プログラムの理論*

伊藤貴康**

1. まえがき

コンピュータ・サイエンスにおいて，直面する最も重要な課題の一つは，与えられた問題に対するプログラムを書いたときに，そのプログラムが望むような計算結果を，与えてくれるか否かを知ることである．このためには，プログラムにより表現される計算過程を記述，解析するに適した体系を構成し，その体系の中で，プログラムに関する諸種の性質を証明する必要がある．この論文は，このような立場からのプログラム理論についてのいくつかの試みにつき概説する．

計算機および計算の理論に関しては，オートマトン理論と数理言語の理論があり，いくつかの基本的概念および性質も明らかにされ多数の研究がなされている．また，アルゴリズムの研究の基礎理論としての回帰関数の理論 (Recursive Function Theory) の分野では，unsolvability についての重要な結果が得られている．しかし，これらは実際のプログラムの持つ数学的性質を理解する上で十分役立っているとはいえない．

プログラムの数学的性質を究明する研究としての，プログラム理論の研究は，理論面だけからでなく，実際面からの重要性にもかかわらず，最近に至るまで，顕著な進展を見せていなかった．これはプログラム理論の基本的概念を抽出することの困難さに起因すると考えられるが，最近 Yanov の結果を拡張するような立場から，いくつかの研究成果が得られ，プログラム理論は急速にその理論としての体裁を整えようとしている．

プログラム理論においては，プログラムの抽象化したモデルとして，プログラム図式というものを考え，それに基づいて，つぎのようなプログラムの基本的性質を研究する．

（1） プログラムの同値性 (Equivalence)
（2） プログラムの正当性 (Correctness)
（3） プログラムの停止性 (Termination)
（4） プログラムの変換と翻訳 (Transformation and Translation)
（5） プログラムの簡単化・最適化 (Simplification and Optimization)
（6） プログラムの複雑性 (Complexity)

これらを証明する公理体系，証明のアルゴリズムやこれらの性質に関する決定問題が研究の中心テーマとなっている．その成果は計算プログラムの自動作成，プログラミング言語のセマンティックスの形式化，計算過程に重点を置いた立場からの計算機の表現などへの応用を通して，一般的な計算の理論 (Mathematical theory of computation) の一つの基礎を与えるものと考えられる．

第2節において，プログラムの単純なモデルの一つとしてグラフ図式を与え，それを正規プログラム図式により代数的に取り扱うとともに，その完全な公理系につき述べる．これは Yanov の結果をオートマトン理論の立場から形式化し直すとともに拡張したものである．また，正規プログラム図式の自然な拡張として，CF 形プログラム図式についても述べ，プログラム理論と言語理論の一つの関係につき述べる．第3節ではグラフ図式の演算子が Assignment Statement 形になっているプログラム図式についての同値問題の決定不能性に対する Luckham-Park-Paterson の結果を詳述する．第4節では述語論理の形式化を用いて，プログラムの性質を証明するアルゴリズムの詳細と正当性，停止性，完全同値性に対する基本定理を与えた．第5節においては，プログラム理論に関する研究の小史を述べながら，他の研究についてもその側面を概説した．

2. グラフ図式と正規プログラム図式

プログラムの論理的構造を記述するのに，直観的に最も自然な仕方はフロー・チャートによる方法であろう．フロー・チャートをグラフ理論的な立場から数学的対象として定義したものはグラフ図式 (Graph schema) といわれる．プログラム理論の基礎になって

* Mathematical theory of programs, by Takayasu ITO (Central Research Laboratories, Mitsubishi Electric Co. Ltd.)

** 三菱電機株式会社・中央研究所

いる Yanov[53] の結果は，本質的にグラフ図式の諸性質を明らかにしたものであることが，その後明らかにされ，また，グラフ図式とオートマトン理論の関係も明らかとなっている．この節では，グラフ図式の定義を述べ，次いで，正規プログラム図式からグラフ図式の性質が証明できることを示す．正規プログラム図式はオートマトン理論における Kleene の正規表現[42]に対応する．言語理論において正規表現の拡張として，CF 形言語 (Context-free language) が考えられるように，CF 形プログラム図式というものが考えられ，この点についても本節で言及する．この節は主として Ito[19] に基づかれている．

2.1 グラフ図式

グラフ図式は

(1) 演算子 (operator) $F=\{e, f_1, \cdots, f_r\}$

(2) 述語 (predicate) $P=\{T, F, \pi_1, \cdots, \pi_s\}$

(3) 矢印 (arrow)，節点 (node)

(4) 演算子ボックス，述語ボックス

から，つぎのような規則により形成されるものである．

(1) 演算子ボックスには演算子，述語ボックスには述語が書き込まれる．

(2) 各節点には入力矢印と出力矢印があり，それぞれの個数を入力数，出力数という．

(3) 始点と呼ばれる入力数が 0 の節点が一つ存在する．

(4) 終点と呼ばれる出力数が 0 の節点が一つ存在する．

(5) 演算子ボックスは一つの入力矢印と一つの出力矢印を有する．

(6) 述語ボックスは正方向矢印と負方向矢印の二つの出力矢印を有する．

(7) 各節点は任意個数の入力矢印を持ちうる．

(8) 矢印は節点間，節点と演算子および述語ボックス相互間を結ぶ．

このように定義されたグラフ図式は，グラフ図式に現われる記号に意味が与えられたときに初めてプログラムとしての意味を持つことになる．演算子ボックスの中の演算子，述語ボックスの中の述語は，interpretation $I=<\hat{D}, \hat{F}, \hat{P}>$ の下に，矢印の方向に従い演算されていく．ここに

(1) $\hat{D}$ は interpretation の領域

(2) $\hat{F}$ は $\hat{D}$ 上での F に対応する関数で，とくに $\xi\epsilon\hat{D}$ に対して $e(\xi)=\xi$

(3) $\hat{P}$ は $\hat{D}$ 上での $\hat{P}$ に対応する述語で，とくに $\xi\epsilon\hat{D}$ に対して $T(\xi)=$true, $F(\xi)=$false

であるとする．

このようなグラフ図式は二次元の幾何学的図形として表現できるため，直観的にはとらえやすいが，そのままの形で取り扱うのは数学的理論を展開するときには，不便な場合が多い．このため次項では，正規プログラム図式というものを導入し，代数的に取り扱う方法につき述べる．

2.2 正規プログラム図式とグラフ図式

前項で定義したようなグラフ図式が，オートマトン理論における遷移図と同様な形で表現されることが，Karp[25] により示された．遷移図との違いは，述語という特殊な性質を有する事象が図式中に現われることで，このような図式を Karp 図式と呼ぶ．このように考えると，オートマトン理論において任意の遷移図に対し，正規表現を与えることができると同様に，任意の Karp 図式に対し，述語表現を含む正規表現を与えることができる．このようなプログラム族を正規プログラム図式ということにする．

