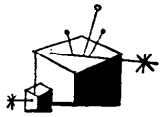
講　座



—抽象データタイプの代数的仕様記述法の基礎（4）—

終代数意味論に基づく抽象型構成子の仕様記述，実現ならびにその検証†

稲垣康善††　坂部俊樹††

1. はじめに

本講座の目標は，データ抽象に基づくプログラミング方法論の中心的概念である抽象データタイプならびに抽象型構成子の代数的仕様記述法の基礎を厳密にしかもできるだけ分かり易く整理して解説することである．

第1回[46]では，データ抽象の理論における基本概念である多ソート代数および等式論理について説明した．それに基づいて，第2回[47]では，抽象データタイプと抽象型構成子との区別を明確にしたうえで，抽象データタイプの代数的仕様記述法，実現および実現検証法について解説した．第3回[48]では，抽象型構成子の代表的な仕様記述法の1つとして，始代数意味論の仕様記述法を解説した．最終回の今回は，抽象型構成子のもう1つの仕様記述法である終代数意味論による代数的仕様記述法，および，抽象型構成子の実現とその検証法について解説する．最後に，代数的仕様記述法の問題点について述べるとともに今後の展望を試み，本講座のまとめとする．

2. 抽象型構成子の終代数意味論に基づく仕様記述

始代数意味論による抽象型構成子の仕様記述法に相対するもう1つの仕様記述法の流儀は，Guttag[37)~40)]流の考え方に従うものである．始代数意味論の場合と同様に細かい違いに注目すればこの流儀に属するアプローチもいくつかの異なる方法として区別されるが，基本的な考え方は共通している．そこで，本稿でも，細かい差異は無視して，本質的な考え方とそれに基づく仕様記述法を本解説の形式とモデルのもとで説明する．そうすることによって，抽象データタイプの仕様記述法（本講座第2回[47]第3章）や始代数意味論による抽象型構成子の仕様記述法（本講座第3回[48]第3章）との比較が容易になり，それらの違いがはっきりする．

まず，仕様の構文は，前回[48]に述べた仕様記述法の構文と同じである．すなわち，前回[48]と今回の仕様記述法の違いは意味論の違いだけである．参照の便宜のため構文の定義を再掲しておく．

[定義 1] 抽象型構成子の仕様は7項組 $\langle S, \Sigma, S', \Sigma', X, \Gamma, B\rangle$ である．ここに，$\langle S, \Sigma\rangle$，$\langle S', \Sigma'\rangle$ は $S\subseteq S'$ かつ $\Sigma_{w,s}\subseteq\Sigma'_{w,s}$ $(w\in S^*, s\in S)$ であるシグニチャであり，$X=\langle X_s\rangle_{s\in S'}$ は変数集合族である．Γ は等式言語 $\langle S', \Sigma', X, \approx\rangle$ の等式の集合である．B は，$S^B\subseteq S$，$\Sigma^B\subseteq\Sigma$ であるようなシグニチャ $\langle S^B, \Sigma^B\rangle$ の代数である．□

次に，仕様の意味論を説明する．そのために，まず，本解説の定式化の枠組に沿って，Guttag 流の仕様記述法の基本的な考え方を説明する．

前回[48]に紹介した始代数意味論での抽象型構成子の仕様記述法では，仕様の等式集合 Γ が仕様の意味を定める際の中心的な役割を果たし，引数データタイプ D の構造はそれほど重要ではなく補助的な役割をするにすぎない．すなわち，$\Sigma'(D)$ 項が互いに等価であるのは，Γ から等しいことが推論できるときかつそのときに限るとして $P[\Sigma'(D)]$ 上の合同関係が定められた．これに対して，Guttag 流の仕様記述法では，意味論の鍵となるのは引数データタイプ D の構造である．すなわち，抽象型構成子 F による D の像 $F(D)$ で D の構造が保存されるという事実に重点を置き，D の異なる要素が $F(D)$ においても互いに区別されるという条件が満たされる限りどの $\Sigma'(D)$ 項も等価であるとみなすのである．

使用者の立場に立って言い換えれば，Guttag 流の仕様記述法では，型構成子（によって得られるデータタイプ）を使うにあたっては，値のソートが引数デー

† Final Algebra Approach to Specification, Implementation and Verification of Abstract Type Constructor by Yasuyoshi INAGAKI and Toshiki SAKABE (Faculty of Engineering, Nagoya University).

†† 名古屋大学工学部

タタイプ D のソートであるような合成演算 ($\Sigma'(D)$ 項) が D のどのような値に対応付けられるかという情報だけが必要で，引数データタイプのソート以外のソートの値を持つ合成演算 ($\Sigma'(D)$ 項) がどのような値を持つかということは知る必要がないという認識を前提にして意味論が定められている．つまり，仕様の等式集合からそのような引数データタイプのソートをもつ $\Sigma'(D)$ 項とそれが持つべき D の値との対応関係——型構成子の D 上での動作と呼ばれる——を求め，その関係に矛盾しない最大の合同関係による $P[\Sigma'(D)]$ の商代数を用いて意味論が定められている．

以上に直観的に説明した Guttag 流の意味論は，形式化すれば，以下のように定められる．

まず，上述のような型構成子の動作の概念を，一般的に，$P[\Sigma'(D)]_s$ 上の同値関係 $H(D)_s$ の族として定式化する．

[定義 2] $\langle S, \Sigma, S', \Sigma', \mathcal{D}, F\rangle$ を型構成子とする．F の**動作** (behavior) H は，任意の Σ 代数 $D\in\mathcal{D}$ に次のような $P[\Sigma'(D)]_s$ 上の同値関係 $H(D)_s$ の族 $H(D)=\langle H(D)_s\rangle_{s\in S}$ を対応付ける写像である．すなわち，任意の $t, t'\in P[\Sigma'(D)]_s$ に対して，

$$tH(D)_s t' \Longleftrightarrow t_{F(D)}=t'_{F(D)}$$

である．$H(D)$ を F の D 上の動作という． □

前回[48]の定義 8 によれば，上の定義の $H(D)_s$ は，$\overline{\overline{F(D)}}$ のソート s の同値関係 $\overline{\overline{F(D)}}_s$ に一致する．すなわち，$H(D)$ は $\overline{\overline{F(D)}}$ の部分族 $\langle\overline{\overline{F(D)}}_s\rangle_{s\in S}$ と同一である．したがって，前回[48]の命題 1 の (2) により，次のことが成り立つ．任意の $\Sigma'(D)$ 項 $t\in P[\Sigma'(D)]_s$ ($s\in S$) に対して，$tH(D)d_t$ となる $d_t\in D_s$ が一意に存在し，この d_t は，$F(D)|\Sigma$ から D への同型写像を h' とすれば $h'(t_{F(D)})$ で与えられる (**図-1** 参照)．このことを言い換えると，$H(D)_s$ は $P[\Sigma'(D)]_s$ の要素 t を $d_t=h'(t_{F(D)})$ へ写す写像を自然に定める (**図-2** 参照)．つまり，$H(D)_s$ はこのような写像とみることもできる．

すでに直観的に述べたように，Guttag 流の意味論では，まず仕様 S の等式集合 Γ から引数データタイプ D の上での動作を定め，その動作に矛盾しない最大の合同関係を用いて D が写像される先のデータタイプを定める．

まず，仕様から Σ 代数上の動作を定めよう．定義 2 で定めたように，抽象型構成子の Σ 代数 D 上での動作は，$P[\Sigma'(D)]_s$ ($s\in S$) 上の同値関係の族であるから，仕様の等式集合 Γ からその同値関係の族 $H(D)=\langle H(D)_s\rangle_{s\in S}$ を定めればよい．その最も自然な定義は，

$$tH(D)_s t' \Longleftrightarrow \Gamma|\overline{\overline{D}} t\approx t'$$

によって $H(D)$ を定めることである．すなわち，$H(D)_s=\overline{\overline{\Gamma, D}}_s$ とすることであろう．しかし，ここで少し注意が必要である．この定義を無条件に用いると，D によっては $dH(D)_s d'$ となる異なる $d, d'\in D_s$ が存在したり，あるいは，どの $d\in D_s$ に対しても $tH(D)_s d$ とならない $\Sigma'(D)$ 項 t が存在する可能性がある．したがって無条件に上の $H(D)$ を抽象型構成子の D 上での動作とするのは不適当である．このような可能性を排除する条件として，前回[48]の定義 13 で与えた仕様の Σ 代数 D に関する無矛盾性と完全性が役に立つ．すなわち，これらの条件が満たされれば，その定義から直接に上のような不都合は排除される．

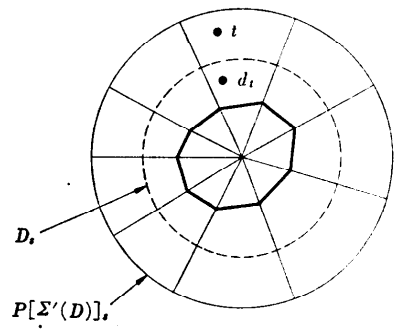


図-1 $H(D)_s$ による $P[\Sigma'(D)]_s$ の同値分割

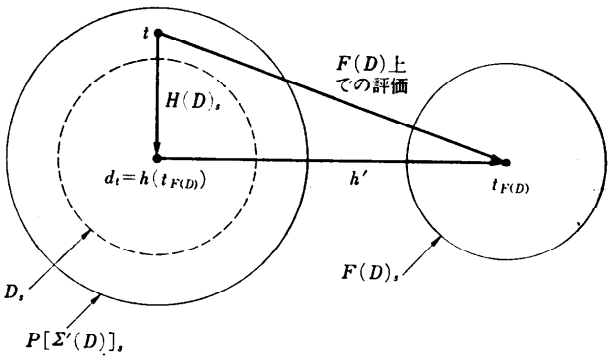


図-2 写像としての $H(D)_s$

以上の議論に基づいて，Guttag 流の仕様記述法の意味論は次のように形式的に与えられる．

[定義 3] 抽象型構成子の仕様 $\mathcal{S}=\langle S, \Sigma, S', \Sigma', X, \Gamma, B\rangle$ が意味する抽象型構成子 $\langle S, \Sigma, S', \Sigma', \mathcal{D}[\mathcal{S}], ATCF[\mathcal{S}]\rangle$ は次のように定義される*．

まず，Σ 代数のクラス $\mathcal{D}[\mathcal{S}]$ を，

$\mathcal{D}[\mathcal{S}]=\{D\in \mathcal{A}lg_\Sigma \mid D|\Sigma^B=B$ かつ $\mathcal{S}$ は D に関して無矛盾かつ完全である$\}$

と定め，$\mathcal{D}[\mathcal{S}]$ の中の任意の Σ 代数 D に対して，$ATCF[\mathcal{S}](D)$を，

$ATCF[\mathcal{S}](D)=P[\Sigma'(D)]/\sim$

で定める．ここに，$\sim$は，すべての $s\in S$ ($s\in S'$ でないことに注意せよ) について，

$\sim_s=\overset{\Gamma, D}{\equiv\equiv}_s$

であるような $P[\Sigma'(D)]$ 上の最大の合同関係である．

また，以下では，$\langle\overset{\Gamma, D}{\equiv\equiv}_s\rangle_{s\in S}$ を $BHV[\mathcal{S}](D)$ で表し，それを仕様 $\mathcal{S}$ から定められる動作と呼ぶ． □

