

プログラムの形式的正当性

京大 工学部 謝 章文

§ 1. 序

文字列としてのプログラムを形式的に取り扱い、そのプログラムの構造および諸性質を考察する目的で、プログラミング言語、プログラミングおよびプログラムの形式的モデルをそれぞれ設定する。形式的取り扱いの基礎を与えるプログラミング言語は、その構文解析によって得られる幾つかの集合の集合として特徴づける。さらに、プロセデュアとしてのプログラムの一般的正当性すなわち全ての正しいプログラムに存在する共通の性質について考察し、与えられたプログラムのモデルについて、形式的に判定できる正当性およびそのアルゴリズムについて述べる。最後に、プログラムの一般的正当性と求められたプログラム形の形式的正当性の関係について述べる。

§ 2. プログラミング言語

ここでは、プログラムの構造を形式的に取り扱うために、必要な表現機能をそなえた形式的言語について考察する。

〔定義1〕 N は $\mathcal{N}_i$ を要素とする可附番無限集合、 D^k は有限ないし可附番無限集合とするとき ($k=1, 2, \dots, m$)、 N の全ての要素 $\mathcal{N}_i$ に対して、ordered $(m+1)$ -tuple $(\mathcal{N}_i, D_i^1, D_i^2, \dots, D_i^m)$ 、ただし $D_i^j \subseteq D_i^j$, $j=1, 2, \dots, m$, が1つただ1つ与えられているとき、 $\mathcal{N}_i$ を m -value name, D_i^j を m -value name の j -range という。また、 $\#D_i^j = 1$ ^{*1} のとき、 $\mathcal{N}_i$ を j -constant, $\#D_i^j > 1$ のとき、 $\mathcal{N}_i$ を j -variable, D_i^j を $\mathcal{N}_i$ の j -domain という。さらに、各 j に対して、 m -value name $\mathcal{N}_i$ に1つただ1つの ordered pair $(\mathcal{N}_i, d)$ $d \in D_i^j$ が与えられることを、 $\mathcal{N}_i$ が j -define されるといい、 $(\mathcal{N}_i, d)$, d をそれぞれ $\mathcal{N}_i$ の j -name relation, j -denotation という。 $\mathcal{N}_i$ が j -define されており、 j -name relation $(\mathcal{N}_i, d)$ が与えられているとき、 $(\mathcal{N}_i, d')$ $d' \in D_i^j \wedge d \neq d'$ でそれを置き換えることを、 $\mathcal{N}_i$ を j -redefine するといひ、また、 $\mathcal{N}_i$ によって d が denote されることを、 $\mathcal{N}_i$ が j -refer されるという。

〔定義2〕 2-value name の recursive set の 1, 2-range がそれぞれ finite, recursive set であるとき、2-value name を Identifier, 前者の要素を Address, 後者の要素を

*1 $\#D$ は集合 D の cardinal number を表わす。

Valueと呼ぶ。

現実のプログラミング言語では、1-constant または dynamic register などは、implicit に表現されている。また、2-constant は literal である。

〔定義3〕 constant を含まない 1-value name の集合の range が name の集合であるとき、その集合の要素を meta-variable と呼ぶ。

〔定義4〕 Statement Schema Form SSF を次のように定める。

$$SSF_i = \lambda v_i^{1d} v_i^{1r} v_i^{2d} v_i^{2r} \overline{SSF_i}^{*2} \quad (1.1)$$

$$\overline{SSF_i} = \langle F_i, v_i^{1d}, v_i^{1r}, v_i^{2d}, v_i^{2r} \rangle$$

ただし、 $v_i^{(1d, 1r)}$: meta-variable の集合。右肩の添字はその meta-variable の位置に代入される Identifier の (1)-denotation が (define) されることを意味する。

F_i : $v_i^{1d} \cup v_i^{1r} \cup v_i^{2d} \cup v_i^{2r}$ の要素の meta-variable に代入される Identifiers 上の function symbol の集合。