任意のグラフ図式は，前節で述べた条件の下に，**図 2.1** に与えた 5 種類のグラフ図式を用いて表現できることが，グラフ図式の定義より容易に知られる．このような 5 種類の基本グラフ図式に対応する Karp 図式，正規プログラム図式，ALGOL 形プログラムは図 2.1 のように与えられる．

これより任意のグラフ図式は，必ず正規プログラム図式で表現でき，したがって，グラフ図式で表現されうるプログラム族は，正規プログラム図式で表現してもよいことがわかる．

上述の議論においては，正規プログラム図式の意味は，オートマトン理論における正規表現の意味を．仮定して話を進めてきたが，正確には下記のような interpretation を与えることにより，上述の状況が保証されることがわかる．まず，正規プログラム図式の形式的な定義を与えることから始める．

定義 2.1 まず述語表現をつぎのように定義する．

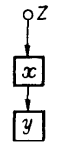
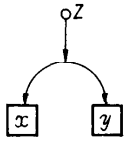
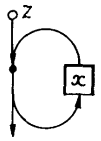
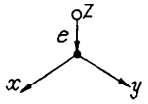
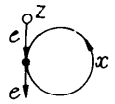
(a) T, F, π_i は述語表現である．

(b) もし p, q が述語表現であれば $(p\vee q)$, $p\cdot q$, $\bar{p}$, p^* も述語表現である．

(c) 上の規則 (a), (b) で定義されるものだけが述語表現である．

定義 2.2 正規プログラム図式は，つぎのように定義される．

(a) $\phi, e, f_i (i=1, 2, \cdots, r)$ は正規プログラム図式

	G_1	G_2	G_3
グラフ図式			
Karp 図式			
正規プログラム図式	$Z \cong x.y$	$Z \cong x \vee y$	$Z \cong x^*$
ALGOL形プログラム	do x ; go to L_1 ; L_1 : do y ;	go to L_1 ; L_1 : do x ; go to L_2 ; L_1 : do y ; go to L_3 ;	go to L_1 ; L_1 : do x ; go to L_1 ; L_1 : go to L_2 ;

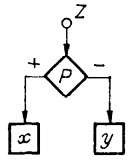
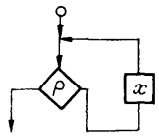
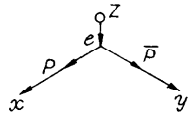
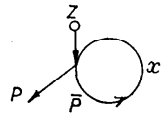
	G_4	
グラフ図式		
Karp図式		
正規プログラム図式	$Z \cong Px \vee \bar{P}y$	$Z \cong (\bar{P}x)^*P$
ALGOL形プログラム	if P then go to L_1 ; else go to L_2 ; L_1 : do x ; go to L_3 ; L_2 : do y ; go to L_4 ;	L_0 : if P then go to L_2 else go to L_1 ; L_1 ; do x ; go to L_0 ; L_2 ; go to L_3 ;

図 2.1 グラフ図式と正規プログラム図式の関係

である.

(b) p が述語表現であれば，p は正規プログラム図式である.

(c) もし x と y が正規プログラム図式であれば，$(x\vee y)$, $(x\cdot y)$, x^* も正規プログラム図式である.

(d) 上の規則(a),(b),(c)で定義されるものだけが，正規プログラム図式である.

このように定義された正規プログラム図式は，基本記号に interpretation が与えられたときに，初めて，実際のプログラムとしての意味を持つことになる. interpretation はグラフ図式に対する場合と同様に，$I=<\hat{D},\hat{F},\hat{P}>$ で与えられる. $\hat{D}$ として，たとえば，実数の集合，自然数の集合，あるアルファベットでのあらゆる有限系列の集合，あるいは計算機の状態集合などをとってもよい. $\hat{D}$ が空集合でなく関数 $\hat{F}$ と述語 $\hat{P}$ が具備されているとき，領域 $\hat{D}$ はアルゴリズム構造を構成するという. アルゴリズム構造は計算構造とも呼ばれ，このとき interpretation は計算機を定義すると考えてよい.

ある interprctation I の下でのプログラム図式 x の値は，プログラム事象と呼ばれ $s.[x](\xi)$ としるされ，つぎのように定義される. $\xi\in\hat{D}$ に対して

(1) $s.[T](\xi)=\{\xi\}$

(2) $s.[F](\xi)=\{\ \}$

(3) $s.[\pi_i](\xi)=if\ \pi_i(\xi)$ then $\{\xi\}$ else $\{\ \}$

(4) $s.[\bar{p}](\xi)=\{\xi\}-s.[p](\xi)$ (p は述語表現)

(5) $s.[\phi](\xi)=\{\ \}$

(6) $s.[e](\xi)=\{\xi\}$

(7) $s.[f_i](\xi)=\{f_i(\xi)\}$

(8) $s.[x\vee y](\xi)=s.[x](\xi)\cup s.[y](\xi)$

(9) $s.[x\cdot y](\xi)=s.[y](s.[x](\xi))$

(10) $s.[x^*](\xi)$
$=s.[e](\xi)\cup s.[x](\xi)\cup s.[xx](\xi)\cup\cdots$

ここに $\cup$ は和集合，"$-$" は差集合の演算記号，$\{\ \}$ は空集合，$\{a,b,\cdots,c\}$ は $a,b,\cdots,c$ よりなる集合である.

2.3 プログラムの同値性

これまでに導入してプログラム図式が与えられた計算構造の上で同じ計算をするか否かを知ることは，プログラムの諸性質が後述するように，同値性の概念を用いて記述できるゆえ，プログラム理論の基礎となっている.

定義 2.3 あらゆる $\xi(\in\hat{D})$ に対し，$s.[x](\xi)=s.[y](\xi)$ なら二つのプログラム図式 x と y は，interpretation $I=<\hat{D},\hat{F},\hat{P}>$ の下で，第1種同値であるという.

定義 2.4 あらゆる $\xi(\in\hat{D})$ に対し，$s.[x](\xi)\neq\{\ \}$, $s.[y](\xi)\neq\{\ \}$ で $s.[x](\xi)=s.[y](\xi)$ なら二つのプログラム図式 x,y は，interpretation I の下で，第2種同値であるという.

定義 2.5 あらゆる $\xi(\in\hat{D})$ に対し，$s.[x](\xi)=s.[y](\xi)$ であるか，$s.[x](\xi)$, $s.[y](\xi)$ のどちらかが空集合のとき，二つのプログラム図式 x と y は，I の下で第3種同値であるという.

定義 2.6 x と y があらゆる interpretation の下で第1種同値のとき，x と y は強同値であるといい，$x\cong y$ としるす. 強同値性のことを単に同値という.

定義 2.7 x と y があらゆる interpretation の下で，第2種同値であるとき，x と y は完全同値であるといい，$x\equiv y$ としるす.

