TM-0069

第五世代コンピュータ入門
—そのハードウェアシステム—

内田俊一

August, 1984

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

第五世代コンピュータ入門
そのハードウェアシステム

財団法人　新世代コンピュータ技術開発機構
内田　俊一

１．機械ができる推論とは？

第五世代コンピュータは、「考えるコンピュータ」などと言われています。それは、このコンピュータが、私達の話す自然言語（に近い言語）を理解したり、それ自身が、限られた分野とはいえ、機械の設計や、オフィスにおける文書管理などについての専門知識を持ち、それを用いて推論をおこない、私達人間が、やらせたい仕事をこと細かに指示しなくともやる能力を持つと考えられていることによっています。このような能力、または機能は、知識情報処理とか人工知能と呼ばれているものです。

第五世代のコンピュータは、従来のコンピュータと異なり、そのソフトウェアもハードウェアも、このような知識情報処理に向いた機能、構成をもつことを目ざしています。具体的には、知識ベースを用いた推論機能を持たせようとしているわけで、ここでは、そのハードウェアに重点を置いて、コンピュータ、すなわち、機械ができる推論とは、どんなものか考えてみたいと思います。

推論とは、論理学の言語で、大きく分けて、帰納的推論と演繹的推論があります。ちょっと難しそうですが、実は、それ程でもありません。図－１を見てもらうとわかるように、いろいろな事実から一般的な規則（ルール）を導き出すのが、帰納的推論です。「すずめは飛ぶ」、「カラスは飛ぶ」なんていう事実から、「鳥は飛ぶ」なんていう一般的規則を得るのは、これにあたり、学習することの根本とも深く関連します。しかし、常に例外的事実、たとえば、ペンギンのような鳥もいるわけで、このようにして得られた一般的規則は、正しくない時もあり、これを使って推論するときは、その結果が正しいか別途検証する必要があります。ともかく、機械、特にハードウェアでやろうとする時は、こういう推論は手に負えませんから、ソフトウェアにまかせることになります。

もう一つの推論は、演繹的推論で、図－２に示すように、一般的規則（ルール）は、完全に正しいものとして、これから具体的事実が導き出されるか否かを求めるものです。この推論は、機械向きですが、実は、これとても、その範囲は広く、機械でやる場合には、その一部を実現するので満足しなければなりません。すなわち、図－３に示すように、推論を効率よく実行する手法（アルゴリズム）のわかっている部分のみを対象とすることになります。しかし、これでも、従来のノイマン型コンピュータに比べれば、ずっと機能は高く、自然言語理解などのためのソフトウェアなどが、ずっと作りやすくなることが期待できます。

また、この図に示すホーン論理の世界を基盤として作られるプログラミング言語は、従

来のコンピュータのように逐次型で実行もできますし、並列にも実行できるという特徴があります。これは、高速のコンピュータを必要とする時、並列処理を利用できることを意味しており、高速のハードウェアを実現するための言語としても都合のよいものです。

第五世代コンピュータの研究開発プロジェクトでは、最終ゴールとしては、並列推論マシン（ＰＩＭ）を目ざしていますが、当面必要なソフトウェアの研究開発ツールとしては、作りやすい逐次型推論マシン（ＳＩＭ）を開発しており、これは、すでに、その中核ハードウェアの試作に成功しています。

２．推論マシンのためのプログラミング言語

推論マシンは、逐次型であれ、並列型であれ、このホーン理論をベースとする言語を実行します。この言語は、論理型言語と呼ばれ、PROLOGと呼ばれる言語が有名です。第五世代コンピュータ・プロジェクトでは、同種の言語を設計していますが、私達は、これを核言語（ＫＬ）と呼んでいます。では、この言語において、推論がどんなふうに行われるかみていきましょう、

図－４に、これまで規則（ルール）とか事実（ファクト）といってきたものを、具体的に、論理型言語で記述してみました。この言語では、人間は、事物の関係を書けばよく、ノイマン型のBASIC などの言語のように実行順序を細かく書く必要はありません。（もちろん書きたい時は、細部について書けます。）このようにして書いた規則や事実に対して、図－４の⑤のような質問をすると、コンピュータは、孫や子についての規則や事実、さらに、これらに含まれるＸ、Ｙなどの引数間の関係や対応を調べて、Ｘに何をあてはめたらよいか求めてくれます。

この答を求めるプロセスは、論理学における定理証明の手続きそのもので、その基本操作は、図－６に示すような“孫”、“子”などという文字と引数の形のパターンマッチングとなっています。特に、引数の一致をみる操作は、ユニフィケーション（単一化）と呼ばれています。図－６に示した、ブロックの色と切り口の一致をとる操作は、三段論法における一段の推論と対応していることから、マシンの実行速度は、この操作が１秒間に何回できるかで計ることができ、その含意をLIPS（リップス）と呼んでいます。

図－５のように関係づけられたプログラムは、逐次的でも、並列的にでも実行できるわけですが、まず、逐次型的に実行する場合を考えてみましょう。引数Ｘ、Ｙに何をあてはまたらよいかが問題となるわけですが、逐次実行では、図－７のように、試行錯誤が繰り返させます。

図－７の①の子（Ｘ，Ｙ）に対し、1a，1b，1cをそれぞれあてはめてみて、それが、②の子（Ｙ，Ｚ）を満足するかみるわけで、これは、三度目のやり直しで成功します。何かずい分と面倒なことをしているようですが、このような解答の探索はすべてマシンがやってくれるわけで、その分、人間が楽をすることができると言えましょう。知識ベースを用いて複雑な推論をする場合には、このようなマシンの働きが、目的とするソフトウェアが

実現できるか否かの鍵となります。

３．推論マシンへのアプローチ

図－６、図－７で示したように、論理型言語のプログラムは、上が根で、下が葉となるように枝分れした木とみることができます。その枝分れには、２種類あり、何本かに分岐した枝の、すべてが正しくマッチングに成功しなければいけないAND 関係を示す枝分れと、少くとも１本正しくマッチングに成功すればよいOR関係を示す枝分れがあります。論理型言語の実行とは、このような枝葉から成る木（ツリー）を、枝から枝へと渡り歩く探索のプロセスにほかなりません。

