言語モデル論（４）

--- 言語と算法の基本性質と帰納的関数 その2---

限界付最小化
Bounded Minimalization

述語p($\vec{x}$,y)に対して

$\mu_{z<y}p(\vec{x},z) = \min(\{z \in N | z < y かつ p(\vec{x},z)\} \cup \{Y\})$

と

おく。 → →

すなわち$\mu_{z<y}p(x,z)$は与えられた自然数の組x,yに対して、y未満でp(x,z)を満たす自然数zがあればそのような最小のzを、もしなければyを値として返す関数である(例えば、$\mu_{z<y}(x < (z+1)^2)$は$\sqrt{x}$の整数部分を表す)

Nプログラム

- 変数の生成
- 式を表現する関数は以下の三つする。
  零関数　　zero: $N^0 \rightarrow N$　ただし zero( ) = 0
  後者関数　succ: $N \rightarrow N$　ただし succ(x) = x+1
  射影関数 P_i^n: $N^n \rightarrow N$　ただし $P_i^n(x_1,x_2,\ldots,x_n) = x_i$,
  $1 \leqq i \leqq n$ は原始的関数である
- 変数への代入命令
- Goto命令
- If-then-else、
- 入力命令、出力命令

基本的にこの三つの命令を使うプログラムは、コンピュータで実行できるアルゴリズム（のすべて）を記述出来ると考えられる。

補題と定理(1)

補題：述語 $p:N^{n+1}\rightarrow$ {真, 偽} が原始的ならば、
$\mu_{z<y}p(x,z)N^{n+1}\rightarrow N$ は原始的関数である。

定理：原始的関数はNプログラムで計算することができる。

証明　全ての原始的関数は、入力変数の値を最後まで保存するNプログラムによって計算できることを、原始的関数の構成に関する帰納法で証明する。 ← 課題！

Nプログラムでは変数、定数、四則演算がはじめから許されていた。

Ackermann関数(1)

問: (Ackermann関数)　関数$f:N^2 \to N$を次のように定義する。

$$\begin{cases} f(0, y) = y+1, \\ f(x,y+1) = f(x, 1), \\ f(x+1, y+1) = f(x, f(x+1, y)). \end{cases}$$

(1) 全ての自然数の組x,yに対してf(x,y)の値　が定義されることを確かめよ。

(2) f(1,y) = y+2, f(2,y) = 2×y+3 であることを示せ。同様に、f(3,y)とf(4,y)をyの関数として具体的に表せ。

(3) Nプログラムを使って、f(x,y)を計算する方法について考えよ。

Ackermann関数(2)

難問：前問の関数f について、次の(1)~(4)を順に示し、その結果を使ってf が原始的関数でないとこを証明せよ。

(1) $x+y+1 \leqq f(x, y)$.

(1) $f(x, y) < f(x, y+1) \leqq f(x+1, y)$

(したがってf はxとyに関してそれぞれ強い意味 で単調増加である)

(3) 任意のa, b≧0に対して

$$f(a, f(b, y)) < f(c, y) \quad (y \in N)$$

を満たすc≧0がある。

(4) $g:N^n \to N$ が原始的関数ならば、あるc≧0に対して

$$g(x_1,x_2,\ldots,x_n) < f(c, \max(x_1,x_2,\ldots,x_n))$$

$$(x_1,x_2,\ldots,x_n \in N)$$

が成り立つ（ただし n=0のとき $\max(x_1,x_2,\ldots,x_n) = 0$とする)

帰納的関数

- 前述の定理の逆は成立しない。即ち、*N*プログラムで計算できるが原始帰納的でない関数が存在する。（例:Ackermann関数）
- 以下、原始帰納的関数を拡張して、*N*プログラムで計算できる関数の族と一致するものを与える。
- そのような関数の族を帰納的関数（の族）と呼ぶ。

最小解関数

述語　$p:N^{n+1}\rightarrow${真, 偽}に対して

$$f(\vec{x}) = \begin{cases} p(\vec{x},y) = \text{真} & \text{となる自然数yがあるとき、その最小値} \\ p(\vec{x},y) = \text{真} & \text{となる自然数yがないとき、未定義} \end{cases}$$