定義 2.8 x と y があらゆる interpretation の下で，第3種同値のとき，x と y は弱同値であるといい，$x\doteqdot y$ としるす.

定義 2.9 $x\cong y$, $x\equiv y$, $x\doteqdot y$ の形の式を，wff (well-formed formula) あるいは方程式であるといい，x と y が強同値のとき wff $x\cong y$ は普遍妥当であるといい，$x\equiv y$, $x\doteqdot y$ はそれぞれ x と y が完全同値，弱同値のときに普遍妥当であるという.

しかし，ここで導入したプログラムの同値性概念のうち，数学における同値関係で要求される反射律，対称律，推移律を満たすのは，強同値性のみで，他の同値性の概念は同値関係にはならない. たとえば，完全同値性は，$x\cong\phi$ のときに $x\equiv x$ とならないゆえ，反射律を満足しない. また，弱同値性が推移律を満足しないのは明白である. なお，$x\equiv y$ は $(x\cong y)\wedge(x\ncong\phi)\wedge(y\ncong\phi)$ と表わせ，$x\doteqdot y$ は $(x\cong y)\vee(x\cong\phi)\vee(y\cong\phi)$ と表わせる (ここに $\wedge,\vee$ は論理積，論理和である). このような点から強同値性は，数学的には基本的なプログラムに関する同値関係となることがわかる. 以下，本論文では，とくにことわらないかぎり，同値性とは強同値性をさすことにする. 第 2.2 項の議論より，二つのグラフ図式の同値性は，それらに対応する二つの正規プログラム図式の同値性になることがわかる. 次章では正規プログラム図式の同値性を証明する完全な公理系を与え，正規プログラム図式の同値問題が決定可能なこともその結果として与える. これよりグラフ図式の同値問題もグラフ図式と正規プログラム

図式の上述の関係により解けることがわかる.

2.4 正規プログラム図式の完全な公理系

この項では，正規プログラム図式の同値性に対する完全な公理系を与える．ここでいう完全性は，公理系の中で誘導可能なwffの集合が同じ計算を意味する正規プログラム図式のwffの集合と一致することを意味している．正規プログラム図式の公理系を T_r と記すことにする．

(1) つぎの wff は T_r の公理といわれる．

$Ax.\ 1$ $x\vee(y\vee z)\cong(x\vee y)\vee z$

$Ax.\ 2$ $x(yz)\cong(xy)z$

$Ax.\ 3$ $x\vee y\cong y\vee x$

$Ax.\ 4$ $x(y\vee z)\cong xy\vee xz$

$Ax.\ 5$ $(x\vee y)z\cong xz\vee yz$

$Ax.\ 6$ $x\vee x\cong x$

$Ax.\ 7$ $ex\cong x$

$Ax.\ 8$ $\phi x\cong\phi$

$Ax.\ 9$ $x\vee\phi\simeq x$

$Ax.\ 10$ $x^*\cong e\vee xx^*$

$Ax.\ 11$ $x^*\cong(e\vee x)^*$

$Ax.\ 12$ $F\cong\phi$

$Ax.\ 13$ $T\cong e$

$Ax.\ 14$ $\bar{p}p\cong F$

$Ax.\ 15$ $p\vee\bar{p}\cong T$

$Ax.\ 16$ $pq\cong qp$

$Ax.\ 17$ $p\vee qr\cong(p\vee q)(p\vee r)$

ここに x, y, z は任意の正規プログラム図式で，p, q, r は任意の述語表現である．

(2) T_r における推論の規則は

$R1$: $\dfrac{x\cong y}{\alpha(x)\cong\alpha(y)}$

すなわち $x\cong y$ のとき，x を含む任意の正規プログラム図式 $\alpha(x)$ の中のいくつかの x を y で置きかえたもの $\alpha(y)$ は，$\alpha(x)$ と同値である．

$R2$: α を任意の正規プログラム図式，β を単位演算子性を有しない正規プログラム図式としたとき

$$\frac{x\cong\alpha\vee\beta x}{x\cong\beta^*\alpha}$$

ここにある正規プログラム図式 x は，つぎの3条件を満たすときにのみ単位演算子性を有するという．

(1)ある正規プログラム図式 y が存在して，$x\cong y^*$ となる．(2)正規プログラム図式 x が $\vee$ に関する和形式になっているとき，和形式の要素の少なくとも一つが単位演算子性を有する．(3)正規プログラム図式 x が・に関する積形式になっているとき，積形式の要素のどれもが単位演算子性を有する．

T_r における証明とは wff の有限系列であって，各 wff が公理であるか推論の規則により前に現われた wff から導かれるものであることを要する．ある wff $x\cong y$ で終わる証明があるときに $x\cong y$ は T_r で誘導可能であるといい $\vdash x\cong y$ としるし，T_r で誘導可能なあらゆる wff が普遍妥当であるときに，公理 T_r は無矛盾であるといわれる．あらゆる普遍妥当な wff が誘導可能なときに，T_r は**完全**であるといわれる．

定理 2.1 公理系 T_r は無矛盾である．

定理 2.2 公理系 T_r は完全である．

これらの定理が Kleene の正規表現の公理系に対する Salomaa の証明法[19]と同様の構成的(constructive)な仕方で Ito[19] においてなされている．証明法が構成的であることからつぎの定理を得る．

定理 2.3 正規プログラム図式の同値問題 $x\cong y$ は決定可能である．

グラフ図式と正規プログラム図式の関係よりつぎの結果を得る．

定理 2.4 グラフ図式の同値問題は決定可能である．

2.5 CF 形プログラム図式

正規プログラム図式が，Kleene の正規表現と類似のものであることから，数理言語の理論において正規表現の拡張として，CF 形言語が考えられるように，CF 形プログラム図式というものをつぎのように導入できる．CF 形言語が集合多項式に関する方程式として定義されるという結果を用いて，CF 形プログラム図式をつぎのように定義する．

$$x_i\cong\varphi_i(x_1,\cdots,x_n)\quad(i=1,2,\cdots,n)\qquad(2.1)$$

ここに φ_i は $\{x_1,\cdots,x_n\}$, $\{f_1,\cdots,f_r\}$, $\{\pi_1,\cdots,\pi_s,\bar{\pi}_1,\cdots,\bar{\pi}_s\}$ から $\vee$, ・ を演算として作られる多項式である．

正規プログラム図式が CF 形プログラム図式となることは，$x^*\cong e\vee xx^*$ と $p\cdot\bar{q}\cong\bar{p}\vee\bar{q}$ の関係より容易に知られる．正規プログラム図式ではない CF 形プログラムの例として，つぎのように回帰的に定義される関数を考えることができる．

$$\varphi(\xi)=\text{if } p(\xi) \text{ then } f(\xi) \text{ else } g(\varphi(h(\varphi(f(\xi)))))\qquad(2.2)$$

