1

Introduction: Medical Artificial Intelligence Programs

William J. Clancey and Edward H. Shortliffe

1.1 Approaching Medical Artificial Intelligence

In approaching most fields of scientific inquiry, it is useful to consider two basic questions:

- What methodologies and assumptions do the researchers share?
- What issues and concerns distinguish the research projects from one another?

This introduction sketches out answers to these questions for the field of artificial intelligence applied to medicine (AIM), as viewed in the early 1980s, approximately a decade after the field's initial development. Our intent is to provide a common ground for appreciating what makes the work reported in this book special as a whole and for understanding the diverse terminology and research emphases of the individual chapters. In contrast with the history-oriented discussion one can find in Szolovits (1982), we discuss the dimensions by which one can recognize and study AIM programs. Thus we have two important goals: introducing the reader to a programming approach and relating the programs to one another through recurring research issues.

After a brief historical introduction, we define knowledge-based programming, provide dimensions for characterizing and comparing these programs, and outline the state of the art.

1.2 What Is Medical Artificial Intelligence?

Artificial intelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior—understanding language, learning, reasoning, solving problems, and so on (Barr et al., vol. 1, 1981, p. 3).

Medical artificial intelligence is primarily concerned with the construction of AI programs that perform diagnosis and make therapy recommendations. Unlike medical applications based on other programming methodologies, such as purely statistical and probabilistic methods, medical AI programs are based on symbolic models of disease entities and their relationships to patient factors and clinical manifestations.

In the early 1960s researchers in AI had focused on problem solving in game playing, image recognition, speech understanding, and language understanding (Feigenbaum and Feldman, 1963). During this time some general problem-solving principles were formalized, such as reducing a complex problem to a network of subgoals. However, it was discovered that most of the difficulty in achieving intelligent behavior was in collecting and storing a large *knowledge base* of facts specific to the problem area. This result was confirmed by the success of a few large programs in scientific (Lindsay et al., 1980) and mathematical (MATHLAB, 1974) areas. At this time, the explosion in medical knowledge was forcing physicians to specialize increasingly and was often overwhelming to those who tried to remain generalists. Medicine was therefore a logical field in which to apply practically the developing knowledge-based techniques.

Large domain-specific problem solvers came to be known as *consultation programs*, for they fit the image of an expert-specialist who is asked to provide advice about some difficult problem. In medicine, the "problem" would typically be a patient with an illness to be diagnosed. By the late 1970s, these programs became known as *expert systems*, although that remains a somewhat generous characterization in light of the limitations of these programs (as described in the following chapters).

Four major systems were developed by 1975—PIP, CASNET, MYCIN, and INTERNIST—all described in this book. As early efforts, they are prototypes directed at two questions: What are the issues involved in designing a consultation program (e.g., what would make such a program acceptable to physician users)? What is the nature of the expertise to be formalized (e.g., how can factual and judgmental knowledge be integrated)?

Most AIM programs are directed at serious practical applications, but there are other reasons for doing the research. Some AIM researchers have constructed programs for the sake of better understanding human problem solving in general. In this respect some of the most important results from this work are in the area of psychology (e.g., Chapter 12; Kassirer and Gorry, 1978; Kuipers and Kassirer, 1983; Swanson et al., 1979). On the other hand, constructing an AIM program is also fundamental medical research. The knowledge that is formalized in these programs does not come straight from textbooks; some is heuristic, developed through experience and passed down by apprenticeship. Thus, in their collaboration with physicians, medical AI researchers are helping to formalize medical knowledge. Importantly, the formalization process goes beyond accumulating facts to include new ways of structuring medical knowledge in general, such as formulating a language for describing diseases on multiple levels of detail (see Chapter 14). There is good reason to believe that such research will ultimately be of benefit to medical educators (Shortliffe, 1983).

Almost all AIM programs have been developed at universities, either in a medical school or in collaboration with a nearby school of medicine. The projects typically involve teams of collaborating computer scientists and physicians. On rare occasions, the computer scientist is also a physician and so can easily understand the medical issues or supply his or her own expertise (see Blum's work in Chapter 17). In many cases, the research has benefited by having computer scientists, initially unfamiliar with the medical field, approach the problem areas freshly, serving as "investigative reporters" who study what physicians know and how they solve problems.

