直観主義自然数論とGödel Interpretationに基づくプログラム合成のLiST処理への拡張

水上達就（北大工学部）

はじめに。

プログラムの仕様からプログラムを抽出することを目的として各種のプログラム合成法が研究されてきた。直観主義自然数とGödel Interpretation に基づくプログラム合成の方法は高須氏［1］，佐藤氏［2］，後藤氏［3］によって研究されている。このプログラム合成法は次のように定義されている。すなわち、xを input vector，yを output vector とし、$\varphi(x)$を input 述語，$\psi(x, z)$を output 述語とし、プログラムの仕様を$\forall x(\varphi(x) \supset \exists z\, \psi(x, z))$とする。この仕様を直観主義自然数論によって証明し、その証明から Gödel Interpretation を用いて$\varphi(x)$が成立する時$\psi(x, f(x))$を満す関数$f(x)$を抽出する。

このプログラム合成法は直観主義自然数論によって仕様の証明を行う為処理するData type は自然数のみに限定されている。本論文はLiST構造を処理可能となるようにプログラム合成法を拡張し、さらに Guttag氏［4］，Goguen氏［5］等により研究されているAbstract Data typeによるプログラム作成の手法をこのプログラム合成法に取入れ各種 Data-type を使用したプログラム合成を可能とする方法の提示を目的とする。

LiST構造は自然数上で定義するが各自然数を記号と見なすことで記号上のLiST構造を考えることができる。LiST構造処理の為の拡張はLiST構造およびその上での operation を Data-typeの operationを使った公理化と同じように公理化し直観主義自然数論に付加しLiST構造および operationを帰納的関数によって解釈しGödel Interpretation によるLiST構造を持つ直観主義自然数論の解釈を可能とすることで行っている。また、Gödel Interpretationにより解釈可能となることで Schütte 氏［5］の直観主義自然数論無矛盾定理と同様にLiST構造を持つ直観主義自然数論は無矛盾であることが証明できる。

Data-typeによるプログラム作りはTop-DownでData-typeを考えそれを使ってプログラムを記述し更にその時使用したData-typeをより詳細なData-typeを作りそれで記述しこれを繰返して計算機で実行可能なData-typeにまで行いプログラムとすることである。また、このData-typeによる記述は Guttag氏等の述べるようにData-type による証明であり、implementationである。Abstract-Data-typeはこの各種Data-type の数学的記述による共通項であり、これらを設定することにより各種Data-typeの定義を容易ならしめている。

プログラムのData-typeによる記述が証明というのは上記プログラム合成の考え方と同一である。無論、証明の仕方で意味論的とsyntactics的の相違があるが証明の基本的考えは同じである。従って、LiST構造を付加するのと同様にAbstract-Data-typeをLiST構造上に構成することで各種Data-type を使ったプログラム合成が可能となる。以下でこのことをDutch National FLagの問題を解くことにより説明する。

定義

直観主義自然数論として高須氏の［1］におけるGNJを用いる。以下に定義す

るLiST構造をGNJに加えたシステムをGNJ+Lとする。さらに、Gödel Interpretationを記述するものとしてSchütte氏[6]のFT，QFTを用いる。

LiST構造

LiST項によりLiSTを構成しcar，cdr，cons，Ord $a_1 \cdots a_n$（要素が自然数 a_1，…，a_nで作られるLinear LiST上の順序関数），$Ord^{-1} a_1 \cdots a_n$（Ord $a_1 \cdots a_n$の逆関数），$\chi_{t=}$（LiST上の＝についての特性関数）なるLiST上のoperationを公理化する。さらに、Manna，Waldinger両氏の[7]で用いられた（Recursive LiST Induction）をLiST構造上のInductionとして導入する。