この $\varphi(\xi)$ は CF 形プログラムとしてつぎのように書ける．

$$\varphi(\xi)=p(\xi)f(\xi)\vee\bar{p}(\xi)g(\varphi(h(\varphi(f(\xi)))))\qquad(2.3)$$

CF 形プログラム図式としては，つぎのように書ける．

$$\varphi \cong pf \vee \bar{p} f \varphi h \varphi g \qquad (2.4)$$

CF 形プログラム図式については，つぎのような性質が知られている[19]．

定理 2.5 いま x, y を任意の CF 形プログラム図式としたとき，x と y の同値問題 $x \cong y$? は決定不能である．

定理 2.6 CF 形プログラム図式 x の停止性問題 $x \cong \phi$? は決定可能である．

定理 2.7 CF 形プログラム図式 x に対し，演算子の有限系列 α が，x の述語に値を割り当てたときに，到達可能か否かは決定可能である．

CF 形プログラム図式と同様，CS 形プログラム図式なども考えることができるが，ここでは言語理論のプログラム図式への応用については，これ以上立入らない．

3. Assignment Statement 形プログラム図式

この節では，assignment statement と，テスト命令からなる ALGOL 形のプログラム図式について，Luckham-Park-Paterson によって得られたプログラムの同値性決定問題についての結果を紹介する．同じく Yanov のモデルの一般化を目的としながら，前節が正規表現，CF 形言語のプログラム理論への応用であるのに対し，この節の理論は Rabin-Scott[42] や Rosenberg[44] のオートマトンの理論のプログラム理論への応用と考えてよい．

3.1 Assignment Statement 形プログラム図式

Luckham-Park-Paterson[28] による Assignment Statement 形プログラム図式（以下，AS 形プログラム図式と略す）は，つぎのように定義される．

（1） AS 形プログラム図式の基本アルファベットは

（a） 自然数

（b） 演算記号：$F_1^1, F_2^1, F_3^1, \cdots, F_1^2, F_2^2, \cdots$

（c） 述語記号：$T_1, T_2, T_3, \cdots, T_k$

（d） 番地記号：$L_1, L_2, \cdots, L_m$

（e） assignment 記号：":="

（f） 括弧，コンマ他

（2） AS 形プログラム図式の statement は，つぎの二つのどちらかである．

（a） 演算命令

$a.\ L_j := F_u^t(L_{k1}, \cdots, L_{kt})$

（b） テスト命令

$a.\ T_u(L_j)\quad b, c$

ここに $a, b, c, k_1, \cdots, k_t, t, u$ は自然数で，とくに a は命令番地といわれる．（a）の L_j は assigment 番地，$L_{k1}, \cdots, L_{kt}$ は retrieval 番地といわれる．

（3） AS 形プログラム図式は，始点番地と終点番地を与えられた statement の有限形系からなっている．AS 形プログラム図式は interpretation I が与えられたときに，初めてプログラムとしての意味を持つことになる．I は，（1）L_i に対し $I(L_i) \in \hat{D}$，（2）F_i^n に対し $I(F_i^n) : \hat{D}^n \to \hat{D}$，（3）$T_u$ に対し $I(T_u) : \hat{D} \to \{0, 1\}$ なる $\hat{D}$ 上の関数を規定する．interpretation I の下でのプログラム P_I は通常の仕方で実行され，P_I の計算が有限で終わるときに，P_I の計算値はその最終値として与えられ $\mathrm{val}(P_I)$ としるされる．

定義 3.1 AS 形プログラム図式 P と Q は，あらゆる interpretation の下で $\mathrm{val}(P_I) = \mathrm{val}(Q_I)$ か，P と Q ともに I の下で不定のときに 1 強調値であるといい，$P \cong Q$ としるす．

定義 3.2 同値性の定義において，interpretation の領域 $\hat{D}$ が有限（finite）であるときに，$P \cong_F Q$ としるされ，有限同値であるという．また I の下で関数記号に割り当てられる関数が，必ず recursive 関数であるときに，recursive 同値であるといわれ，$P \cong_R Q$ と記される．$P \doteq Q$ は弱同値関係を意味する．

3.2 Luckham-Park-Paterson の結果

Luckham-Park-Paterson は，まず multi-head オートマトンに関する決定問題を Turing 機械の停止問題に帰着させ，その決定不能性を証明したのち，AS 形プログラム図式によって multi-head オートマトンをシミュレートすることにより，AS 形プログラム図式の同値問題を Turing 機械の停止問題に帰着させ，その決定不能性を証明した．

定理 3.1 つぎの関係は Partially solvable でない．すなわち，recursively enumerable でない．

（1） $P \cong D$; $P \cong_F D$

（2） $P \cong Q$; $P \doteq Q$; $P \not\doteq Q$; $P \not\cong_R Q$

（3） P があらゆる recursive interpretation の下に収束する；P があらゆる finite interpretation の下に収束する．

ここに P, Q は任意の AS 形プログラム図式であり，D は $L : go\ to\ L$ なる形のプログラム図式であるとする．

これらの結果は，プログラム図式の一般的な公理系や，プログラムの簡単化に対する一般的アルゴリズムの存在問題に対し，否定的な答えを与えるものである．

Luckham-Park-Paterson は，実際には two-head オートマトンで Turing Machine の停止問題のシミュレーションを行ない，二つのメモリー番地を用いる AS 形プログラム図式に対し，上の定理がなりたつことを示している．しかし，つぎのような肯定的な結果も，彼らにより与えられている．

定理 3.2 つぎの関係は Partially solvable である．

（1） $P \not\equiv D$; $P \not\equiv_F D$

（2） $P \not\equiv_F Q$

（3） P があらゆる interpretation の下に収束する．

定理 3.3 つぎの AS 形プログラム図式の同値問題は決定可能である．

（1） メモリー用番地が一つの AS 形プログラム図式．

（2） 関数記号が 1 変数で，いかなる演算命令も一つのループにしか現われない AS 形プログラム図式．

定理 3.3 の第 1 の結果は，Rutlege[45] により示された Yanov の結果と一致するものであり，以上の結果より，一つのメモリー番地しか使わない AS 形プログラム図式の同値問題は決定可能で，二つ以上のメモリー番地を使う AS 形プログラム図式の同値問題は決定不能となることがいえる．

