知識工学と人工知能 47 － 7
（１９８６．７．１１）

Prolog様言語による知識表現の試み

三値Prolog

磯道良典・牛田聡史
広島大学

人工知能の研究と切りはなせないのが、知識の表現方法である。雑多な情報をただコンピュータに入れただけでは、コンピューターはそれを生かすことができない。有用な人工知能の制作には、有効な知識獲得手段とともに、知識を機械に利用させるための適当な表現方法が必要である。そのためにＬＩＳＰ・ＰＲＯＬＯＧをはじめとして、色々なコンピュータ言語の研究がされている。
本論文では、そのうちのPrologを取りあげ、それの拡張を行い、より効果的な知識表現のできるツールにした例を示す。

An experiment to knowledge representation with designing Prolog-like language.

— Tri Value Prolog —

Yoshinori Isomichi, Satoshi Ushita
The University of Hiroshima Nobeoka, Hiroshima 730, JAPAN

A way of knowledge representation is important for researching Artificial Intelligence. Computers cannot use confused information if we simply input them. An effective way for getting knowledge and an proper language for representing knowledge are important. For that, we use various computer languages Lisp, Prolog, and so on.
We'll show you an effective tool that represent knowledge more effectively expand usual Prolog.

人工知能に関する文献の中には、しばしば「知識表現（knowledge representation）」という言葉が使われている。これには、どういう意味がこめられているのであろう。

たとえば、教科書を一冊まるごとコンピューターに記憶させたとして、これが人工知能（ここでは、知識処理を行う機械の意味で用いることにする）といえるであろうか。答はもちろん"No"である。たとえ、それに、検索機能や編集機能など データを扱うための便利なツールが付いて使い易いものになったとしても所詮は メモ帳の域を出ない。教科書の中に不定積分の公式が書いてあっても、面積を求めるのは人間である。病気の症状が書いてあっても、診断を行うのは人間である。すなわち、メモ帳＝情報提供・人間＝問題解決という図式である。人工知能に関する研究は この図式を、人工知能＝情報の保持および問題解決・人間＝監視および利用というように作りかえることだと言って良い。

I. 知識の表現

上で説明したことを、別の側面から見てみよう。メモ帳に入っているのは、雑多な情報である。これから有効な情報（知識）を引き出すのは、あくまで人間の仕事である。それに対して、人工知能は知識を保持し、さらに、それを問題解決に応用するといった作業を行う。そのために 機械にとっては、単なる文字のよせ集めにすぎない情報から、実際に応用のできる知識をぬき出すこと、そして、それを機械が利用しやすく、更に大事なことは、人間が理解しやすい形に表現することが必要になってくる。それが、知識表現というものである。

	機械の役割	人間の仕事
メモ帳	情報提供 ＝ 情報の保持	問題解決 ＝メモ帳から知識を引き出し、応用
人工知能	問題解決 ＝ 知識の保持 知識の利用	監視

実際に知識を表現する場合にはどうすれば良いであろうか、そのために、いくつかの表現方法が考えられている。

a. フレーム
b. 意味ネットワーク
c. プロダクション・システム
d. 論理プログラミング

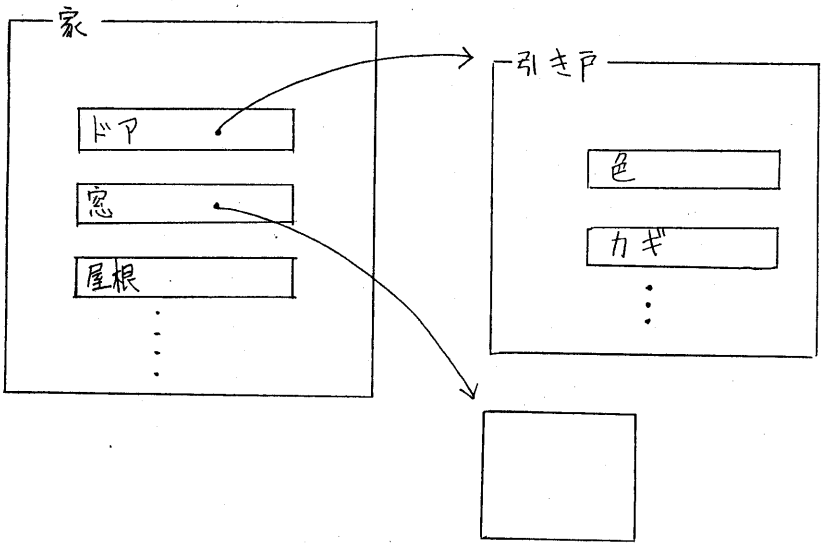
主なものとしては、上の4つがあげられる。
a. フレームは、人間が物事を理解するときのように、その物に対する枠組み