この定義で，$\sim_s=\overset{\Gamma, D}{\equiv\equiv}_s$ $(s\in S)$ である最大の合同関係$\sim$が存在しない場合があると危惧するかもしれないが，そのようなことは決してないことが次に述べる命題1で保証される．

その準備に二，三の記法に約束をしておく．変数集合族 $X=\langle X_s\rangle_{s\in S}$，および，ソートの系列 $w\in S^*$ に対して，$X^w=\langle X_s^w\rangle_{s\in S}$ は次のように定められる変数集合であるとする．すなわち，$w=s_1\cdots s_n$ とおくとき，各 $i=1, \cdots, n$ に対して，1つずつしかもすべて相異なるようにソート s_i の変数 $x_i\in X_{s_i}$ を選ぶ．これらの変数 $x_1, \cdots, x_n$ のうち，ソート s の変数の集合を X_s^w とする．

[命題 1] $\mathcal{S}=\langle S, \Sigma, S', \Sigma', X, \Gamma, B\rangle$ を抽象型構成子の仕様とし，D を $\mathcal{D}[\mathcal{S}]$ の中の任意の Σ 代数とする．各々の $s'\in S'$ に対して，$P[\Sigma'(D)]_{s'}$ 上の関係 $R_{s'}$ を次のように定める．

$t_1 R_{s'} t_2 \Longleftrightarrow$ 任意の $s\in S$ および任意の $\xi\in T[\Sigma'(D)(X'')]_{s'}$ に対して**

$\Gamma|\overset{D}{-}\xi[t_1/x]\approx\xi[t_2/x]$

ただし，$X''=\langle X''_{s''}\rangle_{s''\in S'}$ は $X''_{s'}=\{x\}$，$X''_{s''}=\phi$ $(s''\neq s')$ である変数集合族である．このとき，$R=\langle R_{s'}\rangle_{s'\in S'}$ は，任意の $s\in S$ について $R_s=\overset{\Gamma, D}{\equiv\equiv}_s$ となる最大の合同関係である．

(証明) まず，R_s の定め方より，任意の $s\in S$ について，$X'_s=\{x\}$ とおくと，

$t_1 R_s t_2 \Rightarrow \Gamma|\overset{D}{-} x[t_1/x]\approx x[t_2/x]$

$\Rightarrow \Gamma|\overset{D}{-} t_1\approx t_2$

したがって，$R_s\subseteq\overset{\Gamma, D}{\equiv\equiv}_s$ である．

逆に，$t_1\overset{\Gamma, D}{\equiv\equiv}_s t_2$ すなわち，$\Gamma|\overset{D}{-} t_1\approx t_2$ ならば，推論規則より，任意の $s\in S$ および任意の $\xi\in T[\Sigma'(D)(X'')]_{s'}$ に対して，$X''_{s'}=\{x\}$ とおくと，

$\Gamma|\overset{D}{-}\xi[t_1/x]\approx\xi[t_2/x]$.

すなわち，$t_1 R_{s'} t_2$ である．

よって，$R_s=\overset{\Gamma, D}{\equiv\equiv}_s$ $(s\in S)$ である．

次に，R が合同関係であることを示す．各 $s'\in S'$ に対して，$R_{s'}$ が同値関係であることは明らかであるから，ここで示すべきことは，$\alpha\in\Sigma_{s_1\cdots s_n, s}$，$t_i R_{s_i} t_i'$ $(i=1, \cdots, n)$ とすると，任意の $\xi\in T[\Sigma'(D)(X')]_{s'}$ $(s'\in S)$ に対して，

$\Gamma|\overset{D}{-}\xi[\alpha(t_1, \cdots, t_n)/x]\approx\xi[\alpha(t_1', \cdots, t_n')/x]$

が成立することである．これは，$t_1 R_{s_1} t_1'$ であることより，

$\Gamma|\overset{D}{-}\xi[\alpha(t_1, t_2, \cdots, t_n)/x]\approx\xi[\alpha(t_1', t_2, \cdots, t_n)/x]$

であり，さらに，$t_2 R_{s_2} t_2'$ であることより，

$\Gamma|\overset{D}{-}\xi[\alpha(t_1', t_2, \cdots, t_n)/x]\approx\xi[\alpha(t_1', t_2', t_3, \cdots, t_n)/x]$

等々と順次 $t_1, \cdots, t_n$ を $t_1', \cdots, t_n'$ で置きかえることによって証明される．

最後に，R が $R_s=\overset{\Gamma, D}{\equiv\equiv}_s$ $(s\in S)$ である合同関係のうちで最大であることを示す．いま，Q を，すべての $s\in S$ に対して $Q_s=\overset{\Gamma, D}{\equiv\equiv}_s$ である $P[\Sigma'(D)]$ 上の合同関係とする．合同関係の定義より，任意の $w\in S'^*$，$s\in S$ および $\xi\in T[\Sigma'(D)(X^w)]_s$ に対して，$w=s_1\cdots s_n$ とすると次が成り立つ．

$\forall t_i, \forall t_i'\in P[\Sigma'(D)]_{s_i}$ $(i=1, \cdots, n)$,

$t_i Q_{s_i} t_i'$ $(i=1, \cdots, n)$

$\Rightarrow \xi_{P[\Sigma'(D)]}(t_1, \cdots, t_n) Q_s \xi_{P[\Sigma'(D)]}(t_1', \cdots, t_n')$

ここに，$\xi_{P[\Sigma'(D)]}$ は，第2回[47]の定義10の中で用いた記法で，$P[\Sigma'(D)]_{s_1}\times\cdots\times P[\Sigma'(D)]_{s_n}$ から $P[\Sigma'(D)]_s$ への関数を表す．これは，しばしば，ξ の $P[\Sigma'(D)]$ 上での導出演算と呼ばれる．そこで，$s\in S$ のときは，$Q_s=\overset{\Gamma, D}{\equiv\equiv}_s$ であるから，上の式で $s\in S$ とし，$\xi\in T[\Sigma'(D)(X')]_{s'}$ とすれば，

$t_1 Q_{s'} t_2 \Rightarrow \xi_{P[\Sigma'(D)]}(t_1)\overset{\Gamma, D}{\equiv\equiv}_s \xi_{P[\Sigma'(D)]}(t_2)$

である．$\xi_{P[\Sigma'(D)]}$ の定義より，

$\xi_{P[\Sigma'(D)]}(t_1)=(\xi[t_1/x])_D$

---

* $ATCF$ の ATC は Abstract Type Constructor を表し，F は Final Algebra を表す．

** $T[\Sigma'(D)(X)]_s$ は X の変数を含むソート s の $\Sigma'(D)$ 項の集合を表す．

$\xi_{P[\Sigma'(D)]}(t_2)=(\xi[t_2/x])_D$

であり，また，

$\Gamma \vdash^D (\xi[t_1/x])_D \approx \xi[t_1/x]$

$\Gamma \vdash^D (\xi[t_2/x])_D \approx \xi[t_2/x]$

であるから，結局，

$t_1 Q_{s'} t_2 \Rightarrow \Gamma \vdash^D \xi[t_1/x] \approx \xi[t_2/x]$

である．よって，$Q_{s'} \subseteq R_{s'}$ である． □

以上で Guttag 流の仕様記述法の意味論が定められた．そこで，いわゆる配列に対応する抽象型構成子を仕様記述し，それを Guttag 流の意味論で解釈してみよう．

［例 1］ シグニチャ $\langle S^B, \Sigma^B\rangle$ $\langle S, \Sigma\rangle$ $\langle S', \Sigma'\rangle$ を**図-3**（a），（b），（c）のように与える．ただし，簡単のために $\langle S, \Sigma\rangle$ の図では Σ^B を省略し，$\langle S', \Sigma'\rangle$ の図では Σ を略して書いてある．変数集合族 X を，

$X_{\mathbf{array}} = \{a_1, a_2, \cdots\}$

$X_{\mathbf{data}} = \{d_1, d_2, \cdots\}$

$X_{\mathbf{bool}} = \{b_1, b_2, \cdots\}$

$X_{\mathbf{index}} = \{i_1, i_2, \cdots\}$

からなるものとする．さらに，等式集合 Γ は次の等式からなるものとする．

ACCESS (NEWARRAY, i_1)
≈ERROR

ACCESS (ASSIGN (a_1, i_1, d_1), i_2)
≈IFD (EQ (i_1, i_2), d_1, ACCESS (a_1, i_2))

ISDEF (NEWARRAY, i_1)≈FALSE

ISDEF (ASSIGN (a_1, i_1, d_1), i_2)
≈IFB (EQ(i_1, i_2), TRUE, ISDEF (a_1, i_2))

また，Σ^B 代数 B を，

$B_{\mathbf{bool}} = \{tt, ff\}$

$\mathrm{TRUE}_B = tt$

$\mathrm{FALSE}_B = ff$

$\mathrm{IFB}_B(tt, b, b') = b$

$\mathrm{IFB}_B(ff, b, b') = b'$

とする．

これらの 7 項組 $\langle S, \Sigma, S', \Sigma', X, \Gamma, B\rangle$ を仕様 $\mathcal{S}$ とする．この仕様 $\mathcal{S}$ が Guttag 流の意味論のもとで意味する抽象型構成子がどのようなものであるかをみよう．Σ 代数 D を

$D_{\mathbf{data}} = \{0, 1, 2, \cdots\}$

$D_{\mathbf{bool}} = \{tt, ff\}$

$D_{\mathbf{index}} = \{0, \pm1, \pm2, \cdots\}$

$\mathrm{TRUE}_D = tt$

$\mathrm{FALSE}_D = ff$

$\mathrm{ERROR}_D = 0$

$\mathrm{ZERO}_D = 0$

$$\mathrm{EQ}_D(i, j) = \begin{cases} tt\,; & i=j \text{ のとき} \\ ff\,; & i \neq j \text{ のとき} \end{cases}$$

$$\mathrm{IFD}_D(b, d, d') = \begin{cases} d\,; & b=tt \text{ のとき} \\ d'\,; & b=ff \text{ のとき} \end{cases}$$

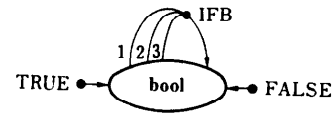
$$\mathrm{IFB}_D(b, b', b'') = \begin{cases} b'\,; & b=tt \text{ のとき} \\ b''\,; & b=ff \text{ のとき} \end{cases}$$

$$\mathrm{ISZERO}(i) = \begin{cases} tt\,; & i=0 \text{ のとき} \\ ff\,; & i \neq 0 \text{ のとき} \end{cases}$$

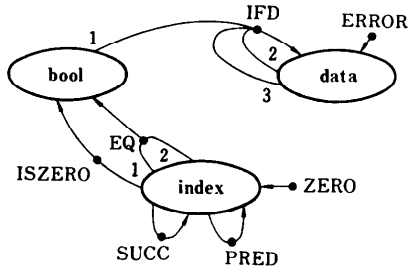
$\mathrm{SUCC}(i) = i+1$

$\mathrm{PRED}(i) = i-1$

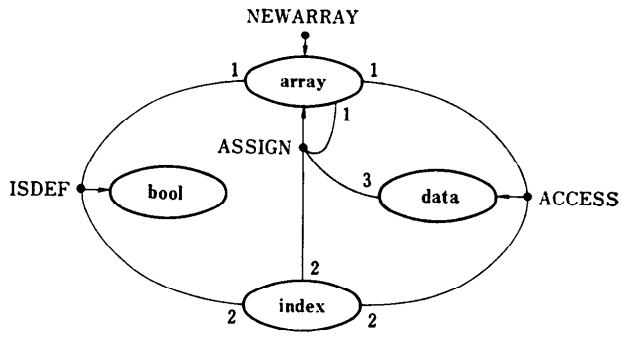
とする．$\Sigma'(D)$ 項に含まれる ASSIGN の個数に関する帰納法を用いれば，$\mathcal{S}$ が D に関して無矛盾か