4. 述語論理によるプログラムの性質の証明

これまでの節において，プログラム図式の同値性を基本的な性質として取り上げ，オートマトン理論，言語理論に近い立場から，プログラム図式の同値問題が一般的には決定不能であるが，いくつかの有意義な決定可能な場合が存在することも示してきた．正規プログラム図式に対する公理系と特別なプログラム図式に対するアルゴリズムは得られたが，任意のプログラムに対して，その性質を証明するのに適用するには限界が明らかである．これに対し，プログラムの性質を述語論理の形式で表現し，述語論理の中で証明できる性質は，古くから知られている述語論理に対する諸種のアルゴリズムにより証明しようとする研究が，Manna[29] および Ito[19),21)] らにより始められている．このような方法は，述語論理が有力な論理体系であるとともに，その証明のアルゴリズムが計算機による定理の証明の研究により，よく知られていることもあり，きわめて将来の実用性に富んでいると考えられ，本節では，これにつき詳述する．

4.1 （第一階）述語論理

この項では(第一階)，述語論理についての簡単な説明を行なう[36]．述語論理において用いられる記号として，

（1） ，（ ）

（2） $\neg \rightarrow \wedge \vee \equiv \exists \forall$ （論理記号）

（3） f_i^n $(i, n \geq 1)$ （n 変数関数）

（4） x_i $(i \geq 1)$ （変数）

（5） a_i $(i \geq 1)$ （定数）

（6） p_i^n $(i, n \geq 1)$ （n 変数述語）

（7） T F (true, false)

述語論理における論理式は，つぎのように帰納的に定義される．まず述語項 (term) はつぎのように定義される．

（1） 変数 x_i と定数 f_i^0 は述語項 (term) であり，（2）$t_1, t_2, \cdots, t_n (n \geq)$ を述語項としたときに，$f_i^n(t_1, \cdots\cdots, t_n)$ も述語項である．

これら二つの規則から作られる式だけが述語項である．

述語原子式 (atomic formula) は，つぎの二つの規則から帰納的に作られるあらゆる式である．

（1） T, F, p_i^0, q_i^0 は述語原子式であり，（2）$t_1, t_2, \cdots, t_n$ が述語項のときに，$p_i^n(t_1, \cdots, t_n)$ は述語原子式である．

このとき述語論理式は，つぎの三つの規則から帰納的に作られる式で，述語論理の wff (well-formed formula) ともいわれる．

（1） 述語原子式は述語論理式 (wff) である．

（2） A が述語論理式のときに，$\neg A$, $(\forall x_i)A$, $(\exists x_i)A$ は述語論理式である．

（3） A と B が述語論理式のとき，$(A \rightarrow B)$, $(A \wedge B)$, $(A \vee B)$ と $(A \equiv B)$ も述語論理式である．

述語論理式に対する interpretation I もプログラム図式におけると同様に空でない集合 $\hat{D}$ (領域) と関数 f_i^n, 述語 p_i^n, 変数 x_i に対する $\hat{D}$ における関数，述語，変数により規定され，I の下での述語論理式 A を $[A, I]$ と表現する．

ある述語論理式 A は

（1） もし，あらゆる interpretation I に対し，$[A, I]$ が普遍妥当 (valid) のときに普遍妥当であるといわれ，

（2） $[A, I]$ がある interpretation I に対し充足

可能なら，充足可能 (satisfiable) であるといわれ，

(3) $[A, I]$ が充足可能でないときに，充足不能といわれる。この定義より明らかに $\neg A$ が充足不能のとき，そのときに限り A は普遍妥当となる．

任意の述語論理式は，前置記号 $(\exists, \forall)$ がすべて否定せられずに，式の一番前に置かれ，その作用域がいずれも式の終わりまで及ぶような標準形に直すことができ，この標準形を冠頭標準形という．とくに，存在記号がすべて全称記号の前にあるような冠頭標準式が必ず対応し，これを Skolem の標準形という．述語論理の決定問題については，つぎのような性質が知られている．

(1) 述語論理の普遍妥当性問題は決定不能である．

(2) 述語論理の普遍妥当性問題は半決定可能 (semi-decidable) であり，半決定アルゴリズム (semi-decision algorithm) が存在する．すなわち，述語論理式が普遍妥当であれば，必ず有限回の手順まで普遍妥当性が決定できることが保証されたアルゴリズムが存在する（普遍妥当でない場合には，有限回で証明が完了する保証はない)．計算機による定理の証明においては，いくつかの半決定アルゴリズムが知られているが，Robinson による Resolution Principle[43] が有力な手法として脚光をあびている．

(3) 普遍妥当性問題が決定可能な特別な場合もいくつか知られており，たとえば

(a) 述語論理式を冠頭標準形にしたときに，$\forall\cdots\forall\exists\cdots\exists$ が冠頭にきて，関数が式中に現われないとき，

(b) $\forall\cdots\forall\exists\forall\cdots\forall$ が冠頭にきて，関数が式中に現われないとき，

(c) $\forall\cdots\forall\exists\exists\forall\cdots\forall$ が冠頭にきて，関数が式中に現われないときには決定可能である．

以上までに説明した述語論理の形式の下にプログラムの性質を表現し，述語論理の普遍妥当性問題に対して知られている半決定アルゴリズム (semi-decision algorithm) を用いて，プログラムの諸性質を証明しようとするのが，この節の目的である．

4.2 述語論理によるプログラムの性質の証明法

この節では述語論理を用いて，プログラムの正当性 (correctness)，同値性 (equivalence)，停止性 (termination) を証明するアルゴリズムについて述べる．プログラムとしては，前節で定義した AS 形プログラム図式を考えて話しを進めるが，第2節の CF 形プログラム図式に対しても，この項の手法が適用できる．

まず，プログラムの正当性の概念を定義することから始める．

定義 4.1 (強い意味の正当性)

$(\forall\xi\in\hat{D})[(P(\xi)$ が停止する$)\wedge\pi[P(\xi)]]$

のときに，プログラム P は π に関して強い意味で正しいといわれる．

定義 4.2 (弱い意味の正当性)

$(\forall\xi\in\hat{D})[(P(\xi)$ が停止する$)\rightarrow\pi[P(\xi)]]$

のときに，プログラム P は π に関して弱い意味で正しいといわれる．

定義 4.3 (テスト可能な正当性)

$(\forall\xi\in\hat{D})[if\ (P(\xi)$ が停止する$)$ then $\pi[P(\xi)]$ else $\gamma[\hat{P}(\xi)]]$

のときに，プログラム P は π と γ に関して，テスト可能な意味で正しいといわれる．$\hat{P}(\xi)$ は $P(\xi)$ より得られるプログラムで，$\xi\in\{\xi|P(\xi)$ が停止しない$\}$ に対し，停止する適当なプログラムとする．

テスト可能な正当性という概念は，強い意味の正当性，弱い意味の正当性という概念の一般化になっていて，$\gamma[\hat{P}(\xi)]\equiv$false のときには，強い意味の正当性となり，$\gamma[\hat{P}(\xi)]\equiv$true のときには弱い意味での正当性となることがわかる．しかし $\hat{P}$ の作り方は問題の性質に依存し，一般に論じることはきわめて困難である．

