エキスパート・システムの将来性

日本アイ・ビー・エム㈱　東京基礎研究所　　戸沢義夫

１．はじめに

本稿ではエキスパート・システムを開発しようとする場合にどう考えると良いのかを中心に述べたい。まず、第２節でなぜＡＩ技術が最近注目を集めるようになったかその理由を考え、第３節でエキスパート・システムに対する期待がＡＩ技術とどのように関係しているかについて述べる。本稿での大きな論点はＡＩ技術を新しいプログラミングとしてとらえることである。第４節で従来のプログラミングの行き詰まりによる問題点を示す。第５節で新しいプログラミングは従来のプログラミングとどう違うかを説明し、それが第４節の問題を解決する可能性があることを述べる。第６節では、今後エキスパート・システムの開発が期待されている新しい応用分野の特徴を指摘し、第７節ではどうすればエキスパート・システムが開発可能になるかについて気付いていることをまとめる。

２．なぜ人工知能なのか

ここ数年の間に人工知能に対する関心は急激に高まってきた。最近はかなり落ち着いてきたと言えるが、『人工知能（ＡＩ）』という用語は広く浸透している。人工知能が何故こんなに注目を浴びるようになったかを考えてみると次のようなことが言えそうである。

人類の永遠の夢

人間にとって生命現象は最も不思議なもので未だ謎だらけである。そして、それが最も進化した人類は知能を持つようになった。知能は生命の究極の目標であるかのようにも見える。生命が発生してから何十億年もかかってようやく到達できた知能を、人間が人工的に作り出すことができるとすればこれ程すばらしいことはないではないか。人間が作った機械が自分で『理解』したり、『判断』したり、『意識』を持ったりしたらすばらしいではないか。『知能とは何であるか』を研究することはそれ自体が学問であり夢がある。

最先端技術

人工知能を達成するためにはコンピュータの力を借りなければならない。しかし、従来のコンピュータの利用技術ではコンピュータに知能を持たすことはできなかった。それはコンピュータが『推論』することができなかったからである。人工知能を達成するためには新しいコンピュータ利用技術を開発することは必要不可欠である。最近になって、コンピュータによる推論メカニズムがある程度使用できる目度がたってきた。新しい技術は夢ではなくやればできる距離にありそうである。そして、日本では第５世代コンピュータ・プロジェクトが１０年計画でスタートした。新技術が日進月歩で開発されていき、絶えず注目していないとすぐに時代に取り残されてしまう。最新技術に携わることもやりがいの

ある仕事だし、最新技術を理解し、その利用の仕方を考えることが必要とされているのである。

新しい応用分野

新しい技術は今まで不可能と考えられていたことを可能にし、機械かができないと思われていた分野の機械化をもできるようにする。人工知能の目標は人間の持つ知能を機械を使って実現する技術であるから、人間のカンや経験にたよって仕事をしている分野の機械化が可能になるはずである。また、人工知能は機械であるので、人間とは異なり老化による判断力の衰えなどを心配する必要はない。企業にとっては企業の持つノウハウが競争力を高める重要な要素であるが、そのノウハウがある社員の持つノウハウであることも珍しくない。その社員が退職すると同時に企業のノウハウも失われるのは困ったことである。人間のカンや経験に基づいて行なわれる判断をＡＩ技術を用いてコンピュータで行なえるようにできれば企業にとってのメリットは計り知れない位大きい。企業が競争力を維持するためには、今まで人間でなければできなかった分野を機械化することは必須であり、ＡＩ技術は新しい応用分野を開拓する原動力になると期待されているのである。

３．エキスパート・システムとＡＩ技術

前節で述べたことからエキスパート・システムを位置付けると次のようになる。

『エキスパート・システムとは、人間の知的な作業に頼って行なっていた仕事をＡＩ技術を応用して機械化したシステムである。』

知的な作業とはカンや経験に基づく判断や意志決定を指すことが多い。このような知的な作業はその分野の専門家でなければできないと考えられている。エキスパート・システムは専門家と同じような働きをすることが期待され、判断や意志決定を行なった場合にはその理由付けをきちんと説明できなければならない。

この考え方に従うと、ＡＩ技術を導入することができれば専門家の作業を機械化できるはずである。ＡＩ技術はエキスパート・システム・シェルとして売られており、シェルを買えばエキスパート・システムを構築できると信じられている。しかし、実用化されたエキスパート・システムの報告はごく僅かしかないし、実際にやってみるとエキスパート・システムを作るのは相当に難しいことがわかる。つまり、ＡＩ技術を勉強してもエキスパート・システムは作れないのである。

