C),    likes (C,B).
```

This rule asserts that persons A, B, and C comprise a transitive triad if they meet one of the four arrangements for a transitive triad. To test this rule, consider the following PROLOG knowledge base:

```
person (alan).
person (betty).
person (carol).
likes (alan, betty).
likes (betty, alan).
likes (alan, carol).
likes (carol, alan).
likes (betty, carol).
likes (carol, betty).
engage_in_social_rel (alan, betty, carol).
triad (A,B,C) :- person (A), person (B), person (C),
                 engage_in_social_rel (A,B,C).
transitive (A,B,C) :-   likes (A,B), likes (B,A),
                        likes (A,C), likes (C,A),
                        likes (B,C), likes (C,B);
```

Figure 4.1. Transitive Triads with PROLOG Statements

```
----------------------------------------------------------------

                 A                                    A
               /   \                                /   \
       +     /       \     +                -     /       \     +
           /           \                        /           \
         /               \                    /               \
       /                   \                /                   \
      B ------------------- C              B ------------------- C
                 +                                    -

likes (A,B), likes (B,A),            dislikes(A,B),dislikes(B,A),
likes (A,C), likes (C,A),               likes(A,C),   likes(C,A),
likes (B,C), likes (C,B).            dislikes(B,C),dislikes(C,B).


                         --------------
                         | +   like    |
                         | -   dislike |
                         --------------


                 A                                    A
               /   \                                /   \
       +     /       \     -                -     /       \     -
           /           \                        /           \
         /               \                    /               \
       /                   \                /                   \
      B ------------------- C              B ------------------- C
                 -                                    +

   likes(A,B),   likes(B,A),         dislikes(A,B),dislikes(B,A),
dislikes(A,C),dislikes(C,A),         dislikes(A,C),dislikes(C,A),
dislikes(B,C),dislikes(C,B).            likes(B,C),   likes(C,B).


----------------------------------------------------------------
```

```
                    likes (A,B), likes (B,A),
                 dislikes (A,C), dislikes (C,A),
                 dislikes (B,C), dislikes (C,B);

                 dislikes (A,B), dislikes (B,A),
                    likes (A,C),    likes (C,A),
                 dislikes (B,C), dislikes (C,B);

                 dislikes (A,B), dislikes (B,A),
                 dislikes (A,C), dislikes (C,A),
                    likes (B,C),    likes (C,B).
```

If we wanted to know whether Alan, Betty, and Carol make up a transitive triad, we would enter the command:

```
?- transitive (alan, betty, carol).
```

The computer should respond either "yes" or "true." This happens because Alan, Betty, and Carol meet the conditions of the first transitive triad arrangement presented in Figure 4.1.

Triads can also be imbalanced or intransitive. This situation occurs when there are two members of a triad who like each other but differ on their feelings for the third member (Stark 1985). This can be illustrated by the following example:

```
likes (alan, betty).
likes (betty, alan).
likes (alan, carol).
likes (carol, alan).
dislikes (betty, carol).
dislikes (carol, betty).
```

Here we have Alan and Betty liking each other, and Alan and Carol liking each other, but Betty and Carol disliking each other. This arrangement is unstable and will likely break up. Betty's jealousy over the relations between Alan and Carol, or Carol's jealousy over the relations between Alan and Betty may cause either Betty or Carol to break away thus disintegrating the triad. As with transitive triads, there are four arrangements of intransitive triads. They are presented in Figure 4.2 along with their PROLOG representations. The PROLOG rule for an intransitive triad, taking all four possible arrangements into account, is as follows:

```
intransitive (A,B,C) :-      likes (A,B), likes (B,A),
                         dislikes (A,C), dislikes (C,A),
                              likes (B,C), likes (C,B);

                              likes (A,B), likes (B,A),
                              likes (A,C), likes (C,A),
                         dislikes (B,C), dislikes (C,B);

                         dislikes (A,B), dislikes (B,A),
                              likes (A,C), likes (C,A),
                              likes (B,C), likes (C,B);

                         dislikes (A,B), dislikes (B,A),
                         dislikes (A,C), dislikes (C,A),
                         dislikes (B,C), dislikes (C,B).
```

To see how this rule operates, consider the following knowledge base:

```
person (alan).
person (betty).
person (carol).
likes (alan, betty).
likes (betty, alan).
likes (alan, carol).
likes (carol, alan).
dislikes (betty, carol).
dislikes (carol, betty).
engage_in_social_rel (alan, betty, carol).
triad (A,B,C) :- person (A), person (B), person (C),
                 engage_in_social_rel (A,B,C).

transitive (A,B,C) :-   likes (A,B), likes (B,A),
                        likes (A,C), likes (C,A),
                        likes (B,C), likes (C,B);

                        likes (A,B), likes (B,A),
                   dislikes (A,C), dislikes (C,A),
                   dislikes (B,C), dislikes (C,B);
```

Figure 4.2. Intransitive Triads with PROLOG Statements

```
-------------------------------------------------------------

                 A                                    A
               /   \                                /   \
      +      /       \     -              -       /       \     +
           /           \                        /           \
         /               \                    /               \
       /                   \                /                   \
      B -----------------   C              B -----------------   C
                 +                                    +

      likes(A,B),    likes(B,A),     dislikes(A,B),dislikes(B,A),
   dislikes(A,C),dislikes(C,A),         likes(A,C),    likes(C,A),
      likes(B,C),    likes(C,B).        likes(B,C),    likes(C,B).


                           ---------------
                          |  +    like    |
                          |  -    dislike |
                           ---------------


                 A                                    A
               /   \                                /   \
      +      /       \     +              -       /       \     -
           /           \                        /           \
         /               \                    /               \
       /                   \                /                   \
      B -----------------   C              B ------------------   C
                 -                                    -

      likes(A,B),    likes(B,A),     dislikes(A,B),dislikes(B,A),
      likes(A,C),    likes(C,A),     dislikes(A,C),dislikes(C,A),
   dislikes(B,C),dislikes(C,B).      dislikes(B,C),dislikes(C,B).