4.2.1 プログラムの述語論理式への変換

プログラムの性質を述語論理で証明するには，プログラムを述語論理の式で表現する必要がある．ここでは Assignment Statement を持つプログラム図式に対し，述語論理の式を作るアルゴリズムを与える．このような述語論理の式は，プログラムを遂行する際の遂行条件を意味しているという観点から，これを遂行条件式と呼ぶ．プログラム P の π に関する遂行条件式は，**図 4.1** のように作られる．

(A-1) プログラム変数 $\boldsymbol{x}=(x_1,\cdots,x_n)$，入力変数 $\boldsymbol{u}=(u_1,\cdots,u_m)$ をリストする．

(A-2) 入力点でプログラム変数，入力変数に条件

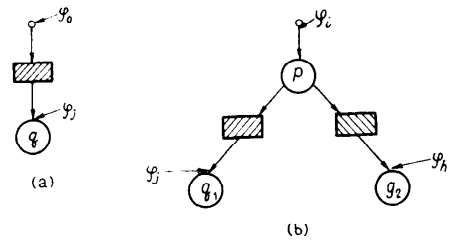


図 4.1 グラフ図式への述語条件の割当

を与えることができるときにはそれを書き，φ_0 と表わす．とくに入力条件が与えられていないときは $\varphi_0 \equiv T$ とする．各述語の入口の所に順番に述語 φ_i を i の番号を増加させながらつけていき，出力点にはテスト条件 π をつける．各 φ_i は n 変数の述語とする．

(A-3) まず図 4.1(a)のようなプログラムの始まりに対して，E_0 をつぎのように作る．

E_0: $\varphi_0 \rightarrow \varphi_j$ (▨)

ここに上式の ▨ はassignment statement が図 4.1(a)の ▨ 部分にあれば，それに対応する変数が変えられることを意味している．

(A-4) 図 4.1(b)のような要素がプログラムに現われるが，これに対しては

E_{i1}: $\varphi_i \wedge p \rightarrow \varphi_j$ (▨)

E_{i2}: $\varphi_i \wedge \bar{p} \rightarrow \varphi_k$ (▨)

を作る．ただし図 4.1 において，q, q_1, q_2 の点がプログラムの終点となるときは，φ_j, φ_k を π で置き換えるものとする．

(A-5) 上述の手順で，$E_0, E_1, E_2, \cdots, E_M$ なる式が得られたとき，つぎのような式を作る．

$$E_P[\pi] \equiv E_0 \wedge E_1 \wedge E_2 \wedge \cdots \wedge E_M$$

$$\hat{E}_P[\pi] \equiv (\forall \underline{x}) E_P[\pi] \equiv (\forall \underline{x})[E_0 \wedge E_1 \wedge \cdots \wedge E_M]$$

以上のように作った $\hat{E}_P[\pi]$ を第1種遂行条件式という．

上記アルゴリズムで q, q_1, q_2 が終点のときに φ_j, φ_k を π で置き換える以外は，そのまま用いたときに得られる

$$E_P^*[\pi] \equiv E_0 \wedge E_1 \wedge \cdots \wedge E_M$$

$$\hat{E}_P^*[\pi] \equiv (\exists \underline{x}) E_P^*[\pi] \equiv (\exists \underline{x})[E_0 \wedge E_1 \wedge \cdots \wedge E_M]$$

とした $\hat{E}_P^*$ を第2種遂行条件式という．

このような遂行条件式により，プログラムの性質が述語論理を用いると証明できることが知られており，次項に二，三の基本定理を述べる．

4.2.2 プログラムの性質と遂行条件式

遂行条件式 $\hat{E}_P, \hat{E}_P^*$ は φ_i を適当に選んだときに真 (true) になりうるならば，充足可能であるといわれる．つぎのような定理が知られている．

定理 4.1 $\hat{E}_P[\pi]$ が充足可能なときそのときに限り，P は π に関して弱い意味で正当なプログラムである．

定理 4.2 $\hat{E}_P^*[\pi]$ が充足可能でないとき，そのときに限り，P は π に関して強い意味で，正当なプログラムである．

定理 4.3 $\hat{E}_P^*[T]$ が充足可能でないとき，そのときに限り，プログラム P は停止する．

また，以上の形式化を用いて，プログラム P と Q の完全同値性（P と Q が停止して同じ値を取る）P と Q を証明する定理が知られている．このために P と Q から，

$E_P[\pi]$ と $E_Q^*[\pi]$

を作り

$$\hat{E}[P, Q] \equiv (\exists \underline{x})[E_P[\pi] \wedge E_Q^*[\pi]]$$

としよう．

この $\hat{E}[P, Q]$ を用いて，つぎの定理が得られる．

定理 4.4 $\hat{E}[P, Q]$ が充足可能でないとき，そのときに限り，P と Q は完全同値である．

以上四つの定理により，AS 形プログラム図式の正当性・停止性・完全同値性が遂行条件式を用いて，述語論理の形式の下に表現されたから，あとは述語論理の証明アルゴリズム——たとえば Robinson の Resolution Principle,[43] Wang のプログラム[52]を用いて，これらの性質を証明することができる．また，**4.1** で述べたように，述語論理においては，いくつかの決定可能な場合が知られているから，上記定理における遂行条件式を，いったん Skolem の標準形に直してから，決定可能な場合になっているか否かを調べることもできる．

なお，この節では AS 形プログラム図式についての遂行条件式の作り方を説明したが，CF 形プログラム図式などについても，同様な形で適用できる[19),30)]．

5. プログラム理論のその他の側面

これまでの節において，プログラム理論において，最近得られた基礎的な数学的結果を，紙面の許す範囲内で詳細に解説した．これらは Yanov の結果を出発点として，それぞれ

(1) 数理言語の理論
(2) オートマトン理論
(3) 述語論理

という確立された手法を適用することにより，プログラムの持つ性質を数学的に形式化し，その基本的性質を明らかにしたものである．しかし，プログラムの性

質を数学的に解析しようとする他のいくつかの研究があり，本節ではプログラム理論研究の小史を概説しながら，それらにつき解説する．

（1） 述語論理の普遍妥当性問題が，一般には決定不能であることを証明するために，Turing 機械のプログラムの停止性と，述語論理の普遍妥当性との関連づけが，Turing により 1936 年になされているが，現代的なプログラム理論の研究は，1958 年の Yanov[53] の研究に始まると考えられている．Yanov (1958) はソロー・チャートで表わされるようなプログラムを，ソ連で使われていたプログラミング言語の記法をもとにして形式化し，そのような形で表わされた二つのプログラムの同値問題を解き，プログラム理論の基礎を置いた．この結果は拡張され，簡明化された形で第2節で述べられている．Yanov 図式のフロー・チャートとの関係については，Karp (1960) は，いわゆる，Karp 図式との関連で述べ，Kaluzhnin (1961) はグラフ図式との関連において明らかにした．グラフ図式はフロー・チャートを形式化したもので，Böhm-Jacopini (1966)，Cooper (1966)，Basu (1968) らのグラフ図式に基づくプログラムの等価変換の基礎をなすものである．また，Kaluzhnin のグラフ図式の interpretation に現われる state vector は，McCarthy (1962) により program state vector と名づけられ使われている．Yanov 図式と有限オートマトンの関係は，Igarashi (1963) と Rutledge (1964) により明らかにされ，Yanov 図式の同値性が有限オートマトンの同値性に帰着できることが示された．