その探索のやり方をみるために、少し手のこんだプログラムを図－８に示しました。ここでは、プログラムと木構造の対応を明確に示すため引数は省略してあります。

このプログラムについて、まず、逐次型の実行順序を示してみましょう。図－９の(a)は、まず、B1で成功したので、C へすすんだところ、ここで、マッチング（ユニフィケーション）に失敗。そこで、残っているORの枝B2へ戻った場合を示しています。失敗したら戻る操作を後戻り（バックトラック）といい、マシンは、このような後戻りに備えて、実行の途中結果をスタックと呼ばれる構造をもつメモリへ蓄えておきます。この中身は、残るORの枝がなくなると、そこの分までが捨てられ、また新しい中身を入れられるようになります。図－９の(b) は、求める解答が、最後の枝にあった場合です。逐次型の実行では、この木の探索は、上から下へ、また、左から右へと進みます。（Depth-first, left to right Search ）逐次型推論マシンは、イメージ的には、図－１０のようなスタックを操作しながら、ブロック合せをやっているといえましょう。しかし、通常その速度は早く、私達の研究所で作られたＰＳＩ（パーソナル逐次型推論マシン）は、このブロック合せを一秒間に３万回実行することができます。（30 KLIPS）

しかし、逐次型でやったのでは、その速度にも、おのずと限界があり、将来の規模の大きな知識情報処理の問題に対処することが難しいと考えられます。また、VLSIなど新しいハードウェア技術を有効に利用するためにも、並列処理の実現が望まれています。

並列推論、すなわち、論理型言語の並列実行は、基本的には、AND-ORの関係で結ばれた木構造の探索を並列に行うことと言うことができます。その第１ステップは、図－１１に示すように、ORの枝分れについて、並列に探索を行うことでしょう。こうすることで、求める解答が早く発見できます。しかし、実際は、このような単純な方法では、探索すべき木の範囲が広くなりすぎ、探す範囲を、もう少し細かく指定する必要があります。また、このような実行方式では、いろいろな応用問題の中で必要とされる処理を記述する能力についても十分とは言えません。たとえば、コンピュータシステムの中では、いくつものプログラムが、互いにメッセージをやりとりしながら、走ったり止ったりして動いています。メッセージによって、プログラムの間で、実行の歩調をあわせて（同期をとって）いるわけです。

汎用の並列推論マシン用の言語としては、このような同期のための機能を言語に組入れる必要があり、このような研究も、最近、活発になってきました。そのような言語の一つに並行PROLOGがあります。これは、ホーン論理の世界の枠組みをできるだけ保ったまま、同期などの制御機能を入れたものです。図－１２に、そのプログラム例を示しておきました。

ここでは、計算プログラムから、リスト構造中の値が送り出されるごとに、これと同期して二乗和プログラムが起動される様子を示しています。このような値の到着によってプログラムが起動されるメカニズムは、ハードウェアの研究者によって、ここ10年程度研究されてきたデータフローマシンなどの並列マシンとも、うまく対応するものです。

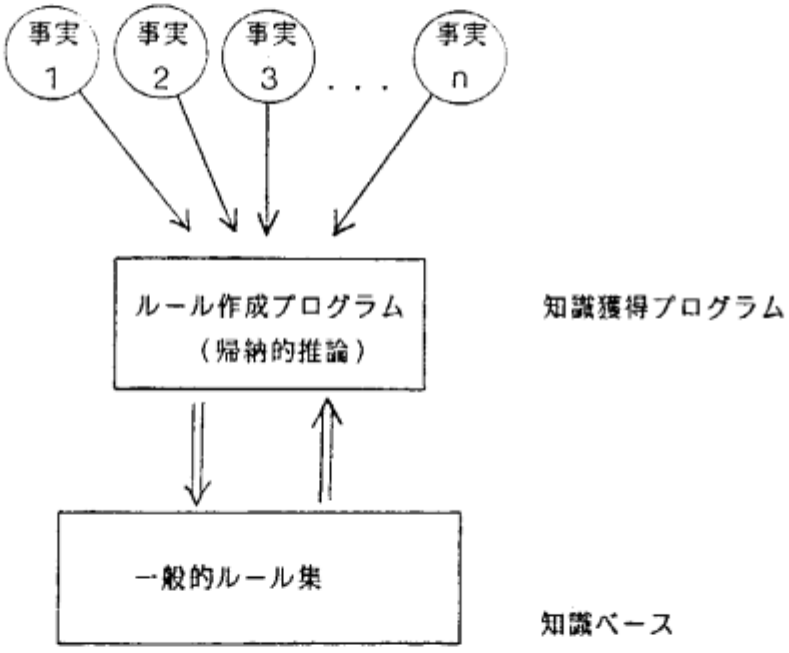
並列推論マシンのハードウェアの実現には、多くの難しい研究上の問題を解決しなければなりませんが、そのイメージは、図－１３に示すような多くのプロセッサや構造メモリがネットワークで結合されたものと考えられています。並列処理用の論理型言語で書かれた知識情報処理の応用プログラムの多くは、並列に動作する大小のプログラム・モジュールから成ると考えられており、それを、この並列推論マシンの各プロセッサへうまく対応づけてのせることも重要な研究テーマの一つとなています。