〔定義5〕 形式的プログラミング言語を次のように定める。

$$\mathbb{L} = \langle N, D \rangle \quad (1.2)$$

ただし、 N : Identifier の集合。 D : SSF の集合。

N は 4 つの部分集合にわけられる。 $N = N_1 \cup N_2 \cup N_{c1} \cup N_{c2}$

$N_{1(2)}$: 1(2)-domain が空集合でない 1(2)-variable の集合。

*2. $\lambda v \in E$ は E 内の集合 v の要素をすべて bound すること。

$N_{C1}(C2)$: $1(2)$ -constantの集合。ここでは、議論を簡単にするため、compiler levelの言語を考えることにする。したがって、 $N_1 \cap N_2 = \emptyset$ ^{*3}と仮定する。

〔定義6〕 Statement Schema SS は次のように定まる。

$$SS_i = \sum_{N'_i \cup N_{C1}, N'_i \cup N_{C2}}^{v_i^{1d} \cup v_i^{1r}, v_i^{2d} \cup v_i^{2r}} |SSF_i|^{*4} \\ = \langle F_i, V_i^{1d}, V_i^{1r} \cup C_i^1, V_i^{2d}, V_i^{2r} \cup C_i^2 \rangle \quad (1.3)$$

ただし、 $N'_i \subset N_i$, $N'_i \subset N_{C1}$, $N'_i \subset N_{C2}$, $C_i^{1(2)}$: $1(2)$ -constantの集合。

〔定義7〕 言語 L の任意の Statement Schema の並びが与えられたとき、一意に各 Statement Schema SS_i が定まり、かつ、 $V_i^{1d}, V_i^{1r}, V_i^{2d}, V_i^{2r}, C_i^1, C_i^2$ と $\lambda v_i^{1d} v_i^{1r} v_i^{2d} v_i^{2r} \overline{SSF_i}$ が求まるとき、言語 L は Unambiguous であるという。

以後の議論において、プログラミング言語とは Unambiguous なものを指すことにする。

§3. プログラミングとプログラム

はじめに幾つかの term を定める。集合 D の要素の重複順列を SSF -seg という。SSF-seg において、各 SSF_i ごとに bound されている meta-variable を、意味上の変更なしに SSF -seg の先頭で bound するようにした形を Prenex SSF -seg という。Prenex SSF -seg に出現する v^1, v^2 の meta-variable の grouping をそれぞれ

*3 $\emptyset$: 空集合。 *4 $\sum_N^v E$: E 内の集合 v の要素を、

全て、集合 N の要素で適当に代置したもの。

Label Linkage, Variable Linkage, 両者を Name Linkage という。
 Name Linkage に従って, Prenex SSF-seg の各 group の meta-variable を代表 meta-variable で置きかえたものを Seg-Form, Seg-Form に区切りマーク (END line) を追加したものを PS-Form という。PS-Form の V^1, V^2 の各 meta-variable へそれぞれ distinct な $N_1, N_2 \vee N_2$ の要素を割り当てることを, Label Assignment, Variable Assignment, 両者を Name Assignment という。

〔定義 8〕 次の 3 つの Process を言語 $\mathcal{L}$ 上の形式的プログラミングと定義する。

1. $\mathcal{L}$ 上の (Prenex) SSF-seg の作成。
2. Prenex SSF-seg の Name Linkage。
3. PS-Form の Name Assignment。

〔定義 9〕 形式的言語 $\mathcal{L}$ 上の形式的プログラミングによって作成される Statement Schema の並びを形式的プログラム形という。

直観的な形式的プログラムの解釈は, "その指令に従って, 計算機がなんらかの動作を連続して行なえるもの" である。

Non deterministic な Program, 無意味な部分, 無意味な Action の指令を含む。ただし, 次の仮定を設ける。

〔定義 10〕 そのプログラム内で全ての denotation (name relation) が定められる Identifier を "Inner Identifier" とし, それ以外のものを "External Identifier" という。

