クラスと部分構造を持つ対象を扱う、論理に基づく単純な枠組の提案

川口雄一*・赤間清**・宮本衛市***

A Simple Computational Framework Based on Logic for Objects with Classes and Substructures

Yuuichi KAWAGUCHI, Kiyoshi AKAMA and Eiichi MIYAMOTO

要　旨

本研究の目的は、クラスと部分構造を持つような対象を論理プログラミングにおいて、柔軟かつ単純に扱う計算体系を構築することにある。我々は既にそのような計算体系を例を用いて示した。本論文の目的は、この計算体系に形式的な定義を与えることにある。

Abstract

The purpose of this study is to construct a new calclus which handle a object with a class and substructures simply and powerfully on the logic programming. We have shown such a calclus by some examples. The purpose of this paper is to give a formal definition for that calclus.

1. はじめに

本研究では，クラスと部分構造を持つような計算の対象を論理プログラム上で柔軟かつ単純に扱う計算体系を提供することを目的としている。そのような計算体系として[6]を提案した。本論文の目的はこの計算体系を，宣言型計算モデル[1]に基づき，形式的に定義することにある。

2. 宣言型計算モデル

宣言型計算モデル[1]では，問題領域を「特殊化システム」という構造の上に据えるところに特徴がある。特殊化システムとは四つ組$<\mathcal{A},\mathcal{G},\mathcal{S},\mu>$で表わされ，以下のように定義されている。

定義1(特殊化システム)

1. $\mathcal{A},\mathcal{G},\mathcal{S}$は集合である。
2. $\mu:\mathcal{S}\rightarrow partial\_map(\mathcal{A})$[†1]
3. $\forall s_1,s_2\in\mathcal{S},\exists s,\in\mathcal{S}:\mu(s)=\mu(s_1)\circ\mu(s_2)$
4. $\exists s\in\mathcal{S},\forall a\in\mathcal{A}:\mu(s)(a)=a$
5. $\mathcal{G}\subset\mathcal{A}$

[†1] $partial_map(\mathcal{A})$は$\mathcal{A}$上の部分写像の全体を表わす。

集合$\mathcal{A}$の元を論理対象，$\mathcal{G}$の元を基底対象，$\mathcal{S}$の元を特殊化と呼ぶ。宣言型計算モデルでは集合$\mathcal{A}$の元の形を特に規定していない。これは例えば通常の論理プログラムの論理で例えば原始論理式の形をpred(arg,・・・)のように固定していることと対照的である。

宣言型計算モデルに基づいて計算を行う場合，まず，計算の対象の形を問題領域に応じて定め($\mathcal{A}$)，そのうちの基底となる対象を定める($\mathcal{G}$)。そして論理対象の特殊化を定める($\mathcal{S}$, μ)。この特殊化を用いて例えば単一化を定めることができる。

本論文では宣言型計算モデルに基づき，この特殊化構造を定める。

3. 論理対象の定義

アルファベットとして，以下の互いに素な集合を用意する。

- **C**：定数の集合(*e.g.* num,list,append,...)
- **V**：対象変数の集合(*e.g.* X,Y,A,...)
- **U**：クラス変数の集合(*e.g.* α,β,...)

例に示したように，以降では英子文字からなる文字列で定数を，英大文字からなる文字列でクラス変数をそれぞれ表わす。

このアルファベットから，特殊化システムにお

＊　講　師　情報工学科
＊＊　北海道大学　工学部
＊＊＊　北海道大学　工学部

ける論理対象の集合$\mathcal{A}$を構築する。

3.1 定数の子孫関係

定数の集合Cの各元は，ユーザにより親子関係を与えられる。この親子関係から，以下に定義する定数間の子孫関係が自然に導かれる。

定義2(子孫の関係)

