知識工学と人工知能 40 － 5
（１９８５．５．７）

幾何学問題解答システム

稲永　健太郎　　志村　正道
（東京工業大学工学部）

１　まえがき

本論文では、学習機能を持つ幾何学問題解答システムについて述べる。一般に幾何の問題を取り扱う場合、その手法として、

ａ．必要最小限の公理を土台とし、様々な定理証明を積み重ねることにより、種々の問題に対処する方法。主として中学レベルにおける初等幾何の考え方である。

ｂ．各図形をなんらかの代数的表現に置き換えることにより、問題そのものを代数的に処理する方法。高校レベルにおける座標上の幾何学問題はこれに相当する。

の２通りが考えられる。このうち、ａの手法に基づいて主に証明問題を解くシステムに関しては過去にいくつか研究されてきたが、ｂの手法に基づくものはほとんど前例がない。ここで採り上げるシステムは、このｂの手法をもとに、ｘｙ座標平面上の幾何の問題を解くものである。

本システムはｐｒｏｌｏｇで記述されており、述語形式で入力された幾何の問題を、方程式抽出により代数的に解く。入力処理から方程式抽出までの段階で、本システムは与えられた問題を、内部に予め備わっている図形・幾何学的性質に関する定義，方程式抽出のための規則等に従って処理する。抽出された方程式を代数的に解く作業は手続き的に行なわれるが、その過程でシステムは、自らの行なう代数的処理が幾何学的にどういう意味を持つのかを推測し、その結果として、システム起動時には備わっていなかったある種の幾何学的知識を学習する。学習された知識は、以降のシステムの問題解決に適用可能な形式になっており、これを用いることにより、同一の問題を代数的処理のみによって解く場合に比べて、より高速かつ見通しのよい解法が可能となる。このように、代数的処理という極めて機械的な方法を土台としながら、知識を踏まえた高度な解法へと進んでゆく一つの例を、本論文は示している。

本文中の例は、すべて DEC-10 prolog の構文規則に準拠した表記によっており、大文字で始まるアトムは、変数を表わす。また本システムでは数式表現にもｐｒｏ－ｌｏｇのオペレータ記法を用いている。

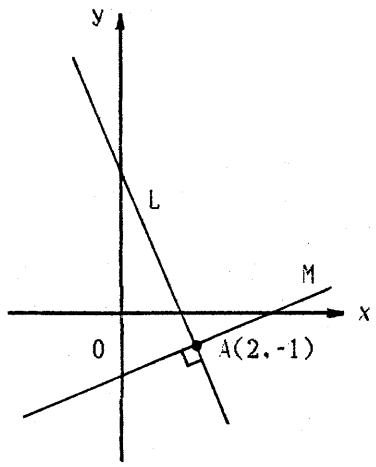
２　問題例とその入力形式

本システムでは、対象とする問題の範囲，システムへの入力の方法，及び動作形式を次のように定めた。

・２次元のｘｙ座標平面上の静的な図形に関する問題を扱う。
・扱う図形は平面上の点あるいは$f(x,y)=0$の形で表現できる物とする。
・平面上の状況記述には、平面上の各図形に関する記述と、個々の図形または複数の図形間に成り立つ幾何学的性質に関する記述の２種類がある。
・与えられた条件下での未定図形の代数的表現や未知変数の値をシステムに質問することにより、システムの問題解決が始まる。

一例として次のような問題が挙げられよう。

例：「２直線Ｌ，Ｍは、点Ａ(2,-1)において直交
　　する。Ｌの傾きが-3のとき、Ｍを求めよ。」



実際の入力は次のような述語形式で行なう。

dot(a,2,-1) ･･･････････････ Ａ(2,-1)は点です。
line(l) ･･･････････････････････ Ｌは直線です。
line(m) ･･･････････････････････ Ｍは直線です。
cross(a,l,m) ･･･････ ＬとＭはＡで交わります。
vertical(l,m) ･･･････････ ＬとＭは直交します。
gradient(-3,l) ･･･････････ Ｌの傾きは-3です。
ask(m) ･･･････････････････････ Ｍの式は？

本システムでは、図形に関する記述は原則として、
１．その図形を他の図形と区別するための図形名（例 d0，l，c1）
２．その図形の形状を示す型名（例　点，直線，円）
３．その図形の代数的表現（例　(2,3),x+3*y-15=0）
の３個の属性で表わされるが、代数的表現が未定の図形の場合、３は省略可能である。

