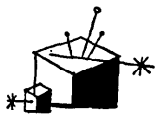
講　座



——抽象データタイプの代数的仕様記述法の基礎（1）——
多ソート代数と等式論理†

稲垣康善††　坂部俊樹††

1. はじめに

抽象データタイプの概念は，ソフトウェアの信頼性ならびにプログラミング方法論の観点からプログラミングにおける強力なツールであると考えられるようになり，多数の研究が活発に行われている．筆者等がこれまでに知っているそれらの文献のうち代表的なものが末尾に掲げてある．特に，文献14)には，ソフトウェア方法論における抽象データタイプとその位置づけの良い解説が含まれている．

抽象データタイプは，**モジュール化**，**抽象化**，**情報隠蔽** (information hiding)，**局所化** (localization) といったプログラミング方法論における概念ないしは原理の実現のための一翼をになう有力な概念として認められる一方で，最近，抽象データタイプを厳密にかつ形式的に仕様記述する方法として**代数的仕様記述法**が提案され，注目を集めている．厳密に仕様記述ができて始めて，客観的な情報交換の手段になりうるばかりでなく，その実現や検証が形式的すなわち機械的にできるようになる．仕様記述ができなければ，何を意図したプログラムであるかさえ分からないのであるから形式的な検証などということは望むべくもないのである．

一般に，仕様記述法に望まれる性質としては，(1) 数学的に厳密な定義が与えられかつ検証の基礎となり，形式的（機械的）な取り扱いに耐える**形式性** (formality) を持つこと，(2) 書き易くかつ読み易いこと，すなわち，**記述性** (constructibility) と**理解性** (conprehensibility) を持つこと，(3) 必要最小限の情報しか含まず，また，具体的な操作手順や実現に関する詳細を含まないこと，などがあげられる．さらに，(4) **広適用性**や (5) **拡張性**を要請していることもある[8]．

抽象データタイプの代数的仕様記述法は，これらの性質，特に，(1)～(3) を有していると考えられ，それに基づいて形式的に抽象データタイプを扱うことができ，また，その厳密な理論的基礎は，種々のアプローチがあるとはいえ，次第に確立されつつあると言えよう．この代数的仕様記述法では，抽象データタイプを代数（の同形なクラス）とみなし，それを公理と呼ばれる等式の集合で仕様記述することが基本となっている．しかし，仕様の意味論，すなわち，仕様が何を表わしているか，言い換えれば，仕様がどのような抽象データタイプを意味しているかの定め方については種種の考え方が提案されている．意味論が異なれば，当然，それに従って，抽象データタイプの実現ならびにその検証の概念も異なってくる．これらについても，種々の提案と考究が行われている．

本講座では，このような抽象データタイプの代数的仕様記述の理論的基礎を整理して，厳密に，しかし，できるだけ分かり易く解説し，抽象データタイプの代数的仕様記述法に関する研究の現状と問題点についてふれることにする．

抽象データタイプの代数的仕様記述法の理論を支えるのは，**多ソート代数**と**等式論理**である．本稿では，本講座で必要な範囲に限ってこれらについて説明し，次回以降の解説の準備をする．すなわち，次の **2.** で多ソート代数について，**3.** で等式論理について，それらの定義と関連する諸概念について解説する．

2. 多ソート代数

2.1 多ソート代数の基礎概念

従来，普通に考えられている代数は，**ユニバース** (universe) ないしは**台** (carrier) と呼ばれる一つの集合と，その上で働くいくつかの演算 (operation) とから定義されている．例えば，2値ブール代数は，真理値 **true** と **false** の集合 {**true**, **false**} とその上で定

† Many-Sorted Algebra and Equational Logic by Yasuyoshi INAGAKI and Toshiki SAKABE (Faculty of Engineering, Nagoya University).

†† 名古屋大学工学部

義される ∧ (and), ∨ (or) と ¬ (not) の演算とから定義されている.

ところが, 例えば, 実数xのn乗を計算する演算をEXP(x, n)と書けば, 演算 EXP は, 実数xと整数nを引数として実数 x^n を値として返す演算である. このようなそれぞれが異なる種類の値をとる引数を持つ演算が含まれているシステムは, ユニバースを一つに定めて議論することはできない. このようなシステムを扱うための一般化として, 複数の台を有する代数が考えられている. このような代数は, **多ソート代数** (many-sorted algebra) ないしは **混交代数** (heterogeneous algebra) と呼ばれている.