-------------------------------------------------------------
```

```
                     dislikes (A,B), dislikes (B,A),
                           likes (A,C), likes (C,A),
                     dislikes (B,C), dislikes (C,B);

                     dislikes (A,B), dislikes (B,A),
                     dislikes (A,C), dislikes (C,A),
                           likes (B,C), likes (C,B).

intransitive (A,B,C) :-    likes (A,B), likes (B,A),
                     dislikes (A,C), dislikes (C,A),
                           likes (B,C), likes (C,B);

                           likes (A,B), likes (B,A),
                           likes (A,C), likes (C,A),
                     dislikes (B,C), dislikes (C,B);

                     dislikes (A,B), dislikes (B,A),
                           likes (A,C), likes (C,A),
                           likes (B,C), likes (C,B);

                     dislikes (A,B), dislikes (B,A),
                     dislikes (A,C), dislikes (C,A),
                     dislikes (B,C), dislikes (C,B).
```

Given the following command:

```
     ?- intransitive (alan, betty, carol).
```

The computer should respond either "yes" or "true" since the conditions for an intransitive triad are satisfied. If the computer were asked whether Alan, Betty, and Carol comprised a transitive triad at this point using the command:

```
     ?- transitive (alan, betty, carol).
```

The computer would respond either "no" or "false" since they do not presently meet the requirements of a transitive triad.

Using previously discussed techniques, we can predict the behavior of actors in a number of social situations provided that all possibilities are included in our model. In these situations, we can create rules of behavior for the actors based on what sociological theorists say should happen in these situations. For example, suppose we create a few actors engaged in an exchange situation. According to social exchange theorists, everything has a price. They explain social behavior in terms of how people engage in exchange relationships for goods and services. Exchange behavior is based on costs and rewards of alternative courses of action. People choose the most attractive exchanges to engage in. We can predict who will exchange with whom in our model by including knowledge of social exchange theory (in the form of rules) as part of our knowledge base.

Suppose we wanted to predict which actors in our knowledge base would become angry if their expectations were not fulfilled or which ones would be pleased if their expectations were fulfilled. We can predict this by representing George Homans' (1974) aggression-approval proposition using the following PROLOG statements:

```
/* Homans' aggression-approval proposition */
become(person,angry) :- expectations(person,disappointed).
become(person,pleased) :- expectations(person,fullfilled);
                          expectations(person,exceeded).
likely_behavior(person,aggressive):- become(person,angry).
likely_behavior(person,approving):-become(person,pleased).
```

When this information is placed in the computer's knowledge base, a user can determine what actors within the knowledge base will become angry and what their likely behavior will be, given the rules listed above.

Knowledge Based Systems for Sociological Decision Making

In addition to the previous logic programming applications, some small knowledge-based systems were also created for this paper. These systems make decisions utilizing sociological knowledge. The knowledge-based systems presented in this paper deal with determining whether a triad is transitive or intransitive, classifying adaptation to anomie (normative confusion), determining whether a type I or type II error has been made in the hypothesis testing process, determining a variable's level of measurement, and classifying individuals on the basis of socio-economic status (S.E.S).

Selection of a Building Tool

VP-Expert was chosen as the tool for building these knowledge-based systems. VP-Expert is a rule-based expert systems shell by Paperback Software International. It runs on IBM PCs and compatibles with a minimum of 384K RAM and two disk drives (of which one may be a hard drive). VP-Expert was chosen because it has a number of attractive features. They include user support, the use of confidence factors for uncertain information, English-like rule construction, an explanation interface, built-in text editor, and the ability to "chain" multiple knowledge bases.

For my problems, I chose an expert system shell over a traditional programming language because the shell had a built-in inference engine. This allowed me to concentrate more on the problem areas than on programming. VP-Expert uses a problem-solving method known as "backward chaining." This means the system's inference engine starts with a goal variable and proceeds through its series of rules until it finds a value to assign to it, thus solving the problem.

TRIADS

The first expert system uses rules from the previous PROLOG triad model to determine whether a triad is transitive or intransitive. This system is known as TRIADS. The rules for this system are shown in Figure 4.3. A consultation with the system is shown in Figure 4.4.

The goal of TRIADS is to find a value for the variable "triad_type." First, the system attempts to satifisfy the conclusion of RULE 0, that the triad type is transitive. To do this it must first satifisfy the rules conditions. It does this by asking "What is the relation between Alan and Betty?" The user is given the choice of "like" or "dislike." For the example in Figure 4.4, "like" was chosen for this question. At this point, the system has determined that "relation_alan_betty=like." The next question it asks is "What is the relation between Alan and Carol?" The response entered for this question was "dislike." The computer knows it can no longer satisfy all the conditions for RULE 0 at this point and thus moves on to RULE 1. The computer then asks the last question "What is the relation between Betty and Carol?" As shown in Figure 4.4, "dislike" was entered as the response. Since all of the conditions for RULE 1 are satisfied at this point, "relation_alan_betty=like," "relation_alan_carol=dislike," and the last condition, "relation_brutus_charlie=dislike," the conclusion "triad_type=transitive" is made and the computer displays "This triad is transitive."

Figure 4.3. Rule Base of TRIADS

```
-----------------------------------------------------------------
RULE 0
IF   relation_alan_betty=like AND
     relation_alan_carol=like AND
     relation_betty_carol=like
THEN triad_type=transitive;

RULE 1
IF   relation_alan_betty=like AND
     relation_alan_carol=dislike AND
     relation_betty_carol=dislike
THEN triad_type=transitive;