幾何学的性質に関しては、本システムでは次の２種類を想定している。

relation：複数の図形間に成り立つ幾何学的位置関係
　　（例　交わる，平行，垂直）
function：一つあるいは複数の図形に対し一意的に定まるスカラ値
　　（例　［直線の］傾き，［２点間の］距離）

それぞれ入力は次のような形式をとる。関係名と関数名の識別はシステムが自動的に行なう。

relation：関係名（図形名１，図形名２，･･･，図形名ｎ）（ｎ≧２）
function：関数名（値，図形名１，･･･，図形名ｎ）　　（ｎ≧１）

システムへの質問は、ａｓｋ（図形名または変数名）という形式により行なう。システムは、問われたのが図形名ならばその図形の代数的表現を、変数名ならばその変数の値を答えようと試みる。答えが未知のとき、システムは与えられた条件をもとに解を探し始める。

３　入力処理

図形に関する入力は、図形名による情報検索を容易にするため、次のような２つの述語形式に変換される。

obj_type（型名，図形名）･･･････････ 図形とその図形の形状を連想させる。
obj_exp（図形名，代数的表現）･･･････ 図形とその代数的表現を連想させる。

図形に関する記述が入力されると、システム内部ではそれをこの２つに分割して登録する。またこのとき入力文中の代数的表現は、システム内の数式処理ルーチンによ

り、カッコを展開して同類項をまとめた形に整理される。その結果、例えば

dot(d) → obj_type(dot,d)

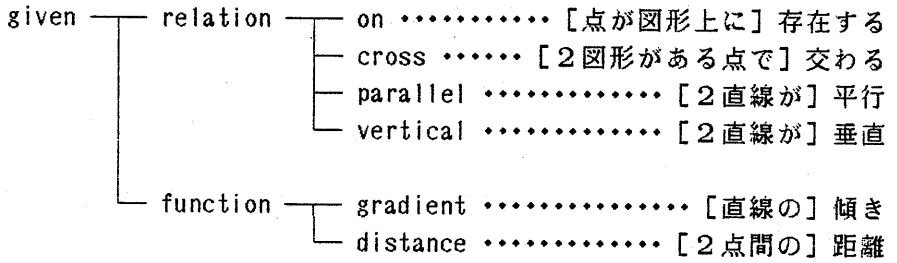
dot(d1,3,-4) → obj_type(dot,d1),obj_exp(d1,(3,-4))

line(l1,x+y=0) → obj_type(line,l1),obj_exp(l1,x+y)

circle(c1,x^2+(y-1)^2=1) → obj_type(circle,c1),obj_exp(c1,x^2+y^2-2*y)

のような変換を経て、各入力は情報として蓄えられる。

一方、relation，function等の幾何学的性質に関する入力に対し、本システムでは次のような階層的概念を備えている。



システムに実際に入力されるのは、上図における最下位の概念である関係名，関数名である。これを一つ上の概念であるrelation，functionと結びつけるために次のように変換する。

relation（関係名，［図形名１，図形名２，・・・，図形名ｎ］）

function（関数名，値，［図形名１，・・・，図形名ｎ］）

いずれの場合も各図形名は、［　］でくくられたリスト形式にまとめられて、

on(d1,l1) → relation(on,[d1,l1])

parallel(l2,l3) → relation(parallel,[l2,l3])

gradient(gr,l2) → function(gradient,gr,[l2])

などのようになる。同時にここで、入力記述に矛盾がないかどうかの検査が行なわれる。検査には次の２種類がある。

ａ．「点Ａと直線Ｌが平行である」のように、その幾何学的性質の定義にそぐわない入力の排除 ・・・・ 記述中の各図形の型と、“各性質がどのような型の図形に対して定義されうるか”というシステム内の知識とを照合させて調べる。

ｂ．「２点P(0,0)，Q(1,1)間の距離は２である」のように、数学的に正しくない入力の検出 ・・・・ 記述中の各図形（functionの場合はスカラ値も含む）が変数を含まない既知のものである場合、その真偽は他の記述とは無関係に直ちに決定可能であるので、この検査の対象となる。後述する方程式抽出の手続きをこの幾何学的性質に対して行ない、“０＝０”の自明な式が得られれば真，“０＝（非零定数）”が出てきたら偽と判断できる。

こうした処理を経て上位概念relation，functionと結びつけられた入力文は、“問題文中で与えられた条件”として統一的に扱うために、

given(relation(・・・・・・・)) , given(function(・・・・・・・))

