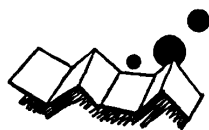


解説

知識表現言語[†]



小山 照夫^{††}

1. 知識表現言語の目的と必要性

知識表現の問題は、人間とコンピュータとの間で、いかにして「知識」と呼ばれる情報を共有するかの問題としてとらえることができる^{1)~4)}。知識として記述された情報を人間の側から見るならば、それは対象とする領域において成り立つ一般的性質を記述するものであって、与えられた状況に応じて、推論や問題解決の具体的方法を与える情報として理解可能なものでなければならない。また逆にコンピュータの側からは、知識の記述に一定のシンタックスが存在し、かつこのシンタックスに対応した知識適用のためのアルゴリズムが存在して、このアルゴリズムに従う知識の適用が、結果として、人間の側から見た知識の意味内容を実現するものとなっていなければならない。このことを言い換えるならば、一つの表現形式が、人間に対しては知識としての意味内容を表し、コンピュータに対しては、あらかじめ用意された一定の手順に従って実行可能な操作を表していると言うことができる(図-1)。

このような知識表現の問題は、知識ベースシステムの開発において、特に重要である。現在までのところ、実用的知識ベースシステムの開発にあたっては、あらかじめ定められた知識表現に従って記述された知識を十分な量だけ蓄積することによって、満足できる機能を持つシステムを実現するのが一般的である^{4), 5)}。ここでどのようにして十分な量の知識を蓄積するかの問題は、知識獲得の問題と呼ばれており、知識ベースシステムを実現するうえでの最大の問題点の一つとなっている。知識獲得の問題に対しては、現在までにさまざまなアプローチが提案されているが^{6), 7), 21), 22)}、少なくとも現在までのところは、この問題を完全に解決する一般的手法は明らかになっていない。このため、十分な量の知識を蓄積するためには、部分的に機

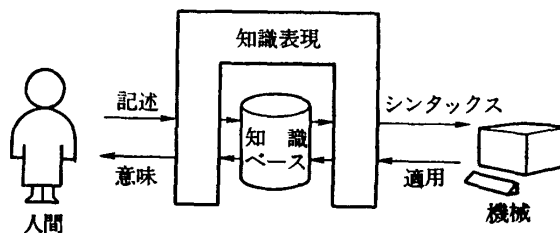


図-1 知識表現の役割

能しているシステムの挙動を、人間が調べあげることを通して、機能的に不足している部分や、誤っている部分を発見し、そのような不備がどのような知識の不足ないし誤りに基づくものであるかを調べ、知識の追加ないし修正を行うことによって、システム性能を満足のいくものに近づけていくという、試行錯誤的方法を採らざるを得ないのが現状であると言える。

このような知識ベース開発のプロセスを考えると、知識ベースシステムに採用される知識表現としては、対象分野において必要とされる知識を記述するための十分な記述能力を持ち、記述された知識をコンピュータによって効率良く処理することが可能であり、部分的試行錯誤の開発が可能となるように、モジュール性のある知識記述を許すものでなければならない。またシステム開発を効率良く行うためには、定義された知識を蓄積、管理し、必要な場合には適当な編集を許す、知識ベース管理機能や、実際に知識を適用してシステム挙動を調べあげるための実行環境およびデバッグ機能も用意する必要がある⁵⁾。

このような知識表現方法の選択や、知識ベース管理などの、システム開発・実行環境は、目的とするシステムに合わせて、そのつど作り上げることも考えられるが、実際には、現在までにかかなり広範な適用範囲を持つ知識表現のモデルがいくつか提案されてきているところから、このようなモデルの一つに対応する、システム開発、実行環境をあらかじめ用意しておくならば、広い範囲にわたる知識ベースシステム開発のた

[†] Knowledge Representation Language Teruo by (KOYAMA Hospital Computer Center Hamamatsu University).