このことから、ＡＩ技術は未だ役に立たないとか、ＡＩ技術はそれ程たいした技術ではないなどの結論を出すのは簡単である。しかし、エキスパート・システムがうまく作れない原因はＡＩ技術にあるのではなくコンピュータ利用技術にあるかもしれない。そこで、ここでは視点を変えて、コンピュータの利用技術（ソフトウェア）の立場からＡＩ技術を見るとどうなるかについて次節以降で考えてみたい。

４．コンピュータ利用技術の行き詰まり

４．１　従来の応用分野

エキスパート・システムがねらっている応用分野と、今までに機械化された応用分野を比べてみよう。従来の応用分野は、問題解決にあたって解き方が極めてはっきり決まっており、その解き方に従うと誰がやっても必ず答がユニークに決まる。従って問題解決のための手順をプログラムすれば機械化が可能であった。機械化のためには、先ず(1)問題解決のための手順を考え、次に(2)その手順をプログラムとして記述すれば良い。プログラマーは普通この(1)と(2)の両方の作業を行なっている。プログラムが書けるようになるためにはそれなりの訓練が必要なため、問題解決の方法がはっきりしている分野はかなり機械化を進めることができたが、問題解決が専門家にまかされている分野では、専門家はあまりプログラムを書くことがないので機械化は進んでいない。このことはプログラミングの作業が機械化を進める上で大きなファクターになっていることを示している。

４．２　ソフトウェア危機

コンピュータを利用するにはソフトウェアは不可欠であり、コンピュータを利用した機械化を行なうにはそのためのプログラミングが必須である。今までに既にいろいろな分野が機械化され、多くのプログラムが書かれてきた。時代が経つにつれ、業務のやり方が変わったり、ハードウェアが新しくなったり、システムへの新しい要求が生まれたりして、過去に書かれたプログラムを変更しなければならなくなってきている。このことをプログラムのメインテナンスと呼ぶが、プログラマーの仕事のうちメインテナンス作業が今では７割を越すようになり、しかもだんだん増え続けている。従って、新しい分野を機械化する作業に労力を費やすことがますます困難になってきている。つまり、プログラマーの生産性がだんだん低下しつつあるのである。この問題はソフトウェア危機として知られている。

４．３　従来のアプローチ

ソフトウェア開発において、メインテナンスが極めて難しいことはかなり以前から気が付いており、その対策として大規模ソフトウェアの開発ではトップ・ダウン・アプローチが有効であることが認められている。このアプローチでは、先ず開発しようとしている分野の要求をきちんと調べ、システムが持つべき機能を完全に定義する。後から不足している機能を追加することを避けるためである。次に、機能を実現できるようにシステムの概略設計を行ない、概略設計が終わると詳細設計を始める。詳細設計では各部分の仕様を実現するための手順が決められる。一度決めた設計はよほどのことがない限り変更しなくて済むように充分検討してから先へ進む。詳細設計が終わったら、詳細設計で決められた手順をプログラムで記述する。

しかし、このアプローチには次のような欠点がある。システムの機能や要件があらかじめきちんと定義できない場合があるのである。例えば、翻訳システムなどがそうである。システムを使用してみて新たな要求がわかり設計変更につながる場合には、このアプローチでは柔軟に対処できない。設計変更をしないことが前提となっているので、システムが成長することがなく、徐々にシステムを良くしていくことはできないのである。また、システムが100%完成するまで動かすことができないのでどうしても開発期間が長くなる。つ

まり、20%完成したところでは20%の機能が動き、50%完成したところでは50%の機能が使えるようにはできない。

4．4　ソフトウェアのライフ・コスト

ある業務を機械化する際にどれだけコストがかかるかを考えてみる。ある業務を機械化しようと決めてから、開発し、実際に使用して廃棄するまでにかかるコストは、開発費用と、実用になってからのコンピュータ使用料とメインテナンス費用である。この合計がソフトウェアのライフ・コストである。ハードウェア技術の進歩により、コンピュータの価格対性能比は年々良くなっている。このため、良いプログラムとは何かという価値観も変更せざるを得ない状況である。