How has the research advanced over the past decade? Most of the progress centers on the problem of *representing* medical knowledge. The first efforts concerned both the formalization of specific disease knowledge (e.g., how to distinguish between bacterial and viral meningitis) and the kinds of relations that physicians make among findings and diseases (e.g., that one disease is a complication of another). Two other areas of research are *knowledge acquisition*—interacting with an expert to formalize his or her knowledge and problem-solving procedures—and *explanation*—tracing back conclusions to the data and justifying the reasoning process. The capability of a program to perform these operations is now understood to depend very much on the adequacy of the underlying knowledge representation. Finding the right kinds of distinctions (for example, how to accurately model a causal process or how to state a diagnostic procedure) has been the focus of much of this research. Given finer-grained, more detailed models, the emphasis then shifts to formulating improved diagnostic operations for *manipulating* the knowledge to solve problems [the thrust of CADUCEUS (Pople, 1982) and developments from ABEL (Patil et al., 1982)].

We have briefly outlined how medical AI grew out of new directions in AI research. It is fair to say that the contributions have been at least as strong in the other direction. Efforts such as INTERNIST (Chapter 8) and

MYCIN (Chapter 5) led to a generalization of techniques that became known as *knowledge-based programming*. The idea of building expert systems like MYCIN has spread to almost every imaginable application (e.g., geology, structural analysis, and tax law), thereby making expert systems currently the largest subarea in the field of artificial intelligence. Moreover, even AI researchers specializing in subareas such as natural language understanding have come to realize that a knowledge base must be the foundation of any reasoning system (Carbonell, 1979). The wave of results is now flowing back to medical applications, as representation and design ideas developed in other scientific areas are picked up and adapted. Within AI, knowledge representation has become an area of study in its own right.

Finally, we should point out that just as AI takes in diverse areas such as signal understanding, image processing, and robotics, medical AI has parallel subfields corresponding to patient-monitoring systems, x-ray and ultrasound imaging systems, and prosthetic devices. In order to focus the collection of papers in this book, we have chosen to restrict the topic to systems concerned primarily with diagnosis and therapy—medical expert systems.

1.3 What Is a Knowledge Base?

The programs described in this book exemplify knowledge-based programming applied to medicine. The goals and techniques of knowledge-based programming are considerably different from other kinds of programming. Ways of analyzing and comparing traditional programs are not always relevant. Moreover, if the basic foundations of these programs are not understood, it is difficult to understand their limitations and potential.

One way to start is to understand that researchers in this field draw a distinction between *knowledge* and *data* (e.g., see Chapter 3). *Data* consist of records of information, such as patient records in a hospital, equipment maintenance records, or scientific measurements such as weather data. Data can be either symbolic (e.g., the names of patients) or numeric (e.g., temperatures). In computer science, the term *record* has generally become synonymous with the pattern that defines what might be recorded for each entry, such as the patient's name, his or her location in the hospital, date of entry, etc. A *data base* contains a set of such records.

By common usage, *knowledge* is anything you know, so it surely includes what we find in data bases. But knowledge also includes how things are related, what general patterns exist, why there are relations and patterns, as well as procedures for solving problems. For example, a data base might record information for a particular patient population from which correlations between drug therapy and adverse reactions or side effects could be derived statistically. A related *knowledge base* might have the general rule "If the patient is a child without a full set of adult teeth, don't prescribe tetracycline," as well as a causal model that explains how the process of chelation occurs, and perhaps a procedure that says "Consider contraindication rules after making a diagnosis and before prescribing therapy." So in an important sense, a knowledge base is general. Its records are about disease processes, diagnosis, and therapy in general, not about particular patients. Some knowledge can be derived analytically from data bases, but some is based on experience and is judgmental or heuristic.

Certain concerns about data bases, such as organization and accuracy, carry over to knowledge-based programming. But a knowledge base is different because it is never complete. A knowledge base is a kind of model: it can be interpreted to predict or explain behavior in the world. Thus diagnosis is based on a causal explanation of what is happening to the patient, and therapy is based on predictions about how the disease process can be modified. As models, knowledge bases are incomplete in that they are *approximate* and *omit levels of detail.*

Simple knowledge bases, like simplified models, might apply only to simple versions of problems; for example, a medical system might not be designed to handle multiple diseases. Handling multiple diseases might require modeling how one disease could cause another. A medical model might also be incomplete because it is based on empirically observed correlations rather than on well-understood causal processes. Since medical science is continuously evolving, new understanding will modify the rules for interpreting symptoms and prescribing therapy. Finally, because medical knowledge bases contain judgmental knowledge relating to social costs and benefits, they always reflect the values of their designers, which might change over time. In general, knowledge bases are incomplete, approximate, and biased models of the world.