（a） シグニチャ $\langle S^B, \Sigma^B\rangle$



（b） シグニチャ $\langle S, \Sigma\rangle$



（c） シグニチャ $\langle S', \Sigma'\rangle$

図-3 例 1 のシグニチャ $\langle S^B, \Sigma^B\rangle$, $\langle S, \Sigma\rangle$, $\langle S', \Sigma'\rangle$ の図表示

つ完全であることを容易に示すことができる．よって，D は $\mathcal{D}[S]$ の元である．S が定める D 上の動作 $BHV[S](D)$ $(=\langle \overset{\Gamma,D}{\equiv}, \rangle_{s\in S})$ を見よう．$\overset{\Gamma,D}{\equiv}_{\mathbf{data}}$ 同値な $P[\Sigma'(D)]_{\mathbf{data}}$ の元の例は，

$0\overset{\Gamma,D}{\equiv}_{\mathbf{data}}$ ACCESS (ASSIGN (NEWARRAY, 9, 0), 9)

$1\overset{\Gamma,D}{\equiv}_{\mathbf{data}}$ ACCESS (ASSIGN (ASSIGN (NEWARRAY, 8, 1), 3, 4), 8)

$2\overset{\Gamma,D}{\equiv}_{\mathbf{data}}$ ACCESS (ASSIGN (ASSIGN (NEWARRAY, 4, 5), 4, 2), 4)

などである．$\overset{\Gamma,D}{\equiv}_{\mathbf{bool}}$ 同値な $P[\Sigma'(D)]_{\mathbf{bool}}$ の元の例は，

$ff\overset{\Gamma,D}{\equiv}_{\mathbf{bool}}$ ISDEF (NEWARRAY, 1)

$ff\overset{\Gamma,D}{\equiv}_{\mathbf{bool}}$ ISDEF (ASSIGN (NEWARRAY, 1, 2), 2)

$tt\overset{\Gamma,D}{\equiv}_{\mathbf{bool}}$ ISDEF (ASSIGN (NEWARRAY, 1, 2), 1)

などである．また，値が **index** であるような演算が Σ' にはないので，$P[\Sigma'(D)]_{\mathbf{index}}=D_{\mathbf{index}}$ であり，当然，$\overset{\Gamma,D}{\equiv}_{\mathbf{index}}$ は $D_{\mathbf{index}}$ 上の恒等関係である（図-4 参照）．

さて，S が意味する抽象型構成子 $ATCF[S]$ による D の像 $ATCF[S](D)$ は $BHV[S](D)$ を保存する最大の合同関係～で定められ，それは命題1によって陽に与えられている．そこでの定め方から，任意の $t\in P[\Sigma'(D)]_{\mathbf{array}}$，$i, i'\in D_{\mathbf{index}}$，$(i\neq i')$ および，$dd'\in D_{\mathbf{data}}$ に対して，

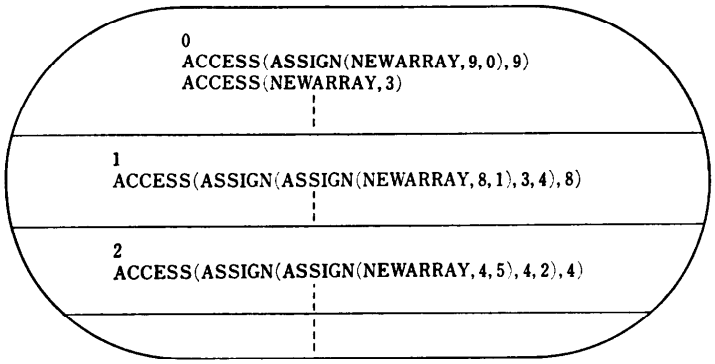
ASSIGN (ASSIGN $(t, i, d), i', d')$
$\sim_{\mathbf{array}}$ ASSIGN (ASSIGN $(t, i', d'), i, d)$

であることは，簡単な帰納法（例えば，$T[\Sigma'(D)(X^{\mathbf{array}})]_{\mathbf{data}}$ の要素に出現する ASSIGN の個数に関する帰納法）で証明できる．したがって，$\sim_{\mathbf{array}}$ のもとでは，代入されるインデックスが異なる限り，配列への代入の順序を入れ換えても等価性が保たれる．

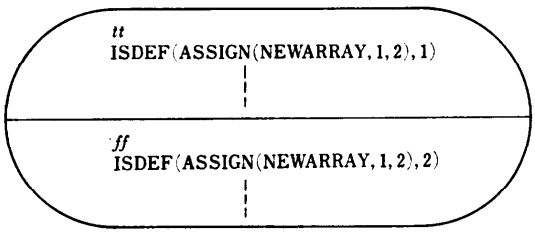
また，同じインデックスへの代入は最新のものだけがアクセスでき，それ以前のものはアクセスできないことは，Γ の中の第2の等式からすぐに分かる．このことから，$P[\Sigma'(D)]/\sim$ が通常の配列に相当していることが理解できるだろう． □

もう一つ型構成子の仕様記述例を与えよう．

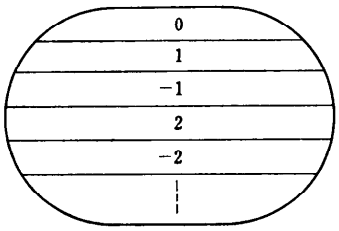
[例 2] コンパイラの中で重要な役割を果たす型構成子としてシンボルテーブル (symbol table) がある．シンボルテーブルは，ブロック構造を持つプログラミング言語のコンパイラの中で，プログラム中に出現する変数とその属性の対応を記憶，管理するデータタイプを構成する型構成子であり，次の 6 つの演算を



(a) $P[\Sigma'(D)]_{\mathbf{data}}$ の $\overset{\Gamma,D}{\equiv}_{\mathbf{data}}$ による同値分類



(b) $P[\Sigma'(D)]_{\mathbf{bool}}$ の $\overset{\Gamma,D}{\equiv}_{\mathbf{bool}}$ による同値分類



(c) $P[\Sigma'(D)]_{\mathbf{index}}$ の $\overset{\Gamma,D}{\equiv}_{\mathbf{index}}$ による同値分類

図-4 $BHV[S](D)=\langle \overset{\Gamma,D}{\equiv}, \rangle_{s\in S}$ の説明図

持つ[40].

INIT：最外側のスコープに対するシンボルテーブルを割り当て初期化する.

ENTERBLOCK：新しいスコープに入るための準備をする.

ADDID：シンボルテーブルに識別子とその属性を付け加える.

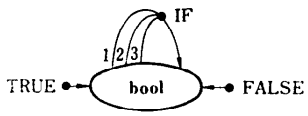
LEAVEBLOCK：最も新しいスコープから出て，その1つ外側のスコープに移る．現在のスコープが最外側のときは何もしない.

ISINBLOCK：現在のスコープ中に指定された識別子が宣言されているか否かを答える.

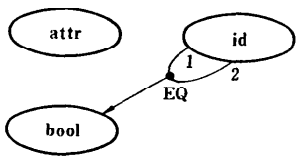
RETRIEVE：指定された識別子の持つ属性のうち，現在のスコープの外側で最も近いスコープに対応付けが宣言されている属性を取り出す.

このような非形式的な説明で与えられるシンボルテーブルを代数的仕様記述法を用いて形式的に仕様記述すると次のようになる.

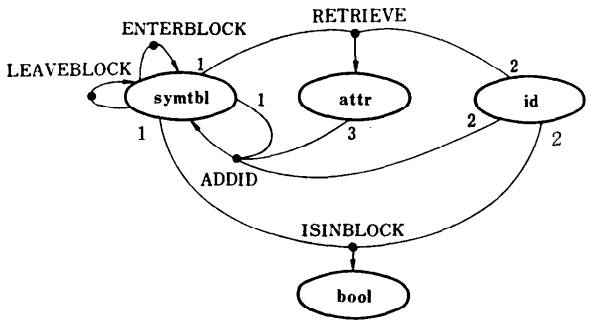
まず上の演算をシグニチャとして形式的に表すと



(a) $\langle S^B, \Sigma^B \rangle$



(b) $\langle S, \Sigma \rangle$



(c) $\langle S', \Sigma' \rangle$

図-5 シグニチャ $\langle S^B, \Sigma^B \rangle$, $\langle S, \Sigma \rangle$, $\langle S', \Sigma' \rangle$ の図表示

図-5 のようになる．図-5 において，(a)はあらかじめ構造が決まっているデータタイプのシグニチャであり，(a)と(b)を合わせたシグニチャはこの型構成子の引数データ型のシグニチャである．(a), (b), (c)を合わせたシグニチャは，この型構成子によって構成されるデータ型のシグニチャである．また，ERROR はエラーを表す属性値に対応する定数演算，EQ は識別子間の等価性を判定する演算，TRUE, FALSE は真，偽を表す定数演算，IF は条件分岐演算である．仕様 $\mathcal{S}=\langle S, \Sigma, S', \Sigma', X, \Gamma, B \rangle$ を次のように定める $\langle S, \Sigma \rangle, \langle S', \Sigma' \rangle, \langle S^B, \Sigma^B \rangle$ は図-5 のとおりとする．X は

$X_{\mathbf{bool}} = \{b_1, b_2, \cdots\}$

$X_{\mathbf{attr}} = \{a_1, a_2, \cdots\}$

$X_{\mathbf{id}} = \{i_1, i_2, \cdots\}$

$X_{\mathbf{symtbl}} = \{s_1, s_2, \cdots\}$

で与えられる変数集合族とする．等式集合 Γ は

LEAVEBLOCK (ENTERBLOCK (s_1))$\approx s_1$

LEAVEBLOCK (ADDID (s_1, i_1, a_1))
$\approx$LEAVEBLOCK (s_1)

LEAVEBLOCK (INIT)$\approx$INIT

ISINBLOCK (INIT, i_1)$\approx$FALSE

ISINBLOCK (ADDID (s_1, i_1, a_1), i_2)
$\approx$IF (EQ(i_1, i_2), TRUE, ISINBLOCK (s_1, i_2))

RETRIEVE (INIT, i_1)$\approx$ERROR

RETRIEVE (ENTERBLOCK (s_1), i_1)
$\approx$RETRIEVE (s_1, i_1)

RETRIEVE (ADDID (s_1, i_1, a_1), i_2)
$\approx$IF (EQ (i_1, i_2), a_1, RETRIEVE (s_1, i_2))

の8個の等式からなるものとする．最後に，B は次のような Σ^B 代数であるとする．

$B_{\mathbf{bool}} = \{tt, ff\}$

$\mathrm{TRUE}_B = tt$

$\mathrm{FALSE}_B = ff$

$$\mathrm{IF}_B(b, b', b'') = \begin{cases} b' ; & b = tt \text{ のとき} \\ b'' ; & b = ff \text{ のとき} \end{cases}$$

この仕様 $\mathcal{S}$ が終代数意味論で意味する $ATCF[S]$ が，直観的なシンボルテーブルに一致することは，Γ に含まれる等式から明らかであるが，例1と同様にして確かめられる.