多ソート代数Aは, 代数の台と呼ばれる集合 A_s の族とそれらの集合の間の演算の集合とで成り立っている. 台 A_s の添字 s は, その台がどの種類の値の集合であるかを示す名前である. この名前を**ソート** (sort) という. 代数Aの台のソートの集合がSであるとき, Sソート代数と呼ばれることもある. 計算機科学の分野では, このような代数は, ごく普通にでてくる. 例えば, ソートの集合Sを {**bool, int, real**} とすると, Sソート代数の一つとして, 台が $A_{\mathbf{bool}}=\{\text{true, false}\}$, $A_{\mathbf{int}}=\{0, \pm1, \pm2, \cdots\}$, $A_{\mathbf{real}}=R$ (実数の集合) であり, それらの間の演算として EXP: $A_{\mathbf{real}}\times A_{\mathbf{int}}\to A_{\mathbf{real}}$, COND: $A_{\mathbf{bool}}\times A_{\mathbf{real}}\times A_{\mathbf{real}}\to A_{\mathbf{real}}$ を有する代数を考えることができる. COND(b, x, y) は **if** b **then** x **else** y に対応する演算である. この代数のソートと演算の関係は, **図-1** のような図で表わすと直観的にわかり易い. この図では, 円の中にソートが記入され, 複数本の尾を持つ矢印が演算を表わす. 矢印の先は, 演算の結果のソートを示しており, 矢印の尾は, 演算の引数のソートを示している. 尾に付いている番号は, 引数の順番を表わしている.

多ソート代数は, 形式的には, 以下のように定義される.

[定義 1] 任意の**ソート** (sort) の集合をSとする. S上の**演算記号領域** (operator domain) Σ は, 集合の族 $\langle\Sigma_{w,s}\rangle_{w\in S^*, s\in S}$ である. ここに, S^* はSの元からなるすべての有限系列の集合である. $\alpha\in\Sigma_{w,s}$のとき, arity$(\alpha)=w$, sort$(\alpha)=s$, あるいは, $\alpha: w\to s$ と書き, α は**アリティ** (arity) w, **ソート** s の**演算記号** (operator symbol) であるという. すなわち, $w=s_1s_2\cdots s_n$ とすれば, α の第 $1, 2, \cdots, n$ 引数のとる値のソートは, それぞれ, $s_1, s_2, \cdots, s_n$ であり, α が返す値のソートは s である. 特に, $w=\varepsilon$ (空系列) のときは, α は**定数記号** (constant symbol) と呼ばれる. ソートの集合 S と演算記号領域との対$\langle S, \Sigma\rangle$ を**シグニチャ*** (signature) という. 混乱の恐れがないときには, $\langle S, \Sigma\rangle$ を単に Σ で表わすことがある. □

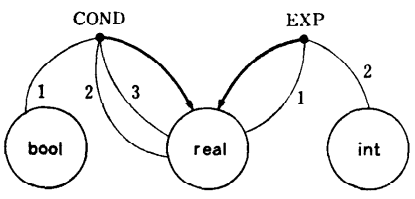


図-1 多ソート代数のソートと演算の関係

実は, 図-1は, ソートの集合 $S=$ {**bool, int, real**}, 演算記号領域 Σ が $\Sigma_{\mathbf{bool\ real\ real, real}}=\{\text{COND}\}$, $\Sigma_{\mathbf{real\ int, real}}=\{\text{EXP}\}$, その他の $\Sigma_{w,s}$ は空集合であるようなシグニチャ $\langle S, \Sigma\rangle$ を表わしている図である.

α が $\Sigma_{w,s}$ の演算記号で, $w=s_1s_2\cdots s_n$ であれば, α は**図-2** のような複数の出発点と一つの行先を持つ矢印で表わされる. Sが有限集合, 各 $\Sigma_{w,s}$ が有限集合であり, しかも, 空ではない $\Sigma_{w,s}$ の個数が有限であれば, シグニチャは**図-2**のような図表示ができる.

例えば,

$S=$ {**int, bool**}

$\Sigma_{\varepsilon,\mathbf{int}}=\{\text{ZERO}\}$

$\Sigma_{\varepsilon,\mathbf{bool}}=\{\text{TRUE, FALSE}\}$

$\Sigma_{\mathbf{int,int}}=\{\text{SUCC, PRED}\}$

$\Sigma_{\mathbf{bool\ int\ int, int}}=\{\text{COND}\}$

$\Sigma_{\mathbf{int\ int, bool}}=\{\text{EQ}\}$

であるようなシグニチャ $\langle S, \Sigma\rangle$ は**図-3**のように表わされる.

多ソート代数はシグニチャの各ソートに集合を割り当て, 各演算記号に, それらの集合上の関数あるいは定数を割り当てて得られるものである. 例えば, 図-3

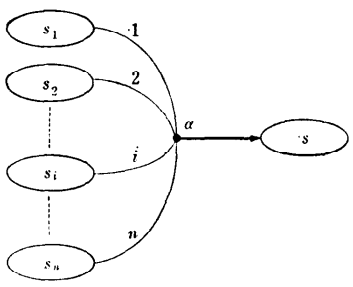


図-2 $\alpha\in\Sigma_{s_1s_2\cdots s_n, s}$ の図表示

* 楽譜の調音記号を signature というが, ここでもそれと同様な意味合で用いられている.