Knowledge bases are also incomplete with respect to level of detail as a model. We can always ask why a statement is true; in medicine we would then delve successively into biology, chemistry, and physics. Since we do not represent everything we know on all levels of explanatory detail (and at some level of detail *everyone* experiences a failure of detailed mechanistic understanding of biologic processes), the knowledge bases we build are necessarily incomplete. One reason for this incompleteness is that there is no practical way to build systems today that know more than a fraction of what any physician knows about the body and how it works. There is just too much knowledge, and we are still struggling to formalize even small portions of it. A second reason for leaving out levels of detail is that useful problem-solving performance can usually be produced even if we leave out pathophysiological knowledge about disease. However, when we push a program to resolve multiple diseases or to deal with an unusual presentation of a disease (one that tends to "violate the rules"), these simplistic models break down.

1.4 Design Criteria for Medical Knowledge Bases

The incompleteness of medical knowledge bases requires that they be built incrementally, both to allow for the difficulty of building a complete model at any time and to allow for improvements to the knowledge base as human experts learn and social judgments change. Perhaps the most important design feature that makes a knowledge base easy to maintain over time is *modularity*: ideally, there should be no side effects or complex interactions among parts of the knowledge base.

Modularity in a sense boils down to a matter of indexing. The level at which we wish to change the system should be easily accessible, so we do not have to wade through complex code to make a change. In knowledge-based systems, one solution is to index knowledge according to how it is used and modified during reasoning. For example, in a rule-based system, it is convenient to index rules by the disease diagnoses they support.

Modularity and indexing suggest that the knowledge base be *structured* according to dimensions that make it easy to use and maintain. To have an indexing scheme, you need primitives for the dimensions of indexing, just as we have the idea of alphabetic ordering for assembling phone books. In medical AI we are led to study and formalize primitives such as *subtype, cause, etiology,* and *specialization.* An important open question is to determine the set of primitives that could be used to describe the temporal properties of any disease.

Of course, creating a well-structured knowledge base is just part of building a consultation system. How will the knowledge be applied to solve problems? To give a very practical example, suppose you want to confirm the presence of a particular disease. Should you seek evidence for all of its manifestations? In what order should you consider them? When should you focus on another hypothesis? You need a *procedure* for doing diagnosis. This procedure is often called *control knowledge* because it controls how the specific knowledge about diseases is applied to solve problems.

The primitives for representing control knowledge are different from those for representing disease knowledge. Concepts like *iteration, steps, subroutines,* and *conditional actions* are useful. Stating control knowledge *explicitly* and *separately* from the disease knowledge offers a big bonus—the disease knowledge can be used in multiple ways; different procedures can be used to interpret it. For example, a knowledge base might be used both to provide consultative advice and to tutor a student (see Chapters 5 and 11). Such a separation also enhances the ability of a program to explain its reasoning, an important concern for the acceptability of the system to its ultimate users. But not all systems are designed this way.

To summarize the important points we have made about the design of knowledge bases:

- A knowledge base is inherently incomplete; it is common for only one level of knowledge to be represented.
- To allow for incremental development, easy maintenance is important.
- Maintenance and explanation are enhanced by modular, well-structured, explicit statements of disease relations and diagnostic procedures.
- A well-structured knowledge base can be used in multiple ways.

1.5 Basic Concepts of Knowledge Representation

In the study of knowledge-based programs, such as the dozen or so systems described in this book, it is useful to consider a number of representation issues.

First, it is important to make a distinction between *what kind of knowledge is represented* and the *representation language* itself. A researcher is indicating what kind of knowledge is represented when she says, "CENTAUR's knowledge base contains descriptions of prototypical patterns of diseases" (Aikins, 1980; 1983). She is describing the representation language when she says, "Disease manifestations are represented as 'prototype components,' as slots in a 'frame.' " In general, it is most helpful to understand what kind of knowledge is represented in a system before trying to grasp the representation language. For example, we can compare CASNET (Chapter 7), ABEL (Chapter 14), and RX (Chapter 17) in terms of the kinds of causal facts about diseases that each represents. The implications for problem solving can then be considered: how might RX use ABEL's multiple-hierarchical representation to better explain data base correlations, for example?

A representation language is just a notational device. The important properties of a representation language include the *brevity* and the *explicitness* with which certain kinds of facts can be stated. For example, when approaching a system for the first time, it is useful to ask how causal, taxonomic, and temporal relations of disease are represented. In a system like MYCIN, such facts are stored only implicitly, but problem-solving behavior can be modified in a direct, concise way. It is also important to ask how the diagnostic procedure is represented. Can the knowledge base be thought of as a network that is interpreted by a separate diagnostic procedure (as in ABEL, CASNET, INTERNIST, NEOMYCIN, PIP, RX, and XPLAIN)? Or is the procedure implicit, inseparable from the knowledge base (as in the original Digitalis Therapy Advisor, MDX, MYCIN, PUFF, and VM)?