定数S, $T \in C$に関して，以下の各条件のいずれかを満たす場合，SをTの子孫と呼ぶ。

(1)$S=T$

(2)Sの親の定数S'が存在し，S'はTの子孫である。

この定義から導かれる定数間の子孫関係は，ループを含む一般的なグラフ構造をなす。しかし，本論文では子孫関係が木構造になっている場合のみを考える。

SがTの子孫である場合，$T \rightarrow S$と表わす。また，省略記法として$T \rightarrow S_1, \ldots, T \rightarrow S_n$をまとめて

$$T \rightarrow S_1 \mid \cdots \mid S_n$$

と表わす。

3.2 クラス対象の定義

定義3(クラス対象) クラス対象をBNFを用いて定義する。

＜定数＞∈ C

＜クラス変数＞∈ U

＜クラス＞::=“(”＜定数＞*“)”

＜クラス対象＞::=

　＜クラス変数＞“：”＜クラス＞

クラスを構成する定数の個数を，クラスの長さという。例えば，クラス(a b c)の長さは3であり，()の長さは0である。クラス対象に対して下線部のクラスのことを，クラス対象の持つクラスという。

定義4(クラス間の階層関係) 定数に関する子孫関係→を用いて，クラス間の階層関係を定める。二つのクラス$S=(S_1 \cdots S_m)$と$T=(T_1 \cdots T_n)$ $(0 \leq n, m)$が$T \geq S$の関係にあることを以下のように定義する。

(1)$n \leq m$かつ

(2)$1 \leq \forall i \leq n$, $T_i \rightarrow S_i$

定数の場合と同様に，$T \geq S$となっているとき，SをTの子孫のクラスと呼ぶ。

あるクラス対象α:(・・・)において，クラス変数αを特に明記する必要のない場合，省略記法としてクラス対象からクラス変数を省いて単に(・・・)と記述することができる。

3.3 論理対象の定義

定義5(論理対象) 論理対象をBNFを用いて定義する。

＜対象変数＞∈ V

＜部分構造＞::=

　(＜クラス対象＞｜＜論理対象＞)*

＜論理対象＞::=＜対象変数＞“：”

　“[”＜クラス対象＞＜部分構造＞“]”

論理対象に対して，下線部のクラス対象のことを，論理対象の持つクラスという。

ある論理対象X:[・・・]において，対象変数Xを特に明記する必要のない場合，省略記法として論理対象から対象変数を省いて単に[・・・]と記述することができる。また，一つの論理対象中において同一の対象変数やクラス変数には同一の情報が付くものとする。この仮定の下，省略記法として，一つの論理対象中における二回目以降の対象変数やクラス変数には，以下の例のように“：”に続く情報を書かないことができる。

$e.g.$

[(・・・) M:[・・・]M], [α:(・・・) [α・・・]]

以上で特殊化システムにおけるAを構成できた。

3.4 論理対象に対する制約

各論理対象は，部分構造の(a)個数，(b)順番，(c)クラスに関して制約を受ける。この制約は，論理対象どうしの単一化を行う時点で参照される。この制約を構造制約と呼ぶ。

構造制約の(c)に関して，部分構造がクラス対象であれば，そのクラスがそのまま制約となる。部分構造が論理対象である場合は，その論理対象の持つクラスが制約となる。部分構造である論理対象の持つ部分構造までは制約としない。また，構造制約に関しては，(a)～(c)の条件のみでなく，さらに様々な制約を記述することも考えられるが，どこまでを制約とするか現在のところ明らかではない。本論文では，上記の(a)～(c)を構造制約とする。

クラスMを持つ論理対象が，部分構造として$S_1, \ldots, S_n$を持つことを次のように記述する。

$$M == c(S_1) \oplus c(S_2) \oplus \cdots \oplus c(S_n)$$

なお，$c()$は部分構造からクラス，及び対象の種別を取り出す関数で，以下のように定義されている。

$$c(S)=\begin{cases} m & \text{where } S=\alpha:m \\ [m] & \text{where } S=X:[\alpha:m\ o_1\cdots o_k] \end{cases}$$