^{††} 浜松医科大学附属病院医療情報部

めの強力なツールとなると考えることができる。このように、特定の知識表現を中心に、知識の定義・編集・検索などの知識ベース管理機能と、知識適用機構とをまとめ上げて知識ベースシステム開発環境を構成したものを、知識表現言語と呼んでおり、知識ベースシステム開発を大いに効率化するツールとして活用されている。

先に述べた通り、知識ベースシステムの開発に際しては、知識獲得の問題が重要な課題となる。知識表現言語が知識ベースシステム構成のための有力なツールとなるためには、この知識獲得の問題を支援する機能を備えていることが重要である。この問題は二つの観点から論ずることができるであろう。すなわち第一の観点は、人間自身が行っているシステムのデバッグを支援するツールを提供することであり²²⁾、第二の観点は、これまでに研究されてきた知識獲得の方法論を活用することにより、部分的にせよ知識獲得を支援することである^{6), 7), 21), 22)}。第一の観点からは、システムの実行過程を記憶して、問題を生じている可能性のある推論部分の動作を調べたり、システムの内部状態を表示したりする機能を備えることなどが考えられるのであり、また第二の観点からは、定義された知識の間に矛盾や冗長性がないかをチェックしたり、システム機能向上に役立つと期待される知識を提案したり、知識定義や編集に際して知識定義形の変更を提案するなどの機能を考えることができる。

以上をまとめれば、知識表現言語は、一定の知識表現モデルを中心として、知識ベースシステム開発、特に知識獲得を支援するべく、知識ベース管理機能や知識適用機構などをまとめあげたソフトウェア環境であり、効率の良い知識ベースシステム開発を可能にするものであると言えるであろう。

2. 知識表現言語の構成

前章でも述べた通り、知識表現言語は、知識ベースシステム開発のツールとして、知識ベースの管理と知識の適用を行うための機能を提供する。したがって知識表現言語は、その基本的構成要素として、知識ベースの定義・編集を行うための一種のエディタ（知識ベースエディタ）と、知識適用のためのメカニズム（推論エンジン）とを持たねばならない。知識ベースエディタは、定義された知識の集合である知識ベースを管理し、新規に定義された知識の蓄積、既定義の知識の編集、検索、表示（リストアップ）など、通常のエデ

ィタの機能を備えるとともに、システムによっては定義された知識のシンタックスをチェックしたり（たとえば EMYCIN）するなどの機能を追加してある場合もある。一方推論エンジンとしては、知識を適用する機能の外に、知識適用の順序や適用結果を記録し、必要に応じて推論プロセスの説明を行ったり、システム内部状態を表示する機能を追加するなど、デバッグのためのツールを用意していることが多い。いずれにしても、これらの機能を使いこなすためには、良好なマン＝マシン＝インタフェースが必要となる。エディタに良好なマン＝マシン＝インタフェースが要求されるのは、ソフトウェア開発環境として当然のことであるが、知識ベースシステムの開発を試行錯誤的に行わざるを得ない現状では、推論エンジン自体も、良好なマン＝マシン＝インタフェースを備える必要がある。推論エンジンのマン＝マシン＝インタフェースは、知識ベースシステム利用の際にも重要な要素となる。知識ベースシステム、特にエキスパート＝システムでは、その利用形態として、人間との対話を通じて意思決定をサポートする形が多いと考えられるから、この意味でも、推論エンジンが良好なマン＝マシン＝インタフェースを持つことが必要である⁵⁾。

知識ベースに蓄積される知識の多くは、対象領域の一般的性質（事実関係や推論規則）を表現するものであるが、知識ベースシステムの実際の利用にあたっては、具体的に与えられた個別の問題について、なんらかの結論を導き出すことが重要である。このようなことが可能であるためには、知識ベースシステムは、対象領域の一般的知識としての知識ベースに加えて、個別の状況に対応する、具体的データを管理する機能を持たねばならない。ただし、このデータは問題解決の間だけ、一時的に管理すればよいデータであって、知識ベースのように、システム内で長期にわたって管理しなければならない情報とは多少性質を異にしている。この意味で、このような一般的データ管理のための領域を、作業記憶（Working Memory）または短期記憶（Short Term Memory）と呼ぶ。これに対して知識ベースそのもののことを長期記憶（Long Term Memory）と呼ぶこともある。純粋な推論の立場からは、このようなデータはまったくの Short Term なものであり、推論プロセスが終了してしまえば不要となる性質のものである。しかしながら知識ベースシステム開発の途中では、これらのデータは、システム挙動を検証するための重要な情報であり、知識の追加

