34. 定理の自動証明法†

永田　守男††

1. はじめに

計算機による定理の自動証明のためのアルゴリズムについて，基本的な事項を解説する．この方面の予備知識をほとんど仮定しないで話をはじめ，こうした分野の他の解説や論文を読めるレベルにまで達することを意図した．

さて，「定理の自動証明」ということばからは，数学者の代りをするプログラムが連想されたり，数学の本に書いてある定理を入力すればこれらを証明できるシステムのことが示されているように思えるかもしれない．しかし，これまでの技術レベルでは，このようなところに至っているわけではない．

従来の定理の自動証明は，明確に定義された書式に則った定理が与えられて，一定の公理と推論規則を使い，自動的にこの定理を証明するという形をとっている．すなわち，数理論理学の「命題計算 (proposional calculus)」や「述語計算 (predicate calculus)」といわれるものをおこなうプログラムのことを，定理の自動証明システム (theorem prover) と呼ぶことが多い[1]．

本稿では，以上の経緯を踏まえて，この分野の研究についての歴史を簡単に振り返ったあと，命題計算のアルゴリズムとして代表的な Wang のアルゴリズム[2),3)]を紹介して，述語計算のアルゴリズムの基になっている導出原理 (resolution principle)[4)] を説明する．そのほか，いろいろな必要があって考案されたアルゴリズムに触れ，最後に，本特集号の他のアルゴリズムと比べたうえでの，本稿のアルゴリズムの特徴をまとめる．

2. 歴史と背景

数学の定理を計算機で自動的に証明してみたいというのは，計算機科学者たちの昔からの夢のひとつで，このための試みは，1950年代に，電子計算機が研究の道具として使えるようになるのとほとんど同時に開始された[5),6)]．日本でも，早い時期から，数学基礎論研究者によって，水準の高い研究がおこなわれていた[7)]．

その結果，1960年代の中頃までに，本稿で説明する主要なアルゴリズムが開発された．そして，1970年代にはいって，プログラムの正当性を自動的に検証するシステムや，述語論理に基礎を置いた論理型プログラミング[9)]，あるいは人工知能技術の実用化[10)]といった流れの中で，定理の自動証明が再認識されて現在に至っている．

なお，本稿の内容は，題材の関係で，数理論理学との関連が深いが，この方面の予備知識がほとんどなくても理解できるように「ふつうのことば」で述べることに努めた．したがって，厳密性を犠牲にした記述が出てきていることは否めない．また，本稿の長さとの関係もあって，論理学におけるいろいろな立場[11)~15)]についてまで言及するに至らなかった．

3. 命題計算のアルゴリズム

命題・命題論理・命題計算の三つを明確にし，命題計算のための Wang のアルゴリズムを説明する．このアルゴリズムは，ドイツの数学者 Gentzen が創りあげた LK[16)] と呼ばれる公理系を基にしているが，このようなシステムは，人間の自然な推論過程を反映しているとして，自然演繹システム (natural deduction system) といわれることもある．

3.1 命題と命題論理

「1と1を足し合わせたものは2だ．」とか「浅間山は富士山より高い．」といった，真 (*t*) と偽 (*f*) が明確に決められるものを命題 (proposition) という*．命題を示す記号として，ここでは大文字のアルファベット (A, B, C, …) を使う．

† Automatic Theorem Proving by Morio NAGATA (Department of Administration Engineering, Faculty of Science and Technology, Keio University).

†† 慶応義塾大学理工学部管理工学科

* この節の中では，命題の中に変数は含まれないものとする．述語論理では，変数を含む命題もある．

命題どうしを，かつ (and, ∧ で表わす)，または (or, ∨)，ならば (imply, ⊃) で結んだものと，命題の前に「～でない」を表わす '¬' (not) を付けたものも命題として考える．ここで，∧，∨, ⊃, ¬を論理記号と呼ぶ．こうして，いくつかの命題から新たに作られる命題の真理値 (truth value, t または f のこと) は，各論理記号ごとに定められた真理表 (truth table) によって決まる．これらの真理表は，ここでは省略するけれども，その内容は，日常の自然言語での解釈とニュアンスを異にする部分も若干ある[17]．しかし，これらは「正しい推論」を進めるために必要な約束ごとである．