また、その実際には、VLSIなどの新しい実装技術が不可欠であると考えられています。

３．おわりに

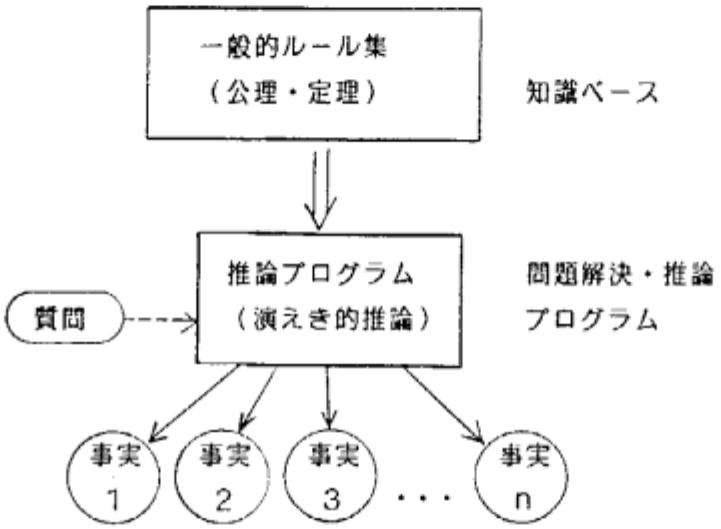
このような推論マシンのハードウェアは、その上に作成されるソフトウェアと組合されて、帰納的推論に基く知識獲得機能、自然言語による対話機能などを実現すると考えられています。また、大量の知識ベースの蓄積と管理のため、これに適した構造をもつ知識ベースマシンも研究されており、これは、最終的には、推論マシンと融合して、第五世代コンピュータのプロトタイプシステムを構成すると考えられています。

新しい論理型の言語、並列処理のマシンの構造、VLSI技術など、第五世代コンピュータは、多くの最先端の技術を結集したものとなっており、まさに夢のコンピュータといえましょう。この夢にどこまでせまれるか、第五世代コンピュータ・プロジェクトは、この夢への大いなる挑戦でもあるわけです。そして、今、世界中から、日本のオリジナルな発想に基く創造的プロジェクトとして注目されているのも、まさにこの理由によっているのです。



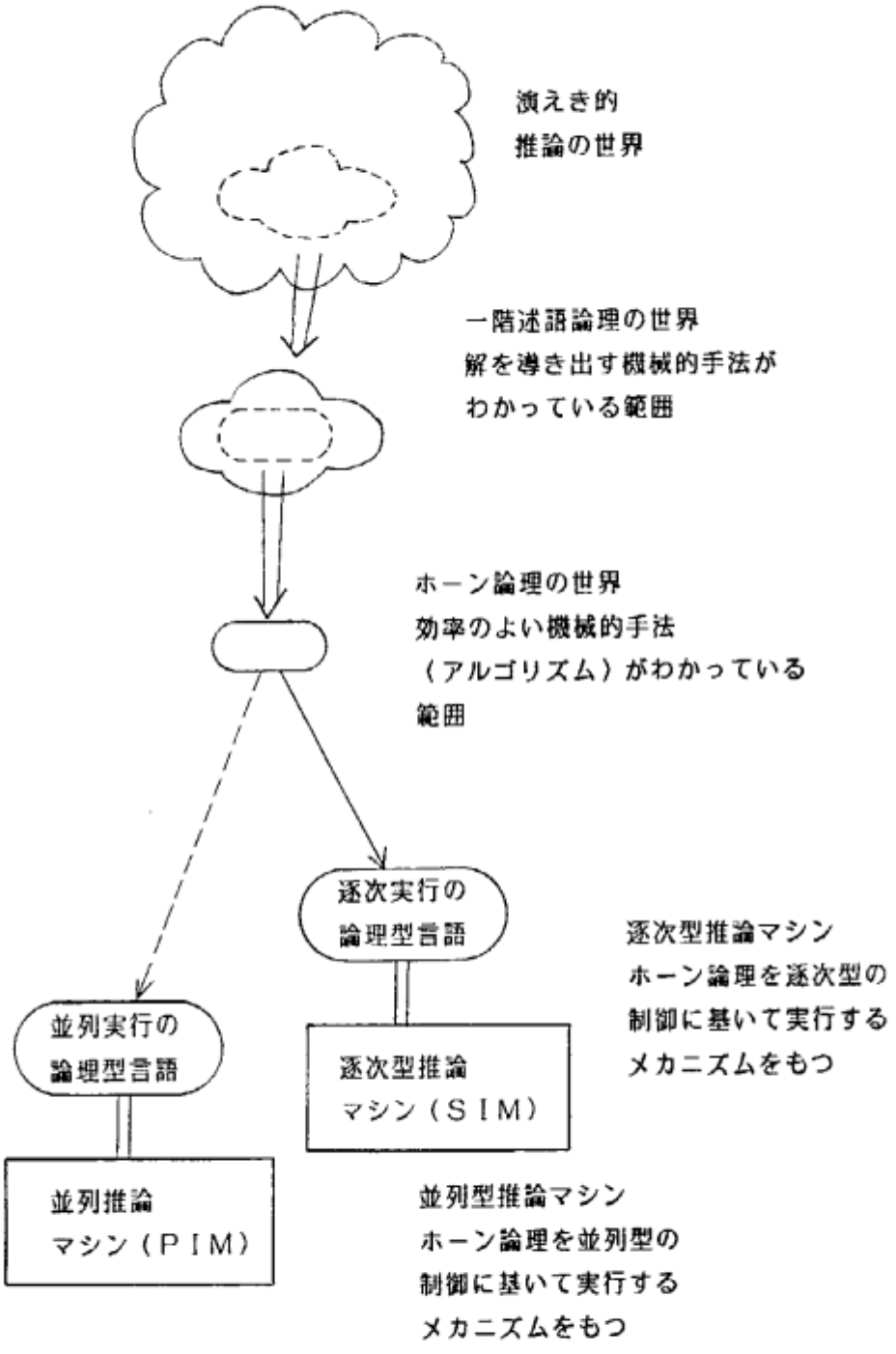
このようにして作られたルール集は、完全に
正しいとは限らない。例外的事実に対しては
正しくない場合がある。

図－１　帰納的推論は与えられた事実から
一般的規則（ルール）を作り出す



演えき的推論では、与えられたルール（規則）を
正しいものとし、与えられた事実がルール集から導き出せるか否か
（真／偽）を判定（証明）する。

図－２　演えき的推論を用いると、与えられた
質問（条件）を満足する答（事実）が得られる



図－３　推論の機械（マシン）による実現

規則（ルール）と事実（ファクト）

規則…　変数を含む。よって、一般的

① XはYの子である。 (1-a)