（イ）LiST項は次の帰納的定義で与えられる。

i) Nil はLiST項である。

ii) $t_1, \cdots, t_n$をGNJの項とすると、list$(t_1, \cdots, t_n)$はLiST項である。

iii) $l_1, \cdots, l_n$をGNJの項又はLiST項とするとすると、list$(l_1, \cdots, l_n)$はLiST項である。

（ロ）operationの型

D_LをLiST項全体の集合，D_NをGNJの項全体の集合，$D(a_1, \cdots, a_n)$を要素が自然数$a_1, \cdots, a_n$で作られるLinear LiST項全体の集合とする。

carの型；$D_L - \{Nil\} \to D_N \cup D_L$

cdrの型；$D_L - \{Nil\} \to D_L$

consの型；$(D_N \cup D_L) \times (D_N \cup D_L) \to D_L$

Ord $a_1 \cdots a_n$ の型；$D(a_1, \cdots, a_n) \to D_N$

$Ord^{-1} a_1 \cdots a_n$ の型；$D_N \to D(a_1, \cdots, a_n)$

$\chi_{t=}$ の型；$D_L \times D_L \to \{0, 1\}$

operationはLiSTの長さおよびGNJの項に対し個々に存在しその合成は上の型に従って行なわれるとする。

（ハ）公理．$l_1, \cdots, l_n$をGNJの項又はLiST項とし、t_1, t_2をGNJの項とする。

i) $\to$ car$($list$(l_1, \cdots, l_n)) = l_1$

ii) $\to$ cdr$($list$(l_1)) = Nil$

iii) $\to$ cdr$($list$(l_1, \cdots, l_n)) =$ list$(l_2, \cdots, l_n)$ $\quad n \geq 2$

iv) $\to$ CONS$(l_1, Nil) =$ list(l_1)

v) $\to$ CONS$(t_1, t_2) =$ list(t_1, t_2)

vi) $\to$ CONS$(l_1,$ list$(l_2, \cdots, l_n)) =$ list$(l_1, \cdots, l_n)$

vii) $\to l_1 = l_2 \supset \chi_{t=}(l_1, l_2) = 0$

viii) $\to \neg l_1 = l_2 \supset \chi_{t=}(l_1, l_2) = 1$

（ニ）（Recursive LiST induction）

Qを量記号と束縛変数の列とする。

$$\frac{\to Q \cdot P[Nil] \;;\; Q \cdot \forall u_1 \cdots u_n P[cdr(list(u_1, \cdots, u_n))] \to Q \cdot \forall u_1 \cdots u_n P[list(u_1, \cdots, u_n)]}{\to Q \cdot \forall u_1 \cdots u_t P[list(u_1, \cdots, u_t)]}$$

tは任意のGNJの項である。

上記LiST構造は list$(t_1, \cdots, t_n)$を$P_0^{t_1+1} \cdots P_{n-1}^{t_n+1}$（$P_i$は$(i+1)$番目の素数を表わす帰納的関数）と解釈することで、各LiSTの長さとGNJの項ごとに、car，cdr，consが公理を満すように帰納的関数で解釈でき、$\chi_{t=}$も帰納的関数$\chi_=$で解釈され、Ord $a_1 \cdots a_n$，$Ord^{-1} a_1 \cdots a_n$も定義を満すように解釈できる。

プログラム合成

プログラム仕様がＧＮＪのFormula $\forall x(\varphi(x) \supset \exists z\, \psi(x, z))$ によって与えられこの証明がＧＮＪで為されるとする。ここで用いるプログラム合成は仕様のFormulaの証明からGödel Interpretationを用いて仕様満すプログラムを抽出することでありその方法を以下で説明する。

$\forall x(\varphi(x) \supset \exists z\, \psi(x, z))$ がＧＮＪにおいて証明可能よりEnd Sequent $\rightarrow \forall x(\varphi(x) \supset \exists z\, \psi(x, z))$ なるＧＮＪの証明図が作成可能である。また、このEnd SequentのGödel Interpretationは $\rightarrow \varphi_*(x) \supset \psi_*(x, bx)$ （φ_*，ψ_* は φ，ψ より量記号を取除いたものである。）となり、Gödelの定理によりすべての x に対して $\rightarrow \varphi_*(x) \supset \psi_*(x, bx)$ がＦＴにおいて証明可能となるＦＴの項 bx が構成的に与えられる。ＦＴの項は計算可能関数であり、ＦＴでの述語は自然数の述語として解釈できることより $\varphi_*(x)$，$\psi_*(x, bx)$ と $\varphi(x)$，$\psi(x, z)$ の述語としての解釈は同一とすることができる。従って、bx は仕様を満す計算可能関数すなわちプログラムとなる。さらに、bx は仕様の証明図のGödel Interpretationに出現する各種の推論規則ごとにGödelの定理の証明で用いられている項の合成によって構成される。