RULE 2
IF   relation_alan_betty=dislike AND
     relation_alan_carol=like AND
     relation_betty_carol=dislike
THEN triad_type=transitive;

RULE 3
IF   relation_alan_betty=dislike AND
     relation_alan_carol=dislike AND
     relation_betty_carol=like
THEN triad_type=transitive;

RULE 4
IF   relation_alan_betty=like AND
     relation_alan_carol=dislike AND
     relation_betty_carol=like
THEN triad_type=intransitive;

RULE 5
IF   relation_alan_betty=like AND
     relation_alan_carol=like AND
     relation_betty_carol=dislike
THEN triad_type=intransitive;

RULE 6
IF   relation_alan_betty=dislike AND
     relation_alan_carol=like AND
     relation_betty_carol=like
THEN triad_type=intransitive;

RULE 7
IF   relation_alan_betty=dislike AND
     relation_alan_carol=dislike AND
     relation_betty_carol=dislike
THEN triad_type=intransitive

ELSE triad_type=unknown;
-----------------------------------------------------------------
```

Figure 4.4. Run of TRIADS

```
----------------------------------------------------------------

What is the relation between Alan and Betty?
  like               dislike

What is the relation between Alan and Carol?
  like               dislike

What is the relation between Betty and Carol?
  like               dislike

This triad is transitive.

----------------------------------------------------------------
```

Underlined answers: Alan and Betty — like; Alan and Carol — dislike; Betty and Carol — dislike.

MERTON

The second expert system was designed to determine the mode of adaptation based on Robert Merton's typology of individual adaptation to anomie. Merton claims that deviance results from obstacles present between a culture's goals and the means for achieving them (Merton 1938, pp. 672-682). According to Merton, there are five modes of adaptation: conformity, innovation, ritualism, retreatism, and rebellion.

Conformists accept both the cultural goals and the institutionalized means for achieving them. This is the most common form of adaptation. The Scout who follows the Scouting handbook rules in order to obtain merit badges is a conformist.

Innovators accept the cultural goals but reject the approved means for obtaining them. Michael Milken and Ivan Boesky are examples of innovators. Both sought the goal of wealth but used illegal means for obtaining it.

Ritualism involves rejecting the cultural goals but accepting the means. Institutional bureaucrats are prime examples of ritualists. They become so involved with the red tape that they lose sight of the goal.

The opposite of conformists, retreatists reject both the cultural goals and the means for achieving them. Drug addicts, alcoholics, and cultists are all examples of retreatists.

Like retreatism, rebellion is the rejection of both the cultural goals and the accepted means for achieving them. But unlike retreatism, new goals and means are substituted for the old ones. Political rebels who seek to overthrow their present form of government are engaging in rebellion.

Each of these modes can be represented as a rule in a knowledge base. The resulting knowledge base is presented in Figure 4.5.

When the system is executed, it knows its goal is to find a value for classification. Possible values are conformity, innovation, ritualism, retreatism, and rebellion. To find a value for classification, the system attempts to satisfy the first rule in the knowledge base. It first seeks the value of "goals." Since it cannot determine this value from information in its knowledge base, the system asks the user for the value of goals as shown in Figure 4.6. Possible choices are "accept" and "reject." If "reject" is entered, the following things occur. First, the system has established the fact that "goals = reject." The system then searches through its rules to find an instance where "goals = reject." It reaches this condition in RULE 2. Rules 0 and 1 tested for conditions where goals had to equal "accept." Since the value of goals is now "reject," we know that the variable "classification" cannot be "conformity" or "innovation."

The next condition the system must satisfy for RULE 2 to fire is "means = accept." Since the system has not yet established a value for "means," it must ask the user for this information. Again the options are "accept" and

Figure 4.5. Rule Base of MERTON

```
-------------------------------------------------------------

RULE 0

IF          goals = accept AND
            means = accept
THEN        classification = conformity;

RULE 1

IF          goals = accept AND
            means = reject
THEN        classification = innovation;

RULE 2

IF          goals = reject AND
            means = accept
THEN        classification = ritualism;

RULE 3

IF          goals = reject AND
            means = reject AND
            replace_goals_and_means = no
THEN        classification = retreatism

ELSE        classification = rebellion;

-------------------------------------------------------------
```

"reject." When "reject" is entered as the value for "means," RULE 2 fails (so we know the value of "classification" isn't "ritualism"); and the system searches for a rule where the value for both "goals" and "means" equals "reject." It finds this situation in RULE 3. At this point, the system has established two facts: "goals = reject" and "means = reject." So we know that our answer will either be "retreatism" or "rebellion." To determine which, the computer tries to establish a value for "replace_goals_and_means." It does this by asking, "Are new goals and means substituted for old?" If "no" is entered, the system will conclude that the mode of adaptation is retreatism. In the test run, "yes" was entered to this question. The system thus concluded that "The mode of adaptation is rebellion," as shown in Figure 4.6. By substituting different values for "goals" and "means," the system will produce different outcomes. A complete listing of this program, called MERTON, can be found in the appendix.

DEES

DEES (Decision Error Expert System) is an expert system designed for determining whether a Type I or Type II error has been made during the hypothesis testing process. A Type I or alpha error occurs when a true null hypothesis is rejected (Loether and McTavish 1988, pp. 534-35). This knowledge can be represented in rule form as follows:

```
IF   null_hypothesis = true AND
     decision_made = reject
THEN outcome = Type_I_Error.
```

Figure 4.6. Run of MERTON

```
What is the value of goals?
 accept                        [✓ reject]

What is the value of means?
 accept                        [✓ reject]

Are new goals and means substituted for old?
 [✓ yes]                        no

The mode of adaptation is rebellion.
```

Type II or beta errors occur when one fails to reject a false null hypothesis. This is represented by the rule:

```
IF   null_hypothesis = false AND
     decision_made = fail_to_reject
THEN outcome = Type_II_Error.
```

There are two other possible outcomes in this decision-making process. A researcher could fail to reject a true null hypothesis or reject a false null hypothesis. In both cases, the researcher would be making a correct decision. These decisions are represented by the following rules:

```
IF   null_hypothesis = true AND
     decision_made = fail_to_reject
THEN outcome = correct_decision;

IF   null_hypothesis = false AND
     decision_made = reject
THEN outcome = correct_decision.
```

Together, these rules comprise a knowledge-based system called DEES. This program can be used by sociologists to teach the concept of Type I and Type II errors to students. A complete copy of DEES is provided in the appendix.

LOMEX

Another knowledge-based system that can be used for teaching social research concepts is LOMEX. LOMEX is an acronym for Level Of Measurement EXpert. It is a system designed for determining a variable's level of measurement. There are four traditional levels of measurement used: nominal, ordinal, interval, and ratio.

Nominal variables have exhaustive, mutually exclusive categories. Religion is an example of a nominal variable. Individuals can be classified into religious categories such as Catholic, Jewish, or Protestant. Race, marital status, and gender are also nominal variables.

Ordinal variables possess the properties of nominal variables, but they can also be ordered or ranked. For example, we can rank athletic teams on the basis of scores or we can rank individuals on the basis of class (such as upper, middle, and lower).

Cases measured on an interval basis possess all the attributes of the previous measures, but in addition, cases or scores can be described in terms of equal units. Examples of interval variables include IQ and prestige scores.

The ratio level of measurement is similar to the interval measure, but it has a true zero point. In fact, ratio is sometimes combined with interval to form one measure. Examples of ratio variables include income, age, and educational level (in years).

Rules have been formulated to represent each of the four levels of measurement. They are presented in Figure 4.7. Figure 4.8 shows an actual run of LOMEX. The goal of LOMEX is to find a value for the variable "Level_of_Measurement." The system begins by asking "Can cases be classified into exhaustive, mutually exclusive categories?" For this run, "yes" was entered. At this point, LOMEX has confirmed the fact that "categories = yes." It then proceeds to determine whether the cases can be ordered or ranked by asking "Can cases be ranked from high to low?" At this point, "no" was entered. Given this information, the system decides that the

Figure 4.7. Rule Base of LOMEX

```
RULE 1

IF          categories = yes AND
            can_be_ranked = yes AND
            equal_units = yes AND
            zero_point = yes
THEN        Level_of_measurement = ratio;

RULE 2

IF          categories = yes AND
            can_be_ranked = yes AND
            equal_units = yes AND
            zero_point = no
THEN        Level_of_measurement = interval;

RULE 3

IF          categories = yes AND
            can_be_ranked = yes AND
            equal_units = no
THEN        Level_of_measurement = ordinal;

RULE 4

IF          categories = yes AND
            can_be_ranked = no
THEN        Level_of_measurement = nominal

ELSE        Level_of_measurement = unknown;
```

Figure 4.8. Run of LOMEX

Can cases be classified into exhaustive, mutually exclusive categories?

<u>yes</u> no

Can cases be ranked from high to low?

yes <u>no</u>

The level of measurement is nominal.

---

level of measurement can't be ratio, interval, or ordinal as defined in rules one through three. When LOMEX reaches rule four, both conditions ("categories = yes" and "can_be_ranked = no") are satisfied and the rule fires. Thus the system concludes the level of measurement is nominal as shown in Figure 4.8.

CLASS-EX

The final expert system created for this paper was CLASS-EX. CLASS-EX (for CLASS EXpert) was designed to classify individuals according to the socio-economic status (S.E.S) indicators: income, educational level, and occupation. The rules it uses to accomplish this task are loosely based on Coleman and Rainwater's Metropolitan Class Structure (Coleman and Rainwater 1978). A description of each class is presented below.

The upper-upper class is known as the "old rich." Their wealth is primarily derived from investments. Much of the wealth of the upper-upper class is inherited. Members of this class tend to be highly educated. Many of them possess degrees from prestigious Ivy-League colleges, often at the graduate level. Annual earnings of the upper-upper class are approximately $100,000 and up. Income figures for this knowledge base were updated to correspond more closely to current conditions.

Lower-upper class Americans make as much money or more than their upper-upper counterparts, but their money is newer. They are often referred to as the "nouveau riche" or the new rich. People such as Donald Trump would fall into this category. Members of this class are typically employed as top professionals and CEOs (Chief Executive Officers) in the corporate world. They tend to be highly educated, often at the postgraduate level, and receive that education at good colleges. As stated previously, lower-upper income is basically equivalent to upper-upper income.

Upper-middle class Americans are usually employed as mid-level managers or professionals. Members of this class have college or more education. Incomes of the upper-middle class range from $60,000 to $149,999.

Members of the middle class are often employed in lower-level management positions. They also comprise the nation's small-business owners, lower-status professionals, and sales and clerical workers. The education level of middle-class Americans tends to be high school with some college. Incomes for this class range from $30,000 to $59,999 per year.

The working class consists of blue-collar workers and low-paid sales and clerical workers. Younger members of this class typically have a high school diploma. Working class income typically ranges from $13,500 to $59,999 annually.

Among the lowest classes of Americans, members of the upper-lower class are concentrated in unskilled labor and service occupations. They have partially completed high school educations and earn from $13,500 to $29,999 a year.

Below them, at the very bottom of the class structure, is the lower-lower class. Members of this class are often unemployed and/or on welfare. They typically have elementary-level education and earn less than $13,500 annually.

This knowledge of social classes has been formulated into a series of rules. These rules are presented in Figure 4.9. Together these rules comprise the expert system CLASS-EX. A sample run of CLASS-EX is presented in Figure 4.10. In it the system has determined that an individual with a sales position, high school education, and earnings between $13,500 and $29,999 falls into the working class.

CLASS-EX does have some shortcomings, however. As with any expert system, its reasoning ability is limited to the amount of knowledge it possesses. If a user were to ask CLASS-EX what class a small business owner with an income of $100,000 a year and a high school education fits in, the system would respond "unknown." Like human experts, computerized experts do not always know the answer either. In this case, the computer would respond "unknown" because its knowledge base lacks information on such an individual. Since the system is based on the expertise of Coleman and Rainwater, knowledge about this situation should come from them. If knowledge from another class expert were placed in CLASS-EX to handle the previous situation, the expert system would no longer be based on the knowledge of Coleman and Rainwater, but on the knowledge of Coleman, Rainwater, and the third class expert. The complete listing for CLASS-EX

Figure 4.9. Rule Base of CLASS-EX