を持っており、それに基いて対象をとらえようとする考え方である。個々のフレームは互いに結合しあっていて、複雑な現実世界に対応させている。それぞれは非常に簡単な事象に関する情報を持っているにすぎず そのため取り扱いも容易である。

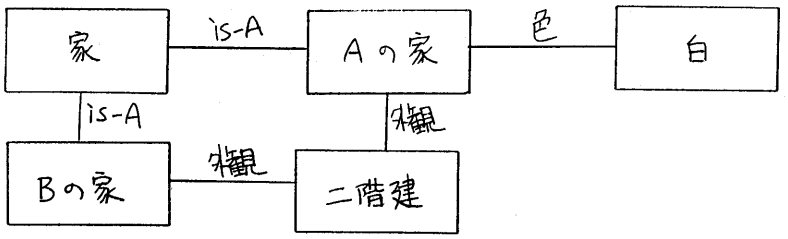
b. 意味ネットワークは、やはり人間の記憶形式をモデルに作られた。意味ネットワークはノードとリンクからなり、ノードは個々の概念に、リンクはその間の関係に対応する。そしてノードとリンクの寄せ集めにより複雑な知識を表現しているわけである。

C. プロダクション・システムは、プロダクションと呼ばれる <条件>→<行動> のルールの集まりである。プロダクションの個々は、非常に簡単なルールであり、しかも他のルールとは切りはなして考えられるので楽である。これは、エキスパートシステムの分野で多く利用されている。

<図I-2 家のフレーム表現>



<図I-3 家を意味ネットワークを用いて表現>



知識の表現方法として最後に残った d. 論理プログラミングとしては Prolog が有名である。人間の思考に比較的近いとされている論理学を利用し、論理学の記法で書くことがプログラミングに、証明手続きが、問題解決の手法に相当する。Prologは、ホーン節に限定された用語によってプログラミングを行い、導出原理によって問題解決をするわけである。

a. フレーム・b. 意味ネットワーク・c. プロダクションシステムの3つは事実を書き並べることによってプログラミングを行うわけだが、論理プログラミングでは、それだけでなく、事実に関する知識・推論を行う・問題解決を行うための知識とかいうものを記述しやすい利点がある。しかし そのような論理プログラミングも、複雑な現実世界を表現しようとすると、曖昧な記述が許されてないことや 知識の増大に伴って、解の探索に組み合わせ的爆発が起こることもあり、十分に使いこなすことは難しい。

ところが、人間の場合を考えてみると、人は不完全で信頼性の低いデータから割合に有効な答を引き出している。また、この推論で、論理学のルールを厳密に適用しているわけでもない。実際問題として、ほとんどの場合には不完全、不正確なデータを用いた推論を要求されるため、当然、機械にもそれを行なわせたい。そこで、論理プログラミングにおいても人間の推論のメカニズムを参考にした改良が行なわれている。このような改良を加えられた論理は、直観主義論理という呼び方をされている。

e. 非単調論理
f. THNOT
g. サーカムスクリプション

古典論理では、公理が完全であることが必要である。言いかえると、完全な知識を持っていて、その知識は永遠に正しい、新しい知識が加わったところで、もとからあった知識が変更されることはない、わけである。これに対して、新しい知識が加わることによって古い知識が否定されてしまうことがある論理は、知識(公理)の増加が、それによって導ける結論(定理)の増加を意味しないことから、非単調論理という。

非単調論理の一種で使われているのが THNOT やサーカムスクリプションである。(THNOT P) というのは、Pが証明できないときに真である。必ずしも(NOT P)が証明される必要はない。これは、そうであることが証明されない限り、そうでないと思うことにしよう。と受けとることができる。これに対してサーカムスクリプションは、証明できないものは、すべて偽である。という考え方である。本言語では、この、証明できない という部分に着目して、真理値を真=証明できる・偽=そうではない・第三の真理値=証明できないが偽ではないの3つに拡張した論理を用いている。

真理値	真	第三の真理値	偽
対応する表現	証明できる	証明できない	そうではない
	P	古典論理にはない	not P
		THNOT P	
		サーカムスクリプションでの not	

（図I－4　真理値とそれを表現する言葉の関係）

II. 閉世界・開世界と第三の真理値

知識をプログラムとして記述する時に、"日本の県名"のように、すべてを書き並べることが可能な場合と"人名"のように、書きつくすことが不可能な場合とがある。

県（青森）
県（秋田）
県（岩手）
　⋮
県（沖縄）.