↓

子（X，Y） (1-b)

② XがYの子であり、YがZの子であれば、XはZの孫である。 (2-a)

↓

孫（X，Z）：－子（X，Y），子（Y，Z） (2-b)

事実…　変数を含まない。よって具体的

③ 秀忠は家康の子である。 (3-a)

↓

子（秀忠，家康） (3-b)

④ そのほか、

子（秀康，家康） (4-1)

子（家光，秀忠） (4-2)

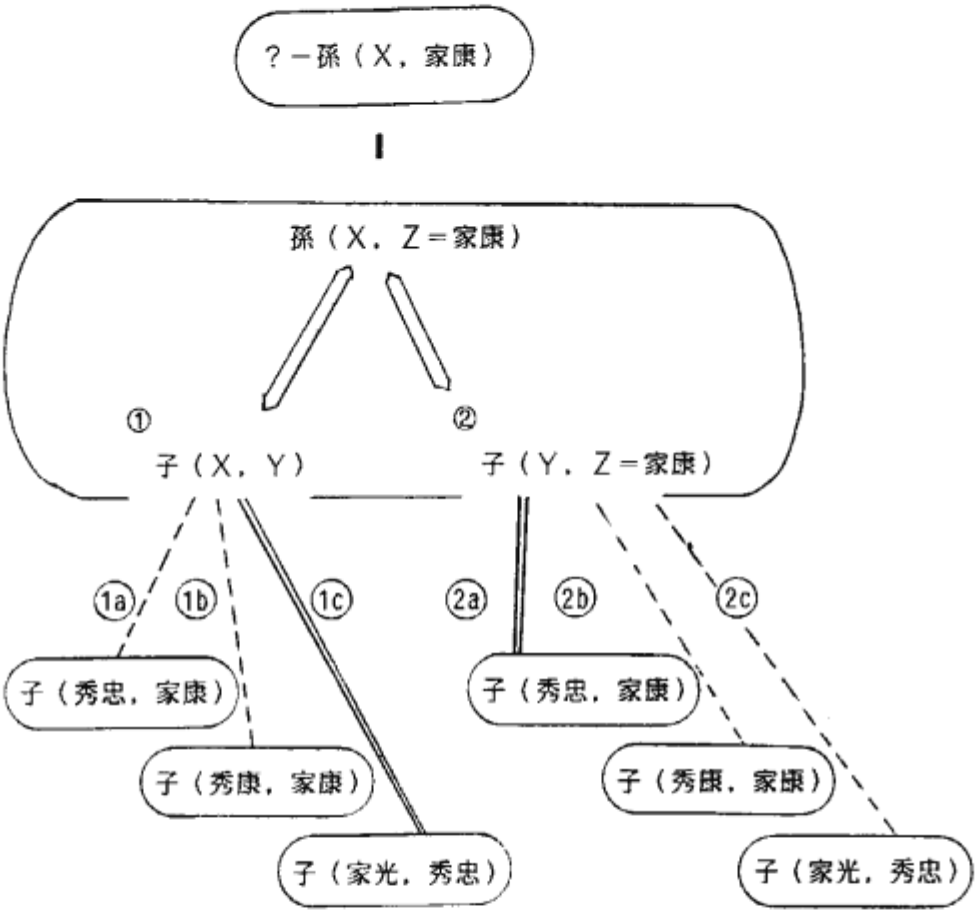
質問…　規則と事実をもとに解を求める。

⑤ Xは家康の孫である。Xを求めよ。

↓

？－孫（X，家康）

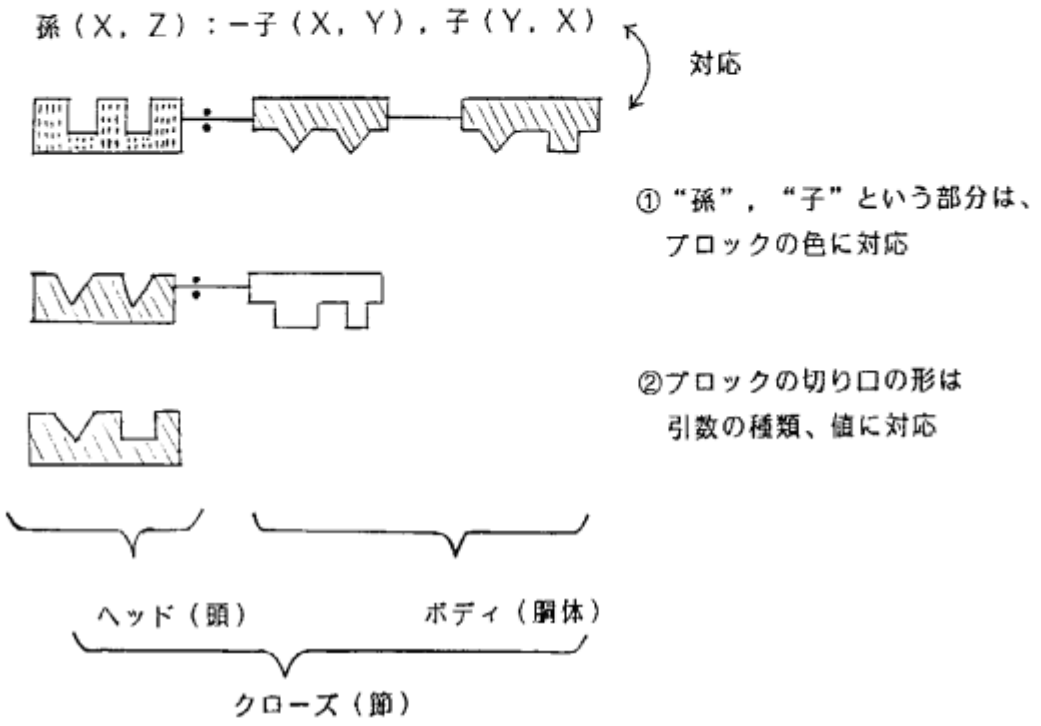
図－4　規則と事実の簡単な例