最後に，前回[48]説明した Thatcher 流の始代数意味論による抽象型構成子の仕様記述法と本稿で解説した Guttag 流の仕様記述法を比較して，それらの相違点ならびにそれらがどのよう

な関係にあるかについて簡単に言及しよう.

両者の構文は共通であるから，それらが違っているところは意味論の違いである．仕様 $S=\langle S, \Sigma, S', \Sigma', X, \Gamma, B\rangle$ が始代数および終代数意味論でそれぞれ意味する抽象型構成子 $ATCI[S]$ および $ATCF[S]$ は，任意の Σ 代数 $D\in\mathcal{D}[S]$ から，

$$ATCI[S](D)=P[\Sigma'(D)]/\overline{\overline{\Gamma, D}}$$
$$ATCF[S](D)=P[\Sigma'(D)]/\sim$$

を構成する抽象型構成子である．したがって，意味論の違いは，合同関係 $\overline{\overline{\Gamma, D}}$ と $\sim$との違いである．S のソート s については $\overline{\overline{\Gamma, D}}_s$ と $\sim_s$ は同一であり，$ATCI[S]$ と $ATCF[S]$ の D 上での動作は等しく，共に，$BHV[S](D)=\langle\overline{\overline{\Gamma, D}}_s\rangle_{s\in S}$ である．$\overline{\overline{\Gamma, D}}$ は $BHV[S](D)$ を保存する最小の合同関係であるのに対して，$\sim$は $BHV[S](D)$ を保存する最大の合同関係である．このことが両者の違いであるといってよい．両者の意味論は，ある意味で双対な関係にある．実際，Σ への制限が D と同型でかつ Γ の等式をすべて満たす $P[\Sigma'(D)]$ の商代数のクラスの中で $P[\Sigma'(D)]/\overline{\overline{\Gamma, D}}$ は始代数になっているのに対して，$P[\Sigma'(D)]/\sim$ は終代数になっている．特に，$P[\Sigma'(D)]/\overline{\overline{\Gamma, D}}$ は，Σ への制限が D と同型でかつ Γ のすべての等式を満たすすべての代数のクラスの始代数にもなっている．これら二つの意味論の名前はこれらのことに由来するものである．

3. 抽象型構成子の実現

3.1 実現の概念の定式化

抽象型構成子の場合にも実現の概念の定式化は，その仕様記述法の理論の重要な課題の一つである．

抽象型構成子の実現の概念の定式化には二つの方法が考えられる．その第一は，抽象データタイプの実現の概念の定式化と同じ方法をとることである．すなわち，抽象型構成子が型構成子の抽象化であることは抽象データタイプがデータタイプの抽象化であるのと同じであるので，抽象型構成子の実現も抽象データタイプの実現[47]と同じ考え方で定式化できる．実現すべき抽象型構成子 F と型構成子 G の抽象化 $abs(G)$ が一致するとき，G は F を実現すると定めるのである．この方法では，抽象データタイプのときがそうであったように，実現の検証が等式論理[46]の推論だけではできず，常に何らかの別の証明手段（例えば，構造帰納法など）が必要である．これは，等式論理では二つの項が等しいことが推論できても等しくないことは推論できないからである．

これに対して，もう一つの実現の定式化の方法として，動作に注目した定式化の仕方がある．これは，型構成子 G の任意の引数データタイプ D 上での動作 $H(D)$ と実現されるべき抽象型構成子 F の D 上での動作 $H'(D)$ とが一致するとき，G は F を実現すると定めるものである．この定式化には，等式論理の推論だけで実現の検証ができる場合があるばかりでなく，実現の定義条件が弱いので，実際に実現する場合の種々の選択の自由度が大きいという利点がある．

第一の実現の定式化については，抽象データタイプの場合とほとんど同じようにして，実現およびその検証に関する議論ができるので，その議論は読者にゆだねる．ここでは，第二のすなわち動作に注目した実現の定式化と実現の検証について解説する．

まず，動作の概念に基づいた抽象型構成子の実現を定式化する．すでに述べたように，抽象型構成子の動作に注目するという立場では，動作が同じであることで実現を定式化するのが自然である．すなわち，

[定義 4] $\langle S, \Sigma, S', \Sigma', \mathcal{D}^{(1)}, F^{(1)}\rangle$, $\langle S, \Sigma, S', \Sigma', \mathcal{D}^{(2)}, F^{(2)}\rangle$ をそれぞれ抽象型構成子，型構成子とし，その動作を，$H^{(1)}, H^{(2)}$ とする．このとき，次の二つの条件

(1) $\mathcal{D}^{(1)}\subseteq\mathcal{D}^{(2)}$

(2) 任意の $D\in\mathcal{D}^{(1)}$ に対して，
$$H^{(1)}(D)=H^{(2)}(D)$$

が成立するときかつその時に限り $F^{(2)}$ は $F^{(1)}$ の実現であるという．□

上の定義の条件 (1) $\mathcal{D}^{(1)}\subseteq\mathcal{D}^{(2)}$ について少し付言しておこう．データ抽象に基づくプログラミングでは，一つのプログラムモジュールの使用者がその使用に際して用いることのできる情報は，そのモジュールの抽象化によって得られる概念（抽象データタイプ，抽象型構成子）についての情報だけであるから，$F^{(1)}$ の使用者が，$\mathcal{D}^{(1)}$ の中にないデータタイプに対して $F^{(1)}$ を施すことはないと仮定してよい．したがって，$\mathcal{D}^{(1)}=\mathcal{D}^{(2)}$ である必要はなく，$\mathcal{D}^{(1)}\subseteq\mathcal{D}^{(2)}$ であれば，$F^{(2)}$ を $F^{(1)}$ とみなして使ったとしても何ら問題は生じない．そればかりか，実現の検証のことを考えると，$\mathcal{D}^{(1)}=\mathcal{D}^{(2)}$ を検証するよりも $\mathcal{D}^{(1)}\subseteq\mathcal{D}^{(2)}$ を検証する方が容易であるという利点もある．

3.2 実現の記述

実現の概念を定式化しただけでは実現の検証につい

て議論をすすめることはできない．抽象データタイプの実現の検証の議論の際に述べたと同じように，抽象型構成子の実現のプログラム化の正当性，すなわち，プログラムが実現になっていることの検証について論じるには，そのプログラムを記述する何らかの手段または言語を固定することが必要である．ここでは，第2回[47]で抽象データタイプのプログラムを記述するのに用いた型記述法に似た簡単な記述法を用いて実現の検証法を説明する．

抽象データタイプのプログラムを記述するためにデライバ* を用いたが，抽象型構成子の実現の候補，すなわち，プログラムの記述にもデライバの概念を使うことにする．すなわち，ここでの記述法では，デライバによって型構成子のシグニチャ変換を記述し，そのシグニチャ変換によって，すでに実現されていると仮定している型構成子のシグニチャを実現されるべき抽象型構成子のシグニチャに変換して得られる型構成子が記述された型構成子であるとする．ただし，ここでのデライバは，型構成子のシグニチャ変換を記述するものとして用いられるから，引数データタイプのシグニチャに関しては恒等変換になっているものに制限される．すなわち，

[定義 5] $\langle S, \Sigma\rangle$ をシグニチャ，また，$i=1,2$ に対して，$\langle S^{(i)}, \Sigma^{(i)}\rangle$ を $S\subseteq S^{(i)}$，$\Sigma_{w,s}\subseteq\Sigma^{(i)}_{w,s}$ $(w\in S^*, s\in S)$ であるようなシグニチャ，$X^{(i)}=\langle X^{(i)}_s\rangle_{s\in S^{(i)}}$ を変数集合族とし，また，$\langle S, \Sigma, S^{(2)}, \Sigma^{(2)}, \mathcal{D}^{(2)}, F^{(2)}\rangle$ を型構成子とする．さらに，$\delta=\langle f, d\rangle$ を次の二つの条件(i)，(ii)を満たす $\langle S^{(1)}, \Sigma^{(1)}\rangle$ から $\langle S^{(2)}, \Sigma^{(2)}\rangle$ へのデライバとする．

(i) $s\in S$ に対して，$f(s)=s$ である．

(ii) $w=s_1\cdots s_n\in S^*$，$s\in S$，$\alpha\in\Sigma_{w,s}$ に対して，$d_{w,s}(\alpha)=\alpha(x_1,\cdots,x_n)$ である．ただし，x_i は互いに異なる変数でそのソートは s_i である．

このようなデライバを，Σ に関して恒等的なデライバと呼ぶ．このとき，型構成子，$\langle S, \Sigma, S^{(1)}, \Sigma^{(1)}, \mathcal{D}^{(1)}, F^{(1)}\rangle$ を次のように定める．

(1) $\mathcal{D}^{(1)}=\mathcal{D}^{(2)}$

(2) 任意の $D\in\mathcal{D}^{(1)}$ に対して，
$F^{(1)}(D)=\delta(F^{(2)}(D))$.

この型構成子 $F^{(1)}$ を $F^{(2)}$ の導出型構成子という．以下では，導出型構成子 $F^{(1)}$ を $\delta(F^{(2)})$ で表すことがある． □

このような，デライバによるシグニチャ変換の記述を用いて，型構成子の記述を形式的に次のように定める．

[定義 6] $\langle S, \Sigma, S^{(2)}, \Sigma^{(2)}, \mathcal{D}^{(2)}, F^{(2)}\rangle$ を型構成子，$\delta=\langle f, d\rangle$ を，$S\subseteq S^{(1)}$，$\Sigma_{w,s}\subseteq\Sigma^{(1)}_{w,s}$ $(w\in S^*, s\in S)$ であるシグニチャ $\langle S^{(1)}, \Sigma^{(1)}\rangle$ から $\langle S^{(2)}, \Sigma^{(2)}\rangle$ への $\langle S, \Sigma\rangle$ に関して恒等的なデライバとする．このとき，デライバ δ と型構成子 $F^{(2)}$ との対 $\langle\delta, F^{(2)}\rangle$ を型構成子記述という．この型構成子記述 $\langle\delta, F^{(2)}\rangle$ が表す型構成子は $\delta(F^{(2)})$ である．

この型構成子記述法を用いて，配列とポインタとの対によるスタックの実現を記述してみよう．

[例 3] まず，例1で与えた配列に対応する型構成子の仕様を少し変更して，ポインタ付配列に対応する型構成子の仕様 $\mathcal{S}^{(2)}=\langle S, \Sigma, S^{(2)}, \Sigma^{(2)}, X^{(2)}, \Gamma^{(2)}, B\rangle$ を次のように定める．シグニチャ $\langle S^B, \Sigma^B\rangle$，$\langle S, \Sigma\rangle$ および $\langle S^{(2)}, \Sigma^{(2)}\rangle$ を図-6 (a)，(b)，(c) のように与える．また，変数の集合族 $X^{(2)}$ は，

$X^{(2)}_{\text{ptr-ary}}=\{p_1, p_2, \cdots\}$

$X^{(2)}_{\text{array}}=\{a_1, a_2, \cdots\}$

$X^{(2)}_{\text{index}}=\{i_1, i_2, \cdots\}$

$X^{(2)}_{\text{data}}=\{d_1, d_2, \cdots\}$

$X^{(2)}_{\text{bool}}=\{b_1, b_2, \cdots\}$

であるとし，等式集合 $\Gamma^{(2)}$ は次の等式からなるものとする．

ACCESS (NEWARRAY, i_1)≈ERROR

ACCESS (ASSIGN (a_1, i_1, d_1), i_2)
≈IFD (EQ(i_1, i_2), d_1, ACCESS (a_1, i_2))