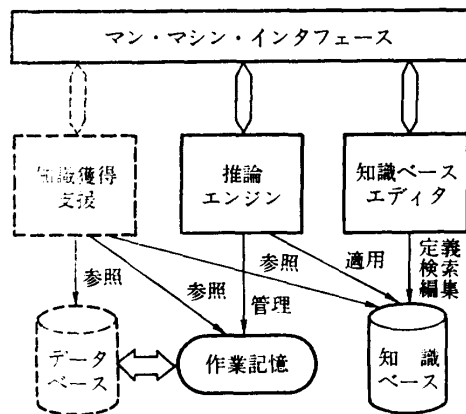


図-2 知識表現言語の構成

や変更によって、システム挙動がどのように変化したかを確認するために再利用が可能である。また帰納推論の試みなどを行う場合には、このようなデータの蓄積が重要な役割を果たすことになる。知識表現言語の中には、このような目的を想定して、個々の具体的なデータを、データベースとして管理する機能を用意しているものもある²⁵⁾。その他知識獲得を支援する機能を積極的に組み入れる試みもなされているが、知識獲得の試みの多くは対象分野への依存性が大きく、一般性の高い形で知識獲得支援機能を統合化したシステムはこれまでのところ実現されていないと言っても良い。

以上をまとめると、知識表現言語の構成は図-2に示すようなものになる。すなわち、知識ベースエディタと推論エンジンを中心として、知識ベース、作業記憶を持ち、良好なマン・マシン・インタフェースを備えている。またシステム挙動をモニタするための機構を備え、将来的には知識ベース内容チェックや帰納推論などの、知識獲得支援機能も組み込まれた構成となると考えられる。

3. 知識表現モデル

先に述べたように、知識表現言語は、特定の知識表現モデルを中心にして、知識ベースシステム開発環境を提供することを目的としている。知識表現言語の基本的特性は、基礎となる知識表現モデルに大きく影響される。現在広く利用されている知識表現のモデルとしては、プロダクション・システム^{11)~3), 15), 19)}、フレーム^{11), 8), 9), 17)}、述語論理など^{3), 18), 20)}、さまざまなものがあるが、これらの詳細は他の論文の中で解説されているので、ここでは省略する。以下では知識ベース

システム開発環境という観点から見て、知識表現言語が採用する知識表現モデルが、その言語の特性にどのように影響するかについて考察する。

3.1 プロダクション・システム

プロダクション・システムにおける知識記述は、基本的には「If~Then~」形のルールとして表現される。このような表現方法は、知識の記述とその適用メカニズムを明確に分離する傾向を持ち、また対象分野の問題解決のための知識として広く用いられる。既知情報から未知情報を推定する推論関係を直接記述するところから、コンピュータの専門家でない、対象領域のエキスパート（たとえば医師）が知識記述を行おうとする場合にも、最も自然な形で知識記述を可能とする知識表現モデルとすることができであろう。このような特徴を持つところから、プロダクション・システムは、多くの知識表現言語の知識表現モデルとして採用されている。

プロダクション・システムにおけるルール型の知識表現は、相対的に知識記述のシンタックスを簡潔かつ均一性の高いものとする。このことがルールに対するさまざまな解析や変換を可能としている。たとえばいくつかのシステムでは、ルールの自然言語による記述と表現が試みられている²²⁾。ROSIE^{11), 21)}ではルールの定義自体、自然言語（英語）を用いて行われるし、またEMYCIN^{11), 21), 12)}のように、ルールの表示に限って自然言語を用いている場合もある。また知識の間の矛盾や冗長性の発見、システム機能を向上させるためのルール追加の提案などの試みも、プロダクション・システムに関連して行われるものが多い²²⁾。