このように，最小単位の命題から出発して，それらを論理記号で結んだものの性質について，真理表を基にして考えるのが命題論理であって，こうした性質を，形式的かつ機械的に調べるのが命題計算の役割である．

3.2 命題計算

命題論理のなかでの性質 (たとえば，「A⊃B と A が成立しているという前提のもとで結論として B が成立する」) を機械的に調べるために Gentzen が考案した方法を，Kleene の説明[18]を参考にしながら解説する．

まず，sequent という概念を導入しよう．$A_1, A_2, \cdots, A_m$ と $B_1, B_2, \cdots, B_n$ を命題としたとき，

$$A_1, A_2, \cdots, A_m \rightarrow B_1, B_2, \ldots, B_n$$

を sequent という．この矢印の左辺を前提，右辺を結論と呼ぶ．この sequent の意味としては，

$A_1 \wedge A_2 \wedge \cdots A_m$ ならば $B_1 \vee B_2 \vee \cdots \vee B_n$

である．したがって，

$$A \supset B, A \rightarrow B \qquad (1)$$

などと書くことができる．$m=0$ のときは，

$$t \rightarrow B_1, B_2, \cdots, B_n$$

と規定し，何の前提がなくても $B_1 \vee B_2 \vee \cdots \vee B_n$ が成立することを示す．また，$n=0$ のときは，

$$A_1, A_2, \cdots, A_m \rightarrow f$$

として，$A_1 \wedge \cdots \wedge A_m$ は成立しないことを示す．

一般的に，sequent を調べるとき，この中にある個個の命題の内容に立ち入らないとすれば，これらの命題全部について t と f のすべての場合の組み合わせについて考えることになる．こうした状況では，常に成立する「公理」として採用できる sequent とは，前提にある命題と同じ命題が結論にも出てくるものだけになる．厳密にいえば，任意の命題を A，命題の並び (命題が 0 個以上コンマで区切られて並んでいるもの) をギリシャ文字 (Γ, Δ など) で表わすと，

$$\Gamma_1, A, \Gamma_2 \rightarrow \Delta_1, A, \Delta_2 \qquad (A\,1)$$

が公理である．

また，sequent の定義と，各論理記号の真理表を考え合わせると，次のような推論規則 (rules of inference) を使うことは納得がいく．ここでは，B も任意の命題を示す．なお，s_1, s_2, s_3 という sequent を使って，推論規則は

$$\frac{s_1 \quad s_2}{s_3}$$

の形になっているが，これは「s_1 と s_2 が成立すれば s_3 が成立する」または「s_3 が成立するためには s_4 と s_2 が成立しなければならない」と読む．ただし，s_1 がない形のものもある．左端の括弧で示してあるのは，各規則の名前である．

$$(\wedge\text{-左}) \quad \frac{\Gamma_1, A, B, \Gamma_2 \rightarrow \Delta}{\Gamma_1, A \wedge B, \Gamma_2 \rightarrow \Delta}$$

$$(\wedge\text{-右}) \quad \frac{\Gamma \rightarrow \Delta_1, A, \Delta_2 \quad \Gamma \rightarrow \Delta_1, B, \Delta_2}{\Gamma \rightarrow \Delta_1, A \wedge B, \Delta_2}$$

$$(\vee\text{-左}) \quad \frac{\Gamma_1, A, \Gamma_2 \rightarrow \Delta \quad \Gamma_1, B, \Gamma_2 \rightarrow \Delta}{\Gamma_1, A \vee B, \Gamma_2 \rightarrow \Delta}$$

$$(\vee\text{-右}) \quad \frac{\Gamma \rightarrow \Delta_1, A, B, \Delta_2}{\Gamma \rightarrow \Delta_1, A \vee B, \Delta_2}$$

$$(\neg\text{-左}) \quad \frac{\Gamma_1, \Gamma_2 \rightarrow A, \Delta}{\Gamma_1, \neg A, \Gamma_2 \rightarrow \Delta}$$

$$(\neg\text{-右}) \quad \frac{A, \Gamma \rightarrow \Delta_1, \Delta_2}{\Gamma \rightarrow \Delta_1, \neg A, \Delta_2}$$