```
------------------------------------------------------------

RULE 0

IF        Occupation = inherited_wealth AND
          Education = college_or_more AND
          Income = $100000_to_$149999 OR
          Income = $150000_and_up
THEN      Social_Class = Upper_Upper;

RULE 1

IF        Occupation = top_professional OR
          Occupation = ceo AND
          Education = college_or_more AND
          Income = $100000_to_$149999 OR
          Income = $150000_and_up
THEN      Social_Class = Lower_Upper;

RULE 2

IF        Occupation = middle_professional OR
          Occupation = middle_manager AND
          Education = college_or_more AND
          Income = $60000_to_$74999 OR
          Income = $75000_to_$99999 OR
          Income = $100000_to_$149999
THEN      Social_Class = Upper_Middle;

RULE 3

IF        Occupation = low_level_manager OR
          Occupation = small_business_owner OR
          Occupation = sales OR
          Occupation = clerical AND
          Education = high_school_&_some_college AND
          Income = $30000_to_$59999
THEN      Social_Class = Middle;

RULE 4

IF        Occupation = higher_blue_collar OR
          Occupation = sales OR
          Occupation = clerical AND
          Education = high_school AND
          Income = $13500_to_$29999 OR
          Income = $30000_to_$59999
THEN      Social_Class = Working;

------------------------------------------------------------
```

Figure 4.9. Rule Base of CLASS-EX (continued)

```
------------------------------------------------------------------

RULE 5

IF          Occupation = unskilled_labor OR
            Occupation = service AND
            Education = some_high_school AND
            Income = $13500_to_$29999
THEN        Social_Class = Upper_Lower;


RULE 6

IF          Occupation = unemployed OR
            Occupation = on_welfare AND
            Education = primary_school AND
            Income = less_than_$13500
THEN        Social_Class = Lower_Lower

ELSE        Social_Class = Unknown;

------------------------------------------------------------------
```

Figure 4.10. Run of CLASS-EX

```
-------------------------------------------------------------

What is the source of your income?

 inherited_wealth      top_professional     ceo
 middle_professional   middle_manager       low_level_manager
 small_business_owner  [✓ sales]            clerical
 higher_blue_collar    unskilled_labor      service
 unemployed            on_welfare

What is your educational attainment?

 college_or_more    high_school/some_col  [✓ high_school]
 some_high_school   primary_school

What is your income level?

 $150000_and_up     $100000_to_$149999    $75000_to_$99999
 $60000_to_$74999   $30000_to_$59999      [✓ $13500_to_$29999]
 less_than_$13500

The social class of individual is working.

-------------------------------------------------------------
```

can be found in the appendix.

Conclusion

The programs presented in this paper show how artificial intelligence and expert systems can be applied to sociology. The programs can be used as teaching tools for instructing students on specific concepts in sociology and for learning the basics of artificial intelligence/expert systems as well. Using similar techniques, students can construct their own models of social situations using PROLOG or develop expert systems for other areas involving sociological decision making. By modeling social situations and social theories using logic programming techniques, students learn to think logically about these situations and theories.

As the field of artificial intelligence advances, so should its potential applications to sociology. Currently, sociologists who wish to apply this technology to their field must acquaint themselves with AI programming techniques. Although expert systems shell programs ease this burden somewhat, you must still become a programmer of sorts to reap the full benefits of the technology.

At this time, there are very few practical applications of AI/Expert systems to sociology. Expert systems are best suited for problems with "cookbook" style solutions. Few problems in sociology fit this description, however. A number of individuals involved in AI research have been overly optimistic as to what could be accomplished with it. This has led to speculation of a truly "thinking" machine --

one that goes beyond simple reasoning capability. With continued progress in the area of artificial intelligence and computer technology, sociologists may one day be able to instill the whole of sociological knowledge into an "intelligent" machine. Afterwards, the sociologist could conceivably present the machine with any situation of sociological interest and receive an analysis from a sociologist's perspective. The machine would be playing the role of sociologist, in effect. Although much of this is science fiction now, increased computation speeds, memory size, and advancements in AI technology may make this science fact in the distant future. The programs created as part of my research can be considered as small "stepping stones" toward such a machine. In the meantime, AI applications to sociology will probably remain a novelty to most sociological researchers.

APPENDIX A

Running a VP-Expert Consultation

The expert system shell VP-Expert is used for executing the knowledge based systems presented here. This program runs on IBM PCs and compatibles. To begin a consultation, follow these steps:

1. At the appropriate DOS prompt (A> for floppy drives, C> for hard disks), type "VPX" and press enter.

2. At this point, the title screen and copyright notice should appear along with a number of options across the bottom of the screen. This is referred to as the "command line." Using the cursor control keys on the keyboard, move the lightbar to the option "FileName" and press enter.

3. VP-Expert will ask you to choose a file at this point. You may select a file by moving the lightbar over it and pressing enter or by typing the name of the knowledge base at the prompt.

4. The name of the knowledge base you have chosen will appear at the top of the screen after "KBS:". This indicates the name of the active knowledge base. If you select MERTON as the desired knowledge base, the message "KBS:MERTON" will appear at the top of the screen.

5. Choose the "Consult" option to load the rule base into memory.

6. The chosen knowledge base is then executed by selecting "Go" on the command line.

7. At this point, a brief introduction to the expert system will be given and you will be asked to press a key to begin.

8. After pressing a key, the expert system will ask you questions so that it can reach a conclusion. You must use the arrow keys to move the lightbar over your selection and press enter. Upon pressing enter, an arrow will appear to the right of your answer to indicate your selection. To move on, press the "END" key on the keyboard and the next question will be asked. This process continues until the computer has enough information to reach a conclusion.

APPENDIX A (continued)

9. To execute the knowledge base again at the end of a consultation, select "Go" on the command line. Choosing "Quit" returns you to the main menu where you may consult another knowledge base by changing the filename and choosing "Consult" or quit VP-Expert by choosing "Quit."

APPENDIX B

TRIADS