〔仮定〕 Program の External Identifier は既に外部でもって、その denotation が定められているとする。

この仮定より、External Identifier は計算機の動作可能性には影響を与えず、プログラムの制御可能性(flow or value)に影響する。

§4. プログラムの正当性の概念

この議論において、プログラムとは、停止することを意図するもののみに限定する。

〔定義11〕 プログラムの一般的正当性とは、全ての正しいプログラムに共有される性質とする。

〔定義12〕 プログラムの一般的正当性とは、それが、ある Algorithm または Semi-Algorithm の表現であることとする。

〔定義11〕は〔定義12〕で置き換えられるものと考える。

しかし、現在 Algorithm のクラスの形式的定義およびそのクラスの要素による Algorithm の形式的定義は提案されているが、Algorithm そのものの直接的定義は与えられていない。従来、提案されている Algorithm の概念にとって本質的であると考えられる幾つかの一般的特徴は、1) 段階性 2) 決定性 3) 要素性 4) 方向性 5) 無限性 等である。これに着目して次節以下でプログラムの形式的正当性を定義する。ここでは、一般的正当性について、次のような定義を採用する。

〔定義13〕 プログラムを P 、その Input Domain を D_x 、

Input Vector を x とするとき. $\forall x (x \in D_x)$ に対して, $P(x)$ の Computation が finite で かつ 常に同一の final result を得るならば, P は totally correct であるという。

〔定義 14〕 プログラムを P , その Input Domain D_x , Input Vector x とするとき, $P(x)$ の Computation が finite で かつ 常に同一の final result を得るような $x (x \in D_x)$ が存在するならば, P は partially correct であるという。

§5. プログラムの形式的正当性 (その1)

5-1 L-Correctness

形式的言語 L 上の形式的プログラム形を次のように表わす。

$$PS = S_1 S_2 S_3 \cdots S_n E \quad (1.4)$$

S_i : Statement Schema の occurrence, 簡単のため, その Statement Schema も表わすものとする。 E : 区切りマーク。

$$S_i = \langle F_i, V_i^{ld}, V_i^{lr} \cup C_i^1, V_i^{zd}, V_i^{zr} \cup C_i^2 \rangle \quad (1.5)$$

$$E = \langle \#, Le \rangle. \# \text{ 終わりマーク. } Le \in N_1$$

〔定義 15〕 $i \neq j \Rightarrow V_i^{ld} \cap V_j^{ld} = \phi$ ならば, PS は L -deterministic である。また, $(\bigcup_{i=1}^n V_i^{lr}) \cup \{Le\} \subseteq \bigcup_{i=1}^n V_i^{ld}$ ならば, PS は L -close である。 PS が L -deterministic かつ L -close ならば, L -correct であるという。

〔命題 1〕 与えられた PS が L -correct かどうかは決定可能である。

〔定義16〕 $(\bigcup_{i=1}^n V_i^{ld} - \bigcup_{i=1}^n V_i^{lr})$ の要素を r -label という。PS の全ての r -label を λ^{*5} で置換したものを L -標準形という。

〔命題2〕 PS' を PS の L -標準形とするとき、 $PS \equiv^{*6} PS'$ 。
これ以後、 PS とは L -correct, L -標準形のことを指すこととする。

5-2 G-Correctness

〔定義17〕 PS から次のプロセスによって作られる Digraph を GPS という。 $GPS = \langle \mathcal{N}, A \rangle$, $\mathcal{N}$: labeled node の finite set, A : arc の finite set.

1. 各 Statement Schema S_i に node n_i を対応づける。
2. $V_i^{lr} = \emptyset$ なる S_i の n_i から n_{i+1} へ arc を出す。
3. $V_i^{lr} \neq \emptyset$ なる S_i の n_i から、各 $L_j \in V_i^{lr}$ に対して、 $L_j \in V_k^{ld}$ なる S_k の n_k へ arc を出す。
4. Start node n_0 を設け、 n_0 から $L \in V_i^{ld}$ なる S_i の n_i へ arc を出す。 $L = \lambda$ のときは n_1 へ arc を出す。

〔定義18〕 次の Induction Definition によって定まる node の集合を、それぞれ 到達可能集合 $\mathcal{N}_S$, 停止可能集合 $\mathcal{N}_h$ という。

1. Start node $n_0 \in \mathcal{N}_S$.
2. $n_i \in \mathcal{N}_S \wedge n_i n_j \in A$ ならば $n_j \in \mathcal{N}_S$.
3. 1, 2 によって定まるものだけが $\mathcal{N}_S$ の要素である。
- 1'. Halt node $n_i \in \mathcal{N}_h$.