上記の方法はＧＮＪ+Ｌにおいても可能となる。ＬｉＳＴおよびＬｉＳＴ上のoperation car，cdr，cons，Ord $a_1 \cdots a_n$，Ord' $a_1 \cdots a_n$，$X_t =$ は帰納的関数によって公理および定義を満すように解釈できる。従って、それらはＦＴの項とみることができその公理もＦＴで証明可能となる。さらに、（Recursive ＬｉＳＴ induction）は次のように考えると他の推論規則によって代用可能である。

$$\dfrac{\dfrac{\rightarrow Q\cdot P[Nil]\ ;\ Q\cdot P[Nil] \rightarrow Q\cdot \forall x_t P[cdr(list(x_t))]}{\rightarrow Q\cdot \forall x_t P[cdr(list(x_t))]}\ ;\ Q\cdot \forall x_t P[cdr(list(x_t))] \rightarrow Q\cdot \forall x_t P[list(x_t))]}{\rightarrow Q\cdot \forall x_t P[list(x_t))]} \quad \text{-----}\ (\alpha)$$

$$\dfrac{Q\ ;\ \dfrac{\dfrac{P[list(a_t)] \rightarrow P[cdr(list(a_{t-1}, a_t))]}{\forall x_t P[list(x_t)] \rightarrow \forall x_{t-1} x_t P[cdr(list(x_{t-1}, x_t))]}}{Q\cdot \forall x_t P[list(x_t)] \rightarrow Q\cdot \forall x_{t-1} x_t P[cdr(list(x_{t-1}, x_t))]}}{\rightarrow Q\cdot \forall x_{t-1} x_t P[cdr(list(x_{t-1}, x_t))]}$$

$$\vdots$$

$$\dfrac{\qquad\qquad \text{（上記操作を } t-1 \text{ 回行う）}}{\rightarrow Q\cdot \forall x_1 \cdots x_t P[cdr(list(x_1, \cdots, x_t))]}$$

従って、推論規則がＧＮＪと同じになり、またGödelの定理が各推論規則についてＦＴの項を構成することで証明されていることよりＧＮＪ+Ｌの場合にもGödelの定理が成立し上記のプログラム合成を実行できる。

ＬｉＳＴ処理の上記方法によるプログラム合成例として、上の（Recursive ＬｉＳＴ Induction）を使用した場合のプログラム合成を行う。

$Q\cdot \forall x_1 \cdots x_t P[list(x_1, \cdots, x_t)]$ を $\forall x \exists z \forall x_1 \cdots x_t R[z, x, list(x_1, \cdots, x_t)]$（Ｒを量記号を含まない述語とする。）とする。この時、output vectorを z として z に相当する項をプログラムとして抽出する。

$\rightarrow \forall x \exists z R[z, x, Nil]$ のGödel Interpretationは $\rightarrow R[R_0 x, x, Nil]$，

$\rightarrow \forall x \exists z \forall x_k \cdots x_t R[z, x, list(x_k, \cdots, x_t)]$ のGödel Interpretationは

$\rightarrow R[R_{t-k+1} x, x, list(x_k, \cdots, x_t)]$，

$\forall x \exists z \forall x_{k-1}, \cdots, x_t\ R[z, x, cdr(list(x_{k-1}, \cdots, x_t))] \longrightarrow$
$\forall x \exists z \forall x_{k-1}, \cdots, x_t\ R[z, x, list(x_{k-1}, \cdots, x_t)]$ の Gödel Interpretation は $R[\bar{k}x, \bar{a}_x \bar{k} x_{k-1} \cdots x_n, cdr(list(\bar{a}_{k-1} \bar{k} x_{k-1} \cdots x_n, \cdots, \bar{a}_n \bar{k} x_{k-1} \cdots x_n))] \rightarrow R[b_{n-k+1}\bar{k}, x, list(x_{k-1}, \cdots, x_n)]$ と表現できる。このGödel Interpretationで求めるプログラムは $k_t x$ である。この $k_t x$ は上記証明のGödel Interpretation中の項より次のように合成される。