人名（太郎）
人名（次郎）
人名（三郎）
　⋮
人名（八郎）

それぞれを Prolog 風にプログラミングすると上のようになる。これに対して

県（New-York）　　人名（Smith）

という問を行ってみると、Prolog ならば　ともに False になる。（図I-4）でいうところの、「サーカムスクリプションでの not」に相当する。しかし、直観的にとらえると、"県名"と"人名"では、少し状態が違うと感じられる。"県"の場合は、本当の意味での False であるが、"人名"の場合は、ただ単に証明できなかっただけにすぎず、いわば消極的な False である。この前者の場合を（図I－4）の"そうではない"、後者を"証明できない"に積極的に割りあてることによって、積極的な False と、消極的な False の区別をつけることができる。知識表現の言葉でいうならば、証明できないものはすべて false である、あるいは、その事象に関しての完全な知識を持っている前者は、閉世界を表わすといえ、それに対して、完全な知識を持っていない後者は、開世界といえる。そういった視点から見ると、Prolog は、閉世界に基いた言語であり、本言語では、2つを自由に使い分けることができるということになる。本言語で閉世界と開世界の区別を

するために、例にあげた"県名"のプログラムの最後のように"."を付けることによって、"県"が、開世界として動作をすることにする。

	定義あり	定義なし
閉世界	真	偽
開世界	真	第三の真理値
Prolog	True	False

(図II-1 本言語の真理値表とPrologの比較)

今の例は「事実の宣言」の場合だけを扱った。これに対してPrologでのもう一つの表現方法である「規則」についても、第三の真理値を取り入れたための解釈の拡張を行っている。これを次に示そう。

ここで、例に変数を取り入れて説明を行っている。本言語での変数は、Prologのそれと特に変わるところはない。変数は"*"で始まる語、と定義する。

```
遠足に行く(*日時) ← 晴れ(*日時).
晴れ(今日)
晴れ(一昨日)
晴れ(去年の今日).
```

上のプログラムに対して、遠足に行く(昨日)という問いを行うと、"晴れ"の述語は開世界として定義されているため("遠足に行く"も開世界である)Prologと同様の動作を行いFalseになる。ところが、次のプログラムに対して、

```
遠足に行く(*日時) ← 晴れ(*日時).
晴れ(今日)
```

遠足に行く(明日)の問いを行った場合、直感的に「わからない」と答えるのが妥当と考えられる。本言語でも、これを第三の真理値にあてはめることにする。いうなれば「規則」に真理値の代入のような効果を持たせるわけである。感覚的には前者は、晴れである日についての完全な知識を持っている。したがって、それ以外の日に遠足に行くの証明はできない。後者は、晴れである日がすべてわかるわけではない。したがって資料のない日に遠足に行くの証明は、現在持っている知識ではできない。

本言語では、この第三の真理値のことを「D」、don't care または doubtful の意、と呼ぶことにする。

III. 解の探索

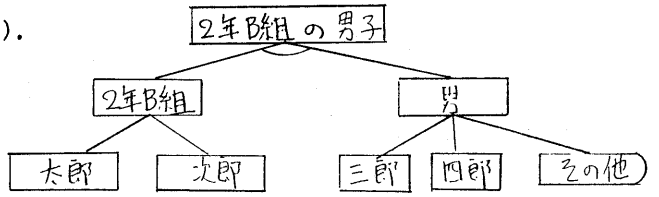
本言語もPrologと同様に、自動バックトラックを含む縦形探索によって解を求める。真理値がTrueまたはFalseである限り、Prologと同様の実行を行う。真理値がDになった時にはそこがTrueであったかのように実行を続けて、一度、解を求めた後に、先のDの選択枝でバックトラックを起こし、別な