It is important not to get confused about external representations (diagrams linking findings and diseases), technical arguments about formalism (rules versus causal networks), and the description of the kind of knowl-

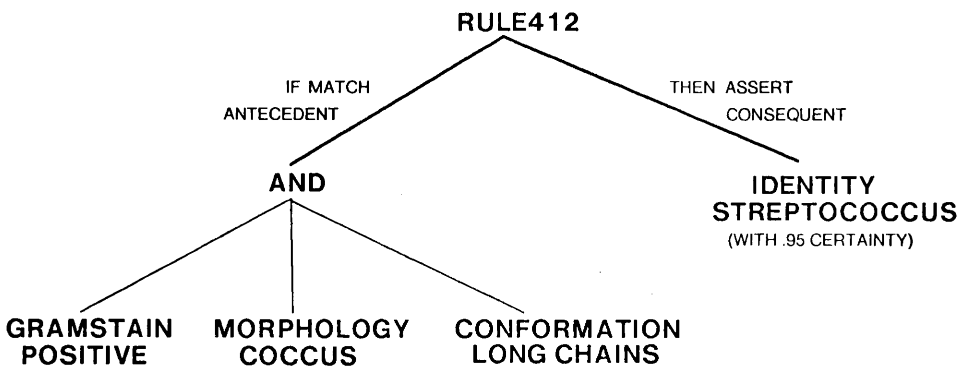

FIGURE 1-1 MYCIN rule viewed as nodes and links.

edge represented (e.g., disease prototypes). All of the jargon aside, what kinds of knowledge are factored out and represented explicitly? A beginner first studying this field should try to understand what the drawings and figures are showing about the kind of knowledge represented before worrying about the technical AI terms (such as production rule and frame). For example, Figures 7-1, 13-1, 14-7, 15-2, and 16-5 describe the kind of knowledge represented. When an internal name is mentioned (such as MYCIN's LABDATA), the important thing to understand is when the label is applied and how it is interpreted by the program (if a finding is marked as LABDATA, MYCIN will ask the user to supply a value before trying to make inferences from what it already knows).

In thinking about an internal knowledge representation, remember that at a basic level a knowledge base is completely describable in terms of nodes and links. You might first figure out what can be a node and how the nodes are linked, then try to pin down how the links are used by the control knowledge. To give a simple example, in a rule-based system such as MYCIN each rule is a conditional expression, which can be represented as a node linking an antecedent (IF part) to a consequent (THEN part) (Figure 1-1). (The rule represented in the figure allows MYCIN to conclude that an organism is almost certainly a streptococcus if it is a gram-positive coccus growing in long chains.) That is the static description. During problem solving, if MYCIN determines that the antecedent of a rule is satisfied, it asserts (adds to its data base) what is specified by the consequent. That is how a rule node and its links are interpreted. The next step is to understand when this operation would be performed on a particular rule node (i.e., when a rule is invoked), what else happens when a new assertion is made, and so on.

It is almost always useful in computer science to think in terms of processes. Ask yourself:

What is the input?
(In MYCIN, a rule);
What is the process?
(Determining if the antecedent is satisfied and making assertions);
What is the output?
(An updated data base of facts and beliefs about the patient).

In AIM knowledge bases, a *node* corresponds in general to medical concepts, for example, patient data and diseases. It is important to understand how the nodes are "marked" as a consultation proceeds. If a particular patient has a disease manifestation, for example, the internal representation is marked to indicate this fact. Furthermore, a link might be added to a particular disease node, indicating that, in this patient, this manifestation is believed to be caused by the disease. In this way a *patient-specific model* (see Chapter 14) is constructed as a constellation of possible findings, diseases, and connections among them. An example of a patient-specific model for the MYCIN domain is shown in Figure 1-2.

Another helpful consideration is to remember that a node might stand for an object (such as a CSF culture), a disease process (an infection), a patient-state description (increased brain pressure), an event (the onset of a headache), or a hypothesis (the belief that the patient has meningitis). Also, a node might stand for a *concrete* entity, such as a particular CSF culture, or an *abstract* one, such as the concept of cultures in general. In this way, general classes or categories can be described in a knowledge base.