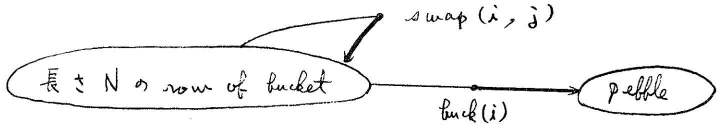
$$k_t x \leftarrow \textit{if}\ list(u_1, \cdots, u_t) = Nil \quad \textit{then}\ k_0 x \quad \textit{else}\ b_{t-1} k_{t-1} x \ .$$

これはManna，Waldinger両氏が[7]で述べた(Recursive LiST induction)を用いた時のプログラム図式と同じものであり、この合成が[7]で与えられる(Recursive LiST Induction)のプログラム図式が妥当であることを示している。このように上記方法によるプログラム合成は各種プログラム合成の理論的基礎付を与えると同時に種々の証明形式によりプログラム図式を作ることで新しいプログラム合成を可能とするものである。

次の節ではData-typeとの関連をDijkstra氏が[8]で解いているDutch National Flagの問題の解き方にならい上記方法によりDutch National Flagプログラム合成を行なうことにより説明する。

Data-type処理について.

Dijkstra氏が[8]で行なっているDutch National Flag作成は次のようなData-typeを設定することから始まっている。



$1 \leqq i, j \leqq N$

上の図はGoguen氏等が[6]で用いる記法を使用している。矢の尾部がoperationの定義域，頭部がoperationの結果得られるものを示し、•の上にoperation名を付けている。

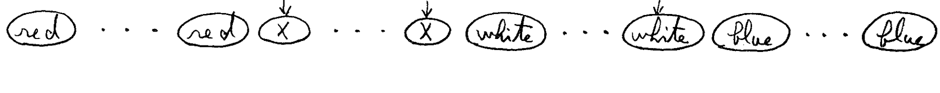
上で定義したプログラム合成を行う為に、このData-typeをLiSTの時と同様に公理化する。また、公理化によってData-typeを表現するというのはAbstract Data typeの記述法である。

公理としては次のものを設定すれば問題の解を得るのに充分である。

αを長さNのrow of bucketでbucketの中にはred，white，blueいずれかのpebbleが入っているとする。

$$\neg(k=i),\ (k=j) \longrightarrow buck(k)(swap(i,j)\alpha) = buck(k)\alpha \ ,$$
$$\longrightarrow buck(i)(swap(i,j)\alpha) = buck(j)\alpha \ ,$$
$$\longrightarrow buck(j)(swap(i,j)\alpha) = buck(i)\alpha \ .$$

次にDijkstra氏は[8]で長さNのrow of bucketの状態をER(established red)，X(as yet uninspected)，EW(established white)，EB(established blue)なるpebbleの並びになっている並びかえの途中の場合として想定し今後の処理を考えている。これは次のような図で表示できる。



上のER，EW，EBは図の r, w, b と長さNの row of bucket α, β によって次のような述語として表現できる。

$ER(r, \alpha) \triangleq \forall i\,(0 < i < r \supset buck(i)\alpha = red) \vee r = 1$ ，

$EW(w, b, \alpha) \triangleq \forall i\,(w < i \leq b \supset buck(i)\alpha = white) \vee w = b$ ，

$EB(b, \alpha) \triangleq \forall i\,(b < i \leq N \supset buck(i)\alpha = blue) \vee b = N$ 。

さらに、プログラム合成に必要な述語として、

$I(\alpha, \beta, r, w, b) \triangleq \forall j \exists i\,(buck(j)\beta = buck(i)\alpha \wedge \forall k \exists l.\,(buck(k)\alpha = buck(l)\beta) \wedge r-1 \leq w \leq b \leq N$

を作る。これらの述語を使うとDutch National Flagのプログラムの仕様を次のように設定できる。この仕様はDijkstra氏の想定した途中の row of bucket の状態から得られる最終状態すなわちDutch National Flagである。

$\forall x\,(\forall i\,(1 \leq i \leq N \wedge (buck(i)x = red \vee buck(i)x = white \vee buck(i)x = blue) \supset$
$\exists z \exists z_r z_w z_b\,(ER(z_r, z) \wedge EW(z_w, z) \wedge EB(z_b, z) \wedge I(x, z, r, w, b)$
$\wedge w = r-1))$ 。

