Computer-Assisted Diagnosis of Orthopedic Gait Disorders

Kathleen B. Tracy
Elaine C. Montague
Richard P. Gabriel
Barbara E. Kent

March 1979

What is this?

In 1976 or so I helped my then-wife, Kathy Tracy, and her friend Elaine Montague with their Physical Therapy Master's dissertation. I wrote a program that did gait diagnosis (walking problems). Barbara Kent was the professor who was expert in such diagnoses. They came up with the rules, tuning them on lots of cases. I wrote the code in Lisp.

The original dissertation was written in POX (*Prototype Overlay Xerographics*), designed and implemented by Robert E. Maas. It became available around the same time as Tex, perhaps a little earlier. I translated the files to Tex.

This file is actually the paper that we published in the Physical Therapy & Rehabilitation Journal. Submitted: June 16, 1977; accepted: April 14, 1978; published: March 1, 1979.

Abstract

A computer program was developed to aid in the diagnosis of orthopedic gait disorders. The processes involved in designing and implementing the program are described, as well as the program's method of operation. The main features of the program include: a knowledge base composed of facts about orthopedic gait organized into premise-conclusion pairs; a *goal directed reasoning chain* which causally relates the facts; and a symbolic structure which allows limited English discourse between the user and the computer. Results of the research indicate that the complex area of gait analysis does lend itself to diagnosis by computer, and that this prototype has potential as an aid to physical therapists in the classroom and in the clinic.

Introduction

Computer technology has become an integral part of the health care system and its influence will continue to grow with technological advancement. A review by Barnett, [Barnett 1968] emphasizing the growth of computer applications in the medical field, offers a thorough explanation of technological implementation in several major areas of health care. Among the areas discussed by Barnett, computerized medical diagnosis seems to present the greatest potential for growth and further development. Such medical programs were originally designed for physicians to aid in the diagnosis of disease. In practice, however, diagnostic programs were primarily used by medical students. Some medical schools have incorporated computerized diagnostic programs as self-instruction aids into their curricula [Entwisle 1963] [Feurzeig 1964].

The two main uses for medical diagnosis programs are clinical and educational. As the diagnostic reliability of such programs increases, and as the cost and size of computers decreases, programs of this type will be able to help the clinician is diagnosing difficult problems. Often the complexity of a situation or a forgotten fact may result in an incorrect diagnosis. Computer diagnosticians would always have all of the relevant information available and would make reliable, repeatable decisions.

Studies showing that self-instruction requires less time to achieve satisfactory performance than traditional lecture-demonstration instruction [Asklund 1976] [Campbell 1970] [Rutan 1973] support the use of computer diagnosis programs for students as a faster method of achieving satisfactory performance. According to the Carnegie Commission's report on computers and higher education, [Rockart 1975] computers offer advantages over traditional education by giving the learner more control over his learning environment and allowing him to be more active in the educational process. Thus, a computerized diagnosis program could offer the student a more efficient and perhaps more effective method of learning disease diagnosis, especially as the cost of computer facilities decreases.

Presently, most computer-assisted diagnosis programs are limited to the domains of physicians and medical students. [Rockart 1975] [Warner 1961] [Overall 1963] Since physical therapy is concerned with the systematic diagnosis of many orthopedic and neurological problems it is also a likely field for a computer-assisted diagnosis program. This research project was designed to create a computer program to aid in the diagnosis of orthopedic gait disorders. The following discussion will familiarize the reader with the growth and change of computer programs used in medical diagnosis, describe the gait diagnosis program and the processes involved in designing such a program, and hypothesize regarding the applicability of such a program in the clinic and in the classroom.

History of Computers in Medical Diagnosis

Early computerized medical diagnosis presented many problems to computer and medical specialists. According to Barnett [Barnett 1968] the major problem resided in inconsistent methods inconsistently applied by physicians while establishing a diagnosis. However, in a limited number of cases medical professionals sufficiently organized their methods of diagnosis to create successful diagnosis programs, as in congenital heart disease, [Warner 1961] thyroid disease, [Overall 1963] and infectious disease [Shortliffe 1976]. Most physicians, however, were skeptical of using computer diagnosis programs in clinical practice, and therefore such use was limited.

Entwistle's program [Entwisle 1963] is an example of an early medical diagnosis program which was used specifically as a teaching aid for medical students. Such early programs merely correlated discrete signs and symptoms with specific diseases. Diagnosis of a disease was postulated only if a statistically significant number of symptoms correlated with the disease. Entwistle's program was equipped to diagnose six possible diseases. Students were given a list of symptoms about which they could inquire; the computer would respond 'yes' or 'no' to specific inquiries depending on which of the six diseases the program had randomly chosen to diagnose. When the student felt sufficiently informed to make a diagnosis he could check himself against the machine's diagnosis.