A *link* is a relation between nodes. A *causal link* between two disease nodes indicates that one disease causes the other. From a simple perspective, links are labeled pointers that group findings and diseases into networks. For example, a taxonomy of diseases might be constructed by linking diseases with a *subtype link*. Links can also indicate spatial relations, levels of detail, examples of a general concept, etc. An issue of major concern is the interpretation of a link during problem solving. If a link indicates that a finding is "caused by" a disease, for example, does this mean that the program will list this disease as a diagnosis in its output? Is the link annotated in some way to indicate the conditions under which the causal process holds? If there are other causal links (and hence explanations) for this finding, how are they taken into account? The complexity of links and how belief about diseases is propagated through a network are major AIM issues.

1.6 Dimensions for Comparing Knowledge-Based Systems

Knowledge-based systems can be studied and compared along the following dimensions (with illustrative systems indicated in parentheses):

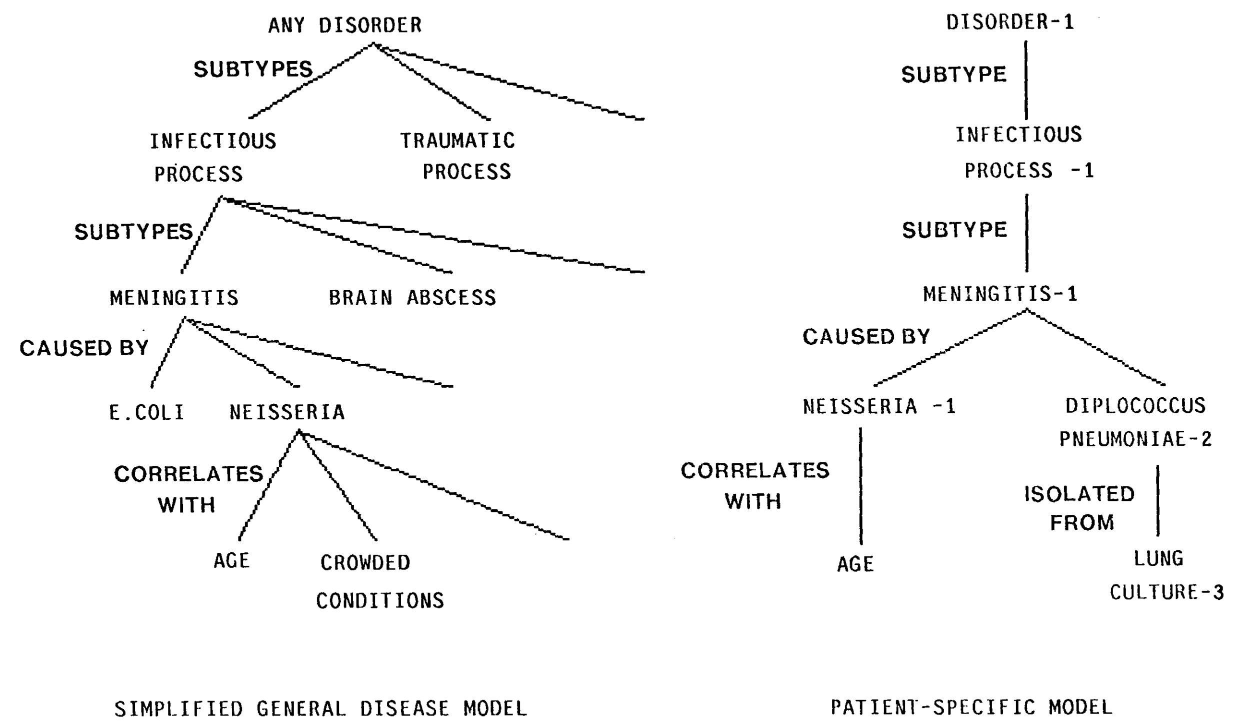

FIGURE 1-2 How a patient-specific model relates to general disease knowledge.