という形に変換され、システム内に登録される。但し、上の検査ｂの対象となった入力は、その真偽が出力されるのみで、内部への登録は行なわれない。仮に真と判断さ

れたものであっても、それをシステム内に登録することは、システム内部の情報を冗長にするに過ぎないからである。

4　方程式抽出

システム内で実際に方程式を立てて解く場合、入力時に代数的表現の与えられなかった未知図形に対して、適当な表現を予め割り当てておく必要がある。このとき重要なのは、どのような代数的表現を用いればその図形を確実に表わせるかということである。例えば、直線を表わすのに y=a*x+bの形式を用いたのでは、ｙ軸に平行な直線を表わせない。本システムでは図形ごとの一般的な代数的表現を、次のように定義した。

・点：　ｘ，ｙ座標を示す独立な２個の変数を割り当てる。本システムではＬＩＳＰのgensymに相当する手続きにより、シンボルを生成して変数として割り当てることができる。

・円：　(x-p)^2+(y-q)^2=r^2が一般的な円の表現式であるが、これを展開・整理して、x^2+a*x+y^2+b*y+c=0 の形で表わしても支障はないので、展開した形の式を用いる。

・直線：上述のように y=a*x+bは一般式としては不完全であり、すべての直線を表わすには a*x+b*y=cの形を使わざるを得ないが、この式でａ，ｂ，ｃの３変数は独立でない。そこで変数の従属性を示すために,a^2+b^2=1という正規化条件を導入した。これにより任意の直線を同一の形式で表現できる。

最後の直線の場合、直線の表現式一つに対して一つの正規化条件を表わす式が生成されるので、これを「直線の表現式に付随する正規化条件式」と呼ぶことにする。この式は、幾何学的性質から抽出された他の方程式と同格に扱われ、解を求めるための代数的処理の対象となる。

方程式抽出規則は、各relation、functionごとに次のように定めた。

on ・・・・・・・・・ ある一点が（点でない）図形上に存在するということだから、その図形の表現式に点の座標を代入した等式が得られる。

cross ・・・・・・ ２つの図形がある点を共有するということである。これはonの連言によって表わすことができる。

parallel ・・・ ２直線の傾きが等しいと考えると、傾き∞の直線の扱いが難しいので、直線の方向ベクトルが線形従属であることを式で表現する。

vertical ・・・ 同様に２直線の方向ベクトルの内積が０であると考える。

gradient ・・・ 方向ベクトルの２成分の比が入力されたスカラ値に等しいという式を抽出する。

distance ・・・ ３平方の定理により、２点のｘ成分の差の２乗とｙ成分の差の２乗の和が、入力されたスカラ値の２乗に等しくなる。

幾何学的性質によっては、得られた解の適・不適の判定を要する場合がある。（例えば、 distance(a,d1,d2)のときａ≧０。）このような判定は、全方程式を解き終えてから最後に一括して行なうのでなく、部分的に変数の値が求まるごとに調べるようにしないと、正しい解に至るまでの効率が著しく低下する恐れがある。そこで、この

種の条件を任意の場所から参照可能にするための一手段としてｃｈｅｃｋという述語を導入した。これは次の形式をとる。

check（満たすべき条件）

条件部には、ｐｒｏｌｏｇの構文による実行可能な手続きを記述する。例えば、上のａ≧０の場合なら、

check(not (eval(a,V),numberp(V),V<0))

のような節を登録する。こう宣言しておけば、代数的処理の段階で、変数の値が求まるごとにこのｃｈｅｃｋで示された各手続きを実行することにより、不適解の検出が可能となる。

５　求解過程での学習

以上のように抽出された方程式を連立させて解けば、与えられた問題に対する答えが求められる。本システムは、文献３に掲載されている数式簡略化プログラムをもとにした数式処理ルーチンを備えている。これは２次式を高々１個含む連立方程式の取り扱いが可能で、直線や円に関するごく簡単な問題なら解くことができるが、ここでその詳細には立ち入らない。ここではそれと同時に本システムの行なう幾何学的知識の学習について説明しよう。具体例として、先に入力例のところに掲げた問題と同じものを用いる。まず、問題入力から方程式抽出に至るまでの経過を図１に示す。