上で定義した公理を加えData-type row of bucket を処理可能となったGNJ+Lを用いて、この仕様をDijkstra氏が[8]で行なっている row of bucket の途中状態から次の状態へ移行させる方法にならって証明することができる。

今、略記法として $\varphi(x)$ を $\forall i\,(1 \leq i \leq N \wedge (buck(i)x = red \vee buck(i)x = white \vee buck(i)x = blue)$ の代りに用いる。この時、状態の移行は次の2つのSequentで表わすことができる。

$\varphi(\beta) \supset \exists z \exists z_r z_w z_b\,(ER(z_r, z) \wedge EW(z_w, z_b, z) \wedge EB(z_b, z) \wedge I(z, \beta, z_r, z_w, z_b) \wedge$
$t = z_w - z_r)$, $t \neq 0 \to \varphi(\beta) \supset \exists \bar{z} \exists \bar{z}_r \bar{z}_w \bar{z}_b\,(ER(\bar{z}_r, \bar{z}) \wedge EW(\bar{z}_w, \bar{z}_b, \bar{z}) \wedge EB(\bar{z}_b, z) \wedge$
$I(\bar{z}, \beta, \bar{z}, z_w, z_b) \wedge t-1 = \bar{z}_w - \bar{z}_r)$ ------ (1)

$\varphi(\beta) \supset \exists z \exists z_r z_w z_b\,(ER(z_r, z) \wedge EW(z_w, z_b, z) \wedge EB(z_b, z) \wedge I(z, \beta, z_r, z_w, z_b)$
$\wedge t = z_w - z_r)$, $t = 0 \longrightarrow \varphi(\beta) \supset \exists z \exists z_r z_w z_b\,(ER(z_r, z) \wedge EW(z_w, z_b, z) \wedge EB(z_b, z)$
$\wedge I(z, \beta, z_r, z_w, z_b) \wedge z_w = z_r - 1)$ ------- (2)

(1)は途中の row of bucket の状態でありそれを最終状態ではない次の状態への移行であり(2)は最終状態への移行である。以下では、略記法として $ER(r, \alpha) \wedge EW(w, b, \alpha) \wedge EB(b, \alpha) \wedge I(\beta, \alpha, r, w, b) \wedge t = w \dot{-} r$ を $\psi(\alpha, \beta, r, w, b, t)$ によって代用する。また、

$$\frac{\to 1 = 1}{\to ER(1, \beta)}\ \text{---(a)} \qquad \frac{\to N = N}{\to EW(N, N, \beta)}\ \text{---(b)} \qquad \frac{\to N = N}{\to EB(N, N, \beta)}\ \text{---(c)}$$

(a)，(b)，(c)と $\to I(\beta, \beta, 1, N, N)$ ， $\to N \dot{-} 1 = N \dot{-} 1$ が証明可能より、

$\to \psi(\beta, \beta, 1, N, N, N \dot{-} 1)$ を得る。

$$\frac{\to \psi(\beta, \beta, 1, N, N, N \dot{-} 1)}{\to \varphi(\beta) \supset \exists z\, \psi(z, \beta, 1, N, N, N \dot{-} 1)}\ \text{--- (3)}$$

より row 中のすべての bucket が inspected である最初の状態に対応するSequentが証明される。従って、(3)から始まって(1)による状態移行を $t-1=0$ となるまで繰返しその後(2)により(cut)推論を行うことで仕様の証明が作成される。

これらのことより、仕様の証明は(1)と(2)を証明することで完了する。

(1)，(2) Sequentの証明は以下のようになる。

$$\frac{1 \leq i < r \supset buck(i)\alpha = red,\ 1 \leq i < r \to buck(i)\alpha = red \quad ; \quad \dfrac{buck(i)\alpha = red,\ i \neq r,\ i \neq w}{\to buck(i)\,swap(r, w)\alpha = red}}{1 \leq i < r \supset buck(i)\alpha = red,\ 1 \leq i < r,\ i \neq w \to buck(i)\,swap(r, w)\alpha = red}$$