①と②について、1cと2aの組合せのみが
“孫”の条件を満たし、X＝家光が解として得られる。

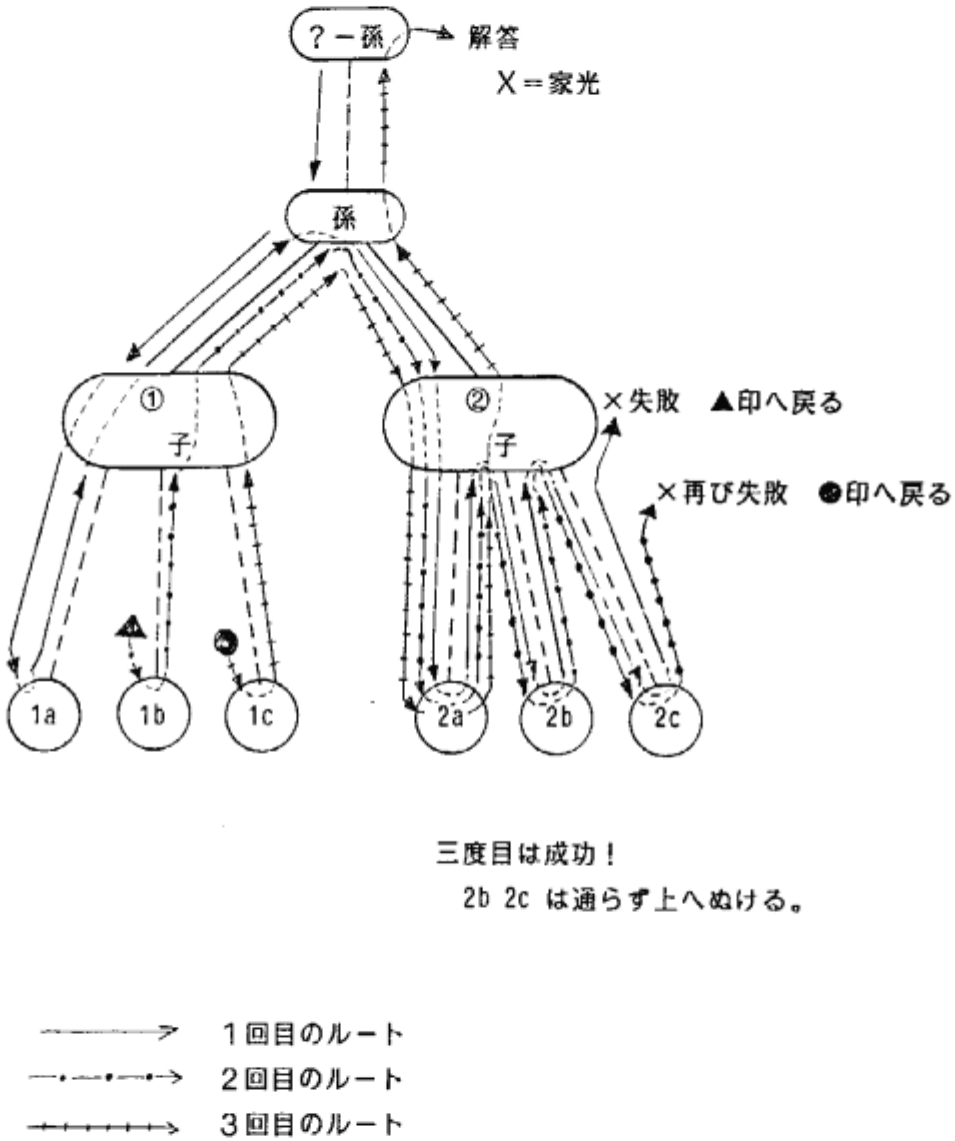
この条件を満たす組合せを得るまで、マシンは、
試行錯誤をくり返す。

図－5　家康の孫Xを求めるやり方

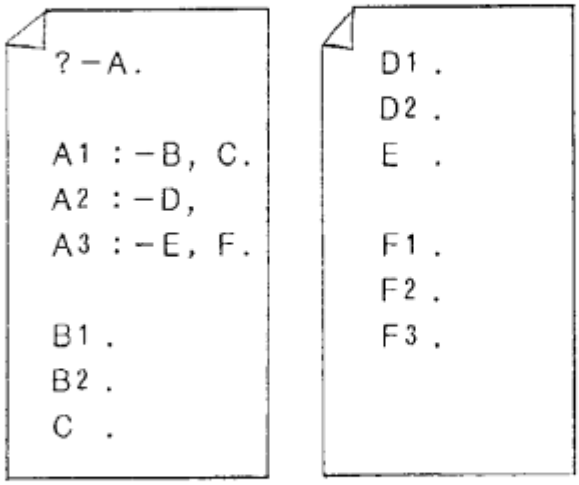


論理型言語によるプログラムの各行は、
節（クローズ）と呼ばれる。ボディに含まれる
“子（X，Y）”は，ボディ・ゴールと呼ばれ、
これと色、切り口の形のマッチするヘッドを探し
出す操作が基本的な推論の単位となる。
特に、切り口（引数）の一致をみる
操作は、単一化（ユニフィケーション）と呼ばれる。