'⊃' についての推論規則は省略するが，'A⊃B' を '¬A∨B' と解釈してもよい．

このような公理と推論規則を利用して，与えられた sequent が成立するか否かを機械的に調べる Wang のアルゴリズムを次に示す．

3.3 Wang のアルゴリズム

Wang のアルゴリズムは，少し簡単にした形で述べると次のようになる．「sequent が与えられたら，これが公理になっているかどうか調べる．公理でなければ，推論規則を利用して，一つまたは二つの sequent に変形し，ここに述べている手順を変形後の sequent に対して適用する．公理であれば，これが本来初めに与えられた sequent のときは証明完了となるが，そうでなければ，この sequent が証明されたものとして一つ上のレベルへ戻り，証明された sequent と兄弟の関係の sequent があるときには，これに対して改めてここに述べてある手順を適用する．」これを，

再帰的に繰り返すのである．

(1)を少し変形した

$\neg A \vee B, A \rightarrow B$

に対して，このアルゴリズムを適用すると，次のような木（証明図という）ができる．

$$\dfrac{\dfrac{A \rightarrow A, B}{\neg A, A \rightarrow B} \qquad B, A \rightarrow B}{\neg A \vee B, A \rightarrow B}$$

図-1　$A \vee B, A \rightarrow B$ の証明図

証明すべき sequent を根として，すべての葉が公理となるこのような証明図ができれば，この sequent は証明可能であるという．

4. 述語計算のアルゴリズム

命題の中にある構造をもう少し詳しく反映するものがあれば，世の中の現象の定式化が，より便利になる．そのようなものが述語である．ここでは，述語論理のための導出原理（もちろん命題論理にも適用できる）を説明する．これについては詳細な解説[19)~21)]があるので，それらへの橋渡しを試る．なお，Gentzen のやり方も述語論理に拡張できるが，計算機の世界では，述語論理のときは導出原理が多く利用されている．

4.1 述語と述語論理

昔から論理学の教科書でよく使われる例を利用して説明する．「人間とは，すべて誤りを犯すものだ」を，ひとつの命題として，たとえば A などと表わしてもよいが，「すべての」を示す記号 '$\forall$'，変数としての x，「x は人間である」と「x は誤りを犯すものだ」をそれぞれ表わす $P(x)$ と $Q(x)$ を使って

$$\forall x(p(x) \supset Q(x)) \qquad (2)$$

とも書ける．x の性質や状態を表わす P や Q のことを「述語」といい，$\forall$ を全称記号，「存在する」を示す $\exists$' は存在記号と呼ぶ．$\forall$ と $\exists$ は論理記号である．また，$\forall x$ を全称作用素 (universal quantifier)，$\exists x$ を存在作用素 (existential quantifier) といって，これらをまとめて限定作用素 (quantifier)* と呼ぶ．

述語論理とは，命題論理に対して，述語と関数，限定作用素を導入し，論理記号として $\forall$ と $\exists$ を追加したものである．したがって，(2)も命題になっている．

4.2 導出原理に基づくアルゴリズム

(2)と「ソクラテスは人間である」ことから，「ソクラテスは誤りを犯すものだ」を導く例を使って，アルゴリズムを説明する．

この例を，あらためて整理すれば，

$$\forall x(\neg P(x) \vee Q(x)) \qquad (2)'$$

$$P(s) \qquad (3)$$

という前提となる事実（公理とここでは呼ぶ）から，

$$Q(s) \qquad (4)$$

を導く問題となる．ここで，s はソクラテスという個人を表わしている．

本節のアルゴリズムの大方針は，これから証明すべきものを否定したものと公理の集合から矛盾を導いてくることである．たとえば，この例では，(4)の否定

$$\neg Q(s) \qquad (5)$$

と (2)', (3) から矛盾を導く．

さて，一般に，論理記号が複雑に入り組んだ論理式を，一つまたはそれ以上の選言標準形というものに変換する手続きがある[21)]．選言標準形とは，最も基本となる命題またはそれに $\neg$ を付けたものを $\vee$ で結びつけたものである．また，限定作用素が，たとえば