*5. λ : 長さゼロの記号。 *6. 全ての Interpretation のもとで強等価であること。

2'. $x_j \in N_h \wedge x_k x_j \in A$ ならば $x_k \in N_h$.

3'. 1', 2'によって定まるものだけが N_h の要素である.

ただし Halt node とは 言語 L によって定まる Halt Statement S_i に対応する node のことである.

〔命題3〕 PS が Halt Statement を含まないとき, $N_h = \phi$.

〔定義19〕 PS の GPS において, $N_s \supset N_h \wedge N_h \neq \phi$ ならば PS は G-correct, とくに $N_s \not\supset N_h$ のとき Weakly G-correct. $N_s = N_h$ のとき Strongly G-correct という.

〔命題4〕 PS が G-correct であるかどうか決定可能である.

〔定義20〕 $N - N_s$ の要素を r-node, 対応する S_i を r-Statement という. PS の全ての r-Statement を入で置換したものを PS の G-標準形, その Digraph を GPS の標準形という.

〔命題5〕 GPS が標準形ならば, それは Connected Graph である. また, G-correct な PS の GPS は Start node から Halt node への path を有する.

〔命題6〕 PS が Weakly G-correct ならば, PS は少なくとも1つ, 形式的 Infinite Loop^{*7} を含む.

証明. GPS を考える. $N_e = N_s - N_h$ とおく. 条件より $N_e \neq \phi$, $N_e \cap N_h = \phi$. だから Halt node $x_h \notin N_e$. 従って $\forall x_i (x_i \in N_e) \rightarrow$ 対して, $\text{Outdegree}(x_i) \geq 1$. かつ $x_i x_j \in A$ ならば $x_j \in N_e$. $\#N_e$: 有限 / a. E. D.

*7 Loop 内の node から Loop 外の node へ出る arc が無い Loop.

〔命題7〕 PS' を PS の G -標準形とするとき, $PS \equiv PS'$.

(L, G) -correct, (L, G) -標準形の PS は Label に関して は, 正しく望ましいものである。Control の flow は PS 内で閉じており, 形式的には, PS 内の全ての path 上を通りうる。 ** L -correct $\wedge$ G -correct の意.

§ 6. プログラムの形式的正当性(その2)

6-1 諸定義 および Notation

PS の Program Variable の集合を V_p とする。 $V_p = \bigcup_{i=1}^n (V_i^{2d} \cup V_i^{2r})$.

〔定義21〕 $X \in V_p$ が $X \in V_i^{2d} (X \in V_i^{2r})$ であるとき, 変数 X の S_i 上の defined (referred) occurrence という。また, まとめて X の S_i 上の occurrence といひ, それぞれ X^d/S_i , X^r/S_i , X/S_i と表わす。

$$\begin{aligned} EDO(S_j, X, S_i) = & (X^r/S_i \wedge X^d/S_j) \wedge \exists V (V = n_{k_1} n_{k_2} \dots n_{k_\ell} \\ & \wedge n_{k_1} = n_j \wedge n_{k_\ell} = n_i \wedge (1 \leq m < \ell \Rightarrow n_{k_m} n_{k_{m+1}} \in A) \wedge \neg \exists n_{kh} (h \neq 1 \\ & \wedge X^d/S_{kh})) \end{aligned} \quad (1.6)$$

〔定義22〕 $EDO(S_j, X, S_i)$ ならば X^d/S_j は S_i の Effective defined occurrence である。このとき, S_j は S_i を支配するといひ, $S_j \succ S_i$ と表わす。