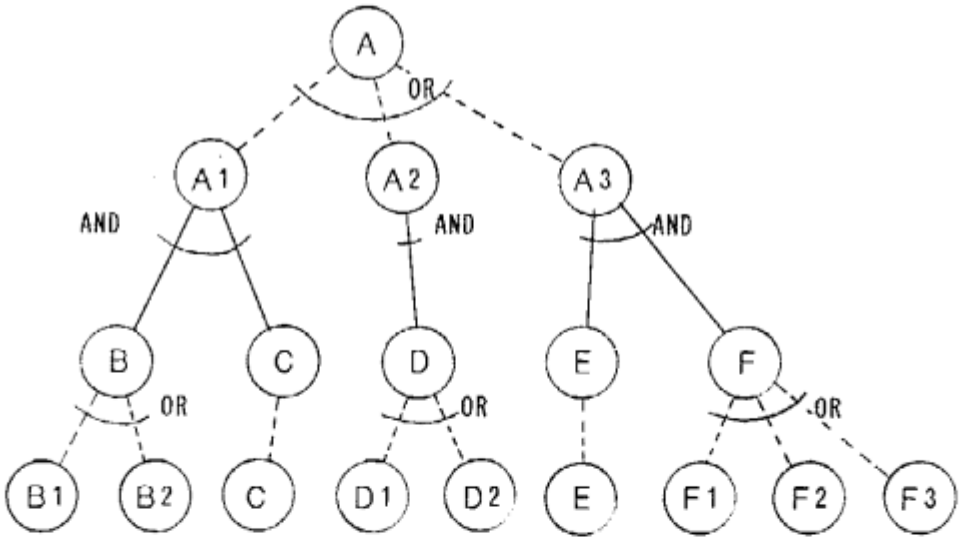
図－6　推論の基本操作と
単一化（ユニフィケーション）



図－７　逐次型の場合の解答の探索ルート

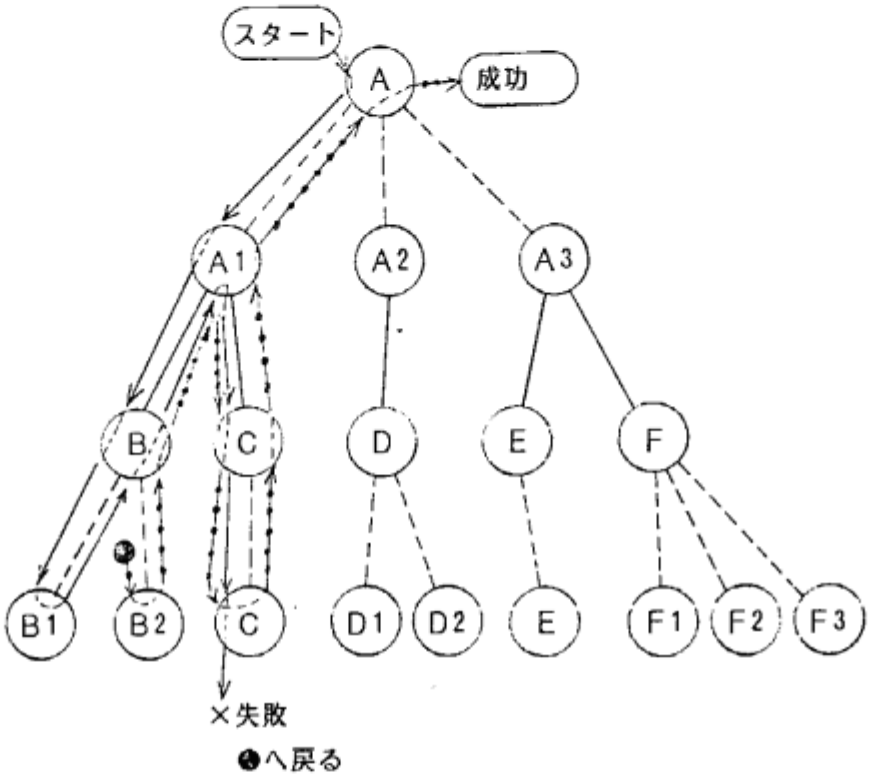


(a) 論理型言語のプログラム（引数は省略してある）



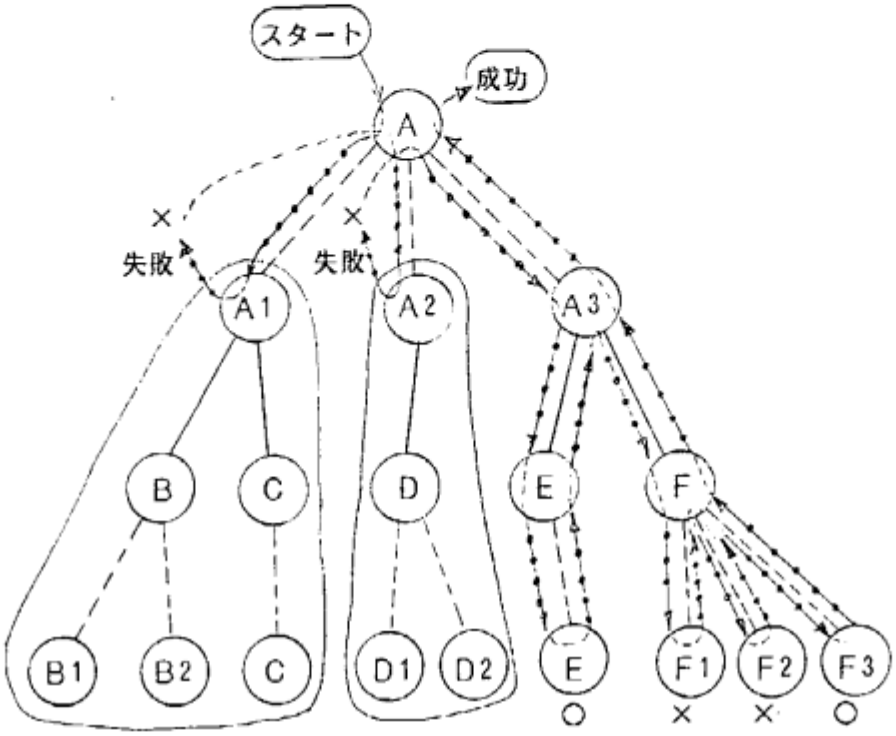
(b) 上のプログラムを、木（ツリー）の形に書いたもの
根が上、葉が下である。AND とORの関係に注意