Later medical diagnosis teaching programs took slightly different approaches. For example, Feurzig [Feurzeig 1964] designed a diagnosis program which required the student to inquire about symptoms as if the student were conducting an orderly physical examination. If the student asked about a symptom out of the predesignated progression of the physical examination, the computer would refuse to give any information about that symptom, and it would direct the student to an appropriate symptom. Feurzig's program took on a more personal character than Entwistle's; the yes-no response was replaced by a short stereotyped phrase and positive reinforcement was given intermittently for correctly proceeding through the predesignated steps of the physical examination.

All of these early programs shared common problems. According to Weber [Weber 1972] a major obstacle was the inability of the computer to interact directly with the user in English. This problem was compounded by the fact that most medical professionals were unfamiliar with programming languages. Shortliffe [Shortliffe 1976] outlined another major limitation in early diagnostic programming: these programs operated on a purely statistical basis. That is, diagnosis was postulated only if a statistically significant number of symptoms correlated with the disease; therefore the program itself had no knowledge or *understanding* of the problem area. The program could not explain to users its rationale for certain conclusions, since that rationale was buried in the statistical analysis and was not part of a logical chain of reasoning. Due to these problems in the early systems, further development of diagnostic programming reached an impasse until the late 60's.

The most important improvement in medical diagnosis programs came with the development of Artificial Intelligence [AI] [Shortliffe 1976] as a subfield of computer science. The contribution to traditional

computer science was the attempt at symbolic reasoning—*thinking*. No longer did a program involve only arithmetic calculations and/or information storage and retrieval. Programs were being written to solve problems in a manner similar to that of the human mind; machines could now be programmed to *think* in a limited sense and to begin to communicate as a human being.

More specifically, AI offered techniques for organizing a knowledge base built of facts which can be causally related by a *reasoning* program and insights into methods which can be used by that program to accomplish the *reasoning*. This causal chain of reasoning can then be used to explain the logical connection between symptoms and diagnoses to the user. Techniques for organizing the knowledge base came from the development of *symbolic* rather than numerical programming languages. A symbolic programming language manipulates abstract symbols rather than arithmetic quantities, and this ability is crucial to the task of building structures which represent facts in a recognizable form in the computer. Operations such as *searching, pattern matching, and backtracking* were found to be useful in modelling human problem solving behavior, and so provided a simple *machine reasoning* framework in which to build a flexible diagnosis program.

These features offered the possibility of exploring the *thought processes* used by the computer in reaching conclusions. Consequently, such a reasoning program could justify to users the rationale for its conclusions. Symbolic representations, as opposed to numerical or statistical representations, lend themselves more readily to English translation, helping remove the language barrier between medical professional and computer.

The most important of the diagnostic programs to use these developments was MYCIN, written by Shortliffe [Shortliffe 1976] in 1974. His program was a rule based program for diagnosing infectious diseases and prescribing anti-microbial therapy. The *knowledge base* consisted of *rules* in the form of *premise-conclusion* or *if-then* pairs. An *inference engine* or *reasoning program* would operate on these rules, asking question concerning the patient's symptoms of the physician, combining these facts with information in the knowledge base, and producing a diagnosis and therapy recommendation. The rules served the purpose of relating symptoms and infectious diseases in a *causal* relationship. In addition, the rules were of a probabilistic nature, and so inferences made by the program were seldom of an "all or nothing" flavor. This program was designed for use by clincians who are unreliable selectors of anti-microbial therapy. [Shortliffe 1976]

The gait diagnosis program was based on the philosophy and methodology of the MYCIN program.

Description of the Gait Diagnosis Program

A gait diagnosis program was envisioned that would: contain the multitude of facts in the knowledge base pertaining to gait deviations; relate the facts in a way that parallels a physical therapist's method of reasoning; diagnose the muscle weaknesses or tightnesses for any particular gait deviation using the knowledge base and information supplied by the user; converse with the user in natural language, English (as opposed to an artificial or computer language); and describe to the user the line of reasoning which lead to a particular diagnosis.

A demonstration of a simple dialogue between the computer and a physical therapist is illustrated in the Chart (at the end of this paper).

Formulation of the Knowledge Base

The process of writing the gait diagnosis program began with the creation of a *knowledge base* consisting of 345 facts relating to orthopedic gait disorders. This knowledge base was derived from relevant literature [Blount 1956] [Chapman 1968] [Inman 1968] [Inman 1966] [Klopsteg 1968] [Murray 1964]

[Murray 1967] [Perry 1967] [Saunders 1953] [Smidt 1974] [Berkeley 1947] [Kent 1976], clinical observation, and consultation with an expert in gait diagnosis, Prof. B. Kent, personal communication, November 1975. The facts in the knowledge base were arranged in premise-conclusion pairs called *rules*. There are three basic types of rules: 1) Descriptions of deviations; 2) Muscle weakness/tightness; and 3) Negatives.