〔定義23〕 次の Induction Definition によって定まる集合を S_i の Dominant Set $D(S_i)$ という。

1. $S_j \succ S_i$ ならば $S_j \in D(S_i)$
2. $S \in D(S_i) \wedge S' \succ S$ ならば $S' \in D(S_i)$

3. 1, 2 によって含まれるものだけが $D(S_i)$ の要素である。

つぎに. Σ を PS の Statement Schema の集合. Statement Schema S_i が. Predicate, Output, Halt Statement Schema のとき. それぞれ P_i, O_i, H_i と表わす. P_i, O_i, H_i は言語 L の D によって, 形式的に定められる.

〔定義 24〕 Output Dominant Set Σ_o , Flow Dominant Set Σ_p , Halt Set Σ_H を次のように定めるとき.

$\Sigma_o = \bigcup_{O_i \in \Sigma} (D(O_i) \cup \{O_i\})$, $\Sigma_p = \bigcup_{P_i \in \Sigma} (D(P_i) \cup \{P_i\})$, $\Sigma_H = \bigcup_{H_i \in \Sigma} H_i$
 $\Sigma - (\Sigma_o \cup \Sigma_p \cup \Sigma_H)$ の要素を r -Statement Schema という. PS の全ての r -Statement Schema を入で置換したものを S -標準形という.

〔命題 8〕 PS' を PS の S -標準形とするとき. $PS' \equiv PS$.

〔定義 25〕 PS の変数の defined occurrence の集合 W^{2d} , Effective defined occurrence の集合 W^{2e} を次のように定める.

$$W^{2d} = \bigcup_{i=1}^n \{X^d/S_i \mid X \in V_i^{2d}\},$$

$$W^{2e} = \bigcup_{i=1}^n \{X^d/S_i \mid \exists S_j \text{ EDO}(S_j, X, S_i)\}.$$

このとき. $W^{2d} - W^{2e}$ の要素を r -occurrence という. PS の全ての r -occurrence を入で置換したものを V -標準形という.

〔命題 9〕 PS' を PS の V -標準形とするとき. $PS' \equiv PS$.

明らかに. PS の (L, G, S, V) -標準形 PS' は. $PS' \equiv PS$ で. Loop 内の一部以外では. 冗長性のないものである.

6-2 Strongly Formal Correctness

〔定義26〕 GPS から次のプロセスによって作られる Digraph を dot 付 GPS. DGPS という. $DGPS = \langle \mathcal{N}', A', D \rangle$, $\mathcal{N}' = \mathcal{N}$. $A' \supseteq A$, D : dot の finite set.^{*8}

1. $\text{Indeg}(\mathcal{N}_i) \geq 1$ の node に in-dot を. $\text{Outdeg}(\mathcal{N}_i) \geq 1$ の node へ out-dot を与える.
2. in-dot から node へ, node から out-dot へ arc を引く.
3. GPS 中の $\mathcal{N}_i$ への in-arc を $\mathcal{N}_i$ の in-dot へ, $\mathcal{N}_i$ からの out-arc を out-dot から書く.
4. $\text{Indeg}(d_i) = \text{Outdeg}(d_i) = 1$ の dot d_i を消去し. in-arc, out-arc を 1 本にまとめる.

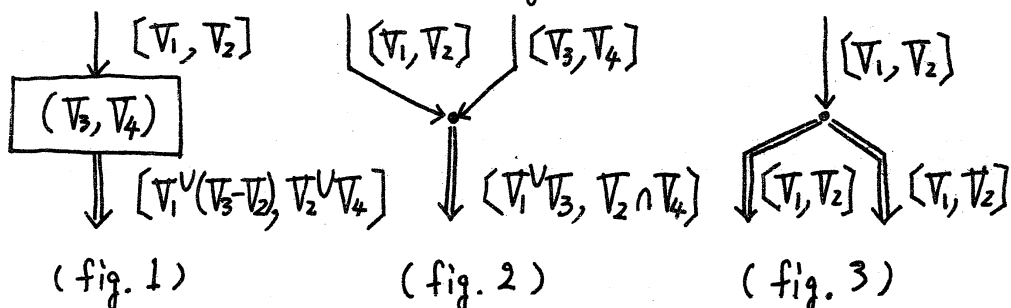
〔定義27〕 node $\mathcal{N}_i$ に対応する $S_i = \langle F_i, V_i^{id}, V_i^{ir} \cup C_i^r, V_i^{2d}, V_i^{2r} \cup C_i^2 \rangle$ によって定まる ordered pair (V_i^{2r}, V_i^{2d}) を $\mathcal{N}_i$ の Node 表現と呼ぶ.