```
!  TRIADS
!  Based on Theodore Caplows work on triad coalitions.
ACTIONS
          DISPLAY "TRIADS can determine whether a triad is
transitive or intransitive based on relations among its
members.

Press the SPACEBAR to begin the consultation.~"
          CLS
          FIND triad_type
          DISPLAY "This triad is {triad_type}.";

RULE 0
IF   relation_alan_betty=like AND
     relation_alan_carol=like AND
     relation_betty_carol=like
THEN triad_type=transitive;

RULE 1
IF   relation_alan_betty=like AND
     relation_alan_carol=dislike AND
     relation_betty_carol=dislike
THEN triad_type=transitive;

RULE 2
IF   relation_alan_betty=dislike AND
     relation_alan_carol=like AND
     relation_betty_carol=dislike
THEN triad_type=transitive;

RULE 3
IF   relation_alan_betty=dislike AND
     relation_alan_carol=dislike AND
     relation_betty_carol=like
THEN triad_type=transitive;

RULE 4
IF   relation_alan_betty=like AND
     relation_alan_carol=dislike AND
     relation_betty_carol=like
THEN triad_type=intransitive;

RULE 5
IF   relation_alan_betty=like AND
     relation_alan_carol=like AND
     relation_betty_carol=dislike
THEN triad_type=intransitive;
```

APPENDIX B (continued)

```
RULE 6
IF   relation_alan_betty=dislike AND
     relation_alan_carol=like AND
     relation_betty_carol=like
THEN triad_type=intransitive;

RULE 7
IF   relation_alan_betty=dislike AND
     relation_alan_carol=dislike AND
     relation_betty_carol=dislike
THEN triad_type=intransitive
ELSE triad_type=unknown;

ASK relation_alan_betty: "What is the relation between Alan
and Betty?";
CHOICES relation_alan_betty: like,dislike;

ASK relation_alan_carol: "What is the relation between Alan
and Carol?";
CHOICES relation_alan_carol: like,dislike;

ASK relation_betty_carol: "What is the relation between Betty
and Carol?";
CHOICES relation_betty_carol: like,dislike;
```

APPENDIX C

MERTON

```
! MERTON is designed to determine mode of adaptation
! based on Robert Merton's typology of
! individual adaptation to anomie.
ACTIONS
          FIND classification
          DISPLAY "The mode of adaptation is
          {classification}.";

RULE 0
IF        goals=accept AND
          means=accept
THEN      classification=conformity;

RULE 1
IF        goals=accept AND
          means=reject
THEN      classification=innovation;

RULE 2
IF        goals=reject AND
          means=accept
THEN      classification=ritualism;

RULE 3
IF        goals=reject AND
          means=reject AND
          replace_goals_and_means=no
THEN      classification=retreatism
ELSE      classification=rebellion;

ASK goals: "What is the value of goals?";
CHOICES goals: accept,reject;

ASK means: "What is the value of means?";
CHOICES means: accept,reject;

ASK replace_goals_and_means: "Are new goals and means
substituted for old?";
CHOICES replace_goals_and_means: yes,no;
```

APPENDIX D

DEES

```
! DEES is an acronym for Decision Error Expert System.
ACTIONS
          DISPLAY "DEES can determine if a Type I or Type II
                   error has been made.

Press the SPACEBAR to begin.~"
          CLS
          FIND outcome
          DISPLAY "A {outcome} has been made.";

RULE 1
IF        null_hypothesis=true AND
          decision_made=reject
THEN      outcome=Type_I_Error;

RULE 2
IF        null_hypothesis=true AND
          decision_made=fail_to_reject
THEN      outcome=correct_decision;

RULE 3
IF        null_hypothesis=false AND
          decision_made=reject
THEN      outcome=correct_decision;

RULE 4
IF        null_hypothesis=false AND
          decision_made=fail_to_reject
THEN      outcome=Type_II_Error;

ASK  null_hypothesis: "What is the value of the null
hypothesis?";
CHOICES null_hypothesis: true, false;

ASK  decision_made: "What is the value of the statistical
decision?";
CHOICES decision_made: reject, fail_to_reject;
```

APPENDIX E

LOMEX

```
! LOMEX is an acronym for Level Of Measurment EXpert System.
ACTIONS
          DISPLAY "LOMEX can determine the level of
                   measurment used in the social research
                   process.

Press the SPACEBAR to start the consultation.~"
          CLS
          FIND Level_of_measurement
          DISPLAY "The level of measurement is
          {Level_of_measurement}.