（2） Wang[52] は Mathematical theory of computation における基本的諸問題とその重要性を強調するとともに，述語論理の定理証明のアルゴリズムをプログラム化し，それによって数学上の結果やプログラムの正当性の機械的なチェックが可能なことを述べた (1960)．これらの考えは，計算機による定理の証明の研究者や，McCarthy により継承されている．定理の証明に関しては Davis-Putnam[9] の研究などがあったが，Robinson は Resolution Principle に基づくすぐれたアルゴリズムを発見した (1965)．Hartmanis-Stearns (1966) はアルゴリズムの複雑性の問題を抽象的な立場から扱ったが，Kolmogrov[9] の情報の複雑性の概念とともに，プログラムに関する定量的な測度を与える道を開くものと考えられる．

（3） Ershov (1961) は Operator Algorithm というアルゴリズムの表現形式を与え，アルゴリズムの表現可能性と等価変換を一般的に論じようとしており，また，Operator Algorithm と Kaluzhnin のグラフ図式や Yanov 図式との関連も論じている．McCarthy[33),34)] は LISP を基礎としてアルゴリズムの表現の問題を Ershov と同様の立場から試みた．また，conditional expression に対する公理系，Recursive Equivalence の概念を応用した Recursion Induction というアルゴリズムの同値性の証明法，質問の論理 (logic of question[16]) を応用した abstract syntax といわれるミンクックスおびびセマンティクスの定義の仕方についても述べた．これらの応用として，コンパイラーの正当性を，source program と object program の計算の同値性から定義し，簡単な場合について，コンパイラーの正当性を Painter とともに証明した (1967)．

Recursion Induction の手法は Cooper により少し拡張され，abstract syntax は PL/I において採用されている．

（4） プログラム図式の同値問題の決定不能性は，1964 年に Luckham-Park[27] により初めて証明されたが，最近に至るまでその成果は，一般には知られていなかった．彼らの結果は 1967 年に Luckham-Park-Paterson により，メモリー番地を二つしか使用しないような場合についてなりたつことが証明され，さらに第3節において述べたような結果が得られた．彼らの結果は一般的な同値問題に対し，決定不能という否定的な答えを与えるが，Paterson[40] は興味深い決定可能な場合をいくつか求めた．

（5） Igarashi (1968) は ALGOL 形のプログラムの同値性に対する公理系を，Yanov，McCarthy の公理系をもとにして，assigment statement に対する公理系 (T_a)，conditional expression に対する公理系 (T_c)，go-to statement に対する公理系 (T_g) を与え，その semantic completeness を証明している．しかし T_a, T_c と T_g の三つを同時に有機的に含むプログラム族に対する公理化の問題は，一般に non-axiomatizable となる．Igarashi はこの場合についてはプログラムがあらかじめあるステップ数の中で完了するという強い仮定のもとに semantic completeness を述べている．

assignment statement に関する公理系は，その後 de Bakker,[3] Kaplan[23] らによっても論じられている．

（6） Ito (1968) は，第2節で述べた正規プログラム図式，CF 形プログラム図式の概念を導入し，その

性質を言語理論を用いて証明し，これと Yanov 図式，グラフ図式の関係も論じた．その後正規プログラム図式の性質は，Kaplan によっても述べられている．Ito[19] は CF 形プログラム図式の停止性，同値性が述語論理を用いて証明できることを述べ，さらに，その手法がプログラムの正当性の証明にも適用できることを示した[20),21)]．同様の試みは Manna[29)~31)] によってもなされ，Manna は述語論理の決定可能な場合の結果から，プログラムの性質についての決定可能な場合を求めた．これらは Naur[38)] および Floyd[14)] のアルゴリズムを数学的に一般化するとともに厳密化したものである．

（7） Karp-Miller[26)] は Yanov のモデルを並列演算 (parallel computation) の立場から拡張し，並列形プログラム図式の概念を導入し，determinacy, boundedness, termination, repetition-freeness に対する決定アルゴリズムを与え，並列形プログラム図式に対する同値問題が一般には決定不能であることを証明した．また，parallel flowchart, decision-free flowchart といった並列プログラム図式の性質についても論じた．上原[50)] は Karp-Miller のモデルを基礎として二，三の結果を得ている．

（8） Scott[46)] はプログラム理論的立場から，オートマトンの統一理論に対する提案を行なうとともに，最近，Mathematical theory of computation の新しい一般的形式化を与えようとする興味深い試みを行なっている[47)]．彼はプログラミング言語の数学的セマンティクスを確立しようとする立場から，data type と function の性質に関する新しい考察を行ない，このために，data type の要素間に新しい関係 "$x \sqsubseteq y$" を導入し，この関係をもとにして新理論に対する基礎を与えようとしている．"$x \sqsubseteq y$" は(1) y は x と無矛盾である．あるいは(2) y は x より正確である．あるいは(3) y は x より多くの情報を有するといった意味を持っている．

6. むすび

この論文では第2節でプログラム図式の同値性に関する公理系，第3節でプログラムに関する各種決定問題，第4節で，プログラムの諸性質を証明するアルゴリズムを述べ，第5節においてプログラム理論研究の小史を概説した．しかし，プログラム理論と重要な関係にあるプログラミング言語のセマンティクス，プログラムの自動作成，その他オートマトン理論，言語理論，コンパイラに関連したいくつかの重要な研究について，紙面の都合から言及できなかった．これらについては，参考文献により補っていただきたい．

なお，本論文の冒頭に "与えられた問題に対するプログラムを書いたときに，そのプログラムが望むような計算結果を与えてくれるか否かを知ること" をプログラム理論の基本的問題としてかかげた．この問題は **定義 4.3** に与えたテスト可能な正当性の概念を用いて形式化できるが，この正当性概念に対しては，一般に partial decision procedure も存在せず，また，プログラムが停止しない場合のテスト条件の作り方が実用上からも困難である．しかし，第4節で与えた証明のアルゴリズムを用いることにより，プログラムの強い意味での正当性，および弱い意味の正当性は証明できる．また，プログラムの正当性の概念は Ito[20)] に見られるように，同値性概念と結びつけられる．これらより，本論文に述べたプログラム理論の成果を用いて，上記の問題をある程度解決できることが知られており，実際問題に対する応用も試みられており，好結果が期待されている．