〔定義28〕 次の規則によって定まる ordered pair を. 対応する arc の Arc 表現と呼ぶ.

1. Start node の out-arc: $\langle \phi, \phi \rangle$
2. Node 表現 (V_3, V_4) の node の in-arc の Arc 表現を $\langle V_1, V_2 \rangle$ とするとき. その out-arc の Arc 表現: $\langle V_1 \cup (V_3 - V_2), V_2 \cup V_4 \rangle$
3. Arc 表現 $\langle V_1, V_2 \rangle, \langle V_3, V_4 \rangle$ の 2 つの arc の joint arc の Arc 表現: $\langle V_1 \cup V_3, V_2 \cap V_4 \rangle$

*8 対応する Statement Schema を持たない node を区別して dot と呼ぶ.

4. disjoint arc の Arc 表現は、その dot の in-arc の Arc 表現と同じ。
規則の 2, 3, 4 をそれぞれ, fig. 1, 2, 3 に示す。



〔命題 10〕 DGPS の全ての node と arc に Node 表現 Arc 表現が定まる。

Node 表現 (V_1, V_2) の意味は、この Statement Schema 以前で、 V_1 の値が定められていけば、この Statement Schema で V_2 の値が定まる。すなわち、この Statement Schema の全ての変数 V_i^2 の値が定まるということである。Arc 表現 $[V_1, V_2]$ の意味は、 V_1 は Start node からこの arc へ到達しうる全ての path 上の全ての node (Statement Schema) の変数の値が、その値が定められると定義可能になるような最小集合、 V_2 はその内のどの path を通って来ても、その値が定義されているような変数の最大集合を表わすということである。

〔定義 29〕 DGPS において、Half Statement Schema H_i (Output Statement Schema O_i) に対応する n_i の In-Arc 表現^{*9}を $[V_1, V_2]$ とするとき、全ての $H_i(O_i)$ の In-Arc 表現において $V_1 = \emptyset (V_2 \supseteq V_i^{2n})$ であるとき、PS は $I(0)$ -close であるという。

*9 in-arc の Arc 表現の意味。

〔定義30〕 (L, G) -correct な PS が (I, O) -Close であるとき、PS は Strongly Formal Correct であるという。

6-3 Weakly Formal Correctness

6-2 Strongly Formal Correctness における Arc 表現に関する〔定義28〕を次のように変更する。

〔定義31〕 z_i を $[v_i, v'_i]$ を表わすものとするとき、Arc 表現を次のように定める。

1. Start node の out arc : $\{[\phi, \phi]\}$
2. Node 表現 (V, V') の node の In Arc 表現を $\{z_1, z_2, \dots, z_n\}$ とするとき、その Out Arc 表現は $\{z'_1, z'_2, \dots, z'_n\}$ である、
ただし、 $z'_i \equiv [v_i \cup (V - v'_i), v' \cup v'_i]$
3. Arc 表現 $\{z_1, z_2, \dots, z_n\}, \{z'_1, z'_2, \dots, z'_m\}$ の2つの arc の joint arc の Arc 表現は $\{z_1, z_2, \dots, z_n, z'_1, z'_2, \dots, z'_m\}$ である。
4. disjoint arc の Arc 表現は、その dot の in-arc の Arc 表現と同じ。