図－8　論理型言語のプログラムと
AND-OR木（アンド・オア・ツリー）



(a) ORの枝は、１本でも成功したら先へ行く。
先の方で失敗(C) したら、残っている枝B2
をやってみる。（後戻り）

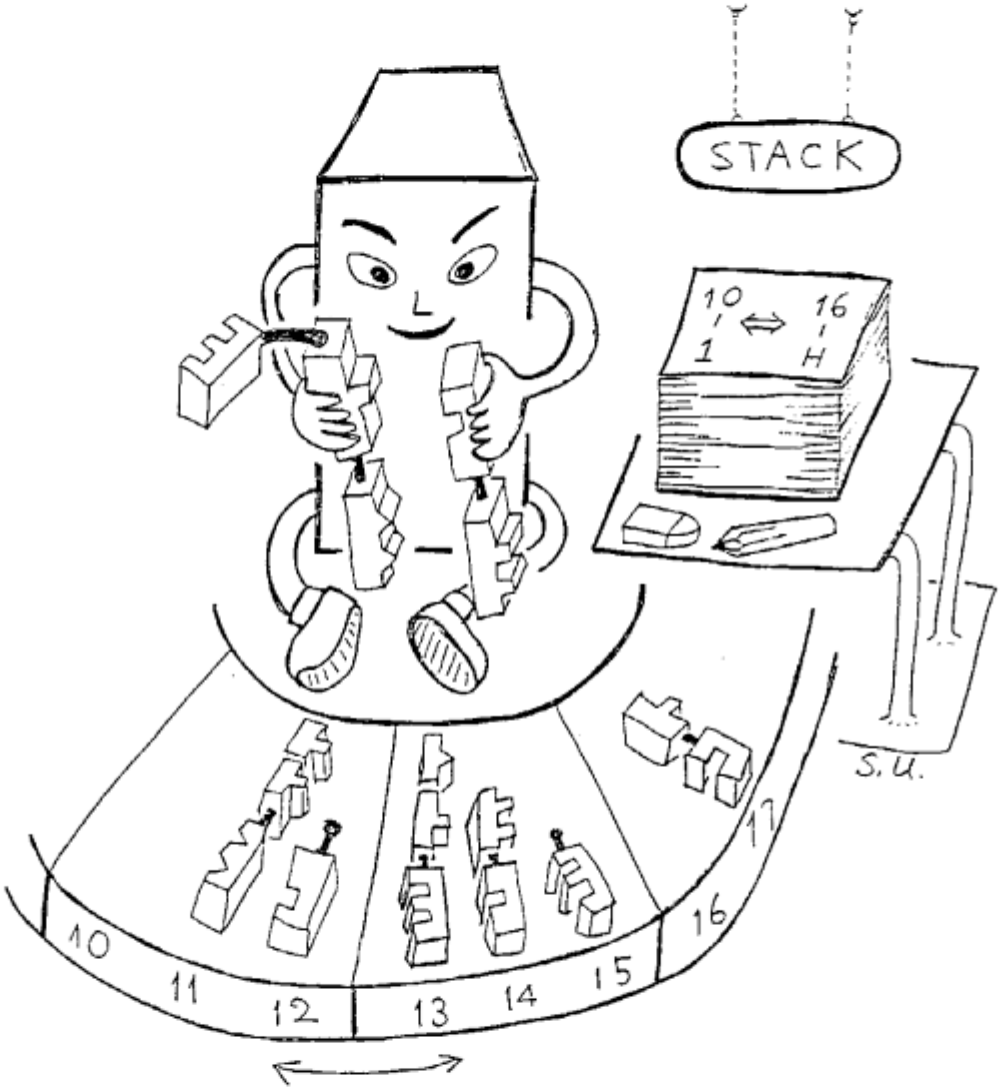
図－9



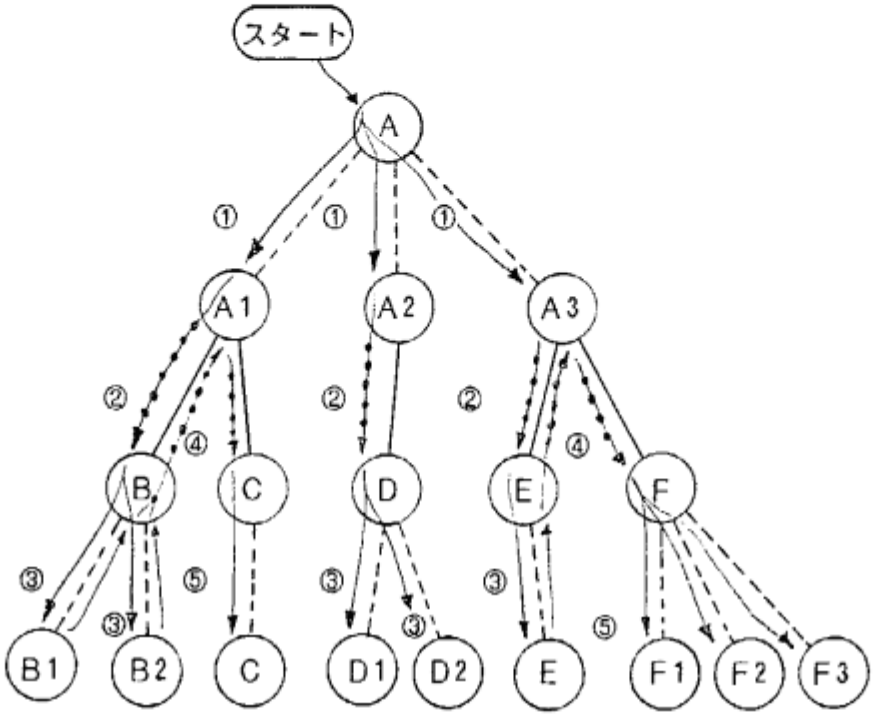
(b)　A1，A2 の探索に失敗し、A3 で
やっと成功した場合の最長ルート

逐次型の実行は、AND-OR木を、まず
左端の枝からはじめ、深さ方向へ末端まで
すすむ。ORの分岐では、左の枝から試み、
成功すれば、残りの枝は、残して進む。
AND の枝は残さずたどる。
（Depth-first, left to rightのアルゴリズム）