Description of deviation rules are definitional in nature: technical terms such as *footslap* are described by a set of positional and dynamic descriptors associated with each involved joint where relevant (e.g. *cannot decelerate plantar flexion* defines *footslap*). These descriptors include symptoms above and below the joint at which the deviation occurs, the side(s) of the body affected, and the phase(s) at which the deviation occurs. The deviation is named in the conclusion. For example:

```
Premise:  If patient cannot decelerate plantar flexion
          on side A
          And this occurs at heelstrike of A
Conclude: Footslap on side A
```

The intent of this rule is that whenever the inability to decelerate plantar flexion during heelstrike on side A is seen, the program can conclude that the patient is manifesting the deviation known as footslap on that side. By means of this rule, the program is informed of the definition of footslap.

Muscle weakness/tightness rules relate deviations such as footslap with the muscle weaknesses and tightnesses which can be a cause of those deviations. These rules include in the premise the named deviation or a description of it; the conclusion states the muscle weakness(es) and/or tightness(es) responsible for the deviation along with a numerical certainty factor for each listing. The certainty factor is assigned according to the likelihood that the particular weakness/tightness caused the deviation:

```
Premise:  If patient exhibits footslap on side A
Conclude: Dorsiflexor weakness side A        .80
          plantar flexor tightness side A    .4
```

Notice that this rule is of the form "<deviations> imply <weakness/tightness>" rather than the other way around. These rules could have been entered into the knowledge base in the inverse format (i.e. <weakness/tightness> implies <deviations>) since it is possible to mechanically transform each form of rule to the other form.

In this example .80 is strongly suggestive evidence that dorsiflexor weakness is responsible for footslap; .45 is mildly suggestive evidence that plantar flexor tightness is responsible. The computation of the certainty factor will be explained more fully in a following section.

Negative rules are primarily used to inform the program that certain conditions are essentially mutually exclusive or to express the necessity of some deviations for particular weaknesses/tightnesses (i.e. <weakness/tightness> always implies <deviations>). Negative rules are of two forms: 1. If x, then not y; and 2. If not x, then not y. The negative rules in the knowledge base are mostly type 1 rules:

```
Premise:  If muscle A is tight
Conclude: Muscle A is not absent 1.0
```

Example of type 2 rule:

```
Premise:  If patient does not have difficulty decelerating
          plantar flexion at heelstrike
Conclude: Dorsiflexors are not weak .95
```

This rules says that dorsiflexor weakness is almost always accompanied by difficulty in decelerating plantar flexion. Thus the absence of this deviation will weigh heavily against weak dorsiflexors, despite a large amount of evidence in its favor.

The negative rules, although not completely valid, were successful in limiting the computer's options without affecting its diagnostic abilities greatly. For instance, there are negatives rules which assert that weakness and tightness of a particular muscle are mutually exclusive, even though this situation is possible; these rules tend to prevent diagnoses which mention both weakness and tightness of a muscle when only one had strongly suggestive evidence. Without such rules, small positive evidence for weaknesses/tightnesses could build up to fairly large support, even in the presence of strong negative evidence (absence of an essential deviation) or contradictory indications (convincing evidence of a contrary diagnosis). Another alternative, formulation of a more extensive knowledge base, was not possible within the time frame of the project.

These rules are called *negative* rules because a conclusion such as "dorsiflexors are not weak .95" is expressed as "dorsiflexors are weak -.95" within the program. These negative values count towards the *measure of disbelief* to be discussed below.

An important point to note is that, as shall be seen in detail later, that it is possible for rules to not be *completely* valid. This results from the use of certainty factors in the rule conclusions. The certainty factors of all relevant rules are combined in the final verdict, and a single certainty factor may not outweigh a bulk of evidence to its contrary. Negative rules point out that the use of these rules is not logical in the strict sense, but heuristic. That is, they are used as rules-of-thumb (via certainty factors) in sorting out the most likely diagnoses.

Problems encountered in formulating the knowledge base revolved around differences of opinion in the literature and between the authors regarding orthopedic gait factors. The problems were resolved by 1) including only information verified in the literature and agreed upon by the investigators; 2) adjusting/averaging numerical certainty factors; and 3) using adjective modifiers when there was too much controversy in determining the numerical certainty factor, such as using *excessive* internal rotation instead of listing exact number of degrees. This had the effect of allowing the user to judge for himself whether a condition was abnormal or significant, rather than strictly defining those terms in the controversial cases.