プロダクション・システムでは、システム挙動の説明を行う機能の組み込みが比較的容易である^{6), 12), 22), 25)}。このことはまず第一に、プロダクション・システムの場合、推論の中間結果をすべて作業記憶上に残すために、これに付随させて推論に用いたルールを記録しておくことにより、推論プロセスの追跡が容易に行える。プロダクション・システムで用いられるルールは、推論規則としての意味付けが明確であるから、このような、適用された推論ルールと推論結果の表示は、コンピュータの非専門家にも理解しやすいという特徴を持ち、デバッグを容易にするという効果も持っている。

ルールのシンタックスの簡潔さと均一さは、当然推論エンジンも簡潔なものとする。このことは推論の効率化を計るうえで有利な点であり、パターンマッチの

効率化アルゴリズムや¹⁴⁾、ルールのコンパイルなどの、推論高速化も試みられている。

3.2 フレーム^{8)-10), 17), 23)}

フレームシステムは、人間の概念構造を記述するための表現方法として、概念間の階層関係とそれに基づく属性の遺伝や、概念に対する手続き付加といった、さまざまな知識記述方法をサポートする知識表現モデルである。フレームシステムはきわめて強力な知識表現の枠組を提供するが、それらの機能の多くは、付加手続き (Attached Procedure) の定義によって実現される。したがってフレームシステムではこの付加手続きをいかに定義するかが問題となる。しかしまたこのことは、逆に、付加手続きの定義と実行のための環境さえ用意しておけば、実に多様なものを付加手続きとして定義できることにもなる。実際推論ルールや述語論理を付加手続きの形で組み込んだフレームシステムも多く見られる^{10), 17)}。

フレームシステムで取り扱われるフレームは、付加手続きにより、アクティブな機能を果たす実体を表している。このようなフレームを用いて、フレームシステムを開発する環境としてのエディタなどをフレームシステム中に組み込むことができるのも、このようなシステムの一つの特徴となっている。

フレームシステムでは、多様な付加手続きを許す関係上、システム挙動をモニタする機能もまた、付加手続きごとに考える必要がある。この点はシステム開発環境としてやや問題が残る点であるが、実際にはフレーム間の相互作用を Message Passing に限定して、Message のシーケンスをモニタしたり、フレーム内の付加手続きをルールの形で表現しておき、ルール適用状況を記録するなどによって、システム挙動のモニタを行うなどが行われる。

3.3 その他の知識表現モデル

知識表現言語の中核となる知識表現モデルとしては、従来プロダクションシステムまたはフレームシステムが利用されることが多かったが、近年になって、述語論理^{18), 20), 21)}およびオブジェクト指向言語^{10), 11)}の利用が目を集めている。

述語論理、特に一階の述語論理は、数学的に整った体系をなしており、知識の間の矛盾や冗長性の発見、帰納推論実現の可能性など、知識獲得の面から、興味深いいくつかのテーマを提供している²¹⁾。また前提となる知識が正しいならば、システムの導く結論は常に正しいという推論の健全性が保証されている点も、

「安全な」システムを作るという観点から、価値が高いと考えられる。しかしながら一方では、確かに Prolog に見られるように、一階述語論理にある程度制約を加えた Horn 節に対しては、非常に効率の良い Resolution (導出) のアルゴリズム³⁾が確立されているものの、一階述語論理の全体系をカバーする十分効率の良い Resolution のアルゴリズムは確立していないという問題点がある。また論理型の知識表現の場合に、健全性や無矛盾性のチェックが可能であるのは、有害な副作用を含まないような論理操作が行われる場合に限られるが、このような状況のもとで、システム挙動の把握をいかにして行うかという問題もある。