例えば$O=X:[\alpha:(\mathrm{a\ b\ c})\mathrm{M1\ M2}]$である場合，

$$c(O)=[(\mathrm{a\ b\ c})]$$

となる。

クラスMを持つ論理対象が，部分構造を全く持たないという構造制約を次のように記述する。

$$M==$$

構造制約の左辺に現れるクラスのうち，$(m\ a\ b)$，$(m\ c\ d)$，...のように，クラスを構成する一つ目の定数が等しいものは，省略記法として$(m\ *s\ *t)$のようにまとめて記述する。ここで$*s$，$*t$は任意の定数を表わす変数である。この変数を構造変数と呼ぶ。構造変数を用いた記法は省略記法である。クラスの長さが不定なクラスを表わすために$(m\ .\ *t)$のようなドット記法を用いることができる。この場合，変数$*t$は任意の長さのクラスを表わす。構造変数は左辺に現れている限り，右辺にも現れることができるものとする。構造変数は構造制約を参照する時点で全て具体化されなければならない。

4. 基底対象の定義

定義6(基底対象) 論理対象のうち，以下の条件を満たすものを基底対象と呼ぶ。

(1) クラスを構成する定数全てが，子孫関係の木において葉の位置に存在する，かつ

(2) 構造制約を満足する。

以上で特殊化システムにおける$\mathcal{G}$を構成できた。この$\mathcal{G}$を用いて，本計算体系に対して宣言的な意味を与えることができる。

5. 特殊化の定義

論理対象どうしの計算は，単一化に基づいて行われる。この単一化を行うためにまず，論理対象を各構成要素である変数，クラス，部分構造毎に操作する特殊化とよばれる演算を導入する。これは特殊化システムにおける$\mathcal{S}$の元である。

この特殊化を定義するために，まず基本特殊化を導入する。基本特殊化の合成により特殊化を定義する。

5.1 基本特殊化の定義

以下の記法で定義される(s1)～(s4)を基本特殊化として導入する。

(s1) ＜クラス変数＞-⊖＜クラス変数＞

(s2) ＜クラス変数＞▷＜クラス対象＞

(s3) ＜対象変数＞-⊖＜対象変数＞

(s4) ＜対象変数＞⟹＜論理対象＞

論理対象をMとし，基本特殊化をθとした場合，θのMへの適用を$M\{\theta\}$と表わす。

定義7(s1) x-⊖yはクラス変数xの名前をyに付け替える。(s1)を用いた例を以下に示す。

$$\mathrm{X}:[\alpha:(\mathrm{cons})\,[(2)]\,[\alpha]]\{\alpha\text{-}\ominus\beta\}$$
$$\triangledown$$
$$\mathrm{X}:[\beta:(\mathrm{cons})\,[(2)]\,[\beta]]$$

定義8(s2) $x\triangleright c$はクラス変数xで示されたクラス対象のクラスを，新しいクラスcに変える。この場合，元々xに付いていたクラスをSとすると，$S\geq c$が成立していなければならない。すなわち，この基本特殊化はクラスをその子孫にのみ変えることができる。(s2)を用いた例を以下に示す。

$$\mathrm{X}:[\alpha:(\mathrm{num})]\{\alpha\triangleright(5)\}$$
$$\triangledown$$
$$\mathrm{X}:[\alpha:(5)]$$

定義9(s3) x-⊖yは対象変数xの名前をyに付け替える。(s3)を用いた例を以下に示す。

$$\mathrm{M}:[(\mathrm{append})\ \mathrm{X}:[(\mathrm{list})]\ \mathrm{X\ Y}]\{\mathrm{X}\text{-}\ominus\mathrm{A}\}$$
$$\triangledown$$
$$\mathrm{M}:[(\mathrm{append})\ \mathrm{A}:[(\mathrm{list})]\ \mathrm{A\ Y}]$$