After the knowledge base was formulated it had to be translated into a language understandable to the computer. At this point a close working relationship between the physical therapists and a computer scientist specializing in AI was necessary. This specialist made the decision to use MacLisp (a dialect of the LISP symbolic processing language) for the implementation of the diagnosis program. LISP is especially well suited for the type of symbolic processing used by AI diagnosis programs, and the particular dialect, MacLisp, is widely available.

After the translation of the knowledge base, the AI expert began to superimpose the knowledge base onto the reasoning structures provided by MYCIN. However, many difficulties were encountered in using MYCIN as a model for gait diagnosis. Although amenable to a diagnostic program in infectious disease, MYCIN lacked three important structures necessary in the analysis of gait: First, there were no provisions for time sequencing of symptoms which would be an important feature in dealing with the phases of gait. Secondly, it was difficult for MYCIN to direct the program to consider a deviation in an organized anatomical manner from the trunk and then sequentially to the pelvis, hip, knee, and ankle. Finally, no convenient structures were available for considering both sides of the body simultaneously, such as weak hip flexors on the right may implying lateral trunk bending towards the left. Due to these problems, it was necessary for the AI specialist to write a new program with the necessary features.

Language Structure

The program converses with the user in a limited subset of English, since it was designed for health care persons who had no specialized knowledge of computers. The problems associated with the language input

(specifying deviations in sentences rather than codified form) were extensive. In fact, research in natural language understanding by computers is an area in which a great deal of effort has been spent with only limited success by scientists in artificial intelligence. The natural language understanding abilities of the gait diagnosis program is quite rudimentary but is of sufficient power to allow therapists to conduct a diagnosis without any specialized knowledge about an artificial computer language.

One language problem which appeared was the difficulty the machine had in understanding various descriptions of the same deviation, e.g. *excessive knee flexion at midstance, 40 degrees flexion at midstance, and knee flexed more than 30 degrees at midstance.* This problem was somewhat resolved by the creation of a *parser*, a subsystem created by the AI expert and incorporated into the gait diagnosis program, which transformed all synonymous sentences into a canonical internal or machine format, which was used by the program to determine the deviations described.

Method of Operation

The program begins by asking the user for the description of a gait deviation. After the deviation is given, the computer checks to see 1) if it knows the particular deviation being specified; 2) if the phase has been specified; and 3) if the side i.e. right or left is known (either specified explicitly or implicitly). If these three criteria are met the program may be instructed to proceed with the diagnosis. If one or more criteria are missing the program directs the user to supply the necessary information.

If the deviation is not found in the knowledge base the program states: "Could not figure out the deviation" and provides the user with a list of joints. The user then chooses the joint where the the deviation occurs. For example:

```
Computer:    Deviation:

Therapist:   Patient walks with a limp.

Computer:    Could not figure out the deviation...
             1       Trunk
             2       Pelvis
             3       Hip
             4       Knee
             5       Ankle
             Select a joint or type "ok":

Therapist:   3
```

The program then displays a listing of the possible deviations occurring at the specified joint.

```
Computer:    Hip

             1       Circumduction
             2       Difficulty decelerating the limb
             3       Excessive external rotation
             4       Excessive internal rotation
             5       The hip extended less than 10 degrees
             6       Excessive hip flexion
             7       Hip hiking
             8       The hip and knee are flexed more than normal
             9       The leg follows a laterally curved path
             10      The patient takes a quick step
             11      The patient takes a short step

             Please specify the desired deviations by typing
             their numbers
             Terminate the list with "ok":

Therapist:   10 ok
```

The therapist has thus described the deviation, quick step, in a manner understandable to the program.

Since one description of a deviation may be recognized more easily than another, some synonymous descriptions are included in the listing. For example, *circumduction* and *the leg follows a laterally curved path* are synonymous.

Phase specification is handled somewhat differently. Most deviations are associated with a particular phase of gait; therefore in the majority of cases the machine can conclude the phase once the deviation is known. However, some deviations, such as excessive knee flexion, occur during different phases of gait depending on the muscle weakness/tightness involved. The program deals with this type of deviation by listing the possible phases during which the deviation may occur and the user then types the appropriate phase. For example:

```
Computer:    Deviation:

Therapist:   Excessive knee flexion on the left side.

Computer:    Improper or missing phase specification...
             1       mid-swing
             2       stance
             Select the desired phase by typing the
             corresponding number:

Therapist:   2

Computer:    Determined deviation:
             The knee is flexed more than 20 degrees
             Determined phase: stance
```

When the side is not specified, the request “Please specify (SIDE)” is given. At this point the user can type “right,” “left,” “left and right,” or “both.”