たとえば DEC-10 Prolog に見られるような、代表的な Prolog 処理などを、そのまま知識表現言語と見るか否かについてはさまざまな議論があるが、一般にはむしろプログラミング言語として考えられることが多いようである。一階述語論理に基づく知識表現言語構成の試みとしては、多世界モデルの導入などにより、Prolog を拡張しようとする Prolog-KR²⁰⁾ や、一階述語論理を拡張した形の知識表現を目指す KA-US⁸⁾ などがある。

オブジェクト指向言語^{10), 11)}は、内部状態 (Instance Variable) と手続き (Method) を備えたオブジェクトと呼ばれる概念要素を導入する点では、フレームシステムと類似の概念に基づいていると言うことができる。しかしながらオブジェクト指向言語では、各オブジェクトは、必要な情報以外は外部に公開しないという、抽象データ型としての情報隠蔽を行うことにより、ソフトウェア部品化を目指している点、またオブジェクト間のインタフェースが、Message Passing のみに限定されている点、コンパイルなどの処理効率の向上に多大な注意が払われている点など、人間の概念のモデルを念頭においたフレームシステムのアプローチとはかなり異なる面を持っている。

代表的なオブジェクト指向言語である Smalltalk-80 では、すべての実体はオブジェクトであり、新しく定義されるオブジェクトのメソッドに至るまで、オブジェクト+Message Passing として取り扱われるが、また一方では Loops¹⁰⁾に見られるように、メソッド定義にそれほど厳密さを要求しない言語も存在する。Loops ではメソッド定義にルール形の定義を許すなど、むしろフレームシステムに近い知識表現と考えることもできるであろう。

4. 知識表現言語のインプリメンテーション

知識表現言語は、知識ベースシステム構成のためのソフトウェア開発環境の提供を目的としている。ところで知識ベースシステムで用いられる知識の多くは、数値的情報と言うよりは、記号を用いて表現される情報が多い。プロダクション＝システムのルールを例に取って見ても、それは最終的には自然言語の形に解釈されて理解されうる記号表記が大部分であることは容易に理解されよう。このような記号表記された知識を管理し、適用する機能を提供する知識表現言語は、それ自体が、記号処理機能を備えている必要がある。このような要求に答える最も直接的な方法は、Lisp や Prolog など、それ自身強力な記号処理機能を備えている基底言語を用いてシステムを実現する方法である。実際に現在までに開発されてきた知識表現言語の多くのものが、Lisp ないし Prolog を基底言語として採用している。また実際の知識適用メカニズムを考えて見ると、たとえばパターンマッチングやバックトラック、データとして表現された知識情報の、手続きとしての適用など、特に人工知能研究の中で開発されてきた、Lisp や Prolog を用いるプログラミングテクニックが有用であることも、これらの言語によるシステム実現を容易にする要因となっている。広い意味で言うならば、Lisp ないし Prolog という言語自体、知識表現言語として要求される多くの機能をあらかじめ備えたプログラミング言語となっており、知識表現言語の中には、これらの基底言語に知識表現モデルをサポートする機能を追加する形で実現されていると考えられるものもある。もちろん Lisp, Prolog 以外の言語を用いて実現された知識表現言語も存在しており¹⁶⁾、特に処理の効率化が重要となる場合には、より機械寄りの言語が用いられることになる。

プロダクション＝システムの実現にあたっては、作業記憶をどのようなものとするかが、第一に問題となる。作業記憶に格納されるデータが、単なるパターンであるのか、あるいはオブジェクトごとに分けられたデータであるかによって、作業記憶へのアクセス方法はまったく異なったものとなる。次に作業記憶に対するアクセスとして、単なるパターンマッチだけで良いのか、それともある種の述語を用いてアクセスすることを許すのかという問題がある。これら2点が決定されるならば、ルールのシンタックスは基本的に決定することとなる。さらに Conflict Resolution の方法