$1 \leq i < r \supset buck(i)\alpha = red,\ 1 \leq i < r,\ I(\alpha, \beta, r, w, b) \longrightarrow buck(i)\,swap(r, w)\alpha = red$ ------ $(r1)$

$$\dfrac{buck(w)\alpha = red,\ i = r \rightarrow buck(i)\,swap(r, w)\alpha = red}{1 \leq i < r \supset buck(i)\alpha = red,\ buck(w)\alpha = red,\ i = r,\ I(\alpha, \beta, r, w, b) \longrightarrow buck(i)\,swap(r, w)\alpha = red}$$

$$\dfrac{;\ (r1)}{1 \leq i < r \supset buck(i)\alpha = red,\ 1 \leq i < r+1,\ buck(w) = red,\ I(\alpha, \beta, r, w, b) \rightarrow buck(i)\,swap(r, w)\alpha = red}$$

$E(r, \alpha),\ I(\alpha, \beta, r, w, b),\ buck(w)\alpha = red \rightarrow ER(swap(r, w)\alpha,\ r+1)$ --- $(r2)$

$$\dfrac{t = 0,\ t = w \dot{-} r \rightarrow w = (r+1) \dot{-} 1}{ER(r, \alpha),\ buck(w)\alpha = red,\ t = 0,\ t = w \dot{-} r \rightarrow w = (r+1) \dot{-} 1}$$ ---- $(S1)$

$$\dfrac{t \neq 0,\ t = w \dot{-} r \rightarrow t \dot{-} 1 = w \dot{-} (r+1)}{ER(r, \alpha),\ buck(w)\alpha = red,\ t \neq 0,\ t = w \dot{-} r \rightarrow t \dot{-} 1 = w \dot{-} (r+1)}$$ --- $(S2)$

$$\dfrac{(r2)\ ;\ (S2)}{ER(r, \alpha),\ I(\alpha, \beta, r, w, b),\ buck(w)\alpha = red,\ t \neq 0,\ t = w \dot{-} r \rightarrow ER(swap(r, w)\alpha,\ r+1) \wedge t \dot{-} 1 = w \dot{-} (r+1)}$$ ---- $(T1)$

$$\dfrac{\dfrac{w < i \leq b \supset buck(i)\alpha = white,\ w < i \leq b,\ i \neq r \rightarrow buck(i)\,swap(r, w)\alpha = white}{EW(w, b, \alpha),\ w < i \leq b,\ I(\alpha, \beta, r, w, b) \rightarrow buck(i)\,swap(r, w)\alpha = white}}{EW(w, b, \alpha),\ I(\alpha, \beta, r, w, b) \rightarrow EW(w, b, swap(r, w)\alpha)}$$ ---- $(T2)$

同様に、 $EB(b, \alpha),\ I(\alpha, \beta, r, w, b) \longrightarrow EB(w, b, swap(r, w)\alpha)$ ---- $(T3)$ が証明できる。さらに、$(T1)$、$(T2)$、$(T3)$から次の Sequent が証明できる。

$\Psi(\alpha, \beta, r, w, b, t),\ t \neq 0,\ buck(w)\alpha = red \rightarrow \exists z\, z_r\, z_w\, z_b\, \Psi(z, \beta, z_r, z_w, z_b, t \dot{-} 1)$。

さらに、 $buck(w)\alpha = white$, $buck(w)\alpha = red$ についても上と同様の Sequent が証明可能である。従って、次の Sequent が証明できる。

$\Psi(\alpha, \beta, r, w, b, t),\ t \neq 0,\ buck(w)\alpha = red \vee buck(w)\alpha = white \vee buck(w)\alpha = blue \rightarrow \exists z\, z_r\, z_w\, z_b\, \Psi(z, \beta, z_r, z_w, z_b, t \dot{-} 1)$ --- $(T4)$

$$\dfrac{\dfrac{buck(w)\alpha = buck(i_0)\beta,\ buck(i_0)\beta = red \vee buck(i_0)\beta = white \vee buck(i_0)\beta = blue \longrightarrow buck(w)\alpha = red \vee buck(w)\alpha = white \vee buck(w)\alpha = blue\ ;\ (T4)}{buck(w)\alpha = buck(i_0)\beta,\ \Psi(\alpha, \beta, r, w, b, t),\ t \neq 0,\ \varphi(\beta) \rightarrow \exists z\, z_r\, z_w\, z_b\, \Psi(z, \beta, z_r, z_w, z_b, t \dot{-} 1)}}{\varphi(\beta) \supset \exists z\, z_r\, z_w\, z_b\, \Psi(z, \beta, z_r, z_w, z_b, t),\ t \neq 0 \rightarrow \varphi(\beta) \supset \exists z\, z_r\, z_w\, z_b\, \Psi(z, \beta, z_r, z_w, z_b, t \dot{-} 1)}$$