Notice in the above example, the program paraphrases its understanding of the deviation so that the user can re-phrase his specification. A response of "No" will cause the program to accept an alternative deviation.

In summary, before a procedure for diagnosis begins, the program must: 1) identify the deviation in the knowledge base, 2) obtain a proper phase specification, and 3) specify the side of the deviation.

Process of Diagnosis

After the deviations are determined the user choses the method he wishes to use in diagnosing the problem. The methods include: infer, diagnose, and menu (diagnose).

Infer

If the person selects "infer," the machine assumes that no more information is needed to make the correct diagnosis and the program proceeds without asking any questions. For example:

```
Computer:    Deviation:

Therapist:   The patient's left foot slaps.

Computer:    Determined deviation: Foot slap
             Determined phase: heelstrike

             Deviation:

Therapist:   ok [means no more deviations to add]

Computer:    Infer, diagnose, or menu (diagnose)?

Therapist:   infer

Computer:    Inferring ...
             Left
             The dorsiflexors are weak  .85
             The plantar flexors are tight  .45
```

Diagnose

If "diagnose" is chosen, the machine assumes additional information may be needed; that is, other deviations may need to be supplied. The user is asked a series of questions about gait deviations requiring a yes or no response. "Diagnose" questions progress from the trunk down to the ankle; therefore questioning begins by trying to determine muscle weakness/tightness responsible for trunk deviations first. For example, the following is the beginning of a Diagnose session with no initial deviations (the program lists the rules it is using as they are used; this way the therapist can check the line of reasoning later):

```
Computer:    Infer, diagnose, or menu (diagnose)? diagnose

Therapist:   diagnose

Computer:    Considering: The back extensors

             Is the center of gravity anterior to the knee on
             the right during end of swing through midstance?

Therapist:   yes

Computer:    Invoking RULE214 [the computer lists the rules invoked]
             Invoking RULE242
             Invoking RULE10
             Invoking RULE318

             Is the upper trunk hyperextended during heelstrike
             through midstance on the right side?

Therapist:   no

Computer:    Invoking RULE326
             Invoking RULE324
             Invoking RULE322

             Does the pelvis tilt anteriorly less than 3 degrees during
             heelstrike through midstance on the right side?
```

Menu (Diagnose)

If "menu (diagnose)" is selected the machine also assumes that there is more information to be gathered. Unlike "diagnose," however, the person can direct the program to find out information regarding a specific muscle weakness/tightness. After a person chooses "menu (diagnose)" a listing appears from which s/he selects the muscle group(s) s/he wishes to consider. For example:

```
Computer:    Infer, diagnose, or menu (diagnose)?

Therapist:   menu

Computer:
          1       The back extensors
          2       The abdominals
          3       The hip flexors
          4       The hip extensors
          5       The hip external rotators
          6       The hip internal rotators
          7       The hip abductors
          8       The gluteus maximus
          9       The quadriceps
          10      The hamstrings
          11      The knee flexors
          12      The posterior tibialis muscle
          13      The anterior tibialis muscle
          14      The peroneals
          15      The plantar flexors
          16      The dorsiflexors
          Select the desired muscles by typing the
          corresponding numbers/
          Terminate the list with "ok":
```

Menu (diagnose) allows the user to consider any number of muscles and to specify the order in which they will be considered.

Method of Reasoning

Backward Reasoning Chain

The program reasons by relating the facts in the knowledge base; it uses a *backward reasoning chain* in which the conclusion is proven by validating the premise. The reasoning chain is in the form: D→C, C→B, B→A.

In order to prove A, the program will try to validate B. B will be validated by proving C. C will be validated by proving D. Thus, given the rule:

```
Premise:  If patient exhibits trendelenberg gait on side A
          during swing
Conclude: Hip abductors weak    .95
```

The system will attempt to prove the hip abductors are weak by verifying that the patient has trendelenberg gait. It will verify that the premise (trendelenberg gait) is true if any of the following conditions exist: 1) if the user has typed in that the patient has trendelenberg gait; 2) if the user responds "yes" when asked by the machine if the patient has a trendelenberg gait; 3) if the machine can prove trendelenberg gait by applying relevant rules from the knowledge base. The machine performs 3 in the following manner: Since the machine has the rule B→A:

```
B    Premise:  If patient exhibits trendelenberg gait on side A
↓              during swing
A    Conclude: Hip abductors weak    .95
```

it searches for rule(s) with B in the conclusion, C→B:

```
Premise:  If C
Conclude: Trendelenberg gait.
```

Example:

```
C    Premise:  If pelvis drops more than 8 degrees
↓              on side A during swing
B    Conclude: Trendelenberg gait on side A
```

The machine will try to verify C (pelvic drop more than 8 degrees) by looking for rules with C in the conclusion, D→C:

```
Premise:  If D
Conclude: Pelvic drop more than 8 degrees on side A
```

Example:

```
D    Premise:  If abnormal pelvic drop on side A during swing
↓
C    Conclude: Pelvic drop more than 8 degrees on side A
```

If D can be validated C is verified. Having proven that trendelenberg gait is true the machine can now verify A i.e. hip abductors are weak.