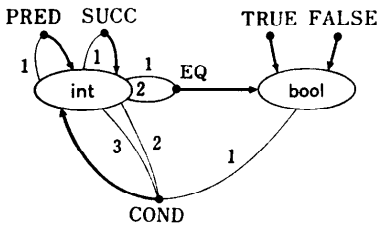


図-3　シグニチャの図式表示

のシグニチャ $\langle S, \Sigma\rangle$ について考える．ソート **int**, **bool** に対して，それぞれ，

$I_{\mathbf{int}} = \{0, \pm1, \pm2, \cdots\}$

$I_{\mathbf{bool}} = \{\text{true}, \text{false}\}$

を対応させ，演算記号 ZERO, PRED, SUCC, EQ, COND, TRUE, FALSE に対応する関数あるいは定数を，

$\text{ZERO}_I = 0$

$\text{PRED}_I(n) = n-1$

$\text{SUCC}_I(n) = n+1$

$$\text{EQ}_I(n, m) = \begin{cases}\text{true} ; & n = m \\ \text{false} ; & n \neq m\end{cases}$$

$$\text{COND}_I(b, n, m) = \begin{cases} n ; & b = \text{true} \\ m ; & b = \text{false}\end{cases}$$

$\text{TRUE}_I = \text{true}$

$\text{FALSE}_I = \text{false}$

として，一つの多ソート代数 I が得られる．

多ソート代数は，形式的には，次のように定義される．

[定義 2] $\langle S, \Sigma\rangle$ をシグニチャとする．Σ **代数** A は次の 3 項目からなる：

（1） 各ソート $s \in S$ に対する非空集合 A_s．A_s は，ソート s の台と呼ばれる．

（2） 各演算記号 $\alpha: s_1 s_2 \cdots s_n \to s$ $(n>0)$ に対する関数 $\alpha_A: A_{s_1} \times A_{s_2} \times \cdots \times A_{s_n} \to A_s$．

（3） 各定数記号 $\alpha: \varepsilon \to s$ に対する A_s の元 α_A．

Σ 代数は，しばしば，多ソート代数，または単に，代数といわれる．□

2.2 多ソート代数の準同形写像と合同関係

多ソート代数に関連する最も重要な概念として**準同形写像**と**合同関係**がある．この両者は密接に関係し合っていて，いわゆる準同形定理に対応する定理が成立する．

まず，合同関係を定義する．Σ 代数上の合同関係は，台となっている集合上の同値関係の族で，演算がその各同値関係を保存するようなものである．すなわち，互いに同値関係にある引数に演算を施した結果はやはり同値であるような同値関係を合同関係という．厳密な定義は次の通りである．

[定義 3] $\langle S, \Sigma\rangle$ をシグニチャ，A を Σ 代数とする．A_s 上の関係 $\equiv_s \subseteq A_s \times A_s$ からなる関係族 $\equiv = \langle \equiv_s \rangle_{s \in S}$ は，次の条件を満たすとき，Σ 代数 A 上の**合同関係** (congruence relation) といわれる．

（1） 各ソート $s \in S$ に対して，$\equiv_s$ は A_s 上の同値関係である．

（2）* Σ の任意の演算記号 $\alpha: s_1 s_2 \cdots s_n \to s$ $(n>0)$ に対して，$a_i \equiv_{s_i} b_i$ $(i=1, 2, \cdots, n)$ ならば，$\alpha_A(a_1, a_2, \cdots, a_n) \equiv_s \alpha_A(b_1, b_3, \cdots, b_n)$ が成り立つ．

Σ 代数 A 上の合同関係 $\equiv$ のもとで同値な要素を同一視することによって，もう一つの代数 B が得られる．この Σ 代数 B はいわゆる**商代数**である．

[定義 4] $\langle S, \Sigma\rangle$ をシグニチャ，A を Σ 代数，$\equiv$ を A 上の合同関係とする．このとき，次に定められる Σ 代数 B を A の $\equiv$ による**商代数**という．

（1） 各ソート $s \in S$ に対して，$B_s = A_s/\equiv_s$.

（2） 各 $\alpha: \varepsilon \to s$ に対して，$\alpha_B = \equiv_s[\alpha_A]$.

（3） 各 $\alpha: s_1 \cdots s_n \to s$ に対して，$\alpha_B(\equiv[a_1], \cdots, \equiv[a_n]) = \equiv_s[\alpha_A(a_1, \cdots, a_n)]$.