$$\forall x \exists y(P(x, y))$$

のようになっていても

$$\forall x(P(x, f(x)))$$

のように，スコーレム関数と呼ばれる f を使って，y を $f(x)$ で置き換えることで全称作用素だけの式に変換できる．こうした形になれば全称作用素を取り除いてもよい．選言標準形でかつ全称作用素だけの論理式から全称作用素を除いたものを節 (clause) と呼ぶ．(2)' は，

$$\neg P(x) \vee Q(x) \qquad (2)''$$

という節になって，これと (3) と (5) の節の集合から，次の i) から v) に示す操作によって，矛盾を示す空節 (NIL) を導き出すのが，ここで示すアルゴリズムである．

i) A と $\neg A \vee B$ から B

ii) $A \vee B$ と $\neg A \vee B$ から B

iii) $A \vee B$ と $\neg A \vee B$ から $B \vee \neg B$ と $A \vee \neg A$

iv) $\neg A$ と A から NIL

v) $\neg A \vee B$ と $\neg B \vee C$ から $\neg A \vee C$

ただし，A, B, C は任意の命題を示す．

例について，この規則を使ったアルゴリズムを具体的に説明する．まず(2)" と(5)の両方の節に対して，x に s を代入して i) が適用できて，新たに $\neg P(s)$ という節が導かれる．次に，この節と (3) に対して iv) を適用すれば，空節が導ける．この過程を図-2 に示すが，このような図を refutation tree といい，前

* 限定作用素の中に関数や述語を含まないものを1階の述語論理といい，本稿では，こうした論理だけを扱う．

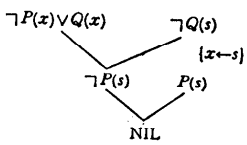

図-2 (2)″, (3), (5) についての refutation tree

に示した証明図に対応する.

導出原理とは，もともと矛盾をはらんだ節の集合があるときに，i)からv)に示した規則だけを適用してNILを導くことが可能だとする主張である.

4.3 述語計算と命題計算の完全性

定理証明のためのアルゴリズムは，本特集号の他のアルゴリズムに比べて，メタなもの，すなわち，より一般的なアルゴリズムだと考えることができる．つまり，証明すべき定理を具体的に入れかえれば，それぞれに応じて，いろいろな計算量のアルゴリズムが得られることになる．したがって，他のアルゴリズムのような計算量を，一般的な定理証明のアルゴリズムについて論ずることには，あまり意味がない.

ただし，「ある式がどんな解釈をしても正しい（妥当）ならば，その式がそのアルゴリズムで証明可能である」という完全性は，定理証明のアルゴリズムで重要である．計算機による定理の自動証明の立場から考えると，完全性が保証されているアルゴリズムに則したプログラムを作れば，正しい式については，このプログラムによって有限回のステップで証明が完了する可能性があるということになる.

本稿で示した命題計算と述語計算のアルゴリズムは，ある範囲内で共に完全であることが証明されている.

4.4 証明戦略

導出原理に基づく自動証明アルゴリズムでは，どの節とどの節で命題を消し合うかによって，与えられた節の集合からNILを導き出すまでの道のりの長さが異なる．したがって，より効率的な証明のための戦略が重要で，これまでに多くの証明戦略が提案されている．「完全性」を念頭において，そのうちの代表的な三つについて解説する.

第1は，set-of-support strategy. これは，証明すべきものの否定として与えられた節(複数のこともある)に注目するもので，消し合う節の一方に必ずこのような節を選ぶか，この節と他の節を消し合って得られた節を使う．公理として与えられた節の集合だけからNILが導けるはずがないので，この戦略は当然ともいえるものだが，プログラムで消し合う節を選び出す選択の幅を狭めるので有効なものである．この戦略を使っても完全性は保証されている.