- *Content:* What kind of medical knowledge does the knowledge base contain? Programs typically contain (heuristic) links between findings and diseases (MYCIN). They sometimes contain pathophysiological descriptions of disease processes (CASNET). They rarely contain anatomical descriptions (CADUCEUS).
- *Structure:* How are the nodes and links organized? Programs typically contain hierarchies of various kinds (CENTAUR, INTERNIST, MDX, NEOMYCIN). They sometimes contain levels of abstraction (ABEL). No current programs contain multiple models or perspectives describing a single disease process.
- *Hypothesis formation and evaluation:* How does the program use links to make inferences (the *reasoning strategy*)? Most programs generate hypotheses from given data and use a hypothesis-directed *questioning strategy* (CASNET, INTERNIST, MDX, PIP). Many consider diagnoses in a focused, nonexhaustive way (INTERNIST, NEOMYCIN, PIP). A few attempt to model human reasoning (MDX, NEOMYCIN, PIP).
- *Management of uncertainty:* How does the program represent and cope with uncertain information? Most programs represent hypotheses with some degree of uncertainty, using a *scoring mechanism* for combining evidence and comparing hypotheses (CASNET, INTERNIST, MYCIN). No current programs cope with inconsistent evidence by reasoning about the justification for inferences.
- *Data collection:* How does the program acquire information about the problem? Most programs ask questions of the user, requiring keyboard input. No current programs allow a true mixed-initiative interaction. No current consultation programs can accept data from on-line medical data bases, although some data interpretation systems have been interfaced with patient monitoring devices (PUFF, VM) and other analytic devices (EXPERT/Electrophoresis).[1]
- *Explanation and knowledge acquisition:* What methods are available for building and testing the knowledge base? Some programs have a means of displaying the network in English form (MYCIN). Many have some form of "audit trail" so the reasoning can be traced back in debugging (MYCIN, NEOMYCIN, XPLAIN). Only a few programs have a user model to facilitate the interaction (GUIDON, XPLAIN). None of the programs truly learn from experience, but several detect patterns in the knowledge base (MYCIN), a patient data base (RX), or a case library (SEEK) as an aid in knowledge acquisition.
- *Meta-knowledge:* What is implicit in the knowledge base? What does the program know about its own design? Most programs attempt to separate

[1]The HELP system (Pryor et al., 1982) is a good example of a non-AI program that assists a physician by accessing an on-line data base. A recent AI system named ONCOCIN (Shortliffe et al., 1981) uses an on-line data base of patient information, but requires that *current* data be entered by the user.

out disease knowledge from the diagnostic procedure. Some programs have explicit knowledge about the principles underlying the network (ABEL). A few programs have abstract, nonmedical knowledge about their procedures and representation (MYCIN, NEOMYCIN).

The list above is meant to illustrate features to look for when studying medical systems. The progression from typical to rare parallels what many researchers would hold to be desirable and helps identify many of the key research areas for the next decade.

1.7 Are Knowledge-Based Systems Textbooks of the Future?

How is an ideal medical knowledge-based system different from a medical textbook? Understanding the differences might help the reader understand what researchers are trying to do in these programs, what makes their task difficult, and what they might potentially achieve for humanity.

We proceed by describing medical textbooks according to the dimensions for comparing knowledge-based systems given above:

- *Content:* A medical textbook typically contains all kinds of anatomic, disease process, and heuristic knowledge. As stated above, AIM systems currently tend to model only high-level associations between findings and diseases.
- *Structure:* Organization of textbooks is always of paramount concern. Different textbooks might organize the same knowledge along different dimensions (e.g., either by disease entity or by presenting complaints). However, sharp distinctions are generally not made about the kinds of knowledge being presented; diagnostic procedures tend to be interwoven with medical facts. Knowledge is usually not clearly stated on multiple levels of detail; a given textbook generally adopts one viewpoint. There is little discussion of the epistemological terms used, such as causality and subtype—they might even be used in a confused way. While this can also be true of programs, the requirements and trend for new systems is to articulate and be precise about these kinds of distinctions.
- *Hypothesis formation and evaluation:* Medical textbooks tend to give disease-specific relations for considering and confirming the presence of a disease. Consideration of the practical problems of diagnosing multiple diseases, separating complication from cause, ruling out diseases, weighing evidence—all of which must be formalized in a knowledge-based system—is generally not treated in a general way in textbooks. Typically, only tips and rules of thumb are given. Most importantly, knowledge-

based systems are *programs* that can solve problems. Textbooks sit there inertly, relying on you to search for the relevant facts and then to infer solutions on your own.