          Press the SPACEBAR to exit LOMEX.~"
          CLS;

RULE 1
IF        categories=yes AND
          can_be_ranked=yes AND
          equal_units=yes AND
          zero_point=yes
THEN      Level_of_measurement=ratio;

RULE 2
IF        categories=yes AND
          can_be_ranked=yes AND
          equal_units=yes AND
          zero_point=no
THEN      Level_of_measurement=interval;

RULE 3
IF        categories=yes AND
          can_be_ranked=yes AND
          equal_units=no
THEN      Level_of_measurement=ordinal;

RULE 4
IF        categories=yes AND
          can_be_ranked=no
THEN      Level_of_measurement=nominal
ELSE      Level_of_measurement=unknown;

ASK  categories: "Can cases be classified into exhaustive,
mutually exclusive categories?";
CHOICES categories: yes,no;

ASK  can_be_ranked: "Can cases be ranked from high to low?";
CHOICES can_be_ranked: yes,no;
```

APPENDIX E (continued)

```
ASK  equal_units: "Can distance between cases/scores be
described in terms of equal units?";
CHOICES equal_units: yes,no;

ASK  zero_point: "Is there a meaningful zero point on the
scale?";
CHOICES zero_point: yes,no;
```

APPENDIX F

CLASS-EX

```
!  CLASS-EX can determine the social class of an individual
!  based on Coleman & Rainwater's Metropolitan
!  Class Structure.

ACTIONS
          DISPLAY "This expert system, CLASSEX, can determine
                   the social class of an individual based on
                   Coleman and Rainwater's Metropolitan Class
                   Structure.

Press the SPACEBAR to start the consultation.~"
          CLS
          FIND social_class
          DISPLAY "The social class of individual is
          {social_class}.";

RULE 0
IF   Occupation=inherited_wealth AND
     Education=college_or_more AND
     Income=$100000_to_$149999 OR
     Income=$150000_and_up
THEN social_class=upper_upper;

RULE 1
IF   Occupation=top_professional OR
     Occupation=ceo AND
     Education=college_or_more AND
     Income=$100000_to_$149999 OR
     Income=$150000_and_up
THEN social_class=lower_upper;

RULE 2
IF   Occupation=middle_professional OR
     Occupation=middle_manager AND
     Education=college_or_more AND
     Income=$60000_to_$74999 OR
     Income=$75000_to_$99999 OR
     Income=$100000_to_$149999
THEN social_class=upper_middle;

RULE 3
IF   Occupation=low_level_manager OR
     Occupation=small_business_owner OR
     Occupation=sales OR
     Occupation=clerical AND
     Education=high_school_&_some_college AND
     Income=$30000_to_$59999
THEN social_class=middle;
```

APPENDIX F (continued)

```
RULE 4
IF   Occupation=higher_blue_collar OR
     Occupation=sales OR
     Occupation=clerical AND
     Education=high_school AND
     Income=$13500_to_$29999 OR
     Income=$30000_to_$59999
THEN social_class=working;

RULE 5
IF   Occupation=unskilled_labor OR
     Occupation=service AND
     Education=some_high_school AND
     Income=$13500_to_$29999
THEN social_class=upper_lower;

RULE 6
IF   Occupation=unemployed OR
     Occupation=on_welfare AND
     Education=primary_school AND
     Income=less_than_$13500
THEN social_class=lower_lower

ELSE social_class=unknown;

ASK Occupation: "What is the source of your income?";
CHOICES Occupation: inherited_wealth,top_professional,ceo,
middle_professional,middle_manager,low_level_manager,
small_business_owner,sales, clerical, higher_blue_collar,
unskilled_labor, service,unemployed,on_welfare;

ASK Education: "What is your educational attainment?";
CHOICES Education: college_or_more,
high_school_&_some_college, high_school,
some_high_school,primary_school;

ASK Income: "What is your income level?";
CHOICES Income: $150000_and_up, $100000_to_$149999,
$75000_to_$99999, $60000_to_$74999, $30000_to_$59999,
$13500_to_$29999, less_than_$13500;
```

APPENDIX G

Running a PROLOG Consultation

The PROLOG programs for this paper were created by using a text editor to input commands. A word processor that writes its output as an ASCII file will also work. The program used to compile and execute the PROLOG statements was PD PROLOG. To load PD PROLOG into memory, type "prolog" at the DOS prompt. To consult a particular file, issue the following command:

```
?- consult ('progname').
```

where "progname" is the name of your program. The program will compile at this point and will return the "?-" prompt for your inquiries. If the program is rejected during compilation, you must leave PD PROLOG and edit your program using a word processor or other text editor. To leave PD PROLOG, issue this command:

```
?- exitsys.
```

You will be returned to the DOS prompt.

APPENDIX H

Triad Model Using PROLOG

```
person (alan).
person (betty).
person (carol).
likes (alan,betty).
likes (betty,alan).
likes (carol,alan).
likes (carol,betty).
likes (alan,carol).
likes (betty,carol).
engage_in_social_rel (alan,betty,carol).
triad (A,B,C):-person (A), person (B), person (C),
               engage_in_social_rel (A,B,C).

transitive (A,B,C) :-   likes (A,B), likes (B,A),
                        likes (A,C), likes (C,A),
                        likes (B,C), likes (C,B);

                        likes (A,B), likes (B,A),
                  dislikes (A,C), dislikes (C,A),
                  dislikes (B,C), dislikes (C,B);

                  dislikes (A,B), dislikes (B,A),
                        likes (A,C), likes (C,A),
                  dislikes (B,C), dislikes (C,B);

                  dislikes (A,B), dislikes (B,A),
                  dislikes (A,C), dislikes (C,A),
                        likes (B,C), likes (C,B).

intransitive (A,B,C) :- likes (A,B), likes (B,A),
                  dislikes (A,C), dislikes (C,A),
                        likes (B,C), likes (C,B);