（以下、@n(n≧1)はシステムが内部で新たに生成した変数である。）

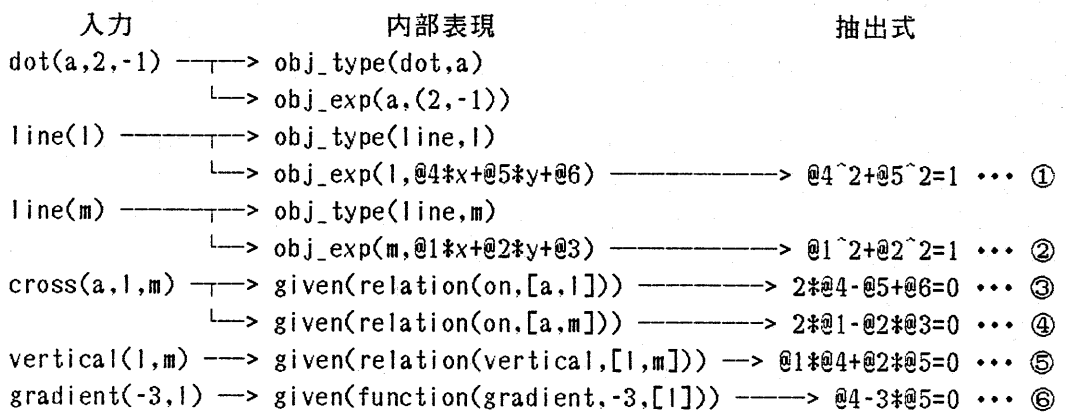


図１　入力から方程式抽出までの経過

Ｍの表現式を求めるには、①～⑥の連立方程式を解き、@1,@2,@3の値を決定すればよいのだが、現在の本システムの数式処理能力には限界があり、⑤のように変数同士の積を含む式がうまく扱えない。そこで本システムではこれを次のように解く。

ａ）この６式中で、他と独立に解ける部分集合がないかどうかを探索する。今の場合、①③⑥の３式には@4,@5,@6の３変数のみが含まれており、この３式を解いて@4～@6が求まる。

ｂ）その結果を⑤に代入する。

c）残りの３式より@1～@3が定まり、Mの表現式が求まる。

ところで、a)で①③⑥より@4,@5,@6すなわちLの表現式中の変数が求まるということは、幾何学的にはどういう意味を持つのだろうか。３式のうち、①はLの表現式に付随する正規化条件であるが、③はAがL上にあることより、⑥はLの傾きが-3であることより得られた式である。したがって、Aを通り傾きが-3であることよりLの表現式が求まることが分かり、一般化すると「傾きと通る一点が決まれば直線は一意に定まる」という幾何学的知識が得られる。システムはLが定まった時点で、これを次のような形式で内部に登録する。

```
rules(line,V) :- relation(on,[V1,V]),function(gradient,V2,[V]).
```

すなわち、ある未知図形が決定されるための幾何学的条件を、システムは問題を解きながら知識として獲得してゆく。これが本システムの行なう学習である。

このことは、c)に於いて②④⑤より@1,@2,@3が求まる場合についても同様にあてはまり、最終的にシステムは、「通る一点と垂直な他の一直線が与えられた直線は一意に定まる」という意味の

```
rules(line,V) :- relation(on,[V1,V]),relation(vertical,[V2,V]).
```

を学習する。

6　学習された知識の適用

求解過程で学習される知識は、未知の図形を決定するための幾何学的条件を示唆するものであるから、これを実際の問題解決に適用できれば、未知図形を求めるうえで有力な手掛かりとなるであろう。本システムでその考え方を導入してみたところ、多くの問題に対して求解速度が向上した。以下その仕組みを、先程と同じ問題を解かせた場合を例に用いて説明する。

ask(m)という質問に対し、Mが直線なのでシステムは直線を決める知識を呼びだす。これは、組み込み述語clauseを用いて、clause(rules(line,m),Body)という形式で行なう。変数Bodyの値と,givenに登録された幾何学的性質との照合により、その知識が適用可能かどうかが調べられる。システム内の知識が、

```
rules(line,V) :- relation(on,[V1,V]),function(gradient,V2,[V]). ···· ①
rules(line,V) :- relation(on,[V1,V]),relation(vertical,[V2,V]). ···· ②
```

の順で登録されているとき、知識の適用は次のようになる。

```
①  変数Bodyの値：relation(on,[V1,m]) ， function(gradient,V2,[m])
                          ⋮                          ×
   given内の情報：relation(on,[a,m])        （照合するものがない）
                                                          backtrack
②  変数Bodyの値：relation(on,[V1,m]) ， relation(vertical,[V2,m])
                          ⋮                          ⋮
   given内の情報：relation(on,[a,m])     relation(vertical,[l,m])
```