- *Management of uncertainty:* Books are often vague about how to interpret evidence, using words such as *often, suggest,* and *rarely seen,* without specifying a consistent interpretation for these terms or how to use them to solve problems. Programs require the discipline of at least some kind of "scoring function" for assigning weights and combining evidence, although this is often an *ad hoc* scheme that is adjusted until it works well enough. Resolving multiple diagnoses with a large number of findings requires information about importance, frequency, risk, cost, discomfort, etc., as well as the development of a more precise understanding of the conditions of causal processes (as in Chapter 14)—crucial information for successful knowledge-based programs, but often ignored in textbooks.
- *Data collection:* Some medical textbooks have very good discussions of the problem of interviewing a patient. But given all of the commonsense knowledge involved, current knowledge-based programs cannot help the consultation user to supply accurate information. For example, they cannot explain how to recognize lethargy in a patient or how to distinguish between coccus in long chains and rods. Certain assumptions about the user population are generally not stated explicitly, so the systems are missing meta-knowledge about their own design. But knowledge-based systems can actually use their knowledge to collect data from a data base or to conduct an interview; a textbook obviously must leave these tasks to the physician.
- *Explanation and knowledge acquisition:* Textbooks use graphic techniques, as well as prose, to explain complex relations and disease processes. While a textbook might have a good glossary, you cannot ask it for clarifications. You cannot pose hypothetical questions or give it new knowledge and see how the answer changes (see Chapter 11).
- *Meta-knowledge:* Some authors do a good job of describing the organization of their book. Arbitrary perspectives are possible for making a physician self-aware, for example, about how to speak to a patient, how to interpret culture results, and how to organize multiple problems on paper. However, medical AI is contributing new meta-knowledge about the structure of disease knowledge and the abstract character of diagnostic procedures. These topics are typically sacrificed in books to an emphasis on facts and are also often ignored in formal medical education.

So, while textbooks and knowledge-based systems are both knowledge repositories, a program has many potential advantages. Because knowledge can be represented independently from its use, a given knowledge base can be interpreted for multiple applications. Of course, in talking about knowledge-based systems above, we are referring to the knowledge base plus the interpretive procedures. These include procedures for doing diagnosis, for teaching, for learning, etc. In most cases, developing such procedures is at least as difficult as developing the knowledge base itself. The construction process usually goes on in parallel, even when the knowledge is represented independently from how it is to be used. Developing these procedures involves substantial research. For example, consider the difficulties of developing the domain-independent teaching rules of GUIDON (Chapter 11) and how this involves basic research in education, and consider how developing the domain-independent diagnostic rules of NEOMYCIN (Chapter 15) involves basic research in psychology and medicine. Each of these is an expert system–building task in its own right.

Taking an optimistic point of view, a knowledge base might be the basis of any *intelligent agent*—a consultant, tutor, librarian, or decision analyst—responding actively to the needs of its user, capable of explaining itself, and, most importantly, capable of learning from experience (Clancey, 1983a). A well-designed knowledge-based program can be easily revised and cheaply copied. With the refinement of knowledge-based systems, textbooks might become like parchments, as antiquated as a mechanical calculator when compared to a personal computer.

1.8 Implementation of Medical AI Systems

Implementation includes the programming language (software) and the computer (hardware) upon which a system is constructed. With only one exception, the programs described in this book are written in the LISP (LISt Processing) programming language. A good introduction to LISP for the layperson is Winston (1977). LISP is chosen by AI researchers as much for its natural capabilities for representing knowledge networks (a *list* is a linked set of nodes) as for the powerful environment for building large programs that is offered by most LISP dialects [e.g., Interlisp (Teitelman and Masinter, 1981)].

The exception described here is EXPERT (Chapter 20), which is implemented in FORTRAN. FORTRAN was chosen because it is faster and runs on many different kinds of machines. However, it should be noted that FORTRAN cannot easily be used directly. Encoding meta-knowledge about the knowledge network design and interpretive procedures requires building a language on top of FORTRAN that makes it easier to reference a node by its name or by the kinds of links it has. Thus some key features of the LISP language must be added to FORTRAN to make it useful.

Medical AI programs generally run on machines with large address spaces. With one exception (INTERNIST, Chapter 8) it is currently the interpretive and maintenance software, not the knowledge bases, that takes up the space. This is likely to change as AIM systems grow in size and complexity. The current situation reflects both the youth of the field and the tediousness of building large knowledge bases. The software for these systems includes not only the program for interacting with the user during a consultation, but also knowledge acquistion and explanation programs. The address limitation makes it impractical to develop these programs on a 16-bit machine, typical of the small personal computers of the early 1980s. However, a system might be "downloaded" after development, particularly if user interaction is minimized (see Chapter 19). Over the past decade hardware costs have plummeted, and new personal machines have become available. The 32-bit "professional workstations" that run LISP and are just appearing on the market provide insight into the kinds of computing environments that will bring AIM systems to physicians in a cost-effective manner in the decades ahead.