コンピュータの使用料が高かった頃は、実行時のコストが少しでも安くなるように、プログラマーが手間をかけて能率の良いプログラムを書いたものである。能率の良いプログラムが『良い』プログラムだったのである。しかし、実行時のコストが安くなり、逆にプログラマーの生産性が下がってきている現状では、実行時のコストがかかっても、ソフトウェアの開発費用やメインテナンス費用を少なくする方法を選んだ方が結果的に得になる場合がでてきている。メインテナンスが容易な読んでわかりやすいプログラムが『良い』プログラムなのである。

5．新しいプログラミングとしてのＡＩ

前節で述べたコンピュータ利用技術上の問題点は次のようにまとめられる。

(1)　誰がプログラムを書くのか
(2)　プログラミングの生産性を上げ、より簡単に機械化する方法はないか
(3)　設計変更に強いアプローチはないか
(4)　実行時にコンピュータ費用が多くかかっても、開発費用とメインテナンス費用が安くなる方法がないか

これらは、機械化すべき応用分野をいかに能率良く簡単に行なうかという問題に対して、従来のプログラミングには限界があることを示している。今求められているのは従来のとは異なった新しいプログラミングなのである。

プログランミングの立場から見ると、ＡＩ技術は新しいプログラミングを提供していると言えるのである。

5．1　プログラミング・パラダイム

従来のプログラミングの行き詰まりを打開するためにとられた方法は大きく分けて２通りある。ひとつは、プログラミング環境を良くするアプローチで、プログラマーの生産性を高めようとするものである。これは、主としてワークステーションで達成されている。マウスやビットマップ表示装置、マルチウィンドウ・システムを持ったシステムを個人が占有して使用する。特定の言語向きにいろいろ便利なソフトウェアをあらかじめ充分に用意しておき使い勝手を良くしてプログラマーの生産性を高めるのである。

もうひとつは、プログラミングの際の考え方（発想）を新しくするもので、プログラミング・パラダイムと呼ばれている。新しく提案されたプログラミング・パラダイムで有名なものに『オブジェクト指向』や『関数型プログラミング』などがある。ＡＩ技術によるプログラミング・パラダイムは『知識プログラミング』と総称されている。

プログラミングの作業は次の２つの部分から成っている。

(1)　手順を決める
(2)　手順を記述する

プログランミング・パラダイムは手順を決める際の発想の仕方に影響を与えるものである。

プログラミング・パラダイムの立場からすると、従来のプログラミングはひとつのパラダイムに基づいていたと言える。つまり、FORTRANやPL/Iなど言語はいろいろあるが、プログラミング作業における手順を決める際の発想は同じだからである。言語間の違いは、手順を決めた後での手順を記述する際の容易さである。

プログラミング・パラダイムが新しくなると当然そのためのプログラミング言語が必要になる。知識プログラミング用の言語として有名なものは PrologとOPS5(OPS83)である。これらの言語を評価する場合には、従来の言語とプログラミング・パラダイムが違うという点を無視してはならない。従来書いていたプログラムと同じものをこれらの言語で書こうとすると書きにくい場合が多いからである。新しいパラダイムが必要になった理由は、従来のプログラミングでは行き詰まったからであり、従来のプログラミングで困難なプログラムが Prologや OPS5でどのように書かれるかを評価しなければならない。

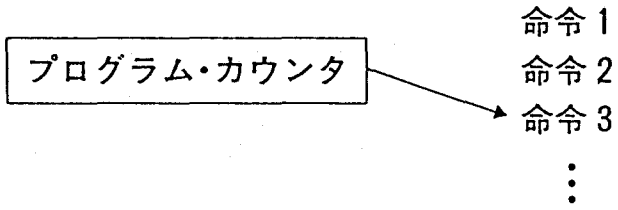
５．３　従来のプログラミングとの違い

知識プログラミングは従来のプログラミングと手順を決める際の発想が違うと述べた。残念ながら、どのように発想が変わるかは実際に体得してわかってもらう以外にうまい説明ができない。しかし、手順の記述法には極めてはっきりした違いがあるので以下にそれを説明する。

従来のプログラミングでは、手順が決まったらそれを順番に並べて書けば良い。図１に示すように、コンピュータ内部にはプログラム・カウンタがあり、命令を１つ実行すると自動的に次の命令を指すようになっている。プログラム・カウンタの働きにより、順番に並べて記述された命令はその順序で実行される。