ここに，$A_s/\equiv_s$ は集合 A_s の同値関係 $\equiv_s$ による商集合 $\{\equiv_s[a] \mid a \in A_s\}$，$\equiv_s[a]$ は a の $\equiv_s$ 同値類 $\{b \mid a \equiv_s b,\ b \in A_s\}$ である．文脈から s や $\equiv_s$ が明らかなときには，$\equiv_s[a]$ を単に $\equiv[a]$ あるいは $[a]$ と書くことがある．また，以下では，A の $\equiv$ による商代数 B を $A/\equiv$ と書く．□

合同関係と商代数の簡単な例を示そう．前述の Σ 代数 I を考える．今，k を任意の正整数に固定して，$I_{\mathbf{int}}$ 上の同値関係 $\equiv_{\mathbf{int}}$ を，

$$n \equiv_{\mathbf{int}} m \Leftrightarrow \begin{cases} n,\ m \text{ を } k \text{ で割ったときの余り} \\ \text{mod}(n, k) \text{ と } \text{mod}(m, k) \text{ が等しい．}\end{cases}$$

で定める．また，$I_{\mathbf{bool}}$ の同値関係 $\equiv_{\mathbf{bool}}$ は

$b \equiv_{\mathbf{bool}} b' \Leftrightarrow b = b'$

によって定める．明らかに，$\equiv = \langle \equiv_{\mathbf{int}}, \equiv_{\mathbf{bool}} \rangle$ は I 上の合同関係である．容易に分かるように，I の $\equiv$ による商代数 $I/\equiv$ は次のようになる．

$(I/\equiv)_{\mathbf{int}} = \{\equiv_{\mathbf{int}}[0], \equiv_{\mathbf{int}}[1], \cdots, \equiv_{\mathbf{int}}[k-1]\}$

$(I/\equiv)_{\mathbf{bool}} = \{\{\text{true}\}, \{\text{false}\}\}$

$\text{ZERO}_{I/\equiv} = \equiv_{\mathbf{int}}[0]$

$\text{TRUE}_{I/\equiv} = \{\text{true}\}$

$\text{FALSE}_{I/\equiv} = \{\text{false}\}$

---

* a_i と b_i の対 $\langle a_i, b_i\rangle$ が関係 $\equiv_{s_i}$ に含まれることを $a_i \equiv_{s_i} b_i$ と書く．

$$\mathrm{SUCC}_{I/\equiv}(\equiv_{\mathrm{int}}[n])=\equiv_{\mathrm{int}}[n+1]$$

$$\mathrm{PRED}_{I/\equiv}(\equiv_{\mathrm{int}}[n])=\equiv_{\mathrm{int}}[n-1]$$

$$\mathrm{EQ}_{I/\equiv}(\equiv_{\mathrm{int}}[n],\equiv_{\mathrm{int}}[m])=\begin{cases}\{\mathrm{true}\}\ ;\ n\equiv_{\mathrm{int}}m\\ \{\mathrm{false}\}\ ;\ \text{その他}\end{cases}$$

$$\mathrm{COND}_{I/\equiv}(b,x,y)=\begin{cases}x\ ;\ b=\{true\}\\ y\ ;\ b=\{\mathrm{false}\}\end{cases}$$

次にΣ代数の間の準同形写像を定義する．よく知られているように，単一ソートの代数間の準同形写像は，その写像と代数の演算とが可換であることで定義される．多ソート代数の場合も，これを自然に拡張して次のように定義される．

[定義 5] $\langle S,\Sigma\rangle$ をシグニチャとし，A,B をΣ代数とする．次の条件を満たす写像の族 $h=\langle h_s: A_s\to B_s\rangle_{s\in S}$ をAからBへの**準同形写像**という．

(1) Σの任意の**演算記号** $\alpha: s_1\cdots s_n\to s$ に対して，$h_s\circ\alpha_A=\alpha_B\circ(h_{s_1}\times\cdots\times h_{s_n})$.

(2) Σの任意の**定数記号** $\alpha:\varepsilon\to s$ に対して，$h_s(\alpha_A)=\alpha_B$.

ただし，$\circ$は関数合成を表わし，$h_{s_1}\times\cdots\times h_{s_n}: A_{s_1}\times\cdots\times A_{s_n}\to B_{s_1}\times\cdots\times B_{s_n}$ は，$h_{s_1}\times\cdots\times h_{s_n}(\langle a_1,\cdots,a_n\rangle)=\langle h_{s_1}(a_1),\cdots,h_{s_n}(a_n)\rangle$ で定められる関数である．すべての $s\in S$ に対して h_s が上への［1対1の］写像であるときhは**上への［1対1の］準同形写像**であるといわれる．さらに，1対1かつ上への準同形写像は**同形写像**といわれる． □

前述のΣ代数Iおよび$I/\equiv$を考える．写像 $h_{\mathrm{int}}: I_{\mathrm{int}}\to(I/\equiv)_{\mathrm{int}}$ および $h_{\mathrm{bool}}: I_{\mathrm{bool}}\to(I/\equiv)_{\mathrm{bool}}$ を，