1.9 What Is the State of the Art?

The dimensions for comparing programs are descriptive, but can be adapted to characterize the best that programs can do today. The following list gives some dimensions of quality with short descriptions of what representative programs have accomplished:

- *Performance:* Several systems that have been formally evaluated in statistical studies of their performance are CASNET, INTERNIST, MYCIN, and PUFF (Duda and Shortliffe, 1983). A typical finding is that program behavior is acceptable to 80% of the evaluators, but evaluators usually disagree as much among themselves. INTERNIST currently has by far the best capability to deal with a wide variety of problems.
- *User interaction:* MYCIN set the standards for user interaction in terms of providing spelling correction, a nicely laid-out question-answer format, and English input of simple questions. The ONCOCIN program (Shortliffe et al., 1981; Bischoff et al., 1983) is adapting standard ways of filling out a patient's chart to the demands of a consultation program.
- *Explanation:* MYCIN was the first consultation program to provide an explanation facility. The therapy program was redesigned to enable it to present concise explanations of the optimization process (Buchanan and Shortliffe, 1984). The tutorial features of GUIDON (Chapter 11) provide more individualized display of reasoning. XPLAIN (Chapter 16) deals with explanation of procedures and methods for providing multiple levels of detail. Structuring the knowledge representation for the purpose of explanation is the focus of NEOMYCIN research (Chapter 15).

- *Representational adequacy:* While representation is at the heart of all medical AI research, some programs have been constructed specifically to solve problems with earlier representations. Perhaps the most complex examples are the multiple disease relations of CADUCEUS (Pople, 1982), the levels of causal detail of ABEL (Chapter 14), the abstract meta-rules of NEOMYCIN (Chapter 15), and the refinement structure of XPLAIN (Chapter 16).
- *Actual use:* Of the AIM systems with which we are familiar, only PUFF (Chapter 19), EXPERT/Electrophoresis (Chapter 20), and ONCOCIN (Bischoff et al., 1983) are developed to the point of being used routinely by physicians.
- *Psychological model:* Programs developed specifically as models of human reasoning include PIP (Chapter 6), MDX (Chapter 13), NEOMYCIN (Chapter 15), and CADUCEUS (Pople, 1982). The program reported by Johnson et al. (1981) has been evaluated to determine its accuracy as a model.
- *Knowledge acquisition:* The state of the art is represented by the packages of EMYCIN/TEIRESIAS (Davis, 1979; van Melle, 1980) and the interactive features of SEEK (Chapter 18).

1.10 Organization of This Book

We close this introductory chapter with a brief discussion of the papers we have selected for the book. Because many excellent projects and papers could not be included, we urge the reader to make use of the extensive bibliography we have provided. The papers cited there will provide valuable additional insights regarding many of the issues raised in this volume.

The first chapter, by Gorry, introduces the rationale and advantages of applying AI approaches to medical problem solving. This is followed by an extensive survey of computer-based clinical decision aids; of particular interest is the description of traditional algorithmic, statistical, pattern recognition, and decision-theory approaches.

Kulikowski then provides an introductory overview to the early AIM systems, placing them in the context of the expert systems subarea of AI that partially grew out of them. These classic systems are then described: MYCIN (known for its use of rules), PIP (use of frames), CASNET (use of a causal-associational network), and INTERNIST (handling of multiple problems). The chapter by Szolovits and Pauker makes detailed comparisons of representation issues handled by these four programs.

The next chapters describe two medical developments from the MYCIN program: VM and GUIDON. [Buchanan and Shortliffe (1984) provide a complete survey of the MYCIN project and its spinoffs.]

Representation issues in modeling physicians' reasoning are considered by Feltovich's psychological study, MDX, and ABEL.

NEOMYCIN and XPLAIN are contemporaneous, second-generation systems designed to enhance explanatory capability. The approaches are complementary, so it is useful to consider these programs together.

Knowledge acquisition in the form of partially automated learning is considered by RX (based on statistical analysis of a data base) and SEEK (based on analysis of case experience).

The development of two practical, routinely used programs is described: the implementation of PUFF in BASIC running on a minicomputer and of the EXPERT/Electrophoresis system running on a microprocessor.

In the last chapter, we take a step back to consider some of the practical issues regarding the AIM field. To what extent will the complicated systems we are developing ever be used? What are the principal research challenges that remain? Will medical expert systems be viewed as beneficial tools, or as threats to physicians or to the sanctity of the physician-patient relationship? Difficult questions such as these can be answered more realistically now that we have had a decade of solid experience with AIM research and have had more time to observe the evolution of society's attitudes toward computers and the remarkable revolution in hardware technology. Both of these developments are changing our predictions about the future, and they permit an optimistic view of medical AI and its potential for beneficial clinical use.