                        likes (A,B), likes (B,A),
                        likes (A,C), likes (C,A),
                  dislikes (B,C), dislikes (C,B);

                  dislikes (A,B), dislikes (B,A),
                        likes (A,C), likes (C,A),
                        likes (B,C), likes (C,B);

                  dislikes (A,B), dislikes (B,A),
                  dislikes (A,C), dislikes (C,A),
                  dislikes (B,C), dislikes (C,B).
```

APPENDIX I

Program Disks

BIBLIOGRAPHY

Alker, Hayward R. Jr. and Cheryl Christensen. 1972. "From Causal Modelling to Artificial Intelligence: The Evolution of a U.N. Peace-making Simulation." Pp. 177-224 in Experimentation and Simulation in Political Science, edited by Laponce, J.A. and Paul Smoker. Toronto: University of Toronto Press.

Alker, Hayward R. Jr., James Bennett and Dwain Mefford. 1980. "Generalized Precedent Logics for Resolving Insecurity Dilemmas." International Interactions 7: 165-206.

Banerjee, Sanjoy. 1986. "Reproduction of Social Structures: An Artificial Intelligence Model." Journal of Conflict Resolution 30:221-52.

Barr, Avron, and Edward A. Feigenbaum (eds.). 1981. The Handbook of Artificial Intelligence. Vol 1. Reading MA: Addison-Wesley.

Benfer, Robert A. 1989. "Individual Differences in Rule Based Systems of Knowledge with Behavioral Implications." Anthropological Quarterly 62: 69-81.

Brent, Edward E. 1984. "Qualitative Computing: Approaches and Issues." Qualitative Sociology 7: 34-60.

-------. 1985. "Relational Data Base Structures and Concept Formation in the Social Sciences." Computers and the Social Sciences 1: 29-49.

-------. 1986. "Knowledge-Based Systems: A Qualitative Formalism." Qualitative Sociology 9: 256-82.

-------. 1988a. "Is There a Role for Artificial Intelligence in Sociological Theorizing?" The American Sociologist 19: 158-66.

-------. 1988b. "New Approaches to Expert Systems and Artificial Intelligence Programming." Social Science Computer Review 6: 569-78.

-------. 1989a. "ERVING: A Program to Teach Sociological Reasoning from the Dramaturgical Perspective." Teaching Sociology 17: 38-48.

-------. 1989b. "Designing Social Science Research with Expert Systems." Anthropological Quarterly 62: 121-30.

Harmon, Paul and David King. 1985. Expert Systems. New York: John Wiley & Sons.

Hayes-Roth, Frederick, Donald A. Waterman, and Douglas B. Lenat (eds.). 1983. Building Expert Systems. Reading, MA: Addison-Wesley.

Hinze, Kenneth E. 1987. "Computing in Sociology: Bringing Back the Balance." Social Science Microcomputer Review 5: 439-51.

Hofstadter, Douglas R. 1979. Godel, Escher, Bach: An Eternal Golden Braid. New York: Basic Books.

Homans, George C. 1974. Social Behavior: Its Elementary Forms. Revised Edition. New York: Harcourt Brace Jovanovich.

Kippen, Jim and Bernard Bel. 1989. "Can the Computer Help Resolve the Problem of Ethnographic Description?" Anthropological Quarterly 62: 131-44.

Levine, Robert I., Diane E. Drang, and Barry Edelson. 1986. A Comprehensive Guide to AI and Expert Systems. New York: McGraw-Hill.

Loether, Herman J. and Donald G. McTavish. 1988. Descriptive and Inferential Statistics: An Introduction. Boston: Allyn and Bacon.

McGraw, Bruce A. and Karen L. McGraw. 1985. "Artificial Intelligence Finds Military Applications." Systems & Software, August, pp. 79-82.

Merton, Robert K. 1938. "Social Structure and Anomie." American Sociological Review 3: 672-82.

Minsky, Marvin. 1975. "A Framework for Representing Knowledge." Pp. 211-77 in The Psychology of Computer Vision, edited by P. Winston. New York: McGraw Hill.

Mishkoff, Henry C. 1986. Understanding Artificial Intelligence. Fort Worth, TX: Texas Instruments.

Read, Dwight W. and Clifford Behrens. 1989. "Modeling Folk Knowledge as Expert Systems." Anthropological Quarterly 62:107-20.

Schank, Roger, and Robert Abelson. 1977. Scripts, Plans, Goals, and Understanding. Hillsdale, NJ: Lawrence Erlbaum.

Schrodt, Philip A. 1988. "Artificial Intelligence and Formal Models of International Behavior." The American Sociologist 19:71-85.

Schutz, Alfred. 1962. Collected Papers I: The Problem of Social Reality. The Hague: Martinus Nijhoff.

Simmel, Georg. 1955. Conflict and the Web of Group Affiliations. New York: Free Press.

Stark, Rodney. 1985. Sociology. Belmont, CA: Wadsworth.

Sylvan, David and Barry Glassner. 1985. A Rationalist Methodology for the Social Sciences. Oxford, England: Basil Blackwell.

Thorson, Stuart J. and Donald A. Sylvan. 1982. "Counterfactuals and the Cuban Missile Crisis." International Studies Quarterly 26:539-71.

Turing, Alan M. 1950. "Computing Machinery and Intelligence." Mind 59:433-60.

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

Winfield, M.J., S.K. Toole, R.T. Griffin and P.M. Davies. 1986. "An Expert System Assistant for Human Services Personnel." Pp. 253-59 in Artificial Intelligence for Society, edited by Karamjit S. Gill. New York: John Wiley & Sons.

Wood, Sharon. 1986. "AI and Theories of Social Situations." Pp. 237-44 in Artificial Intelligence for Society, edited by Karamjit S. Gill. New York: John Wiley & Sons.

Woolgar, Steve. 1985. "Why not a Sociology of Machines? The Case of Sociology and Artificial Intelligence." Sociology 19: 557-72.