∴　点Aと直線Lが既知ならば、②の知識よりMが求まることが分かるが、Lが未知

なので、Ｌを決定するために知識を再帰的に呼び出す。

① 変数Bodyの値：relation(on,[V1,l]) , function(gradient,V2,[l])

given内の情報：relation(on,[a,l]) function(gradient,-3,[l])

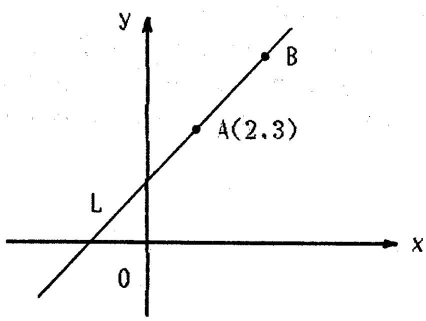
∴ relation(on,[a,l]) と function(gradient,-3,[l])より、Ｌの表現式が求まり、Ｌが定まれば同様にしてＭの表現式も求められる。

すなわち、ある未知図形を求めるにあたり、既存の知識と照らし合わせて必要な情報のみを取り出し、それを方程式化して解くことが、本システムでは可能である。知識を用いた解法は、知識を使わない代数的処理のみによるものに比べて、次の点で有利である。

1. 方程式中で独立に解ける部分集合を探す手間が省ける。
 …… 知識なしの解法は、抽出された式の中から部分的に解ける式集合を探し出す作業を伴うので、式の数が多いときには探索による求解効率の低下が著しい。知識を用いれば、一度に抽出される式の数が必要最小限になるので、そのような問題は起こらない。
2. 余分な情報を含む問題に対処できる。 …… 次に示すのが、その一例である。

例：「２点Ａ(2,3),Ｂを通り、傾きが３の直線Ｌの式を求めよ。」

dot(a,2,3) …… Ａ(2, 3)は点です。
dot(b) ……………… Ｂは点です。
line(l) …………… Ｌは直線です。
on(a,l) …… ＡはＬ上にあります。
on(b,l) …… ＢはＬ上にあります。
gradient(l,l) … Ｌの傾きは１です。
ask(l) ……………… Ｌの式は？



この問題ではＢの座標は定まらないが、傾き１でＡを通ることより、Ｌの表現式は求まる。当然ながらこのことが分かるためには、「傾きと通る一点が決まれば直線は一意に定まる」という知識が必要となる。そのような知識のない状況では、Ｂに関する余分な情報もが方程式化され、その結果システムは、方程式の数よりも変数の数が多くて解けないと判断してしまう。

3. 計画に基づいたより高度な問題解決ができる。
 …… 学習した知識は何重にも再帰的に適用でき、システムは「ある図形を求めるためには別のある図形を求めねばならない。」というような計画を立てながら、問題を解いてゆくことができる。この方法は理にかなったものであり、機械的な方法に比べて、人間がシステムの動作内容を理解することも容易である。工夫次第で、文献１にあるようなQuestion Answeringの機能を導入することも可能であろう。

本システムはSORD M685 K-prolog、及びNEC PC-9801F Prolog-KABA上にインプリメントされている。本文中で再三にわたって例に掲げた「直線Mの式を求める問題」を前者を用いて解かせた場合、知識なしのときには80秒程度を要するが、知識を伴う場合には60秒程度で解ける。より抽出式の数が多くなる問題の場合、知識を用いることにより、問題を解く速度は 1.5～2 倍程度速くなる。

7　あとがき

学習機能を持つ幾何学問題解答システムについて述べた。本システムは方程式抽出による代数的処理を問題解決の基本方針とするが、個々の代数的処理は幾何学的にもなんらかの意味を持つ場合がある。このことに着目し、それをシステムに知識として学習させ、以降の問題解決に適用させることにより、システムの行なう問題解決を高度なものにできることを、本論文では示した。現在のところ本システムの解くことのできる問題はごく限られており、代数的処理能力，学習機能の両面に亘る改良が今後の課題である。

なお、本研究は文部省科学研究費特定研究(1) 課題番号59118003によるものである。

参考文献

1．P.H.Winston：
Artificial Intelligence (Second Edition), Addison Wesley(1984)
2．中島秀之：
ＰＲＯＬＯＧ，産業図書(1983)
3．萩野達也，桜川貴司，柴山悦哉：
Prolog-KABA Reference manual，岩崎技研工業(1984)