によってN^nからNへの部分関数(すなわち、定義域がN^nのある部分集合である関数) f が定義できる。

このf をp($\vec{x}$,y)の最小解を与える関数という意味で、p($\vec{x}$,y)の最小解関数とよび、$\mu_y(p(\vec{x},y))$で表す。

最小化は「探索」に対応する

定義

定義：帰納的関数(recursive function)を次の(1)~(4)により再帰的に定義する。

(1) 原始的関数は帰納的関数である。

(2) 原始的述語の最小解関数は帰納的関数である。

(3) 帰納的関数の合成関数は帰納的関数である。

(4) 帰納的関数から原始帰納法によって得られる関数は帰納的関数である。

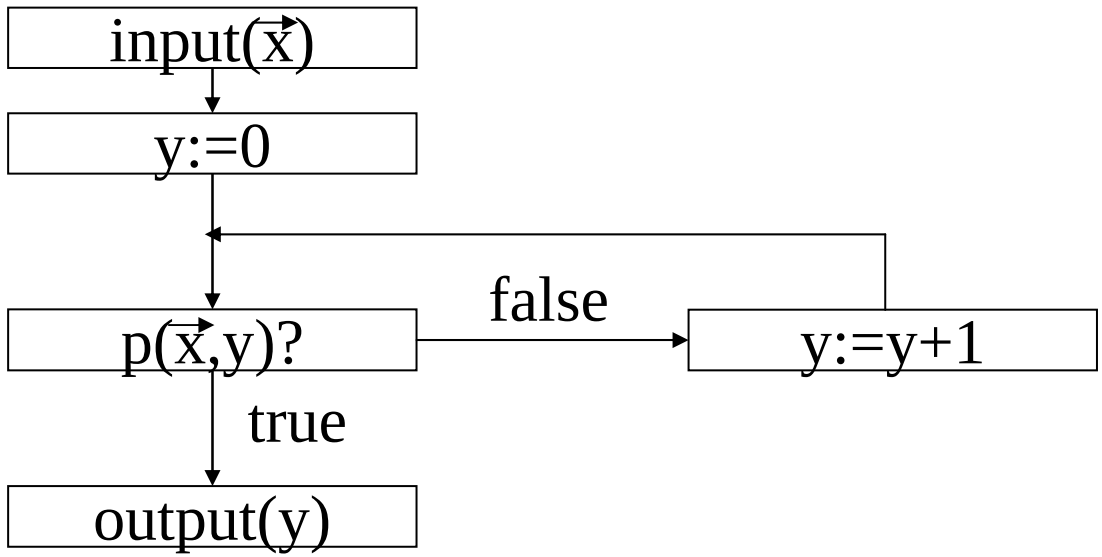
定理

定理：任意の帰納的関数はNプログラムで計算することができる。

証明　帰納的関数fの構成に関する帰納法により、f を計算するNプログラムで入力値を保存するものがあることを示す。

(1)　f が原始的関数の場合は前述の定理が使える。

(2)　$f(\vec{x}) = \mu_y(p(\vec{x},y))$ (ただし pは原始的述語)のとき、f は次のNプログラムで計算できる。



(3)　合成関数および(4)原始帰納法で定義される関数の場合は、それぞれ前述の定理の証明が適用できる。

前述の定理：原始的関数はNプログラムで計算できる

帰納的関数族 ＝ Nプログラムの族

- 前の定理の逆、即ちNプログラムで計算できる関数は、帰納的関数であることを証明できる。(ここでは、証明はしない)
- 証明には、自然数列のゲーデル算術化などの概念が必要となる。

関数$G:N^m \to N$ (ただし $m \geqq 0$) が1対1関数のとき、すなわち

$$G(x_0, x_1,\ldots,x_{m-1}) = G(y_0, y_1,\ldots y_{m-1})$$

$$\Longleftrightarrow \quad x_i = y_i \quad (i = 0, 1,\ldots,m-1)$$

のとき、$G(x_0,x_1,\ldots,x_{m-1})$は数列 $x_0,x_1,\ldots,x_{m-1}$ を一つの数で表したもの(一種のコード)とみなすことができる。このGに対してさらに

$$G_i(G(x_0, x_1,\ldots,x_{m-1})) = x_i$$

を満たす関数 $G_i:N\to N$ $(i=0,1,\ldots,m-1)$ があれば、これらの関数を使って $G(x_0,x_1,\ldots,x_{m-1})$ から元の列 $x_0,x_1,\ldots,x_{m-1}$ が復元できる。