と、推論の向きが決定すれば、システムの基本的骨格が定まることとなる。

フレームシステムないしオブジェクト指向の考えを取り入れたシステムでは、フレームないしオブジェクトの実現方法と、付加手続きないしメソッドの定義方法が大きな問題となる。純粋な意味でのオブジェクト指向言語では、システムがプリミティブとして用意するオブジェクトと、それらに対して有効なメッセージとを用いて全システムが構築されることとなるが、Lisp や Prolog を基底言語とする実現では、フレームないしオブジェクトをリストないし述語の集合として実現し、付加手続きやメソッドは基底言語の関数または述語として実現する方法も採用されている。このような実現方法では、Lisp による、関数の中からの関数呼び出し、または Prolog の述語の中からの述語呼び出しの機能が、付加手続きやメソッドの実行のうえで便利な機能を提供している。

5. 将来の展望

知識ベースシステムの普及について、その開発環境としての知識表現言語の重要性は、ますます大きなものとなると考えられる。知識表現言語では、知識表現モデルの記述能力、効率的な推論エンジン、知識獲得の支援が重要な問題となる。またこれらの機能を活用するための、良好なマン＝マシン＝インタフェースの必要性も忘れてはならない。これらの問題を考えると、知識表現のモデルとしては、これまでに開発されてきた種々のモデルを統合する形での、いわゆる Mixed Paradigm¹⁰⁾ を許すようなシステムが開発されてきており、諸概念の整理が進むとともに、十分な記述力を持つと同時に、コンピュータの非専門家にも理解しやすい形での知識表現モデルが実現されていくと考えられるし、推論エンジンに関しては、ハードウェアの進歩、特に並列計算モデルの導入などにより、ますます高速の処理が可能となると期待される。マン＝マシン＝インタフェースの問題についても、最近のワークステーションの進歩により、大幅な改善を期待することができるであろう。

そこで最後に問題として残るのが、知識獲得の問題である。この知識獲得の問題の解決こそが、知識ベースシステムの将来を左右することと言っても過言ではあるまい。知識獲得の問題の多くは、対象領域に依存した形でしか整理されていないことが多く、また一般性のある形で定式されたものについては、現在

までのところ比較的限定された有効性しか持っていない²¹⁾。今後、より一般性のある、有効範囲の広い知識獲得の手法と、その知識表現言語への組み込みが研究されねばならない。同時に、最終的には人間がシステムの動作を確認し、その正当性を証明したり、不足ないし誤った知識を発見することを支援する機能についても、より一層の検討を必要とすると考えられる。

実用的知識ベースシステムは、相当程度大規模な知識ベースを必要とすることが予想される。そこで将来的には、完成した知識ベースシステムをどのように保守していくかが問題とされねばならないであろう。特に進歩の激しい分野における知識ベースシステムでは、新しく追加された知識を、既存の知識と適切な形で統合するための方法論が問題となる可能性がある。

大規模知識ベースの開発と保守にあたっては、単一の人間ではなく、複数の人間による共同作業が必要となるであろう。このような状況のもとでは、複数の人間から、知識ベース変更の要求を受け、しかも知識ベース全体の整合性を保証するための方法論が要求されると予想される。すでにデータベース管理システムでは、同時更新に対して Integrity を保証する方法論が確立されているが、知識ベースシステムについても、将来的には類似の機能が必要とされると考えられ、当然知識表現言語の中にも反映されねばならないであろう。

知識ベースシステム実用化の期待が高まるとともに、知識ベースシステム開発環境としての知識表現言語の重要性もまた高まってきている。近年のコンピュータの、ハードウェア、ソフトウェア両面の進歩も、強力な知識表現モデルとその適用機構の実現、良好なマン=マシン=インタフェースの提供などの面で、有力な知識表現言語の実現を可能とする環境を整えつつある。現に強力な機能を備えた知識表現言語も出現してきている。しかしながら知識ベースシステムの将来とその実用化を考えるならば、知識表現モデルとその適用メカニズムに関する一層の研究とともに、大規模知識ベースの開発と保守に関する方法論の確立が必要になることが予想される。この問題に関しては、知識表現言語による、知識獲得の支援、複数人の共同作業による知識ベースシステムの開発と保守などが検討されねばならないと考えられる。