$t = 0$の場合にも、上と同様の方法により$(r2)$、$(S1)$から(2)を求めることができる。

この証明は Dijkstra 氏の $buck(w)\alpha$ を red, white, blue の場合に分けて pebble操作を構成する問題解法を証明図へ焼直したものである。さらに、焼直しにより証明図が構成されるということは同氏の証明を考慮しながら行うプログラム作成の傍証となる。また、row of bucket と pebble の Data-type が LiST関数を基本関数に組込れた帰納的関数（GNJ+L で作られる関数）によって構成され公理は証明される。従って、Gödel Interpretation からプログラム合成ができる。この時合成されるプログラムは Dijkstra 氏のものと同一となる。

Data-type を考え Top-Down 方式でプログラムを作成することは Dijkstra 氏も提示している。上の Dutch National Flag もその手法に沿って行なわれている。Guttag, Goguen 氏等の Abstract-Data-type の考え方は各種の Data-type で共通に使用できる構造を持つもの（例えば string, array, etc）を特に Abstract-Data-type として代数的にまたは公理的に表現し、各 Data-type をこれらによって記述しようとするものである。この時プログラムの記述は Abstract-Data-type を使った証明とな

る。この証明は上の National Dutch Flag と同様に syntactical システムによる証明すなわち上記プログラム合成での証明に焼直しができる。従って、各種の Abstract-Data-type を公理化し（数学的定義が与えられている為容易に公理化できる。）GNJ+L システムに GNJ+L と同様の方法であらかじめ付加し Gödel の定理を満すように解釈を行っておくことにより、Data-type の証明すなわち implementation という考えがここで用いたプログラム合成と同一視することができる。

おわりに

直観主義自然数論と Gödel Interpretation に基づくプログラム合成の LiST 処理への拡張と一般の Data-type 処理の方法を Dutch National FLag の例を用いて提示した。LiST 処理については LiSP の基本的な operation の公理化によって行った。本論文中で証明はしていないが Linear LiST 上の LiSP-プログラムはすべて合成可能である。Data-type 処理は Dutch National Flag の例のみであるが Abstract-Data-type を各種公理化することで種々の Data-type をプログラム合成において処理可能とすることができる。
今後の課題として Linear LiST 以外の LiST 処理はどこまで可能か？という問題と Data-type に関する Induction（Multi-Set 上の順序に関する termination に対応するようなもの），他の Data-type 個有の証明形式を Data-type の公理化と GNJ+L への付加を行ないつつ考え出すということが残されている。

謝辞

日頃、御指導頂く北海道大学工学部竹村教授，宮本助教授，桃内助手に感謝いたします。

文献

[1] Takasu, s., Proof and Programs, Report of Research Institute for Mathematical Sciences Kyoto University.

[2] Sato, M., Towards a Mathematical Theory of Program Synthesis, IJCAI-79, pp 757-762.

[3] Goto, S., Program Synthesis from Natural Deduction, IJCAI-79, pp 339-341.

[4] Guttag,J. V., Horowitz, E. and Musser, D. R., The Design of Data Type Specifications, Current Trends in Programming Methodology volume lV, pp 60-79.

[5] Goguen, J. A., Thatcher, J. W. and Wager, E. G., An Initial Algebra Approach to The Specification, Correctness, and Implementation of Abstract Data Types, Current Trends in Programming Methodology volumelV, pp 80-149.

[6] Schütte, K., Proof Theory, Springer-Verlag.

[7] Manna, Z. and Waldinger, R. J., Towards Automatic Program Synthesis, Springer, Lec, Notes in Math, No 188, pp 270-310.

[8] Dijkstra, E. W., A Discipline of Programming, Prentice-Hall.