解の探索を開始する。
その例を（図Ⅲ-1）以下で説明する。AND/OR木では ∧ がAND ∨ がORである。

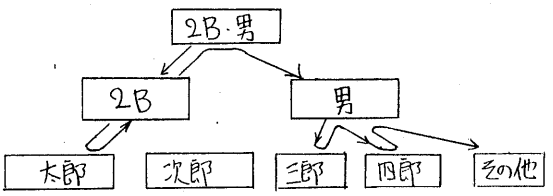
（表Ⅲ-1）

True	そのまま実行を続ける。
False	ただちにバックトラック。
D	そのまま実行し解を求めた後に D の発生した枝でバックトラック。

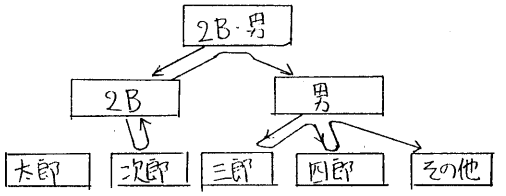
2年B組の男子（*X）
←2年B組（*X），男（*X）．
2年B組（太郎）
2年B組（次郎）．
男（三郎）
男（四郎）



（図Ⅲ-1 プログラムと AND/OR 木）



a: 実行第一段階



b: 実行第二段階

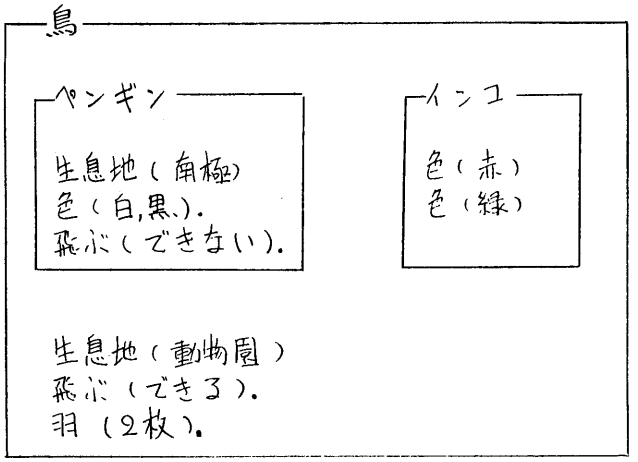
（図Ⅲ-2 実行のトレース）

実行の第一段階では，"男"の枝でDになる。そこで（表Ⅲ-1）に従い，解を求めた後に（すでにDが発生した段階でまっている）バックトラックして"2年B組"の枝へもどる。そこで第二段階へ移るが，またも"男"の枝でDになり，2つめの解を求める。その後，バックトラックする選択枝がないためにこれが最終的な解になる。結果が"D"になった時の出力は，以下のようなものとする。

？2年B組の男子（太郎） ← ？男（太郎）
？2年B組の男子（次郎） ← ？男（次郎）
↑ ↑
・Dであることを示す ・Dが発生した述語呼び出し
・見つかった解をすべてならべる

Ⅳ. 多重世界

Prolog/KR（cf. 参考文献）で実現されている多重世界の考え方を本言語でも取り入れた。多重世界では，個々の概念を各々の世界に当てることによって，概念の間での属性の継承が非常に簡単に記述できる。



述語呼び出し	ペンギン	インコ
生息地（南極）	True	D
生息地（動物園）	True	True
色（赤）	False	True
色（黄）	False	D
飛ぶ（できる）	False	True

（図Ⅳ－1　多重世界でのプログラミングと実行例）

世界は、（図Ⅳ－1）のように入れ子構造を持ち、それぞれの世界で述語が定義されている。原則として内側の世界から外側の世界の述語は参照できる。その逆はできない。

ある世界で述語呼び出しが行なわれた場合、それがTrueまたはFalseの時には、それをそのまま述語呼び出しの結果にする。もしもそれがDならば、一つ外側の世界に同じ呼び出しを行う。こうして、最も外側の世界に到達してもDだった場合に、その述語呼び出しをDだと解釈する。

Ⅴ．おわりに

人工知能の作成は、ほとんどが非常に複雑な世界を対象にしたものになる。そして、扱う対象も明確に記述できるものだけでなく、あいまいなもの、すべてを書き並べられないもの、あるいは、データ数が膨大で、普通に書き並べると非常に能率の悪いものなど、色々な性格がある。

この三値Prologは、特に、不完全な知識から、ある程度有効な推論を行おうという考えで設計し、Prologの拡張という形をとっている。扱うべきデータが十分に集められない時、完全な表現ができない時などには有効なツールとなるだろう。

参考文献

- 中島秀之　「知識表現とProlog/KR」，産業図書（1985）
- 白井良明・辻井潤一　「人工知能」，岩波書店（1982）
- 中島秀之　「論理に基づく知識の表現」，情報処理学会誌 Vol.26 1985 No.12
- 後藤滋樹　「PROLOG入門」，サイエンス社（1983）
- 淵一博　「述語論理的プログラミング――EPILOGの提案」情報処理学会誌 Vol.26 1985 No.11