知識プログラミングでは、プログラムはプロダクションの集まりとして記述される。プロダクションは通常『知識』と呼んでいる　ＩＦ　．．．　ＴＨＥＮ　．．．　の形をしたルールである。プロダクションはどんな順序に並べて記述しても良い。(ただし Prologでは実行結果が記述の順序に依存する場合がある)　コンピュータ内部にはプログラム・カウンタの代りにパターン・マッチ・メカニズムがあり、次にどのプロダクションを実行するかを実行時に決めていく。パターン・マッチにより次に実行すべきプロダクションを決めるメカニズムのことを『推論メカニズム』と呼ぶことがある。プログラムを実行すると実行順序にプロダクションが並べられる。この実行時の順番があらかじめ意図した手順と合っていれば良いのである。

◦ プログラム・カウンタが次に実行する命令を決める



◦ パターン・マッチ(推論メカニズム)によって次に実行されるプロダクションが決まる

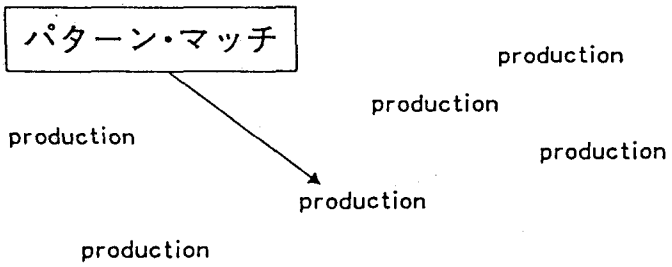


図1　従来のプログラミングと新しいプログラミングの違い

知識と推論の枠組

◦ 事実を記述する

◦ 知識(プロダクション)を記述する

◦ 推論メカニズムが問題を解く

図2　知識と推論の枠組によるプログラミング

言い換えれば、従来のプログラミングに比べて、ＡＩ技術は手順の記述法を変革し、手順を順番に並べて書かずに、プロダクションの形で自由な順序に書けるようにしたのである。自由な順序に書くための代償はパターン・マッチ・メカニズムに必要なコンピュータの計算量である。

５．４　知識と推論の枠組

ＡＩ技術はプログラミングに於ける手順の記述法を変革した。プログラムを構成する基本単位は　ＩＦ　．．．　ＴＨＥＮ　．．．　の形をしたプログラミングである。ひとつひとつのプロダクションは『知識』を表現していると解釈でき、パターン・マッチ・メカニズムは、いろいろある知識の中から次にどの知識を適用すれば良いか『推論』していると言える。従って、このようなプログラミングを『知識と推論の枠組によるプログラミング』と呼ぶことがある。

プログラムがプロダクションの集まりで記述されているということは、プログラムのどの構成要素ひとつをとってもそれを知識として単独で意味を理解できる良さがある。従来のプログラミングではプログラムの構成要素は命令であるので、単独で取り出すとどういう意味があるかわからず、プログラム全部を理解しなければその役割がわからなかった。プログラムの読み易さ、わかり易さという観点からは、プログラムの構成要素そのものの意味が理解できるという意味で、知識プログラミングの方が従来のプログラミングより優れている。

知識と推論の枠組による理想的なプログラミングを考えてみる。普遍的な知識はプロダクションで書かれ、自由な順序で記述されている。その他に現実を表わした『．．．は　．．．である』という事実が記述される。すると、推論メカニズムが自動的に問題を解いてくれる(図2)。理想的な状態では、プログラミングは知識の記述と事実の記述だけで済み、手順は推論メカニズムが自動的に決めてくれる。もちろん、現在の推論技術ではこれはとうてい不可能であるけれども。

プログラムを書くことが問題解決に必要な知識の記述の集積で済むのであれば、プログラマーでなくても専門家がじかにプログラミングできるようになる可能性がある。また、手順を決める作業も推論メカニズムが行なえるようになれば、プログラマーは不要になり、プログラマーの生産性の問題も生じなくなる。知識を持っている人が断片的に知識を記述し、事実を知っている人が自分の知っていることだけをコンピュータに教えると、それがプログランミングのすべてである。コンピュータに知らせる知識や事実の量が少なければコンピュータはそれなりに答を出してくれる。全体の20%の知識と事実しかコンピュータに入力されていなければ20%の範囲内で答えが出される。知識や事実をどんどんコンピュータに教えると、難しい問題に答えられるようになるし、答えられる範囲も広くなる。

理想的な状態では、前節で述べた従来のプログラミングで問題になっているコンピュータ利用技術上の問題点がすべて解決されるのである。知識と推論の枠組はこのような可能性を秘めているが、現実は理想からはかなり掛け離れていると言わざるを得ない。