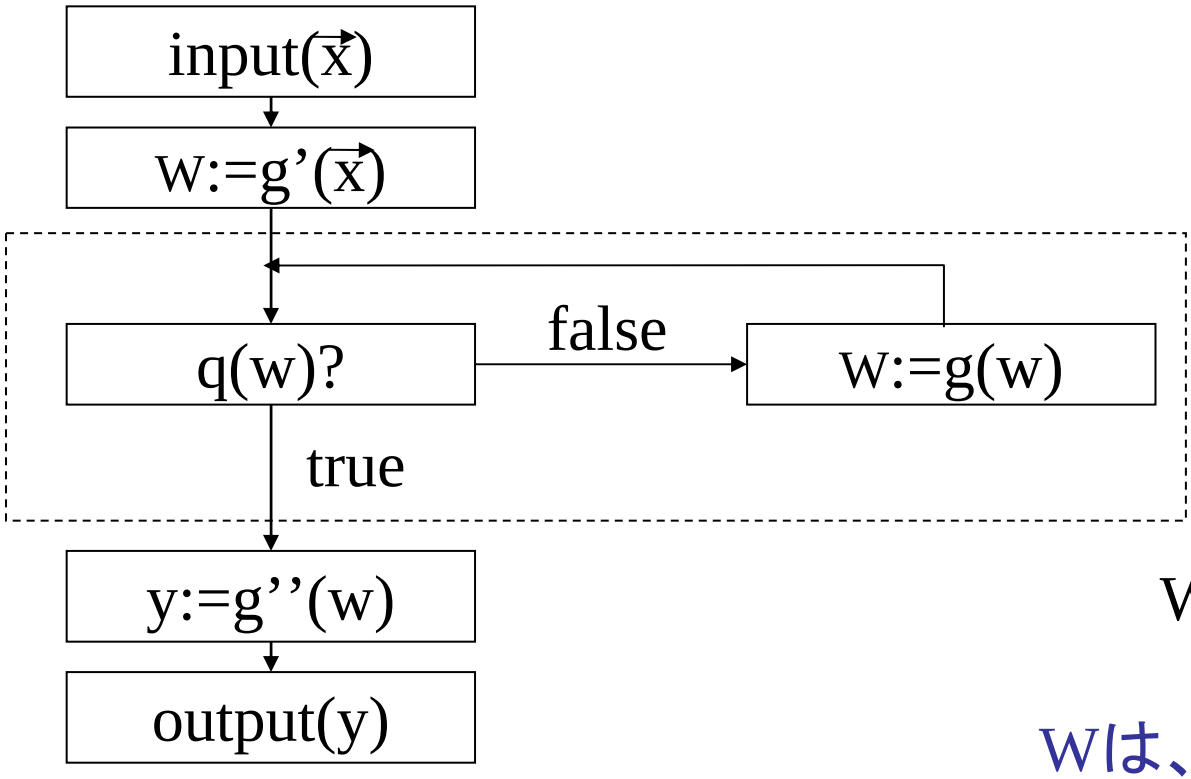
そのようなGおよび$G_0,G_1,\ldots,G_{m-1}$ が原始的関数のとき、Gを(長さmの自然数列の)ゲーデル関数、$G_0,G_1,\ldots,G_{m-1}$をその逆関数とよぶ。また、$G(x_0,x_1,\ldots,x_{m-1})$を数列 $x_0,x_1,\ldots,x_{m-1}$ のゲーデル数と