- すべての $i\in I_{\mathrm{int}}$ に対して，$h_{\mathrm{int}}(i)\equiv_{\mathrm{int}}[i]$
- すべての $b\in I_{\mathrm{bool}}$ に対して，$h_{\mathrm{bool}}(b)=\{b\}$

と定める．このとき，容易に確かめられるように，写像族 $h=\langle h_{\mathrm{int}},h_{\mathrm{bool}}\rangle$ はIから$I/\equiv$の上への準同形写像である．

上述の事柄は，一般に任意のΣ代数について成立する．すなわち，A,BをΣ代数とするとき，次の(1)，(2)，(3)が成り立つ．

(1) $\equiv$がA上の合同関係ならば，上述の例のようにして，$\equiv$から自然に定められるAから$A/\equiv$の上への準同形写像 $h^{\equiv}$ が存在する．

(2) hがAからBへの準同形写像ならば，hから自然に定められるA上の合同関係 $\equiv^h$ が存在する．すなわち，$\equiv^h=\langle\equiv^h_s\rangle_{s\in S}$ は，任意の A_s の要素 a,b に対して，$a\equiv^h_s b\Leftrightarrow h_s(a)=h_s(b)$ によって定められる．

(3) hがAからBの上への準同形写像ならば，$A/\equiv^h$ とBは同形である．

特に，これらのうちの(3)は，いわゆる準同形定理に対応する結果である．

これらに関するさらに厳密な議論は，本講座の範囲では必要ないので省略するが，詳しくは例えば文献11)などを参照されたい．

2.3 始代数と終代数

抽象データタイプの代数的仕様記述の理論，特に，その意味論を論じる際には，**始代数** (initial algebra) および**終代数** (final algebra，または，terminal algebra) の概念がしばしば重要な役割をする．これらの用語は，カテゴリ論における**始対象**と**終対象**の概念からでてきたものである．

同じシグニチャ $\langle S,\Sigma\rangle$ を持つΣ代数はいくつでも無数に考えられるが，Σ代数の任意のクラス$\mathcal{A}$は，$\mathcal{A}$に含まれるΣ代数間のすべての準同形写像のクラス$\mathcal{M}$を**射** (morphism) のクラスとする**カテゴリ** (category) $C=\langle\mathcal{A},\mathcal{M}\rangle$ をなす．

事実，(1)準同形写像の合成はやはり準同形写像である，(2)準同形写像の合成は結合律を満たす，さらに，(3)$\mathcal{A}$の各代数Aの台の上の恒等関数の族 $1_A=\langle 1_{A_s}\rangle_{s\in S}$ は準同形写像であり，かつ，準同形写像の合成に関する単位元である，すなわち，任意の準同形写像 $h: A\to B$ に対して，$h\circ 1_A=1_B\circ h=h$，であることを示すことができる．

始代数と終代数は次のように定義される．

[定義 6] Σ代数のカテゴリ $C=\langle\mathcal{A},\mathcal{M}\rangle$ において，Σ代数Aから，$\mathcal{A}$のすべてのΣ代数への唯一の準同形写像が存在するとき，AはCの**始代数**であるといわれる．これと双対に，$\mathcal{A}$のすべてのΣ代数からΣ代数Bへの唯一の準同形写像が存在するとき，BはCの**終代数**であるといわれる． □

例えば，すべてのΣ代数からなるカテゴリの終代数は，各ソートの台が単一要素集合であるΣ代数である．また，始代数は，各ソートの台がそのソートに属するすべての項*からなる集合で，演算αが，引数 $t_1,\cdots,t_n$ を項 $\alpha(t_1,\cdots,t_n)$ へ写す写像として解釈される代数（後に $T[\Sigma]$ と書かれる）である．

始代数ならびに終代数は，同形な代数を同一視すれば，いずれも唯一に定まる．抽象データタイプの代数的仕様記述の理論で始代数，終代数の概念が有用であるのは，この性質によるのである．

ここで，これ以上，カテゴリ論を説明することは，

* 項の定義は次章の定義8を参照されたい．

本講座の範囲を越えるので，カテゴリ論については例えば文献5),10)を参照されたい．本講座では，いたずらに道具だてを大きくすることを避け，分かり易さを第一に考えて，カテゴリ論の言葉をできるだけ使わないで解説する方針である．後に，始代数，終代数についても，それらに対応する概念が必要になるところで別に定義されるであろう．しかし，それらが，定義6で与えた始代数，終代数になっていることを確認することは，抽象データタイプの代数的仕様記述法の理解を深める上で必要である．