現在の技術レベルを考えると、知識表現法は未熟であり、推論メカニズムも決まりきったパターン・マッチ以上のことはできない。しかし、現在の技術レベルでも『診断』などの応用分野では、知識と推論の枠組が理想に近い状態で達成されている。MYCIN[1]はその良い例である。成功した理由は、診断で使われる知識が現在の推論メカニズムで扱えるパ

ターンで記述できたからである。従って、知識と推論の枠組によるプログラミングが可能かどうかは、機械化しようとしている応用分野で使用している知識が、(1)システムの提供する知識記述法で簡単に記述できるかどうかと(2)推論メカニズムが適用すべき知識を的確に選び出せるかどうかにかかっている。

6．新しい応用分野

本節では、今まで機械化されていなくて今後機械化が望まれている応用分野がどのようなものであるかについて述べる。4.1節で述べたように、従来の応用分野は誰が解いても解がユニークに決まる世界であった。しかし、現実の世界では解がユニークに決まらない場合もかなり多いのである。

その典型的な例はセールスである。もしも、いくつもの製品の中のどれを買うのがベストであるかがユニークに決まるとしたらセールス活動は成り立たない。セールスのポイントは売り込みたい商品が良いと結論するような買手の価値観を探しだし、その価値観に焦点を当てて買手を説得することにある。このようなセースル活動を支援するエキスパート・システムを開発しようとする努力が行なわれており、そのプロトタイプはYES/FAME[2]として発表されている。

解がユニークに決まらない他の例としては、プランニングがある。一般にある状況下ではいろいろなプランが可能である。それらのプランの中で、どれがベストであるかは簡単には決まらないのがふつうである。どのプランが良いかはどんな価値観で物を考えるかによって決まる。現実の世界ではいろいろな価値観があり、互いに相反した結論に導くことも多いのである。例として工場における生産計画を考えてみる。工場の目標は(1)納期を守り、(2)コストを下げ、(3)工場の生産能力限界まで受注し、(4)品質の良い物を作ることである。しかし、これらの目標は互いに競合しているのである。生産過程ではいつ不良品が発生するかわからない。納期を正しく守るためには、不良品が発生した時に対応できるようにできるだけ早く生産を開始したほうが良い。一方、コストを下げるためには、仕掛在庫を減らした方が良くできるだけ遅く生産を開始したほうが良い。いつ生産開始するかはコストを下げるか納期を守るかどちらに重点を置くかに依存する。人間は生産開始する時期を決定しているけれども、これは実は解がユニークに決まらない問題なのである。

解がユニークに決まらない問題をどのように機械で解くか(どのようにプログラムするか)はまだわかっていないのである。人間はいろいろな価値観をうまくバランスし答を決めている。人間ができることだからエキスパート・システムでもできなければならないと考えたくなるが、プログラムの方法がわかっていないのであるからエキスパート・システムが開発できなくて当然なのである。筆者達はプランニングの問題を中心に解がユニークに決まらない問題をプログラムする方法の研究を開始したところである。

新しい応用分野を機械化するには、まずどのようにプログラムするかがわかっていなければならない。どのようにプログラムするかがわかっているかいないかで機械化できるかできないかが決まる。プログラムの方法がわかっている場合には、従来のプログラミングでも新しいプログラミング(ＡＩ技術)でも必ずエキスパート・システムは作れるのである。つまり、ＡＩ技術を用いないと構築が不可能なエキスパート・システムはない。ただし、ＡＩ技術を利用した新しいプログラミングで行なうか従来のプログラミングで行なうかの

差は、システム開発に必要な労力、システムが稼働するまでの期間、メインテナンスの容易さとなってあらわれる。ＡＩ技術がエキスパート・システム開発に不可欠なのではなく、新しいプログラミングがエキスパート・システムの開発を簡単にするのである。解がユニークに決まらない分野を扱うエキスパート・システムを作ろうとしても、プログラムのしかたがわからない限り、それは極めて困難なのである。

７．エキスパート・システムの将来

ＡＩ技術を応用すれば専門家でなければ行えなかった業務を機械化できるという期待から、エキスパート・システムは注目を集めている。エキスパート・システムは判断や意志決定を機械が行えるようになると期待されている。しかし、実際にエキスパート・システムを作ろうとすると大きな困難に直面し、たいていはおもちゃのようなシステムしか作れない。この理由は、現在の技術では理想的な『知識と推論の枠組によるプログラミング』が達成できないからである。ＡＩ技術をコンピュータ利用技術の立場から見ると、新しいプログラミングを提案し、プログラムの記述法に変革をもたらした。それは、手順を順序通り並べて書くのではなく、プロダクションにより任意の順序で書けるようにした点である。