図－9　逐次処理の実行順序



図－１０　逐次型推論マシン



点線で示されたノードの分岐（ORノード）の
探索は、並列に実行される。

A1，A2，A3 の実行、B1，B2 の実行、
D1，D2 の実行、F1，F2，F3 の実行は
並列に行われる。

図－１１　ORノードに着目した並列処理の実行順序

X＝［１，２，３］，Y＝［４，５，６］の
ようなリスト（値を一列に並べたもの）の要素
を２乗したものを加えて、それぞれ印刷する。

二乗和の計算と印刷は、並列に実行される。
印刷が遅いと、計算は先に進む。
“？”マークは、待ちあわせを示す。

プログラム（並行PROLOG）

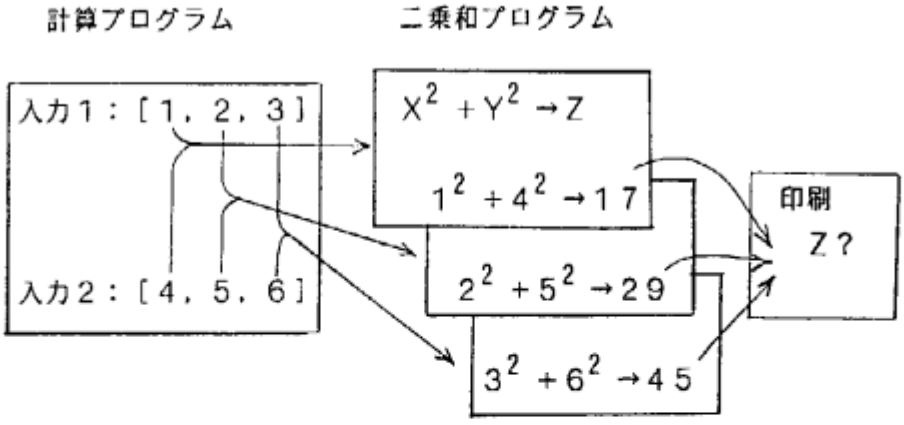
待ち合せる。

```
計算：－二乗和（［１，２，３］，［４，５，６］，Ｚ），印刷（Ｚ？）．

二乗和（［　］，［　］，［　］）．
二乗和（［Ｘ｜Ｘｔ］，［Ｙ｜Ｙｔ］，Ｚ）：－
        （Ｘ２　：＝Ｘ＊Ｘ）＆（Ｙ２　：＝Ｙ＊Ｙ）＆
        （Ｔ：＝Ｘ２＋Ｙ２）＆（Ｚ＝［Ｔ｜Ｔｔ］）．
         二乗和（Ｘｔ？，Ｙｔ？，Ｔｔ）．
```

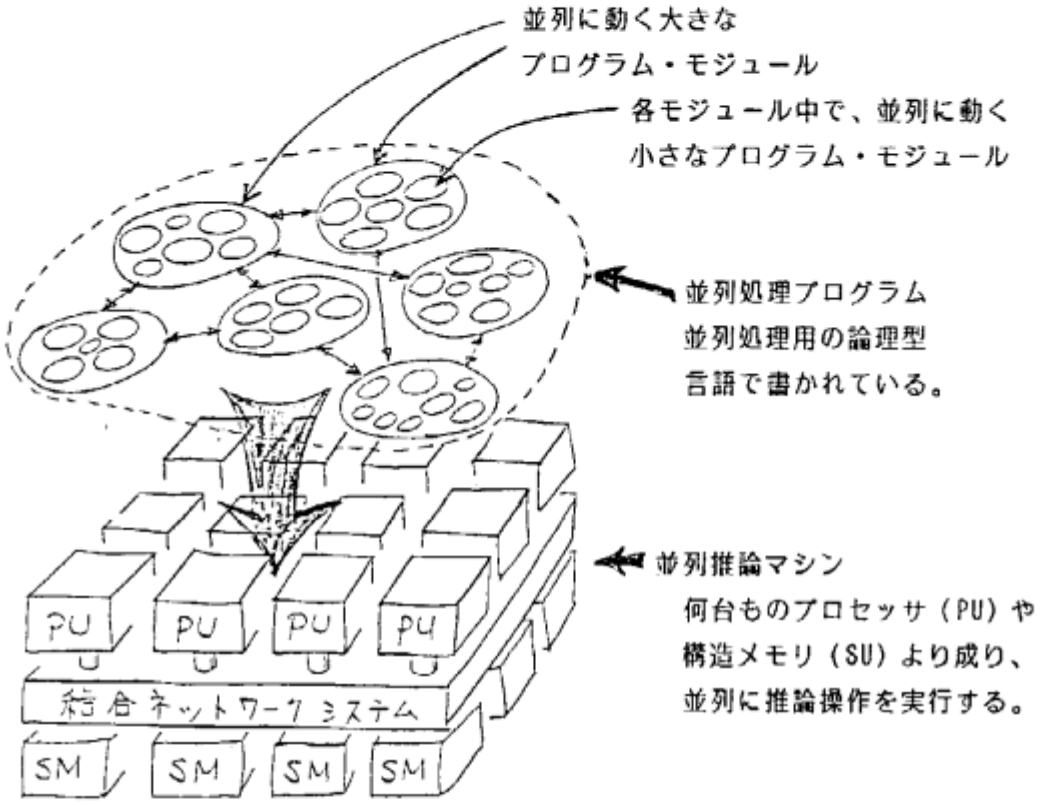
$Z = X^2 + Y^2$ を計算する。

データ到着ごとに
次の計算が
はじめられる。



入力の値のペアが、次々と流れてきて、二乗和の
入力Ｘ，Ｙへ到着するたびに計算が起動される。
（ストリーム処理とデータ駆動）

図－１２　プログラムの同期をとる操作と並行PROLOG



自然言語処理やエキスパート・システムなどは、
並列処理用の論理型言語で記述され
並列推論マシン上で実行される。

図－１３　並列推論マシンのイメージ