ISDEF (NEWARRAY, i_1)≈FALSE

ISDEF (ASSIGN (a_1, i_1, d_1), i_2)
≈IFB (EQ(i_1, i_2), TRUE, ISDEF (a_1, i_2))

PR1 (PAIR (a_1, i_1))≈a_1

PR2 (PAIR (a_1, i_1))≈i_1

PAIR (PR1 (p_1), PR2 (p_1))≈p_1

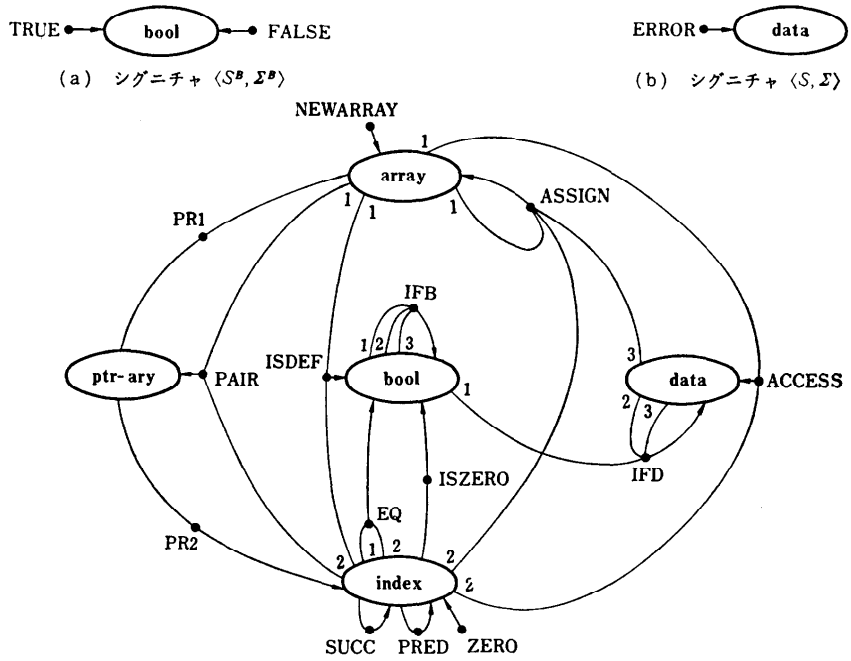
PRED (ZERO)≈ZERO

PRED (SUCC (i_1))≈i_1

ISZERO (ZERO)≈TRUE

---

* 前回[46]に与えたデライバの定義（定義 9）に不完全な点があったので，ここにおわびをするとともに正しい定義を与える．
[定義] $\langle S, \Sigma\rangle$，$\langle S', \Sigma'\rangle$ をシグニチャ，$X=\langle X_s\rangle_{s\in S}$ および $X'=\langle X_s'\rangle_{s\in S'}$ を変数集合族とする．写像 $f: S\to S'$，写像族 $d=\langle d_{w,s}: \Sigma_{w,s}\to T[\Sigma'(X^w)]_{f(s)}\rangle$，および，写像族 $v=\langle v_s: X_s\to X_s'\rangle_{s\in S}$ からなる3項組 $\delta=\langle f, d, v\rangle$ を $\langle S, \Sigma\rangle$ および X から $\langle S', \Sigma'\rangle$ および X' へのデライバという．δ から自然に $T[\Sigma(X)]_s$ から $T[\Sigma'(X')]_{f(s)}$ への写像が定まる．この写像を δ_s 又は δ で表す．なお，$\langle X_s\rangle_{s\in S}$ から $\langle X's'\rangle_{s'\in S'}$ への写像を略しても曖昧さが生じないときは，f と d との対 $\langle f, d\rangle$ を $\langle S, \Sigma\rangle$ から $\langle S', \Sigma'\rangle$ へのデライバということがある．



(a) シグニチャ $\langle S^B, \Sigma^B \rangle$　　(b) シグニチャ $\langle S, \Sigma \rangle$

(c) シグニチャ $\langle S', \Sigma' \rangle$

図-6 シグニチャ $\langle S^B, \Sigma^B \rangle$, $\langle S, \Sigma \rangle$, $\langle S', \Sigma' \rangle$ の図表示

ISZERO (SUCC(i_1))≈FALSE

EQ(i_1, i_1)≈TRUE

EQ(SUCC^n(i_1), RRED^m(i_1))≈FALSE

($n=1, 2, \cdots$, $m=0, 1, 2, \cdots$)

IFA (TRUE, a_1, a_2)≈a_1

IFA (FALSE, a_1, a_2)≈a_2

IFB (TRUE, b_1, b_2)≈b_1

IFB (FALSE, b_1, b_2)≈b_2

IFD (TRUE, d_1, d_2)≈d_1

IFD (FALSE, d_1, d_2)≈d_2

ただし，SUCC^n(i_1) は $\overbrace{\text{SUCC}(\cdots(\text{SUCC}}^{n\text{個}}(i_1))\cdots)$ を表し，PRED^m(i_1) も同様である．

この仕様 $S^{(2)}$ が整数ポインタを伴う配列に対応する型構成子を意味することは上の等式から直観的に分かるであろう．厳密には，それは仕様の意味論の定義に基づいて確認すべきであるが，それをすることは読者にまかせることにする．

さて，次に，前回[48]の例3に示したスタックの仕様を $S^{(1)}=\langle S, \Sigma, S^{(1)}, \Sigma^{(1)}, X^{(1)}, \Gamma^{(1)}, B \rangle$ として，$\langle S^{(1)}, \Sigma^{(1)} \rangle$ および $X^{(1)}$ から $\langle S^{(2)}, \Sigma^{(2)} \rangle$ および $X^{(2)}$ へのデライバ $\delta=\langle f, d, v \rangle$ を次のように定める．

f(**data**)=**data**

f(**bool**)=**bool**

f(**stack**)=**ptr-ary**

d(ERROR)=ERROR

d(TRUE)=TRUE

d(FALSE)=FALSE

d(NEWSTACK)

　=PAIR (NEWSTACK, ZERO)

d(PUSH)=PAIR (ASSIGN (PR1(p_1),

　SUCC (PR2(p_1)), d_1), SUCC (PR2(p_1)))

d(POP)=PAIR (PR1(p_1), PRED (PR2(p_1)))

d(TOP)=ACCESS (PR1(p_1), PR2(p_1))

d(ISNEW)=ISZERO (PR2(p_1))

$v(x_i)=p_i$　($i=1, 2, \cdots$)

$v(y_i)=d_i$　($i=1, 2, \cdots$)

この δ と $ATCF[S^{(2)}]$ から得られる型構成子記述 $\langle \delta, ATCF[S^{(2)}] \rangle$ の表す型構成子 $\delta(ATCF[S^{(2)}])$ が $ATCF[S^{(1)}]$ の実現であることは直観的に理解できるであろう．厳密な検証は次節で与えられる．

3.3 実現の検証

これまでに抽象型構成子の実現の概念の定式化とデライバにもとづく型構成子記述法を与え，実現の検証について論じる準備が整った．

抽象型構成子の実現も，第2回[47]の **4.3** で述べた抽象データタイプの場合と同じように階層的になっているのが普通である．実現したい抽象型構成子の仕様 $\mathcal{S}^{(0)}$ に対して，それが意味する $ATCF[\mathcal{S}^{(0)}]$ を実現する型構成子を，仕様 $\mathcal{S}^{(1)}$ で与えられる別の抽象型構成子 $ATCF[\mathcal{S}^{(1)}]$ 上の型構成子記述 $\langle\delta^{(1)}, ATCF[\mathcal{S}^{(2)}]\rangle$ で記述する．次に，その中で用いられた抽象型構成子 $ATCF[\mathcal{S}^{(1)}]$ を実現する型構成子の記述 $\langle\delta^{(2)}, ATCF[\mathcal{S}^{(2)}]\rangle$ を与える．この過程を次々と必要な回数だけ，すなわち，最終的にすでに存在する型構成子に到達するまで繰り返す．このような階層的な実現に対する検証は，第2回[47]の命題2〜5と同じことが成り立つので，各ステップでの $ATCF[\mathcal{S}^{(i-1)}]$ が $\delta^{(i)}(ATCF[\mathcal{S}^{(i)}])$ で実現されることの検証に帰着される．この各ステップごとの検証においては次の命題が成り立つ．

[命題 2] $\mathcal{S}^{(i)}=\langle S, \Sigma, S^{(i)}, \Sigma^{(i)}, X^{(i)}, \Gamma^{(i)}, B\rangle$ $(i=1,2)$ を抽象型構成子の仕様とし，$\delta=\langle f, d\rangle$ を $\langle S, \Sigma\rangle$ に関して恒等的な $\langle\mathcal{S}^{(1)}, \Sigma^{(1)}\rangle$ から $\langle\mathcal{S}^{(2)}, \Sigma^{(2)}\rangle$ へのデライバとする．このとき，次の条件 (R1) および (R2) が成立すれば $\delta(ATCF[\mathcal{S}^{(2)}])$ は $ATCF[\mathcal{S}^{(1)}]$ の実現である．

(R1) $\mathcal{D}[\mathcal{S}^{(1)}]\subseteq\mathcal{D}[\mathcal{S}^{(2)}]$

(R2) 任意の $D\in\mathcal{D}[\mathcal{S}^{(1)}]$ に対して，
$\delta(ATCF[\mathcal{S}^{(2)}])(D)\models\Gamma^{(1)}$ □

この命題2の条件 (R1) は，$ATCF[\mathcal{S}^{(2)}]$ の定義域が $ATCF[\mathcal{S}^{(1)}]$ のそれより広いことを要請する条件である．また，条件 (R2) は，任意の Σ 代数 $D\in\mathcal{D}^{(1)}$ に対して，$\Gamma^{(1)}$ の中のすべての等式が $\Sigma^{(1)}$ 代数 $\delta(ATCF[\mathcal{S}^{(2)}](D)$ 上で成立することを要請している．条件 (R2) の検証には次の命題が役に立つ．

[命題 3] $\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \delta$ を命題2と同じとし，さらに，(R1) が成立していると仮定する．また，$\lambda\approx\rho$ を $\langle\mathcal{S}^{(1)}, \Sigma^{(1)}, X^{(1)}\rangle$ 上の任意の等式とする．このとき，

(R3) 任意の $\xi\in\bigcup_{s\in S^{(1)}} T[\Sigma^{(1)}(X^{(1)})]_s$ において，
$sort(\xi)\in S$, $sort(x)=sort(\lambda)$
$\Rightarrow\Gamma^{(2)}\vdash\delta(\xi[\lambda/x])\approx\delta(\xi[\rho/x])$

ならば，任意の $D\in\mathcal{D}[^{(1)}]$ に対して，

$\delta(ATCF[\mathcal{S}^{(2)}])(D)$

は $\lambda\approx\rho$ のモデルである．すなわち，

$\delta(ATCF[\mathcal{S}^{(2)}])(D)\models\lambda\approx\rho$

である．

上の条件 (R3) は，$sort(\lambda)\in S$ のときには，それより簡単な条件

(R4) $\Gamma^{(2)}\vdash\delta(\lambda)\approx\delta(\rho)$

と同値である．しかもそのときには，(R3) の検証で，特に $\xi=x$ とすることができるから，そのときの検証条件は (R4) に一致する．しかし，$sort(\lambda)\notin S$ のときは，条件 (R3) は (R4) より弱い条件である．すなわち，(R4) ならば (R3) である．したがって，$sort(\lambda)$ が S のソートであるか否かに従って，(R3) または (R4) の条件を選んで用いるべきである．結局，以上の議論は次の系のようにまとめられる．