証明の概要（雰囲気）

1. Ｎプログラムで計算できる関数は、必ず図６（次スライドにあり）の形のプログラムで計算できる。

2. 図６の形のNプログラムで計算される関数は、帰納的関数であることを示す。

図6



wの初期値設定

wに対する計算

wから出力値を求める

wは、全てのプログラム変数の値を算術化して記憶しておくための変数

- g, g', g''は原始的関数
- q　は原始的述語

← これらはあとで定義

1. 任意の*N*プログラム*P*に対して、まず f_Pを計算する*N*プログラム*P*’で図7のものが構成できることを示す。（Nプログラムの変換）

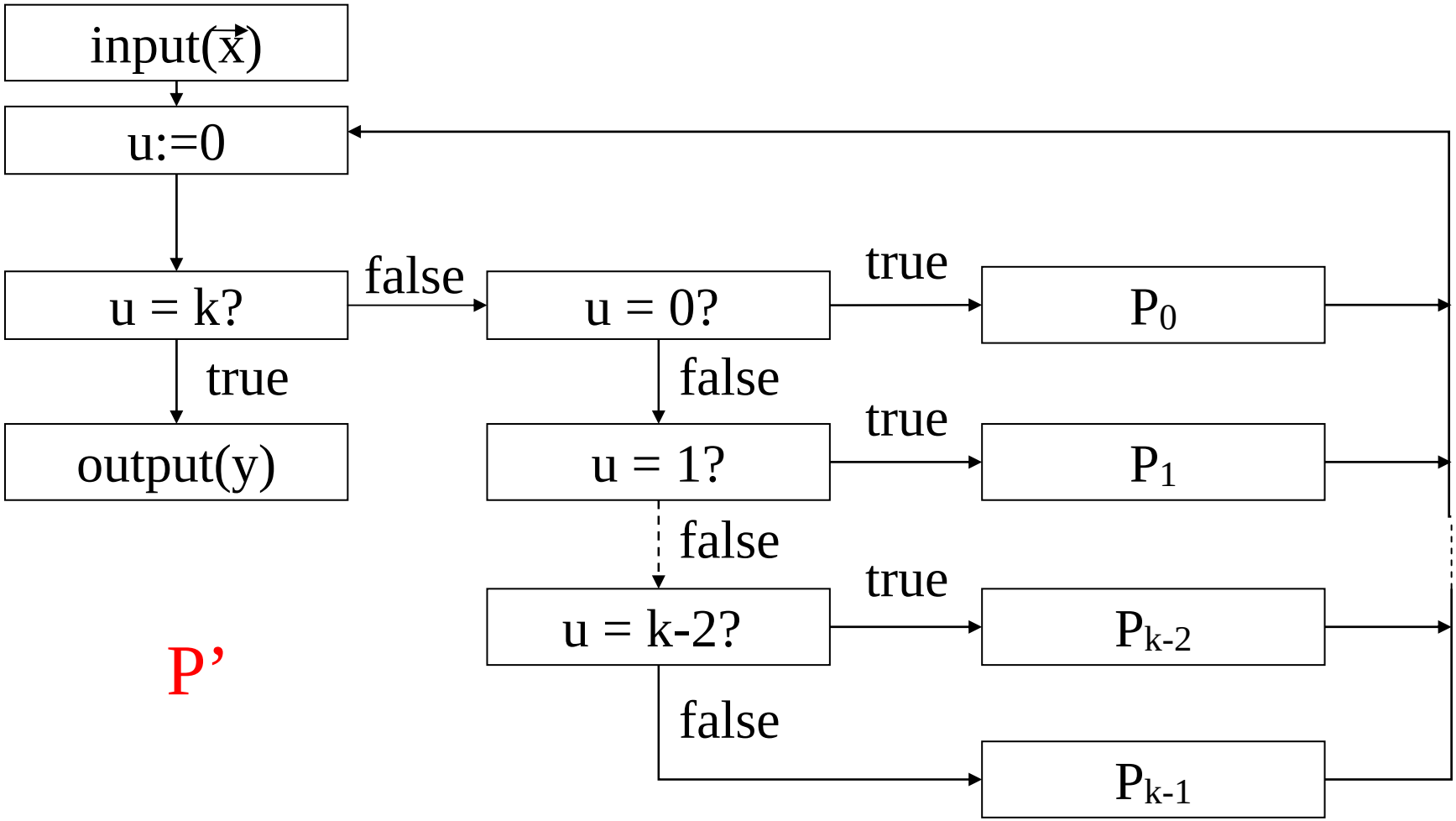
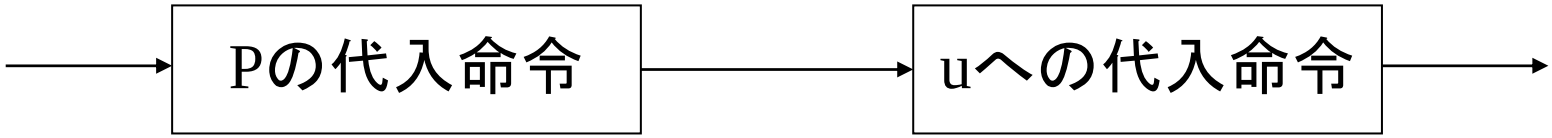
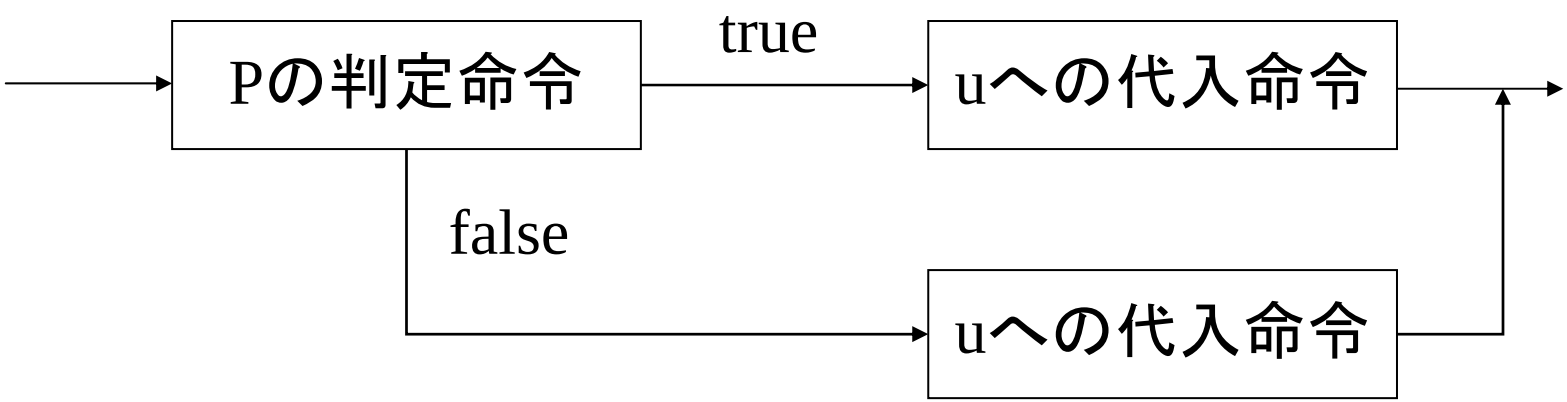