第2は，節の中で∨によって結ばれた命題の個数が1個のもの（このときは∨もない）を最優先して選択する unit preference strategy. これも，最終的にNILを導き出すためには，途中で大きな節を作らないようにするという自然なアイデアで，前の戦略と併用でき，完全性が保証される．この戦略を使った定理証明のためのLISPプログラムが載っている教科書[20]もある.

第3は，linear-input-form strategy. これは，最初に与えられた節の集合中の節を必ず選択するもので，完全性は保証されないが，プログラムを作るのが簡単で効率的な戦略である.

この他にも，無数の戦略が導出原理による定理証明アルゴリズムのために提案された.

5. その他の話題

5.1 数学的帰納法の利用

定理の証明アルゴリズムに数学的帰納法を採り入れて成功したものにBoyerとMooreのLISPプログラム検証システムがある.

数学的帰納法は，もともと自然数についての述語Pに対し，$\forall x P(x)$が成立することを証明する手段として考えられたものである[22]．これを拡張し，自然数の上の関係をLISPにおけるリスト構造の間の関係に置き換えると，アトムを含むリストlについての述語Qに対し，$\forall l Q(l)$が成立することを証明するためには，任意のアトムをa，アトム以外のリストをmとすれば，

(I) $Q(a)$が成り立つ

(II) $Q(\mathrm{cdr}(m))$が成り立つことを仮定すれば $Q(\mathrm{cons}(a, \mathrm{cdr}(m)))$が成り立つ

ことを示せばよいことになる*.

この論法を採用したうえで，いくつかの公理（たとえば$m=\mathrm{cons}(\mathrm{car}(m), \mathrm{cdr}(m))$）を出発点として，

$$\mathrm{reverse}(\mathrm{reverse}(m))=m$$

のような定理（ここでreverseはリストの要素の順番を逆にしたリストを作る関数）を証明しているのがBoyerとMooreのシステムである．彼らの著書[24]には，このシステムで証明できた定理がたくさん載っている.

* このアイデアは Burstall の構造帰納法[23]を基にしている.

Boyer と Moore のものとは少し観点が異なるが，数学的帰納法を利用して同様の定理を証明したシステム[25]もある．

5.2 応　用

定理の証明プログラムをいろいろなところに応用することが考えられているので，それらの代表例をいくつか紹介する．

まず，与えられた定理を証明するプロセスを利用した質問応答システム[26]について．

このシステムのアルゴリズムを，例を挙げて説明する．これは，導出原理による方法の変形である．

4.1 と **4.2** で示した例を少し変えて，「すべての人間は誤りを犯すものだ」と「ソクラテスは人間である」という事実から，「誰が誤りを犯すか？」と聞かれたときの答を引き出すことを考える．

このとき，(2)′と(3)はそのまま使い，(4)を，

$$\exists y Q(y) \qquad (4)'$$

とすれば，(4)′の否定は

$$\forall y(\neg Q(y))$$

となるから，refutation tree は**図-3**に示すとおりである．ここで，yには最終的に s (すなわちソクラテス) が代入されているから，答はソクラテスになる．

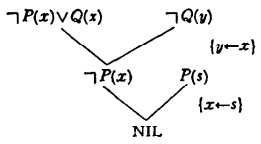

図-3 refutation tree から答を引き出す過程

この方法を使うと，簡単なプログラムの自動合成などもできる．

もうひとつの応用は，論理型プログラミングである．最近いろいろと話題になっている prolog の処理系[27]の中心は，導出原理による方法を使った，特殊な定理証明プログラムということもできる．

6. おわりに

本稿で紹介したアルゴリズムの特徴は，述語論理では「完全性」を保証しているだけであって，その「計算量」は，極めて特殊な場合についてのみ検討されている[28]ことであろう．また，基本的な研究が 1960 年代になされている古い問題であるにもかかわらず，論理型あるいは関数型プログラムという現代的な観点[29]からも一層の進歩が期待されている分野でもある．