3. 等式論理

等式論理は無矛盾かつ完全であるばかりでなく，きわめて直観的である．そのため，その構文と意味論の区別がしばしばあいまいにされる．しかし，実現の正当性証明システムの具体化などを考える際には構文と意味論の区別を常に意識していることが必要である．また，等式論理の存在をはっきり意識すれば，より複雑な論理に基づく仕様記述法や実現の検証法も考え易くなるであろう．そこで，本講座でも，等式論理の構文と意味論をはっきり区別して解説をすすめるので，この章で，等式論理についてやや詳しく説明する．なおこの章の内容は主に文献11)に従ったものである．

等式論理は，第一階論理の式 (formula) を等式だけに制限した論理である．まず，その構文である等式言語を定義する．

[定義 7] **等式言語**は4項組 $\langle S, \Sigma, X, \approx\rangle$ である．ここに，$\approx$は**等号** (equality symbol) と呼ばれる論理記号である．S, Σ は，$\langle S, \Sigma\rangle$ がシグニチャとなる集合と集合族である．すなわち，Sはソートの集合，Σ は演算記号領域である．X は $X_s \cap X_{s'} = \phi$ $(s \neq s')$ であるような記号の可算無限集合 X_s の族 $\langle X_s\rangle_{s\in S}$ である．X_s の元をソート s の変数という． □

この定義からすぐ分かるように，等式言語は第1階言語（第1階論理の構文）から等号以外の論理記号 ($\wedge, \vee, \neg, \forall, \exists$ など) および関係（又は述語）記号を取り除き，演算記号を多ソートの場合へ拡張したものになっている．したがって，等式言語における項 (term) や式 (formula) は上の制限あるいは拡張を除外すれば第1階言語と同様に定義される．すなわち，

[定義 8] $\langle S, \Sigma, X, \approx\rangle$ を等式言語とする．Y を $Y_s \subseteq X_s$ であるような任意の変数集合族 $\langle Y_s\rangle_{s\in S}$ とする．このとき，各ソート s に対して，$T[\Sigma(Y)]_s$ を次の(1),(2)を満たす最小の集合とする．

(1) $\Sigma_{\varepsilon,s} \cup Y_s \subseteq T[\Sigma(Y)]_s$

(2) $\alpha \in \Sigma_{s_1\cdots s_n, s}$ かつ $\xi_i \in T[\Sigma(Y)]_{s_i}$ $(i=1,\cdots,n)$ ならば記号系列 $\alpha(\xi_1,\cdots,\xi_n)$ は $T[\Sigma(Y)]_s$ の元である．

$T[\Sigma(Y)]_s$ の元をソート s の $\Sigma(Y)$ **項**又は Y 上の項という．特にYがXであるときは，略して，ソート s の項ということがある．任意の項 $\xi \in T[\Sigma(X)]_s$ に対して，ξの中に出現するソート s の変数の集合を $\mathrm{var}_s(\xi)$ で表わす．また，$\mathrm{var}(\xi) = \langle \mathrm{var}_s(\xi)\rangle_{s\in S}$ とする． □

[定義 9] 等式言語 $\langle S, \Sigma, X, \approx\rangle$ における**式**は記号系列 $\xi \approx \eta$ である．ここに，ξ, η は同じソートの $\Sigma(X)$ 項である．以下では，この式を**等式**と呼ぶ． □

$\Sigma(X)$ 項は Σ の演算の合成を表わす．例えば，図-3 のシグニチャ $\langle S, \Sigma\rangle$ に対する等式言語 $\langle S, \Sigma, X, \approx\rangle$ においては，$Y_{\mathbf{int}} = \{y_1, y_2\}$，$Y_{\mathbf{bool}} = \{y_3\}$ とすると，$\mathrm{SUCC}(y_2)$, $\mathrm{COND}(y_3, \mathrm{SUCC}(y_1), y_1)$ などはソート **int** の Y 上の項であり，$\mathrm{EQ}(\mathrm{SUCC}(y_2), y_1)$, $\mathrm{EQ}(y_1, \mathrm{ZERO})$ などはソート **bool** の Y 上の項である．また，$\mathrm{SUCC}(y_2) \approx \mathrm{COND}(y_3, \mathrm{SUCC}(y_1), y_1)$ は等式の例である．

なお，$\Sigma(X)$ 項は，木の形で図表示すると分かり易い．例えば，上に例示した項は**図-4** のように表わされる．

次に，等式論理における推論を形式化する．よく知られているように，等式論理では，反射則，対称則，推移則，代入則および置換則に基づいて推論が行われる．ここでは，それを次のように定義する．

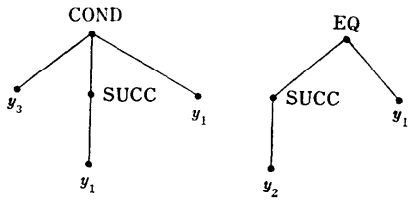
[定義 10] $\mathcal{L} = \langle S, \Sigma, X, \approx\rangle$ を等式言語，Γ を $\mathcal{L}$ の等式の集合とする．このとき，等式の集合 $\mathcal{E}(\Gamma)$ を，次の条件を満たす最小の等式集合 Δ と定める．