[系 1] $\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \delta$ を命題2と同じとし，条件 (R1) が成立しているとする．このとき，$sort(\lambda)\notin S$ であるすべての $\lambda\approx\rho\in\Gamma^{(1)}$ に対して，(R3) または (R4) が成立し，かつ，$sort(\lambda)\in S$ であるすべての $\lambda\approx\rho\in\Gamma^{(1)}$ に対して (R4) が成立するならば，$\delta(ATCF[\mathcal{S}^{(2)}])$ は $ATCF[\mathcal{S}^{(1)}]$ の実現である． □

この系に基づいて例3の実現の検証をしてみよう．

[例 4] 前回[48]のスタックの仕様を $\mathcal{S}^{(1)}=\langle S, \Sigma, S^{(1)}, \Sigma^{(1)}, X^{(1)}, \Gamma^{(1)}, B\rangle$ とおき，$\mathcal{S}^{(2)}$ は **3.2** の例3のポインタ付配列の仕様，δ は例3のデライバとする．まずソートが S に含まれる $\Gamma^{(1)}$ の等式を δ で変換すると以下のようになる．ここで，$|\xrightarrow{\delta}$ は，その左側の等式をデライバ δ で変換するとその右側の等式になることを示す記号である．また，式を短くするため，

$\delta(\text{PUSH}\ (x_1, y_1))=\text{PAIR}\ (\text{ASSIGN}\ (\text{ASSIGN}$
$(\text{PR1}(p_1), \text{SUCC}\ (\text{PR2}(p_1)), d_1), \text{SUCC}\ (\text{PR2}(p_1)))$

を ξ とおく．

$\text{TOP}\ (\text{PUSH}\ (x_1, y_1))\approx y_1$

$|\xrightarrow{\delta}\ \text{ACCESS}\ (\text{PR1}(\xi), \text{PR2}(\xi))\approx d_1$

$\text{TOP}\ (\text{NEWSTACK})\approx\text{ERROR}$

$|\xrightarrow{\delta}\ \text{ACCESS}\ (\text{PR1}\ (\text{PAIR}(\text{NEWARRAY}, \text{ZERO})),$
$\text{PR2}\ (\text{PAIR}(\text{NEWARRAY}, \text{ZERO})))$
$\approx\text{ERROR}$

$\text{ISNEW}\ (\text{NEWSTACK})\approx\text{TRUE}$

$|\xrightarrow{\delta}\ \text{ISZERO}\ (\text{PR2}\ (\text{PAIR}\ (\text{NEWSTACK}, \text{ZERO})))$
$\approx\text{TRUE}$

$\text{ISNEW}\ (\text{PUSH}(x_1, y_1))\approx\text{FALSE}$

$|\xrightarrow{\delta}\ \text{ISZERO}\ (\text{PR2}(\xi))\approx\text{FALSE}$

これらの変換後の等式が $\Gamma^{(2)}$ から，すなわち例3で与えた等式の集合から導かれること，すなわち (R4) が成立することは容易に確かめられるであろう．残る等式は $\text{POP}\ (\text{PUSH}\ (x_1, y_1))\approx x_1$ であり，これに対して (R3) が成立することが，ξ に関する構造帰納法で証明できる．この証明は簡単だがかなり長くなる

ので省略する．証明に際しては，ソート **stack** の $\Sigma^{(1)}(X^{(1)})$ 項 η は，必らず，変数または定数に，

POP　を n_0 回，
PUSH を m_1 回，
POP　を n_1 回，
PUSH を m_2 回，
⋮
PUSH を m_k 回，
POP　を n_k 回

適用した形をしており，それを δ で変換して得られる $\Sigma^{(2)}(X^{(2)})$ 項 $d(\eta)$ と，**図-7** のような形の $\Sigma^{(2)}(X^{(2)})$ 項 η' に対して，$\Gamma^{(2)} \vdash d(\eta) \approx \eta'$ となることを用いるとよいだろう．なお，図-7 では，項 η' を表す木を直接に示すのは大変であるので，共通な部分木をまとめて dag (**directed acyclic graph**) として，項 η' を表している．

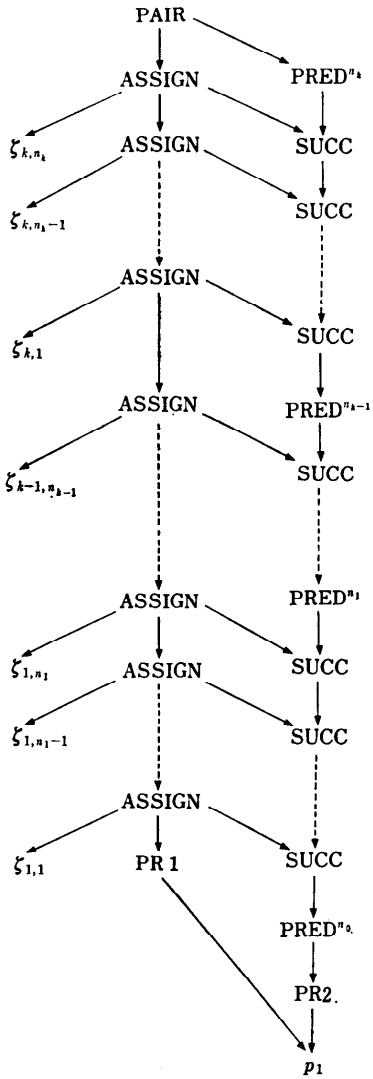


図-7　$d(\eta)$ と等価な $\Sigma^{(2)}(X^{(2)})$ 項 η' の dag 表現

4. 代数的仕様記述法の問題点と今後の展望

本解説では，データ抽象によって得られる概念の代数的仕様記述法に関して現在までに提案された様々な考え方や方法論を整理して，それらに共通する理論的基礎を抽出して解説した．ここで扱ったのは，主として，抽象データ型および抽象型構成子の代数的仕様記述法，実現の概念の定式化，実現の記述法，および，実現の検証法であった．これらはデータ抽象に基づくプログラミング方法論の骨格をなす理論であり，末尾の文献リストにみられるように，それらに関して現在までに数多くの研究がなされている．しかし，それら以外にも多くの重要な問題がある．

ここでは，本文では触れなかった，データ抽象の代数的アプローチに関する重要な問題の代表的なもののいくつかに言及するとともに，それらの問題について今後の展望を試み，本解説のむすびに代える．

4.1 項書き換えシステムと代数的仕様記述法

項書き換えシステムは，書き換え規則と呼ばれる項の対 $\langle \lambda, \delta \rangle$ の集合である*．項書き換えシステム $\mathcal{R}$ に規則 $\langle \lambda, \rho \rangle$ が存在して，$\xi = \zeta[\lambda[\eta_i/x_i]_{i=1}^n/y]$，$\eta = \zeta[\rho[\eta_i/x_i]_{i=1}^n/y]$ であるとき，ξ と η は関係 $\overset{1}{\underset{\mathcal{R}}{\to}}$ にある．すなわち，$\xi \overset{1}{\underset{\mathcal{R}}{\to}} \eta$ であるとして項の間の関係 $\overset{1}{\underset{\mathcal{R}}{\to}}$ が定められる．言い換えると，$\xi \overset{1}{\underset{\mathcal{R}}{\to}} \eta$ であるのは，ξ の部分項で，λ の変数 x_i に η_i を代入した形のものが存在し，それを ρ の変数 x_i に η_i を代入した形の項で書き換えた項が丁度 η になっているときである．この関係 $\overset{1}{\underset{\mathcal{R}}{\to}}$ を含む最小の同値関係を $\overset{*}{\underset{\mathcal{R}}{\to}}$ とする．このとき，$\mathcal{R}$の規則 $\langle \lambda, \rho \rangle$ を等式 $\lambda \approx \rho$ とみなし，規則の集合 $\mathcal{R}$ を等式の集合とみなせば，

$$\xi \overset{*}{\underset{\mathcal{R}}{\to}} \eta \iff \mathcal{R} \vdash \xi \approx \eta$$

である．すなわち，項書き換えシステムは，等式論理の推論を行う体系であると言える．このことから，等式論理に基づく抽象データタイプおよび抽象型構成子の仕様記述法において項書き換えシステムが重要な役割を果たす．

実際，実現の検証では，$\Gamma \vdash \lambda \approx \rho$ を示すことが基本的なステップであり，これは，Γ を項書き換えシ

* 項書き換えシステムについては，文献 27), 43) に良い解説があるので参照されたい．

ステムとみなして，$\lambda \overset{*}{\underset{R}{\to}} \rho$ を示すことである．また，項書き換えシステムの動作を模倣するシステムを構成すれば，仕様から直接的にそれが表す抽象データタイプあるいは抽象型構成子の実現を得ることができる．後で述べる仕様記述言語の処理系は，基本的にはこの考えに基づいて得られる．

しかしながら，代数的仕様記述とそれに基づくソフトウェア開発支援システムへ項書き換えシステムを応用する際に最も重要になる $\lambda \overset{*}{\underset{R}{\to}} \rho$ であるか否かという判定問題（等価性判定問題），$\xi_1 \underset{R}{\to} \xi_2 \underset{R}{\to} \xi_3 \underset{R}{\to} \cdots$ という無限の系列が存在するか否かを判定する問題（非停止性判定問題），あるいは，$\xi \overset{*}{\underset{R}{\to}} \eta$ ならば，$\xi \underset{R}{\to} \xi_1 \underset{R}{\to} \cdots \underset{R}{\to} \zeta \underset{R}{\to} \cdots \underset{R}{\to} \eta_1 \underset{R}{\to} \eta$ であるような ζ が存在するか否かの判定問題（合流性判定問題）はいずれも決定不能であることが知られている[27),43)]．これらの事実は，項書き換えシステムを用いた，実現検証が常に成功するとは限らないこと，また，仕様からの直接的な実現のプログラムが正しく働くことが，すなわち，与えられた引数に対してプログラムが停止し，結果が一意に定まることが一般には保証されないことを意味している．

そこで考えられることは，抽象データタイプ，抽象型構成子の実現の検証や直接実現が首尾よく項書き換えシステムに基づいて行えるための十分条件を与えることである．幸いにして，項書き換えシステムが合流性，停止性を持つための十分条件については詳しく調べられており，いくつかの有用な結果が得られている[27),43)]．それらの結果の種々の十分条件を応用して実現検証システムや直接実現システムを開発する試みはすでにいくつかなされている[31),41),49),66)]．今後もこの方向の研究は，項書き換えシステムの研究と並行して発展していくと考えられる．

4.2 演算の未定義性と例外動作の記述

抽象データ型あるいは抽象型構成子の演算の未定義性や例外動作の厳密な仕様記述は，データ抽象に基づくプログラミングには不可欠である．ところが，現在までに提案された代数的仕様記述法の多くでは，抽象データタイプや抽象型構成子のモデルの基礎として用いる代数の演算が全域関数であるため，演算の未定義性の取り扱いが不自然にならざるを得ず，広く一般に認められる演算の未定義性や例外動作の記述法，取扱い法は見い出されていない．