参 考 文 献

1) ed. by Barr, A. and Feigenbaum, E. A.: The

- Handbook of Artificial Intelligence, Vol. I-III, William Kaufmann Inc. (1981-1982).
- 2) ed. by Hayes-Roth, F., Waterman, D. A. and Lenae, D. B.: Building Expert Systems, Addison Wesley (1983).
- 3) Nilsson, N. J.: Principles of Artificial Intelligence, Tioga Publishing Co. Palo Alto (1980).
- 4) Feigenbaum, E. A.: The Art of Artificial Intelligence, Proc. IJCAI (1977).
- 5) Stefic, M. et al.: The Organization of Expert Systems, Artificial Intelligence, Vol. 18, pp. 135-137 (1982).
- 6) Clancy, W. J.: The Epistemology of a Rule-Based Expert System-A Framework for Explanation, Artificial Intelligence, Vol. 20, pp. 215-251 (1983).
- 7) Swartout, W. R.: XPLAIN: A System for Creating and Explaining Expert Consulting Program, Artificial Intelligence, Vol. 21, pp. 285-325 (1983).
- 8) Winston, P. H.: The Psychology of Computer Vision, McGraw-Hill (1975).
- 9) Smith, R. G. and Friedland, P. E.: Unit Package User's Guide, HPP-80-28, Stanford University (1980).
- 10) Bobrow, D. H. et al.: The LOOPS Manual, memo KB-VLSI-81-13, Xerox Palo Alto Research Center (1981).
- 11) Weinreb, D. and Moon, D.: Flavors: Message Passing in the Lisp Machine, A. I. Memo-602, A. I. Labo. M. I. T. (1980).
- 12) VanMell, W. et al.: The Emycin Manual, HPP-81-16, Dept. Computer Science, Stanford University (1981).
- 13) Nii, H. P. and Aiello, N.: AGE: A Knowledge-Based Program for Building Knowledge-Based Program, Proc. of 6th IJCAI (1979).
- 14) Forgy, C. L. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, Vol. 19, pp. 17-37 (1982).
- 15) 小山照夫, 開原成允: 時間経過の概念を含む汎用医療コンサルテーションシステム, 情報処理学会論文集, Vol. 23, pp. 414-420 (1982).
- 16) 上野晴樹: COMEX マニュアル, 東京電機大学理工学部 (1982).
- 17) 伊藤秀昭, 上野晴樹: ZERO: Frame+Prolog, Proc. of the Logic Programming Conference (1985).
- 18) 大須賀節雄: 知識ベースおよびデータベースを用いたモデル構築技法, 情報処理学会, アドバンス・データベース・システム・シンポジウム資料 (1982).

- 19) 小野典彦, 小林重信: 手続き指向型プロダクションシステム Po-Ps とその応用, 情報処理学会第 36 回知識工学と人工知能研究会資料 (1982).
- 20) 中島秀之: PROLOG/KR USER'S MANUAL, 東京大学工学部計数工学科 (1981).
- 21) Shapiro, E. Y.: Inductive Inference of Theories from Facts, Yale University, Report No. 192 (1981).
- 22) Davis, R. and Lenat, D. B.: Knowledge-Based Systems in Artificial Intelligence, McGraw-Hill (1982).
- 23) Friedland, P.: Knowledge-Based Hierarchical Planning for Molecular Genetics, doctoral dissertation, Stanford University (1979).
- 24) McDermott, J.: RI: A Rule Based Configurer of Computer Systems, Artificial Intelligence, Vol. 19, pp. 39-88 (1982).
- 25) Shortliffe, E. H. et al.: ONCOCIN: An Expert System for Oncology Protocol Management, Proc. of 7th IJCAI (1981).

(昭和 60 年 8 月 6 日受付)