In short, reasoning proceeds by starting at a goal and trying to achieve it by either knowing it to be true or by finding a rule which establishes this goal in its conclusion. If the premise of that goal can be established then the desired goal is achieved. The process thus sets up the premise of the rule as a *subgoal*, and the reasoning continues in an the same manner until it finds a premise which is known to be true. At this point all intermediate subgoals, as well as the desired goal are satisfied.

In the gait diagnosis program, an attempt is made to show that every muscle under consideration is weak and/or tight. This constitutes the original *goal* of the program.

Certainty Factors

The reasoning process adds items to a *data base* as they are established. The data base represents the knowledge that the program has about the patient under consideration at any given time; an item represents one fact about that patient, so the data base is simply a collection of facts. Every item is either a deviation, part of a description of a deviation, or a muscle weakness/tightness.

Items which have been considered in the reasoning process have *belief values* or certainty factors associated with them; these certainty factors reflect the confidence that the program has in the truth of the associated item, ranging from absolute certainty to absolute disbelief. As the program proceeds through the backward reasoning chain, it bases its decision regarding the truth of an item on the numerical value of the certainty factor, as designed by Shortliffe [Shortliffe 1976], rather than on a rigid all or nothing scale.

Items in the data base and rules have certainty factors associated with them; rules, as they are invoked, effect the certainty factors of items in the data base. The certainty factors of items and rules are represented differently within the program.

The certainty factor of an item is a number from +l.0 to -l.0 and is defined as the *measure of belief* plus the *measure of disbelief* (generated from negative rules) of that item. The measures of belief and disbelief of an item are stored separately for each item. Rules have certainty factors which have been assigned by the researchers and which consist of single positive or negative number with values between 0.0 and 1.0 or 0.0 and -1.0. respectively. Also, each rule, when it is invoked, has a certainty factor; this computed certainty factor either modifies the measure of belief or the measure of disbelief of an item, depending on whether this certainty factor is a positive or a negative number. Thus the certainty factor of a rule effects the certainty factors of the items in its conclusion by modifying their measures of belief and disbelief. And the measures of belief and disbelief of an an item are added together to form its certainty factor.

The closer the certainty factor is to 1.0, the more belief the program has in an item, and the closer it is to -1.0, the more the program is certain that the item is false.

The certainty factor is determined in two ways: first, it may be assigned during the acquisition of the initial deviations—every deviation has a certainty factor of 1.0 associated with it when it is entered into the data base; second, it may be calculated by the program during the backward chaining process. Thus, given the rule:

```
Premise:  If Trendelenberg on side A during swing
Conclude: Hip abductors weak side B    .95
```

the system will determine the certainty factor for the conclusion by computing the certainty factor of "Trendelenberg on side A during swing." and using this rule to determine the certainty factor of "hip abductors weak side B" as specified by this rule (there may be other factors which will modify the final certainty factor).

If the final certainty factor of trendelenberg gait is less than .20, the machine does not believe the premise; therefore the premise is not validated, the conclusion is not proven, and the rule is not applied. If the determined certainty factor for the premise is .20 or more, the premise is validated, the conclusion proven, and the rule applied. The value .20 has been empirically established by Shortliffe [Barnett 1968]. The certainty factor for the entire rule is determined by multiplying together the certainty factors in the premise and conclusion:

```
Premise:  If trendelenberg gait on side A
          during swing        .90
Conclude: Hip abductors weak  .95
```

$(.90) \times (.95) = .855$ certainty factor for the entire rule.

Computation of Certainty Factors

Once the certainty factor of a conclusion has been determined, the certainty factor for that item in the data base must be computed. As items are added to the data base, the system's belief in that item is modified by other rules effecting it. If several rules mention a particular item in their conclusions, then the certainty factors generated by those rules interact to define the final certainty factor for the item.

Computation of the final certainty factors is done in the following manner: given three certainty factors .60, .50, .40, the machine computes their value between 0.0 and 1.0. Thus, .60 is 60% of the distance between 0.0 and 1.0; .50 is 50% of the distance between .60 and 1.0, or .20, and is added to .60 ($.60 + .20 = .80$); .40 is 40% of the distance between .80 and 1.0, or .08, and is added to .80 ($.80 + .08 = .88$). Positive certainty factors modify the measure of belief in the item; negative certainty factors modify the measure of disbelief.