定義10(s4) $x\Longrightarrow m$は対象変数xで示された論理対象に，部分構造としてmを追加する。部分構造を追加する場合，既に含まれている部分対象の末尾に付け加える。(s4)を用いた例を以下に示す。

$$\mathrm{X}:[(\mathrm{cons})\,[(\mathrm{a})]]\{\mathrm{X}\Longrightarrow\mathrm{Z}:[(\mathrm{nil})]\}$$
$$\triangledown$$
$$\mathrm{X}:[(\mathrm{cons})\,[(\mathrm{a})]\,\mathrm{Z}:[(\mathrm{nil})]]$$

5.2 特殊化の定義

基本特殊化の0個以上の列を特殊化と呼ぶ。例えば$\theta_1,\ldots,\theta_n(n\geq 0)$をそれぞれ基本特殊化とした場合，これらから構成される特殊化を以下のよう

に表わす。

$$<\theta_1, \ldots, \theta_n> \in \mathcal{S}$$

n＝0の場合，＜＞を単位特殊化と呼ぶ。

特殊化sの論理対象Mへの適用は特殊化システムにおいては$\mu(s)(M)$と表わされる。これを簡単のため$M \circ s$と表わすことにする。この適用の結果は以下のように定義される。

定義11(特殊化の適用)

$$M \circ s = \begin{cases} M \quad \text{where } s = <> \\ (M\{\theta_1\}) \circ <\theta_2, \ldots, \theta_n> \\ \quad \text{where } s = <\theta_1, \ldots, \theta_n>, n \geq 1 \end{cases}$$

定義12(特殊化の連結演算) 二つの特殊化$s = <\theta_1, \ldots, \theta_n>$，$t = <\sigma_1, \ldots, \sigma_m>$ $(n, m \geq 0)$に対して，連結演算“・”を以下のように定義する。

$$s \cdot t = <\theta_1, \ldots, \theta_n, \sigma_1, \ldots, \sigma_m>$$

6. 論理対象同士の単一化の定義

本章では特殊化を用いて，論理対象同士の単一化を定義する。論理対象同士の単一化演算子を，次のように定義する。

定義13(単一化演算子) 計算対象A, Bに対する単一化演算子とは，以下を満たす特殊化の順序対(Φ, Ψ)のことである。

$$A \circ \Phi = B \circ \Psi$$

次節以降では，この単一化演算子を求めるための手続きを示す。

6.1 最汎共通子孫(MGCD)

二つのクラスの共通な子孫のクラスのうち，最も一般的なものを最汎共通子孫(MGCD, Most General Common Descendant)という。最汎共通子孫は次のように定義される。

定義14(最汎共通子孫) クラスA, Bに対し，以下の条件を満たすクラスUをAとBの最汎共通子孫という。

(1) $A \geq U, B \geq U$
(2) $\forall U' \in \{M \mid A \geq M, B \geq M\}, U \geq U'$

クラスA, Bの最汎共通子孫を次のように表わす。

$$\Upsilon(A, B)$$

二つのクラス$S = (s_1 \cdots s_m)$, $T = (t_1 \cdots t_n)$の最汎共通子孫$\Upsilon(S, T)$は以下の手続きにより求められる。

1. $k = \min(m, n)$, $l = \max(m, n)$
   $l = 0$の場合$\Upsilon(S, T) = ()$となり，手続き成功，終了。
2. $1 \leq \forall i \leq k$に対し，以下のように各U_iを定める。
   - $s_i \rightarrow t_i \Rightarrow u_i := t_i$
   - $t_i \rightarrow s_i \Rightarrow u_i := s_i$
   - それ以外⇒手続き失敗、終了。
3. $k < \forall i \leq l$に対して，以下のように各u_iを定める。
   - $n < m \Rightarrow u_i := s_i$
   - $m < n \Rightarrow u_i := t_i$
4. $\Upsilon(S, T) = (u_1 \cdots u_l)$を得る。手続き成功，終了。