図7

uは、元のプログラムの実行ポイントに対応するプログラム・カウンタの目的で新しく導入する変数、kは定数(Pの出力命令の番号)、各P_iは



か、または



•プログラム*P'*（図7）は*P''*（図8）の形に書き換えられる。

この形は帰納関数で
表現できることは明白。

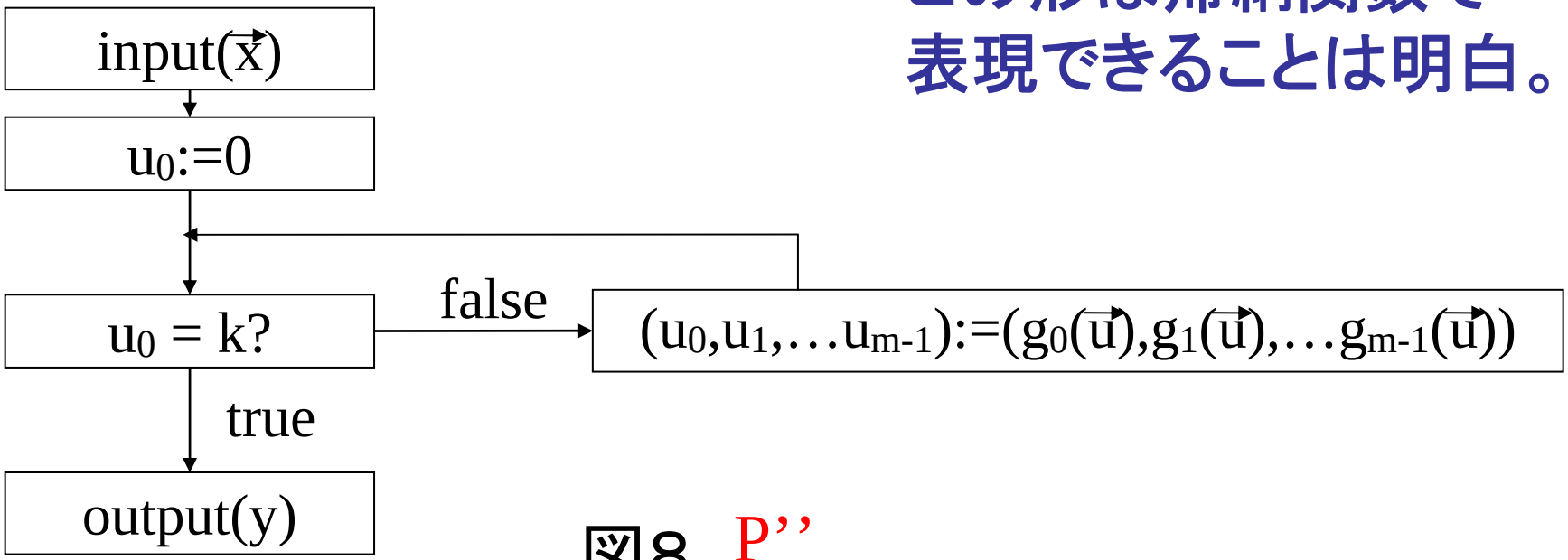


図8　P''

図8は、図6の形をしている。今$G:N^m \to N$を長さmの数列に対するゲーデル関数とすると、補題1.4.5(次スライドに定義あり)により

$$g(G(\vec{u})) = G(g_0(\vec{u}),g_1(\vec{u}),\ldots,g_{m-1}(\vec{u})) \quad (\vec{u} \in N^m)$$

を満たす原始的関数$g:N \to N$が存在する

P''の基本アイデアは、変数***w***に，　***P'***の全変数$u_0(=u),u_1,u_2,\ldots,u_{m-1}$の値のゲーデル数を記憶させること。

補題1.4.5

補題1.4.5 $G:N^m \to N$がゲーデル関数で、$f_i:N^m \to N (i=0,1,\ldots,m-1)$ が原始的関数のとき、次の条件を満たす原始的関数$f:N \to N$がある。

$$f(G(x)) = G(f_0(x),f_1(x),\ldots,f_{m-1}(x)) \quad (x \in N^m)$$

すなわち、fは数列xのゲーデル数を数列 $f_0(x),f_1(x),\ldots,f_{m-1}(x)$ のゲーデル数に変換する関数である。

- この関数 g と、以下に定義する g', g'',および
  述語 q を使って、プログラムP''を図6の形に構成する。

（ 但し、関数g, g', g'',および述語 q は、図6で使われた。）

$$g'(\vec{x}) = G(0, \vec{x}, 0, 0, \ldots, 0)$$

$$q(w) = \text{真} \overset{\text{def}}{\Longleftrightarrow} \quad G_0(w) = k$$

$$g''(w) = G_l(w)$$

ただし、G_i(i=0,1,…,m-1) はゲーデル関数Gの逆関数を表す。Gおよび各G_iは原始的関数であるから、関数g’,g’’および述語qは原始的である。

Kleene標準形定理

- 系: $N^{(n)}$からN への任意の部分関数$f^{(n)}$ について、$f^{(n)}$が Nプログラムで計算されるための必要十分条件は$f^{(n)}$が帰納的関数であること。

- 系(Kleeneの標準形定理) 全ての帰納的関数は適当な原始帰納的関数 f と原始帰納的述語 p を使って

$$f\ (x\ , \mu z\ (\ p(x,\ z)))$$

の形に表される。即ち、すべての帰納的関数は、原始帰納的関数と(原始的述語の)最小解関数の合成によって得られる。