(1) $\Gamma \subseteq \Delta$

(2) $\xi \approx \xi \in \Delta$ (反射則)

(3) $\xi \approx \eta \in \Delta$ ならば，$\eta \approx \xi \in \Delta$ (対称則)

(4) $\xi \approx \eta \in \Delta$ かつ $\eta \approx \zeta \in \Delta$ ならば，$\xi \approx \zeta \in \Delta$



(a) COND $(y_3, \mathrm{SUCC}(y_1), y_1)$　(b) EQ $(\mathrm{SUCC}(y_2), y_1)$

図-4　項の木表現の例

(推移則)

(5) $\xi\approx\eta\in\Delta$ ならば，$\xi[\zeta/x]\approx\eta[\zeta/x]\in\Delta$ (代入則)

(6) $\xi\approx\eta\in\Delta$ ならば，$\zeta[\xi/x]\approx\zeta[\eta/x]\in\Delta$ (置換則)

ここに，$\zeta[\xi/x]$ は，項ζの中に出現する変数xを同時に項ξで置き換えて得られる項を表わす．等式φが$\mathcal{E}(\Gamma)$に属するとき，φはΓから**導出**されるといい，$\Gamma\vdash\varphi$と書く．

例として，図-3のシグニチャ$\langle S,\Sigma\rangle$に対する等式言語$\langle S,\Sigma,X,\approx\rangle$を考えよう．今，$\Gamma=\{$PRED(SUCC($x$))$\approx x$, COND(TRUE, x, y)$\approx x$, COND(FALSE, x, y)$\approx y\}$とおく．ただし，x, yはソート **int** の変数である．このとき，例えば，$\Gamma\vdash$COND(TRUE, PRED(SUCC(ZERO)), SUCC(ZERO))$\approx$ZEROとなる．これは，Γの中の第1の式のxにZEROを代入し(条件(5))，その結果得られる等式の両辺をΓの中の第2の式のxにそれぞれ代入する(条件(5), (6))ことによって得られる．

最後に，等式論理の意味論を定義する．等式論理は述語記号のないような第1階論理であるので，その解釈を与えることは，多ソート代数を与えることに他ならない．そして，等式が多ソート代数上で成り立つのは，その代数上での等式の両辺の評価値が互いに等しいときかつその時に限るとして等式の解釈が定められる．これを形式的に定義するために，変数へ値を割り当てたときの項の評価値を定める．

[定義 11] $\langle S,\Sigma,X,\approx\rangle$を等式言語，$A$を任意の$\Sigma$代数とする．任意の$Y(\subseteq{}^*X)$に対して，写像族$\theta=\langle\theta_s: Y_s\to A_s\rangle_{s\in S}$を$Y$の$A$上での**割当** (assignment)という．θから次のようにして，写像族$\bar{\theta}=\langle\bar{\theta}_s: T[\Sigma(Y)]_s\to A_s\rangle_{s\in S}$を定める．すなわち，任意の$\xi\in T[\Sigma(Y)]_s$に対して，

$$\bar{\theta}_s(\xi)=\begin{cases}\theta_s(x);\ \xi=x\in Y_s\ \text{のとき}\\ \alpha_A;\ \xi=\alpha\in\Sigma_{\varepsilon,s}\ \text{のとき}\\ \alpha_A(\bar{\theta}_{s_1}(\xi_1),\cdots,\bar{\theta}_{s_n}(\xi_n));\ \xi=\alpha(\xi_1,\cdots,\xi_n)\ \text{かつ}\ \mathrm{arity}(\alpha)=s_1\cdots s_n\ \text{のとき}\end{cases}$$

ここに，α_Aは演算記号αに対するA上の演算である． □

$\bar{\theta}_s(\xi)$がξの中に出現する変数にθで割り当てられる値を代入してξをA上で評価して得られる値であることは，直観的に明らかであろう．したがって，等式の解釈は次のように定義される．

[定義 12] $\mathcal{L}=\langle S,\Sigma,X,\approx\rangle$を等式言語，$A$を$\Sigma$代数，$\lambda\approx\rho$を$\mathcal{L}$のソート$s\in S$の等式とする．$X$に対する$A$上での任意の割当$\theta$に対して$\bar{\theta}(\lambda)=\bar{\theta}(\rho)$であるとき，かつ，その時に限り，$A$は$\lambda\approx\rho$の**モデル** (model)である，$A$は$\lambda\approx\rho$を**満たす**，あるいは，$\lambda\approx\rho$は$A$上で**成り立つ**といい，$A\models\lambda\approx\rho$と書く．等式の集合$\Gamma$に対しては，$\Gamma$のすべての等式が$A$上で成り立つときかつその時に限り$A$は$\Gamma$のモデルである，$A$は$\Gamma$を満たす，あるいは，$\Gamma$は$A$上で成り立つといい，$A\models\Gamma$と書く．さらに，$\Gamma$のすべてのモデルが等式$\varphi$のモデルであるとき，すなわち，$\Gamma$のいかなるモデル上でも$\varphi$が成り立つとき，$\Gamma\models\varphi$と書く． □