なお，本稿と内容は重複するが，要領のよい解説がある[30]ので，一読をおすすめしたい．

参考文献

1) 中西正和，永田守男：コンピュータによる定理の証明，数理科学，No. 159, pp. 54-60 (1976).
2) Wang, H.: Toward mechanical mathematics, IBM Journ. of R & D, Vol. 4, No. 1, pp. 2-22 (1960).
3) McCarthy, J. et al.: LISP 1.5 programmer's manual, The MIT Press (1962).
4) Robinson, J. A.: A machine-oriented logic based on the resolution principle, J. ACM, Vol. 12, No. 1, pp. 23-41 (1965).
5) Newell, A. and Simon, H. A.: The logic theory machine, IRE Trans. Information Theory, Vol. 2, No. 3, pp. 61-79 (1956).
6) Gelernter, H.: Realization of a geometry theorem-proving machine, Proc. ICIP, pp. 273-282 (1959).
7) 西村敏男：定理の証明，情報処理学会誌，Vol. 24, No. 3, pp. 267-276 (1983).
8) Boyer, R. S. and Moore, J S.: Proving theorems about LISP functions, J. ACM, Vol. 22, No. 1. pp. 129-144 (1975).
9) Kowalski, R. A.: Predicate logic as programming language, Proc. of IFIP 74, North-Holland pp. 569-574 (1974).
10) Stefik, M. et al.: The organization of expert systems A prescriptive tutorial, Xerox PARC (1982).
11) 沢田允茂：現代論理学入門，岩波書店 (1962).
12) 前原昭二：数理論理学序説，共立出版 (1966).
13) 吉田夏彦：論理と哲学の世界，新潮社 (1977).
14) 竹内外史：層・圏・トポス，日本評論社 (1977).
15) 松本和夫：数理論理学，共立出版 (1970).
16) Gentzen, G.; Unterschungen ueber das logische Schliessen, Mathematishe Zeitshrift, Vol. 39, pp. 176-210, pp. 405-431 (1934, 1935).
17) 福原満州雄ほか：数学と日本語，共立出版(1981).
18) Kleene, S. C.: Introduction to metamathematics, North-Holland (1952).
19) 佐藤泰介：導出原理による定理証明，情報処理，Vol. 22, No. 11, pp. 1024-1036 (1981).
20) Chang, C. and Lee, R. C.: Symbolic logic and mechanical theorem proving, Academic Press (1973).
21) Nilsson, N. J.: Principles of artificial intelligence, Tioga Publishing Co. (1980).
22) 廣瀬　健：数学的帰納法，教育出版 (1975).
23) Burstall, R. M.: Proving properties of programs by structural induction, Computer Jour., Vol. 12, pp. 41-48 (1969).
24) Boyer, R. S. and Moore, J S.: A computa-

tional logic, Academic Press (1979).
25) Nishimura, T., Nakanishi, M., Nagata, M. and Iwamaru, Y.: Gentzen-type formal system representing properties of functions and its implementation, Proc. of 4th IJCAI, pp. 57-64 (1975).
26) Green, C.: Theorem-proving by resolution as a basis for question-answering systems, Machine Intelligence 4, Edinburgh University Press, pp. 183-205 (1969).
27) 中島秀之: Prolog とその処理系, 情報処理, Vol. 23, No. 11, pp. 1049-1062 (1982).
28) 山崎 進, 岸 知二, 石橋稔彦, 堂下修司: いくつかのクラスにおける Horn 節集合の充足不能性問題の計算量, 電子通信学会論文誌 D, J 65, No. 6, pp. 711-718 (1982).
29) 田村浩一郎, 大谷木重夫: 論理と関数, 情報処理, Vol. 23, No. 11, pp. 1034-1048 (1982).
30) Cohen, P. R. and Feigenbaum, E. A. (Eds.): The Handbook of Artificial Intelligence, Vol. III, William Kaufman, Inc. (1982).

(昭和57年12月10日受付)