6.2 制約充足の手続き

本節では論理対象が，その論理対象に対する構造制約を満足するように，論理対象を特殊化する手続きを示す。次に示す論理対象

$$X : [\alpha : (c_1 \cdots c_n)\ O_1 \cdots O_m]$$

に対する制約充足の手続きを以下に示す。制約充足の手続きに成功した場合，構造制約を充足させるために用いた特殊化を得る。

1. $m = 0$の場合は制約充足の手続きは成功し，特殊化$\theta = <>$を得る。
2. クラス$(c_1 \cdots c_n)$に対応する構造制約を探す。見つからなかった場合は制約充足の手続きは成功し，特殊化$\theta = <>$を得る。
3. 見つかった構造制約を

$$\mathrm{M} = = \mathrm{S}_1 \oplus \cdots \oplus \mathrm{S}_k$$

とする。$k \neq m$の場合は，構造制約に違反するものとし，手続きは失敗する。以降では$k = m$とし，添字としてmを使用する。
なお，Mや$S_i (1 \leq i \leq m)$の中に構造変数が現れていたとしても，構造制約を探索する時点でM中の構造変数が各$c_i (1 \leq i \leq n)$で具体化され，それに伴いS_i中の構造変数も具体化され，最終的には構造変数は全て具体化される。
4. 構造制約の右辺から，次の特殊化を得る。各部分構造$O_i (1 \leq i \leq m)$の持つクラスに付けら

れたクラス変数をα_iとする。

$$\theta := <\alpha_1 \triangleright M_1, \ldots, \alpha_m \triangleright M_m>$$

ここで$M_i(1 \leq i \leq m)$は次のように定義されている。

$$M_i = \begin{cases} \Upsilon(O'_i, S_i) \text{ where } S_i\text{はクラス。} \\ \qquad O_i\text{はクラス対象で} \\ \qquad \text{クラス}O'_i\text{を持つ。} \\ \Upsilon(O'_i, A_i) \text{ where } S_i[a_i]\text{。} \\ \qquad O_i\text{は論理対象で} \\ \qquad \text{クラス}O'_i\text{を持つ。} \\ \text{失敗} \quad \text{otherwise} \end{cases}$$

M_iが失敗である場合，制約充足手続きも失敗する。$\Upsilon()$が失敗した場合も，制約充足手続きは失敗する。

5. 各特殊化$<\alpha_i \triangleright M_i>(1 \leq i \leq m)$は，各部分構造の持つ部分構造に対する制約充足を再帰的に引き起こす。
6. 全ての制約充足手続きが成功した場合，元々の制約充足手続きも成功し，得られた特殊化全てを連結し，特殊化θを得る。
   失敗した制約充足手続きが一つでもあった場合，元々の制約充足手続きも失敗する。

6.3 単一化手続き

以下に示す二つの論理対象に対する単一化演算子(θ, σ)を求める手順を示す。

$$\begin{cases} X : [\alpha : (c_1 \cdots c_m)\ S_1 \cdots S_k], m, k \geq 0 \\ Y : [\beta : (d_1 \cdots d_n)\ T_1 \cdots T_l], n, l \geq 0 \end{cases}$$