〔定義32〕 DGPSにおいて、Halt (Output) Statement Schema $H_i(O_i)$ の In Arc 表現を $\{z_{i1}, z_{i2}, \dots, z_{in}\}$, $z_{ij} \equiv [v_{ij}, v'_{ij}]$ とするとき、全ての $H_i(O_i)$ の In Arc 表現において、 $v_{ij} = \phi$ ($v_{ij} = \phi \wedge v'_{ij} \geq v_i^{2r}$) なる z_{ij} が少なくとも1つ存在するとき、PS は Weakly $I(O)$ -Close であるという。

〔定義33〕 (L, G) -correct な PS が Weakly O -Close であるとき、PS は Weakly Formal Correct であるという。

〔命題11〕 PS が Strongly or Weakly Formal Correct であるかどうかは決定可能である。

§7. 一般的正当性と形式的正当性の関係.

全てのプログラムの集合: Ω , 全ての入力に対して停止するクラス: Ω_{VF} , 停止する入力が存在するクラス: $\Omega_{\exists F}$, 同一入力に対し同一の result (undefine も含む) を得るクラス: Ω_S とする。

〔命題12〕 Totally (partially) correct なプログラムのクラスは、 $\Omega_{VF} \cap \Omega_S$ ($\Omega_{\exists F} \cap \Omega_S$) である。

全てのプログラム形の集合: Π , (L, strongly (weakly) G)-correct なクラス: $\Pi_{sq} (\Pi_{wg})$, Strongly (Weakly) (IO)-Close なクラス: $\Pi_{sc} (\Pi_{wc})$, Strongly (Weakly) Formal Correct なクラス: $\Pi_{sfc} (\Pi_{wfc})$ とする。

〔命題13〕 $\Pi_{sfc} = \Pi_{sc} \cap (\Pi_{sq} \cup \Pi_{wg})$, $\Pi_{wfc} = \Pi_{wc} \cap (\Pi_{sq} \cup \Pi_{wg})$.

Interpretation を I , interpretation を与えられたプログラム形 PS を $\langle PS, I \rangle$ と表わす。

〔定理1〕 $PS \in \Pi_{sc}$ ならばかつそのときのみ $\forall I (\langle PS, I \rangle \in \Omega_S)$ である。また $PS \in \Pi_{wc}$ ならばかつそのときのみ $\exists I (\langle PS, I \rangle \in \Omega_S)$ である。

〔定理2〕 $\forall I (\langle PS, I \rangle \in \Omega_{VF})$ ならば $PS \in \Pi_{sq}$ である。

また $\exists I (\langle PS, I \rangle \in \Omega_{\exists F})$ ならば $PS \in \Pi_{wg} \cup \Pi_{sq}$ である。

〔定理3〕 $\forall I (\langle PS, I \rangle \in \Omega_{\text{totally correct}})$ ならば $PS \in \Pi_{sfc}$ である。また $\exists I (\langle PS, I \rangle \in \Omega_{\text{partially correct}})$ ならば $PS \in \Pi_{wfc}$

である。

§ 謝 辞

日頃ご指導いただき、京大工学部 清野 武教授 なさむに
矢島脩三教授 に深謝いたします。

参考文献

1. Church, A., (1956). "Introduction to Mathematical Logic", Princeton
univ. press, vol. 2.
2. Church, A., (1941), "The Calculi of Lambda-Conversion", Princeton
univ. press.
3. Bakker, J. W. (1969), "Semantics of Programming Languages",
Advances in Inf. Sys. Sci. (Tou, J. T., ed), vol. 2.
4. Manna, Z. (1969), "The Correctness of Programs", J. of
Computer and System Science 3, pp. 119-126
5. Engeler, E. ed (1971), "Symposium on Semantics of Algorithmic
Languages", Lecture Notes in Math. 188 Springer-Verlag.
6. 相沢輝昭 (1970), "計算理論の基礎", 統合図書.
7. 謝. 清野, (1971). "λR²の正当性について", 電気関係学会
関西支部連合大会.
8. Harary, F., (1969), "Graph Theory" Addison-wesley.