例えば，$\overbrace{\mathrm{SUCC}(\mathrm{SUCC}(\cdots\mathrm{SUCC}}^{k\text{個}}(x)\cdots))\approx x$という等式は$I$上では成り立たないが，$I/\equiv$上では成り立つ．また，{PRED(SUCC($x$))$\approx x$}$\models$SUCC(PRED(SUCC($x$)))$\approx$SUCC($x$)であることも容易に確かめられる．

等式論理に関する最も重要な定理はその無矛盾性と完全性を主張している次の結果である．

[定理 1] 等式言語$\langle S,\Sigma,X,\approx\rangle$の任意の等式集合$\Gamma$と等式$\varphi$に対して，$\Gamma\vdash\varphi$であれば$\Gamma\models\varphi$である(無矛盾性)．逆に，$\Gamma\models\varphi$であれば$\Gamma\vdash\varphi$である(完全性)．すなわち，$\Gamma\vdash\varphi$と$\Gamma\models\varphi$は互いに必要十分な条件である． □

この定理の証明は，例えば文献11)第24章を参照されたい．

$\Gamma\vdash\varphi$ならびに$\Gamma\models\varphi$の意味を考えると分かるように，等式論理では，構文と意味論をはっきり意識しなくても問題が生じないことは，この定理が保証している．

4. おわりに

本稿では，抽象データタイプの理論の基礎である多ソート代数と等式論理について解説をした．これらの概念に基づいて次回とその次の回で，抽象データタイプの代数的仕様記述法の基礎を与える理論を解説する予定である．

なお，今回，末尾に掲げた抽象データタイプに関する参考文献は，代表的なものだけであるが，その他にも多くの関連論文がある．それらの文献は次回以降にとりあげる．

---

* $Y\subseteq X$ は，$\forall s\in S, Y_s\subseteq X_s$ を表わす．

参考文献

1) Birkhoff, G. and Lipton, J. D.: Heterogeneous Algebra, Journal of Combinatorial Theory, Vol. 8, pp. 115-133 (1970).
2) Burstall, R. M. and Goguen, J. A.: Putting Theories Together to Make Specifications, Proc. 5-th Int. Confr. on Artificial Intelligence, pp. 1045-1058 (1977).
3) Ehrich, H. D.: On the Theory of Specification, Implementation, and Parameterization of Abstract Data Types, J. ACM, Vol. 29, No. 1, pp. 206-227 (1982).
4) Ehrig, H., Kreowski, H. J. and Padwitz, P.: Stepwise Specification and Implementation of Abstract Data Types, in Lecture Notes in Computer Science 62 (eds. G. Goos and J. Hartmanis), pp. 205-226 (1978).
5) Goguen, J. A., Thatcher, J. W., Wagner, E. G. and Wright, J. B.: An Introduction to Categories, Algebraic Theories and Algebras, IBM Research Report, RC 5369 (1975).
6) 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 Research Report, RC 6487 (1976).
7) Guttag, J. V., Horowitz, E. and Musser, D. R.: Abstract Data Types and Software Validation, Comm. ACM, Vol. 21, pp. 1048-1063 (1978).
8) Liskov, B. and Zilles, S. N.: Specification Techniques for Data Abstraction, IEEE Trans. Softw. Eng., Vol. SE-1, No. 1 (1975).
9) Majster, M. E.: Data Types, Abstract Data Types and Their Specification Problem, Theor. Comput. Sci., Vol. 8, pp. 89-127 (1979).
10) Mitchel, B.: Theory of Categories, ACADEMIC PRESS, New York, London (1965).
11) Monk, J. D.: Mathematical Logic, Springer-Verlag, New York, Heidelberg, Berlin (1976).
12) 杉山，谷口，嵩：基底代数を前提とする代数的仕様，信学論 (D), Vol. J 64-D, No. 4, pp. 324-331 (1981).
13) Thatcher, J. W., Wagner, E. G. and Wright, J. B.: Data Type Specification: Parametrization and Power of Specification Techniques, IBM Research Report RC 7757 (1979).
14) 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).
15) Wand, M.: Final Algebra Semantics and Data Type Extensions, J. Comput. Syst. Sci., Vol. 19, pp. 27-44 (1979).

（昭和58年8月8日受付）