1. まず最初に$\theta := <>$，$\sigma := <>$とする。
2. 最汎子孫$U = \Upsilon((c_1 \cdots c_m), (d_1 \cdots d_n))$を求める。$U$を求める手続きに失敗した場合，単一化手続きは失敗する。Uを得た場合には，$(c_1 \cdots c_m) \geq U$，$(d_1 \cdots d_n) \geq U$を満たすので，特殊化$<X \triangleright U>$と$<Y \triangleright U>$はどちらも適用可能である。
3. $\theta := <X \triangleright U>$，$\sigma := <Y \triangleright U>$とする。
4. ある$\gamma \in \mathbf{U}$を用意し，$\theta := \theta \cdot <\alpha \ominus \gamma>$，$\sigma := \sigma \cdot <\beta \ominus \gamma>$として，クラス変数の名前を同一にする。
5. $1 \leq \forall i \leq \min(k, \ell)$に対して，以下の手続きを繰り返す。
   (a) S_iとT_iを単一化する。この単一化の過程で得られた単一化演算子を(θ', σ')とする。
   (b) S_iとT_iの単一化に失敗した場合，元々の単一化も失敗する。
   (c) $S_i \circ \theta' = T_i \circ \sigma' = U_i$とする。
   (d) $\theta := \theta \cdot \theta'$，$\sigma := \sigma \cdot \sigma'$とする。
6. $\min(k, l) < \forall i \leq \max(k, l)$に対して，以下のように$U_i$，$\theta$，$\sigma$を定める。
   - $k < l \Rightarrow U_i := T_i$，$\theta := \theta \cdot <X \Rrightarrow U_i>$
   - $1 < k \Rightarrow U_i := S_i$，$\sigma := \sigma \cdot <Y \Rrightarrow U_i>$
7. 以上の手続きにより新しい論理対象$Z : [\gamma : U U_1 \cdots U_x]$，$x = \max(k, l)$，$Z \in \mathbf{V}$を得る。
8. 手順7で得た論理対象に対し，構造制約を充足する。構造制約充足の手続きが失敗した場合，単一化の手続きも失敗する。構造制約充足の手続きが成功した場合，得られた特殊化をφとする。このとき，

$$\theta := \theta \cdot \varphi, \sigma := \sigma \cdot \varphi$$

により新たにθ，σを得る。

9. $\theta := \theta \cdot <X \ominus Z>$，$\sigma := \sigma \cdot <X \ominus Z>$とし，対象変数の名前を同一にする。

以上の手続きにより得られた(θ, σ)が，求めるべき単一化演算子である。

7. プログラムの記述と実行

本節では，論理対象を用いてプログラムを記述する方法を示す。一般的な論理プログラムの場合と同様に，原始論理式から節を構成し，節の集合でプログラムを構成する。原始論理式は論理対象を用いて記述する。プログラム中で扱う論理対象を定めるために，定数の子孫関係と構造制約をプログラム中に記述する。

以上のように，本論文におけるプログラムは(1)定数の子孫関係，(2)構造制約，(3)節の集合の三つの部分からなる。

7.1 原始論理式の記述

一般的な論理プログラムにおいて，原始論理式はpred(arg, ・・・)のように，述語とその引数からなる形をしている。これを論理対象と比較してみると，

$$\text{pred}(\text{arg}, \cdots) \longleftrightarrow [\text{pred}\ (\text{arg} \cdots)]$$

のように対応付けることができる。したがって，論理対象を用いて原始論理式を表現する場合，(1)論理対象の持つクラス中の定数に述語を，(2)論理

対象の部分構造に引数を，それぞれ置くことにする。このとき引数は再び論理対象である。

7.2 節の記述

$H, B_1, \ldots, B_n$を論理対象で表わされた原始論理式とする。一般的な論理プログラムと同様に，節を以下の形で表現する。

$$H \leftarrow B_1, \ldots, B_n. \ (n \geq 0)$$

特殊化は各論理対象に対して作用するものと定義したが，ここにおいて，これを以下に示すように節に対するものへ拡張する。

$$H, B_1, \ldots, B_n \in A, \ \theta \in S$$
$$(H \leftarrow B_1, \ldots, B_n.) \circ \theta =$$
$$H \circ \theta \leftarrow B_1 \circ \theta, \ldots, B_n \circ \theta.$$

また，対象変数とクラス変数に関する省略記法も，単一の論理対象の範囲から，一つの節の範囲へ拡張する。

7.3 プログラムの記述(例)