System Validation

System validation or debugging was the final step in program design and consisted of running the program to see how it made use of the facts in the knowledge base, making adjustments and corrections necessary for smooth operation and correct analysis of data. Since the program is completely driven by the rules in the knowledge base and since it *reasons* by relating those rules in a logical manner, the program's ability to reason correctly is only as good as the rules supplied to it by the physical therapists. Debugging allowed the physical therapists to follow the *thought processes* used by the computer in relating the rules and drawing conclusions. Weaknesses in the computer's reasoning and conclusions reflected weaknesses in the knowledge base and pointed out inconsistencies in the physical therapists' reasoning. Ultimately, debugging provided a lesson in the process of logical reasoning.

The system validation procedure was carried out by running each deviation in isolation through the computer in order to see what conclusions were reached; and how they were reached. This was done by examining the rules used in reaching the conclusion in the order in which they were applied: the program could print out its entire reasoning chain or only the parts of it relevant to a single conclusion. When the authors were satisfied with the results and the reasoning performed by the program, the deviation was considered *debugged*. After each deviation had been debugged separately, random groupings of three or more deviations together were run through the same procedure to insure consistency of results.

The AI specialist played an integral part in the debugging procedure and provided the technical knowledge necessary for tracing the rules, modifying them in their machine format, and for locating *bugs* or problems with the reasoning program itself (as opposed to the rules).

Incorrect or contradictory conclusions were due primarily to inaccurate certainty factors, insufficient negative rules, and a limited knowledge base. The subsequent readjustment of certainty factors was a fairly trivial process accomplished by reviewing the relevant rules and empirically making the appropriate changes. The need for a more sophisticated layer of negative rules was a more difficult problem to solve and in fact is one of the program's greatest weaknesses. The issue was resolved, at least in the short run, by addition of more type 2 negative rules.

Discussion

The cost of computer time involved in the development of this diagnostic program was estimated at $5000. This figure does not include AI staff time which has been estimated at 200 man-hours. Cost has been a significant factor in limiting the widespread use of computers. However, since computers are becoming more cost-effective, the feasibility of implementing such a program increases [Pauker 1976]. If the cost of computers continues to decrease it is foreseen that this prototype could serve several different purposes as outlined below.

The program has the potential to be used for instruction in gait analysis in academic settings. The program's problem-solving format could serve as a valuable tool in helping the student to integrate information acquired in classroom instruction. The individualized nature of computer instruction permits the student to progress at his own rate and allows him a choice of learning methods i.e. infer, diagnose, menu (diagnose). By increasing individualized independent study, use of such a program could decrease faculty teaching time. This would allow a more effective allocation of personnel in other teaching, clinical, or research capacities.

The program also has the potential to assist the clinician in a physical therapy setting. The system provides a ready storehouse for the multitude of facts associated with orthopedic gait with the ability to dispense these facts to the user as well as to accept and integrate new facts into the system. As this prototype increases in complexity by integrating additional information, its validity will increase and it will become a more viable clinical tool.

The program design and implementation process suggests the expansion of interdisciplinary cooperation as a valuable means for introducing innovative changes into the profession. Drawing upon the expertise of other disciplines such as computer science could provide impetus for physical therapists to explore areas beyond the scope of traditional physical therapy.

Areas for Future Research

Utilization of the program would be enhanced by the development of several components.

An English language interface would allow the user to easily review the rules invoked during the reasoning process and to make program modifications. At the present time knowledge of MacLisp is required for these tasks. This natural language interface would make it possible to perform these procedures using English thus increasing the the program's accessibility and manipulation by health care personnel.

Videotaping a patient and having the program know all of his deviations would be a valuable validation procedure for Diagnosis and Menu (Diagnosis). Program correction of wrong answers given by the user would be helpful in making the program a more effective teaching tool and could be specified at two points during the program's operation: 1) during the diagnostic process, thus allowing immediate and ongoing correction of errors made by the user in his dialogue with the computer; or 2) at the end of the diagnostic process, thus allowing the user a final check of his answer.

Inclusion of additional layers of rules would increase the program's validity, sophistication, and usefulness. These are envisioned as being primarily of three types: 1) facts about normal gait; 2) more extensive negative rules to further refine the program's reasoning process, thus increasing its reliability; 3) justifications in English for the conclusions reached during the diagnostic process which would increase the program's *communication potential* with health care professionals.

The addition of gait disorders of different etiology would widen the program's applicability. Now that this prototype has been developed for orthopedic problems, it is hoped that the format can be followed to add other areas of gait disorders e.g. gait disorders resulting from a neurological etiology.