演算の未定義性あるいは例外動作を取り扱うに際しては二つの立場が考えられる．一つは，未定義である演算，あるいは例外動作を引き起こす演算を適用したとき何が起こるかということを，仕様を書く人が責任を持って記述すべきだとする立場である．この立場に立てば，仕様記述法に特別の工夫をする必要はないが，仕様を書く側に負担がかかる．この負担を軽減するために，例外動作に関する仕様を組織的に与える方法などの研究が行われ，それを応用した仕様記述言語も開発されている[29),31)]．

もう一つは，演算の適用結果が未定義であることを記述できるように仕様の構文と意味論を定める立場であり，等式論理を拡張して部分的代数を扱うことのできる意味論を定め，それに基づいて仕様記述法を与える研究も行われている[9),73)~76)]．このような仕様記述法では，未定義演算を施したとき，あるいは例外動作をさせたとき，何が起こるかということが仕様には記述されない．したがって，抽象データタイプの使用者は，仕様から得られる知識だけに依存してそれを使用するかぎり，決して，未定義演算を適用したり例外動作を引き起こさないと仮定できる．この仮定のもとでは，実現に際して，未定義演算や例外動作を自由に実現できるので，実現の自由度を上げることができる．しかしながら，現実のプログラムの中では，誤って未定義演算を適用することがあるかもしれないと考えるのが普通である．したがって，この立場の仕様記述法には，実行時のエラー検出の機構を用意すべきであり，それに関する研究は今後進める必要があろう．

4.3 仕様記述法，実現記述法の能力の改善

本文で解説した仕様記述法は，その基礎論理として非常に単純な等式論理を採用している．そのことは，意味論が簡単であるとか，検証の機械化が容易であるとかの長所につながるが，反面，記述能力が低いという短所の原因にもなっている．そこで，等式論理に基づく仕様記述法の良さをなるべく保ったまま，その表現能力を改善しようという試みもいくつかなされている．例えば，式として，等式ばかりでなく，

$$\lambda_1 \approx \rho_1, \cdots, \lambda_n \approx \rho_n \Rightarrow \lambda_{n+1} \approx \rho_{n+1}$$

の形の条件付等式を許すものが考えられている[8),20),82)]．この式は，$\Rightarrow$ の左辺にある，$\lambda_i \approx \rho_i$ $(i=1, \cdots, n)$ が成立すれば $\lambda_{n+1} \approx \rho_{n+1}$ が成立するという意味を持つ．このような拡張をしても，仕様の意味論は大きな変更の必要がないことが知られている[20)]．しかし，この拡張された仕様記述法をそのまま形式化した仕様記述言語を考えると，その収理系の開発は等式だけの仕様記述言語の処理系ほど簡単ではない．

また，本文で述べた実現記述法についても，その表

現能力の低さが問題になる．その例としてよく引き合いに出されるのは，第2章の例3で示した仕様が表す抽象型構成子 Symboltable の配列のスタックによる実現であり，本文で与えたデライバの概念に基づく実現記述法では，Symboltable の演算 RETRIEVE を実現できない．このような表現能力の低さを改善するためには，実現記述法に何らかの新しい機能を組み込む必要があり，実際いくつかの試みも行われている．しかし，あまり多くの機能を加えると実現の検証を機械的に行うことが難しくなる．このような，表現能力と機械的検証可能性とのトレードオフを考慮に入れて実現記述法を考えることが肝要である．

いずれにせよ，仕様記述法，実現記述法の妥当な記述能力の増強とそれに対する処理と検証のシステムを開発することは今後の課題である．

4.4 仕様記述言語およびその処理系

本文で紹介した抽象データタイプおよび抽象型構成子の仕様記述法は，具体的な構文が与えられておらず，単に仕様記述の原理を説明しているにすぎない．具体的な仕様記述言語はその原理に基づいて設計されねばならない．形式的に仕様記述を与えることの目的の一つは，検証の自動化と自動プログラム合成であり，それらを実現する際には，仕様を記述するための言語を形式的に定義することが欠かせない．

本文で述べた仕様記述法をそのまま形式化しただけの仕様記述言語では，仕様が書きづらい．仕様記述言語の設計に際しては，仕様の書き易さを考慮に入れるべきであろう．例えば，仕様を階層的に記述できる機能などは是非必要である[12),26),31),81)]．このように，単に等式集合を記述するだけの言語ではなく，仕様の書き易さのための種々の機能を備えた言語の意味論を与えるには，本文で解説した意味論に加えて，その言語に含まれる機能についての意味論も厳密に与えられねばならない．

仕様記述言語は，その構文と意味論が与えられただけでは不十分で，その処理系も与えられてはじめて本来の働きをする．仕様記述言語の処理系の主な仕事は，与えられた仕様から，その仕様の意味する抽象データタイプあるいは抽象型構成子を実現するプログラムを生成することである．すなわち，処理系は，既存のプログラミング言語をターゲット言語とするコンパイラとみなすことができるであろう．このような処理系があれば，具体的な実現を開発するまえに，仕様の段階で，記述されたソフトウェアのテストをすることができ，便利であろう．しかし，一般に，仕様記述言語の処理系を通して得られるプログラムは効率の面で不満な点が多いが，この点を解決すれば，プログラムの自動生成系として充分に役立つようになると思われる．実際に，井上等[49)]は，かなり効率のよいプログラムが得られる処理系について報告している．

参考文献

抽象データタイプの基礎理論関係の主な文献を以下に掲げる．便宜のために，この文献集には，本文中で参照してないものも含まれている．なお，これらの文献は，著者等が手にすることのできたものに限っているので，参照すべきものでこのリストから外れているものがあるが，その点についてはお許しいただきたい．

前回に引き続き，次の略記号を用いる．

CACM : Communications of the ACM
FOCS : Proc. Ann. Symp. Foundations of Computer Science, IEEE
ICALP : Proc. Int. Colloquium on Automata, Languages, and Programming
JACM : Journal of ACM
JCSS : J. Computer and System Sciences
LNCS : Springer Lecture Notes in Computer Science
MFCS : Proc. Int. Symp. Math. Foundations of Computer Science
MST : Mathematical Systems Theory
POPL : Proc. Ann. Symp. Principles of Programming Languages, ACM SIGACT/SIGPLAN
STOC : Proc. Ann. ACM Symp. Theory of Computing
TCS : Theoretical Computer Science

1) Ausiello, G. and Mascari, G. F.: On the Design of Algebraic Data Structures with the Approach of Abstract Data Types, LNCS 72, pp. 514-530 (1979).
2) Bergstra, J. A. and Tucker, J. V.: A Characterization of Computable Data Types by Means of a Finite Equational Specification Method, ICALP 80, LNCS 85, pp. 76-90 (1980).
3) Bergstra, J. A. and Klop, J. W.: Algebraic Specifications for Parameterized Data Types with Minimal Parameter and Target Algebras, ICALP 82, also LNCS 140, pp. 23-34 (1982).
4) Bertoni, A., Mauri, G. and Miglioli, P. A.: A Characterization of Abstract Data as Model-theoretic Invariants, 6th ICALP, LNCS 71, pp. 26-37 (1979).

5) Berztiss, A. T. and Thatte, S.: Specification and Implementation of Abstract Data Types, In "Advances in Computers 22" (M. C. Yovits ed.), Academic Press, pp. 295-353 (1983).
6) Blum, E. K. and Parisi-Presicce, F.: Implementation of Data Types by Algebraic Methods, JCSS, Vol. 27, pp. 304-330 (1983).
7) Brown, E. J.: On the Application of Rothon Diagrams to Data Abstraction, SIGPLAN Notices, Vol. 18, No. 12, pp. 17-24 (1983).
8) Broy, M., Dosch, W., Partsch, H., Pepper, P. and Wirsing, M.: Existential Quantifiers in Abstract Data Types, 6th ICALP, LNCS 79, pp. 73-87 (1979).
9) Broy, M. and Wirsing, M.: Partial Abstract Types, Acta Informatica, Vol. 18, pp. 47-64 (1982).
10) Burmeister, P.: Quasi-equational Logic for Partial Algebras, Fundamentals of Computation Theory, Proc. of the International FCT-Conference (1981), also LNCS 117, pp. 71-80 (1981).
11) Burstall, R. M. and Goguen, J. A.: Putting Theories Together to Make Specifications, Proc. 5th Int. Confr. Artificial Intelligence, pp. 1045-1058 (1977).
12) Burstall, R. M. and Goguen, J. A.: Semantics of CLEAR, A Specification Language, Proc. 1979 Copenhagen Winter Sch. Abstr. Softw, Spec, LNCS 86 (1980).
13) Cristian, F.: Robust Data Types, Proc. of a Workshop on Program Specification (1981), also LNCS 134, pp. 215-254 (1981).
14) Ehrich, H.-D.: Extensions and Implementations of Abstract Data Type Specifications, 7th MFCS (1978), LNCS 64, pp. 155-164 (1978).
15) Ehrich, H.-D.: Specifying Algebraic Data Types by Domain Equations, Fundamentals of Computation Theory, Proc. of the 1981 International FCT-Conference, also LNCS 117, pp. 120-136 (1981).
16) Ehrich, H.-D.: On the Theory of Specification, Implementation, and Parameterization of Abstract Data Types, JACM 29, No. 1, pp. 206-227 (1982).
17) Ehrig, H., Kreowski, H.-J. and Padawitz, P.: Stepwise Specification and Implementation of Abstract Data Types, 5th ICALP, LNCS 62, pp. 205-226 (1978).
18) Ehrig, H., Kreowski, H.-J., Mahr, B. and Padawitz, P.: Compound Algebraic Implementation: An Approach to Stepwise Refinement of Software Systems, 9th MFCS, LNCS 88, pp. 231-245 (1980).
19) Ehring, H., Kreowski, H.-J. and Padawitz, P.: A Case Study of Abstract Implementations and their Correctness, 4e Coll. Int. sur la programmation, Paris, LNCS 83, pp. 282-297 (1980).
20) Ehrig, H., Kreowski, H.-J., Thatcher, J. W., Wagner, E. G. and Wright, J. B.: Parameterized Data Types in Algebraic Specification Language, 7th ICALP, LNCS 85, pp. 157-168 (1980).
21) Ehrig, H. and Mahr, B.: Complexity of Algebraic Implementations for Abstract Data Types, JCSS 23, pp. 223-253 (1981).
22) Ehrig, H., Kreowski, H.-J., Thatcher, J. Wagner, E. and Wright, J.: Parameter Passing in Algebraic Specification Languages, Proc. of a Workshop on Program Specification (1981), also LNCS 134, pp. 322-369 (1981).
23) Ehrig, H., Kreowski, H.-J., Mahr, B. and Padawitz, P.: Algebraic Implementation of Abstract Data Types, TCS 20, pp. 209-263 (1982).
24) Ehrig, H. and Kreowski, H.-J.: Parameter Passing Commutes with Implementation of Parameterized Data Types, ICALP 82, also LNCS 140, pp. 197-211 (1982).
25) Engels, G., Pletat, U. and Ehrich, H.-D.: An Operational Semantics for Specification of Abstract Data Types with Error Handling, Acta Informatica, 19, pp. 235-253 (1983).
26) Futatsugi, K. and Okada, K.: A Hierarchical Structuring Method for Functional Software Systems, The 6th International Conference on Software Engineering, pp. 393-402 (1982).
27) 二木, 外山: 項書き換型計算モデルとその応用, 情報処理, Vol. 24, No. 2, pp. 133-146 (1983).
28) Giarratana, V., Gimona, F. and Montanari, U.: Observability Concepts in Abstract Data Type Specifications, 5th MFCS, LNCS 45, pp. 576-587 (1976).
29) Goguen, J. A., Thatcher, J. W. and Wagner, E. G.: An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types, IBM Reserch Report RC 6487 (1976), also Current Trends in Programming Methodology IV: Data Structuring (R. Yeh ed.), Prentice Hall, pp. 80-144 (1978).
30) Goguen, J. A.: How to Prove Algebraic Inductive Hypotheses without Induction, 5th Confr. on Automated Deduction, LNCS 87, pp. 356-373 (1980).
31) Goguen, J. A.: OBJ-1, a Study in Executable Algebraic Formal Specification, Technical Report, SRI International (1980).
32) Goguen, J. A. and Meseguer, J.: Completeness of Many-sorted Equational Logic,