プロダクションは『知識』であると解釈できるが、実際には、推論メカニズムによってプロダクションが意図した通りの順序に並ぶようにしなければならず、その制御のために書かなければならないプロダクションもかなりある。純粋に知識を表現している以外のプロダクションをかなり書かなければならないことは、プログランミングのために相当な努力が必要であることにつながる。知識プログラミング用言語である Prologや OPS5のプログラミング・スキルをきちんと身につけるためにはかなり訓練しなければならない。しかし、これらの言語は新しいプログラミング・パラダイムを提供しているので、手順を決める際に従来のプログラミングでは気付きにくかった新しい発想が生まれることがある。解がユニークに決まらない分野を機械化する時に、どうやってプログラムしたら良いかが、新しいプグラミングによると考え易くなるのである。

ここで、エキスパート・システム開発の方法について、私が気付いていることを少し述べておきたい。

(1) 機械化しようとしている業務を徹底的に理解すること。特に、どのような過程を経て決定が行われるか、どのようにして判断が行われるかを完全に把握しなければならない。それには専門家へのインタビューは不可欠である。しばしばこの作業は知識エンジニアによる知識獲得と呼ばれるが、『診断』のようなアプリケーションでない限り、専門家のインタビュー結果をそのままルール(プロダクション)で記述することは、ほとんどの場合不可能である。業務を理解するのは、『手順を決める』ための第１歩なのである。

(2) エキスパート・システム開発はプログランミング作業であると認識すること。エキスパート・システムを開発するためには、エキスパート・システム・シェルを使わなければならないと誤解されているような傾向があり、しばしば『どのシェルが良いでしょうか』と質問される。エキスパート・システムの開発に成功するかどうかは(1)の作

業の結果、手順を決められるかどうかにかかっている。どのシェルを使うか、どんな言語を使用するかは、手順の記述のし易さ、読み易さ、変更のし易さの違いである。シェルや言語の選択はエキスパート・システムの開発にとってあまり本質的ではない。

(3) できるだけ人間が使用している概念をコンピュータ上で実現し、扱っている世界をすなおに記述できるようにすること。人間が使用している概念には定義が曖昧なものも少なくない。例えば、『緊急状態』とか『要注意』という概念は具体的にどう定義したら良いか困ることが多い。しかし、MVSのオペレータ作業を自動化したエキスパート・システム YES/MVS[3]が成功した理由のひとつは、このような概念が使用されたからである。MVSから出力されるメッセージを見ながら、エキスパート・システムはMVSが今どういう状態であるかを絶えず判断している。それを、人間が使用している概念に直し、アクションをとらなければならない状態になるとそれを解決する方法が呼び出されるのである。

まだまだ機械化したいけれども機械化されていない分野は多い。従って、これからもエキスパート・システムがどんどん開発されていくことは間違いない。しかし、エキスパート・システムとそうでないシステムとの区別はあまりされなくなると思われる。というのは、エキスパート・システムもプログラミングされたものだからである。より簡単にアプリケーションを開発するための技術はまだまだ研究の余地がある。特に、知識表現や推論メカニズムの現状はまだまだ未熟であり、今後の研究が必要な分野である。それと同時に、解がユニークに決まらない分野をどうプログラミングするか、という問題を解決しない限り、多くの人が期待しているエキスパート・システムは実用にならないのである。

参考文献

[1] E.H.Shortliffe, Computer-Based Medical Consultant:MYCIN, New York:Elsevier, 1976

[2] John Kastner, Chidnand Apte, James Griesmer, Se June Hong, Maurice Karnaugh, Eric Mays, and Yoshio Tozawa, "A Knowledge-Based Consultant for Financial Marketing", AI Magazine, 7(5):71-79, Dec. 1986

[3] R.L.Ennis, J.H.Griesmer, S.J.Hong, M.Karnaugh, J.K.Kastner, D.A.Klein, K.R. Milliken, M.I.Schor, and H.M. VanWoerkom, "A continuous real-time expert system for computer operations", IBM Journal of Research and Development, 30(1):14-28, Jan. 1986