References

[Asklund 1976] Asklund S, Brown S, Fiterman C Slide-tape versus lecture demonstration of thermal agents in a physical therapist assistant program. Phys Ther 56: 1361–1364, 1976

[Barnett 1968] Barnett GO Computers in patient care. N Engl J Med 279: 1321–1327, 1968

[Berkeley 1947] Berkeley Report to National Research Council Committee on Artificial Limbs: University of California Berkeley, (chap 3), 1947

[Blount 1956] Blount WP Don't throw away the cane. J Bone Joint Surg [Am]38: 695–708, 1956

[Campbell 1970] Campbell SK, Kohl MA Audio-tutorial independent study of goniometry. Phys Ther 50: 195–200, 1970

[Chapman 1968] Chapman MW, Kurokawa KM Some observations on the transverse rotations of the human trunk during locomotion. Biomechanics Laboratory, Univ of California, San Francisco and Berkeley, Technical Memorandum, May 1968

[Entwisle 1963] Entwisle G, Entwistle DR The use of a digital computer as a teaching machine. J Med Educ 38: 803–812, 1963

[Feurzeig 1964] Feurzeig W, Munter P, Swets J, et al Computer-aided teaching in medical diagnosis. J Med Educ 39: 746–755, 1964

[Inman 1966] Inman VT Human locomotion. Canad Med Assoc J 94: 1047–1054, 1966

[Inman 1968] Inman VT Conservation of energy in ambulation. Bull Prosthet Res 10: 26–35, 1968

[Kent 1976] Kent BE Lecture notes, Human Motion and Therapeutic Procedures III, Division of Physical Therapy, Stanford University, 1976

[Klopsteg 1968] Klopsteg PE, Wilson PD Human Limbs and Their Substitutes. New York, McGraw-Hill Book Co, 1954, pp 437–471, 1968

[Murray 1964] Murray MP, Drought BA, Ross CK Walking pattern of normal men. J Bone Joint Surg 46A: 335–360, 1964

[Murray 1967] Murray MP Gait as a total pattern of movement. Am J Phys Med 46: 290–333, 1967

[Overall 1963] Overall JE, Williams CM Conditional probability program for diagnosis of thyroid function. JAMA 183: 307–313, 1963

[Pauker 1976] Pauker SG, Gorry GA, Kassirer JP, et al Towards the simulation of clinical cognition: taking a present illness by computer. Am J Phys Med 60: 981–996, 1976

[Perry 1967] Perry J Mechanics of walking. Phys Ther 47: 778–801, 1967

[Rockart 1975] Rockart JF, Morton MS Computers and the Learning Process in Higher Education: A Report Prepared for the Carnegie Commission on Higher Education. New York, McGraw-Hill Book Co, 1975

[Rutan 1973] Rutan FM Comparison of self-instruction and lecture-demonstration in learning a physical therapy skill. Phys Ther 53: 521–526, 1973

[Saunders 1953] Saunders JB, Inman VT, Eberhart HD The major determinants in normal and pathological gait. J Bone Joint Surg [Am]35: 543–558, 1953

[Shortliffe 1976] Shortliffe EH Computer-based Medical Consultations: MYCIN. New York, American Elsevier Publishing Co, 1976

[Smidt 1974] Smidt GL Methods of studying gait. Phys Ther 54: 13–17, 1974

[Warner 1961] Warner HR, Toronto AF, Veasy LG, et al Mathematical approach to medical diagnosis: application to congenital heart disease. JAMA 177: 177–183, 1961

[Weber 1972] Weber JC, Hagamen WD ATS: a new system for computer-mediated tutorials. J Med Educ 47: 637–644, 1972

Chart

```
Therapist:  The patient leans his trunk laterally.

Computer:   Determined deviation: Lateral trunk bending
            Determined phase: midstance
            Please specify:
            (SIDE)

Therapist:  left.

Computer:   Determined deviation: Lateral trunk bending
            Determined phase: midstance
            Deviation: (Type any new deviations or terminate
                        list with "ok")

Therapist:  ok

Computer:   Infer, diagnose, or menu (diagnose)?  *

Therapist:  Infer.

Computer:   Inferring ...  [program is formulating a diagnosis]
            Left
            The hip abductors are absent (strongly suggestive)
            The hip abductors are extremely weak (strongly suggestive)

            Right
            The hip flexors are weak (suggestive)
            The knee flexors are weak (weakly suggestive)
```

* Infer, diagnose,and menu diagnose are three different methods of arriving at a diagnosis. The user chooses one of the three. These are explained in the report.

Acknowledgments

We would like to thank the Stanford Artificial Intelligence Laboratory for providing computational and publication facilities which made this thesis possible.