プログラム理論に限らず，一般に論理的問題においては，単純なシステムを除いて，必ず論理的決定不能命題 (logically undecidable proposition) という悲観的な結果に遭遇しゆきづまる．これを克服することは，今後のコンピュータ・サイエンス全体にとっての重要課題である．これに対する論理学的な接近があるが，本論文の範囲をこえることであり，これ以上は立ち入らない．

本論文によって，プログラム理論を中心とした計算の理論 (Mathematical theory of computation) の研究に興味をいだく研究者が増えれば幸いである．

最後に，本論文を執筆するにあたり，ご激励いただいた日本大学数学科小林孝次郎助教授に深謝の意を表したい．

参考文献

1) Ackerman, W.: Solvable cases of the decision problem, North-Holland (1954)
2) Aho, A. V. and J. Ullman: Theory of languages, Mathematical System Theory, **2** (1968)
3) de Bakker, J. W.: Axiomatics of simple assignment statements, Mathematisch Centrum Report, Amsterdam (1968)
4) Basu, S. K.: Transformation of program schemes to standard forms, Proc. IEEE Symp. on Switching and Automata Theory (1968)

5) Böhm, C. and G. Jacopini: Flow diagrams, CACM, **9** (1966)
6) Cooper, D. C.: On the equivalence of certain computations, Comp. J., **9** (1966)
7) Cooper, D. C.: Reduction of programs to a standard form by graph transformation, Proc. International Seminar on Graph Theory (1966)
8) Davis, M.: Computability and Unsolvability, McGraw-Hill, New York (1958)
9) Davis, M. and H. Putnam: A computing procedure for quantification theory, JACM, **7**, 3 (1960)
10) Engler, E.: Algorithmic properties of structures, Mathematical System Theory, **1** (1967)
11) Ershov, A. P.: Operator Algorithms I, Problem of Cybernetics, vol. 3, Pergamon Press (1962)
12) Feldman, J. and D. Gries: Translator writing systems, CACM, **11** (1968)
13) Fels, E., Kaluzhnin Graphs and Yanov Writs: Logik und Logikkalkül, Verlag Karl Alber (1962)
14) Floyd, R.: Assigning meanings to programs, Proc. Symposia in Applid Mathematics, vol, 19, AMS (1967)
15) Ginsburg, S.: Mathematical Theory of Context-free Languages, McGraw-Hill (1966)
16) Harrah, D.: A logic of questions and answers, Philosophy of Science, **28** (1961)
17) Igarashi, S.: On the logical schemes of algorithms, Information Proessing in Japan, vol. 3 (1963)
18) Igarashi, S.: An axiomatic approach to the equivalence problems of algorithms whith applications, Report of Computer Centre, Univ. of Tokyo (1968)
19) Ito, T.: Some formal preperties of a class of program schemata, Proc. IEEE Symposium on Switching and Automata Theory (1968)
20) Ito, T.: On proving the correctness of programs, Fourth Princeton Conference on Information Sciences and Systems (1970)
21) 伊藤貴康: プログラム理論について, 電子通信学会オートマトン研究会 (1969)
22) Kaluzhnin, L. A.: Algorithmization of mathematical problems, Problems of Cybernetics, vol. 2, Pergamon Press (1961)
23) Kaplan, D. M.: Some completeness results in a mathematical theory of computation, JACM, vol. 15 (1968)
24) Kaplan, D. M.: Regular expression and the equivalence of programs, JCSS, vol. 3 (1969)
25) Karp, R. M.: A note on the application of graph theory to digital computer programming, Information and Control, **3** (1960)
26) Karp, R. and R. E. Miller: Parallel Program Schemata, J. CSS, vol. 3 (1969)
27) Luckham, D. and D. Park: The undecidability of the equivalence problem for program schemata, Report of Bolt Beranek and Newman Inc. (1964)
28) Luckham, D., D. Park and M. Paterson: On formalized computer programs, JACM (to appear)
29) Manna, Z.: Properties of programs and the firstorder predicate calculus, JACM, vol. 16 (1969)
30) Manna, Z.: Formalization of recursively defined functions, ACM Symposium on Theory of Computing (1969)
31) Manna, Z.: The correctness of programs, JCSS, vol. 3 (1969)
32) Marill, T.: Computational chains and simplification of computer programs, IRE Trans. on Electronic Computers, vol. EC-11, No. 2 (1962)
33) McCarthy, J.: Towards a mathematical science of computation, Proc. IFIP '62 (1963)
34) McCarthy, J.: A basis for a mathematical theory of computation, Computer Programming and Formal Systems (1963)
35) McCarthy, J. and J. Painter: Correctness of a compiler for arithmetic expressions, Proc. Symposia in Applied Mathematics, vol. 19, AMS (1967)
36) Mendelson, E.: Introduction to Mathematical Logic, Van Nostrand (1964)
37) Narasimhan, R.: Programming languages and computers, Advances in Computers, vol. 8, Academic Press (1967)
38) Naur, P.: Proof of algorithms by general snapshots, BIT, **6** (1966)
39) Painter, J.: Semantic correctness of a compiler for an ALGOL-like language, Dissertion, Stanford University (1967)
40) Paterson, M.: Equivalence problems in a model of computation, Dissertation, Cambridge Univ. (1967)
41) Paterson, M.: Program Schemata, Artificial Intelligence, vol. 3 (1968)
42) Rabin, M. and D. Scott: Finite automata and their decision problems, IBM Journal R & D, vol. 3 (1959)
43) Robinson, J. A.: A machine-oriented logic based on the resolution principle, JACM, vol. 12 (1965)
44) Rosenberg, A.: On multi-headed finite auto-

mata, IBM Journal R & D, vol. 11 (1966)
45) Rutledge, J.: On Ianov's program schemata, JACM, 11 (1964)
46) Scott, D.: Some definitional suggestions for automata theory, JCSS, vol. 1 (1967)
47) Scott, D.: Outline of a mathematical theory of computation, Fourth Princeton Conference on Information Sciences and Systems (1970)
48) Steel, T.: Formal language description languages for computer programming, Proc. IFIP Working Conference (1966)
49) Turing, A. M.: On computable numbers, Proc. London Math. Soc. **42** (1936)
50) 上原貴夫：プログラムの等価性について，電子通信学会誌，vol. 53-C, No. 4 (1970)
51) Waldinger, R. J. and R. C. Lee, PROW: A step toward automatic program writing, Proc. International Joint Conference on Artificial Intelligence (1969)
52) Wang, H.: Toward Mechanical Mathematics, IBM Journal (Jan 1960)
53) Yanov. Yu. I.: The logical schemes of algorithms, Problem of Cybernetics, vol. 1 (1960)

(昭和45年7月17日受付)