SIGPLAN Notices 16, No. 7, pp. 9-17 (1981).
33) Goguen, J. A. and Parseye-Ghomi, K.: Algebraic Denotational Semantics Using Parameterized Abstract Modules, Proc. International Colloquium on Formalization of Programming Concepts (1981), also LNCS 107, pp. 292-309 (1982).
34) Goguen, J. and Meseguer, J.: Universal Realization, Persistent Interconnection and Implementation of Abstract Modules, ICALP 82, also LNCS 140, pp. 265-281 (1982).
35) Greiter, G.: A Data Type Theory, SIGPLAN Notices 17, No. 5, pp. 46-53 (1983).
36) Gristian, F.: Robust Data Types, Acta Informatica 17, pp. 365-397 (1982).
37) Guttag, J. V., Horowitz, E. and Musser, D. R.: The Design of Data Type Specification, 2nd Int. Confr. on Software Engineering, ACM/IEEE 1976, pp. 414-420 (1976), also University of Southern California, Research Report ISI/RR-76-49 (1976).
38) Guttag, J. V.: Abstract Data Types and the Development of Data Structures, CACM 20, pp. 396-404 (1977).
39) Guttag, J. V. and Horning, J. J.: The Algebraic Specification of Abstract Data Types, Acta Informatica 10, pp. 27-52 (1978).
40) Guttag, J. V., Horowitz, E. and Musser, D. R.: Abstract Data Types and Software Validation, University of Southern California, ISI/RR-76-48 (1976), also CACM 21, pp. 1048-1063 (1978).
41) Hoffmann, C. M. and O'Donnell, M. J.: Programming with Equations, ACM Trans. Prog. Lang. Sys., Vol. 4, No. 1 (1982).
42) Hornung, G. and Raulefs, P.: Terminal Algebra Semantics and Retractions for Abstract Data Types, 7th ICALP, LNCS 85, pp. 310-323 (1980).
43) Huet, G. and Oppen, D. C.: Equations and Rewrite Rules, Formal Language Theory: Perspectives and Open Problems (Edited by R. V. Book), Academic Press (1980).
44) Hupbach, U. L.: Abstract Implementation of Abstract Data Type, 7th MFCS, LNCS 88, pp. 291-304 (1980).
45) Hupbach, U. L., Kaphengst, H. and Reichel, H.: Initial Algebraic Specification of Data Types, Parameterized Data Types and Algorithms, VEB Robotron ZFT, Techn. Rep., Dresden (1980).
46) 稲垣, 坂部: 一抽象データタイプの代数的仕様記述法の基礎(1)一多ソート代数と等式論理, 情報処理, Vol. 25, No. 1, pp. 47-53 (1984).
47) 稲垣, 坂部: 一抽象データタイプの代数的仕様記述法の基礎(2)一抽象データタイプ, 情報処理, Vol. 25, No. 5, pp. 491-501 (1984).
48) 稲垣, 坂部: 一抽象データタイプの代数的仕様記述法の基礎(3)一抽象型構成子のモデルと始代数意味論, 情報処理, Vol. 25, No. 7, pp. 708-716 (1984).
49) 井上, 関, 杉山, 嵩: 関数的プログラミング言語 ASL/F のコンパイル時における最適化, 信学会, 技術研究報告, EC 82-18 (1982).
50) Julliand, J.: Algebraic Specification of Communication between Parallel Processes, Technique et Science Informatiques, Vol. 2, No. 4, pp. 257-269 (1983).
51) Kamin, S.: Final Data Types and their Specification, INRIA Rapports de Recherche, No. 47 (1980).
52) Kanda, A.: Data Types as Initial Algebras: A Unification of Scottery and ADJery, 19th FOCS, pp. 221-230, (1978).
53) Kaphengst, H.: What is Computable for Abstract Data Types, Fundamentals of Computation Theory, Proc. of the International FCT-Conference (1981), also LNCS 117, pp. 173-181 (1981).
54) 嵩, 谷口, 杉山, 萩原, 鈴木, 奥井: プログラムの仕様記述の代数的方法について, 信学会, 技術研究報告, AL 78-5 (1978).
55) 川原林, 太田, 大山口, 稲垣: 代数的仕様記述に基づいたソフトウェアの正当性証明システム, 信学会, 技術研究報告, AL 81-107 (1982).
56) Laut, A. and Partsch, H.: Tuning Algebraic Specifications by Type Merging, Proc. of 5th International Symposium on Programmming (1982), also LNCS 137, pp. 283-304 (1982).
57) Lehmann, D. J. and Smyth, M.: Data Types, 18th FOCS, pp. 7-12, (1977).
58) Lehmann, D. J. and Smyth, M.: Algebraic Specification of Data Types: A Synthetic Approach, MST 14, pp. 97-139 (1981).
59) Leszczylowski, J. and Wirsing, M.: A System for Reasoning within and about Algebraic Specifications, Proc. of 5th International Symposium on Programming (1982), also LNCS 137, pp. 257-282 (1982).
60) Liskov, B. and Zilles, S. N.: Specification Techniques for Data Abstraction, IEEE Transaction on SE, Vol. SE-1, No. 1 (1975).
61) Majster, M. E.: Data Types, Abstract Data Types and their Specification Problem, TCS, Vol. 8, pp. 89-127 (1979).
62) Mayoh, B. H.: Data Types as Functions, 7th MFCS (1978), LNCS 64, pp. 56-70 (1978), Extended Version: Aarhus Report DAIMI pb-89.

63) Moitra, A.: Direct Implementation of Algebraic Specification of Abstract Data Types, IEEE Transaction on Software Engineering, Vol. SE-8, No. 1 (1982).
64) Musser, D.R.: On Proving Inductive Properties of Abstract Data Types, 7th POPL (1980).
65) Musser, D.R. and Kapur, D.: Rewrite Rules Theory and Abstract Data Type Analysis, Proc. of European Computer Algebra Conference (1982), also LNCS 144, pp. 77-90 (1982).
66) Musser, D.R.: Abstract Data Type Specification in the AFFIRM System, IEEE, Trans. Softw. Eng., Vol. SE-6, No. 1, pp. 24-32 (1982).
67) Nourani, F.: A Model-theoretic Approach to Specification, Extension, and Implementation 4e Coll, Int. sur la Programmation, Paris 1980, LNCS 83, pp. 282-297 (1980).
68) Nourani, F.: Abstract Implementations and their Correctness Proof, JACM, Vol. 30, No. 2, pp. 343-359 (1983).
69) 大山口：抽象データタイプの代数的仕様とその実現，信学会，技術研究報告，AL 82-26 (1982).
70) Oyamaguchi, M.: Relationship between Abstract and Concrete Implementations of Algebraic Specifications, Proc. IFIP 83.
71) Padwitz, P.: New Results on Completeness and Consistency of Abstract Data Types, 9th MFCS, LNCS 88, pp. 460-473 (1980).
72) Pair, C.: Abstract Data Types and Algebraic Semantics of Programming Languages, TCS 18, pp. 1-31 (1982).
73) 坂部，稲垣，本多：部分関数を演算とする抽象データタイプの仕様記述と実現，信学会，技術研究報告，AL 80-6 (1980).
74) 坂部，稲垣，本多：データタイプ構成子の代数的仕様記述，信学会，技術研究報告，AL 81-106 (1982).
75) Sakabe, T., Inagaki, Y. and Honda, N.: Specification of Abstract Data Types with Partially Defined Operations, The 6th International Conference on Software Engineering (1982).
76) 坂部，稲垣，本多：パラメタ付抽象データタイプの代数的仕様記述と実現，信学会，技術研究報告，AL 82-41 (1982).
77) Sannella, D. and Wirsing, M.: Implementation of Parameterized Specifications, ICALP 82, also LNCS 140, pp. 473-488 (1982).
78) 杉山，谷口，嵩：代数的記述言語とその部分言語としての関数的プログラミング，信学会，技術研究報告，AL 79-99 (1980).
79) 杉山，谷口，嵩：基底代数を前提とする代数的仕様記述，信学会，論文誌 (D), Vol. J64-D, No. 4, pp. 324-331 (1981).
80) 杉山，鈴木，谷口，嵩：あるクラスの項書き換え系の効率のよい実行，信学会，論文誌 (D), Vol. J 65-D, No. 7, pp. 858-865 (1982).
81) 杉山，谷口，嵩：代数的仕様記述言語 ASL/1 ―文法とその意味の定義―，信学会，技術研究報告，AL 82-62 (1982).
82) Thatcher, J.W., Wagner, E.G. and Wright, J.B.: Data Type Specification: Parameterization and Power of Specification Techniques, 10th STOC (1978), pp. 119-132 (1978), IBM Res. Rep. RC-7757 (1979).
83) 富樫，原尾，野口：データタイプの代数的仕様記述に関する決定問題，信学会，技術研究報告，AL 82-25 (1982).
84) 鳥居，二木，真野：プログラミング方法論の展望，情報処理，Vol. 20, No. 1, pp. 22-43 (1979).
85) 和田，萩原，荒木，都倉：代数的仕様記述に関する一考察，信学会，技術研究報告，AL 79-111 (1980).
86) Wand, M.: Final Algebra Semantics and Data Type Extensions, JCSS 19, pp. 27-44 (1979).
87) Wand, M.: Specifications, Models, and Implementations of Data Abstractions, TCS 20, pp. 3-32 (1982).
88) Weihl, W. and Liskov, B.: Specification and Implementation of Resilient, Atomic Data Types, ACM SIGPLAN NOTICES, 18, 6, pp. 53-60 (1983), Proc. of the SIGPLAN '83 Symposium on Programming Language Issues in Software Systems.
89) White, J.R.: On the Multiple Implementation of Abstract Data Types within a Computation, IEEE Transaction on SE, Vol. SE-9, No. 4, pp. 395-411 (1983).
90) Wirsing, M. and Broy, M.: Abstract Data Types as Lattices of Finitely Generated Models, 9th MFCS, LNCS 88, pp. 673-685 (1980).
91) Wirsing, M., Pepper, P., Partsch, H., Dosch, W. and Broy, M.: On Hierarchies of Abstract Data Types, Acta Informatica 20, pp. 1-33 (1983).
92) Zilles, S.N.: An Introduction to Data Algebras, LNCS 86, pp. 249-272 (1979).
93) Zilles, S.N., Lucas, P. and Thatcher, J.W.: A Look at Algebraic Specifications, IBM Research Report, RJ 3568 (1982).

(昭和59年7月16日受付)