例として図1にmapプログラムを示す。mapプログラムでは，クラス変数の利用により高階的なプログラムを，高階性を特に意識することなく記述できている。

図1において(cons . *t)クラスは，要素のクラスが*tであるコンスを表現している。同様に(list . *t)は要素のクラスが*tであるリストを表現している。

```
;;; 定数の子孫関係
pred → func | map
func → inc | not
data → bool | num
bool → true | false
num → 0 | 1 | 2
list → cons | nil

;;; 構造制約
(inc) == [(num)] ⊕ [(num)]
(not) == [(bool)] ⊕ [(bool)]
(cons .*t) == [*t] ⊕ [(list .*t)]
(map) == (func) ⊕ [(list)] ⊕ [(list)]

;;; 節
[(map) α:[()] [(nil)] [(nil)]] ← .
[(map) α:[()] [(cons) A:[()] X:[()]]
              [(cons) B:[()] Y:[()]] ←
     [α A B], [(map) α X Y].
```

図 1: mapプログラム

7.4 プログラムの実行

プログラムは一般的な論理プログラムの場合と同様に，問い合わせを行うことにより，単一化とバックトラックを繰り返して実行される。このとき，問い合わせは原始論理式であり，したがって論理対象で表現される。

8.考　　察

本章では，本論文で示した計算体系と関連する研究との比較を簡単に行う。ほとんどのプログラム言語の表現能力は「チューリング等価」という意味においてほぼ同等である。したがって，表現能力の比較は無意味と考える。ある概念をどれだけ単純明解に扱うことができるかという点を中心に比較を行う。

論理プログラミング上でクラス階層や型を扱ったものにLOGIN[2], DOT[4], 型仕様付き論理型言語[7], Typed Feature Stucture[8]などがある。LOGIN, Typed Feature Stuctureにおいては，クラス(型)は単なるラベルである。これに対して本論文におけるクラスは列という構造を持ち，複雑なクラスおよびその階層関係を柔軟に表現できる。DOTではクラスの階層(IS－A)関係のみに着目し，部分構造を扱わない。型仕様付き論理型言語では，述語はクラス階層の中に統合されておらず，プログラムの高階的な扱いに難がある。論理プログラムの高階的な扱いを目的とした言語にはHilog[3]，λProlog[5]などがある。Hilogでは高階のプログラムを一階のものに翻訳した後，計算を行う。このため実行時には，述語が変数であってはならないという制約を持つ。本論文のプログラムではそのような制約はない。λPrologと本論文のプログラムとの関係は，現在研究中である。

9.おわりに

本論文の目的は最初にも書いた通り，[6]に示した計算体系を，宣言型計算モデルに基づいて形式的に定義することである。宣言型計算モデルでは特殊化システム$<\mathcal{A}, \mathcal{G}, \mathcal{S}, \mu>$の上に問題領域を捉える。アルファベットの集合C,V,Uから$\mathcal{A}$(論理対象)を構成し，$\mathcal{S}$(特殊化)，μを用いて単一化を定めた。

最後に，特殊化システム上のプログラムに対す

る宣言的意味の定義を以下に示す。

定義15(解釈)

Iは特殊化システム$<\mathcal{A},\mathcal{G},\mathcal{S},\mu>$上の解釈。

$$\Updownarrow$$

$$I\subset\mathcal{G}$$

定義16(解釈に関して真)

節$H\leftarrow A_1,\ldots,A_n$が解釈$I$に関して真。

$$\Updownarrow$$

$$\left\{\begin{array}{l}\theta\in\mathcal{S}\text{が}H,A_1,\ldots,A_n\text{に適用可能かつ、}\\ \{H\circ\theta,A_1\circ\theta,\ldots,A_n\circ\theta\}\subset\mathcal{G}\\ \qquad\Downarrow\\ (H\circ\theta\in I)\vee\\ (A_1\circ\theta\notin I)\vee\ldots\vee(A_n\circ\theta\notin I)\end{array}\right.$$

定義17(モデル)

解釈Iは，プログラムPのモデル。

$$\Updownarrow$$

$$\forall p\in P,\ p\text{は}I\text{に関して真。}$$

定義18(宣言的意味) プログラムPのすべてのモデルの共通部分(最小モデル)をPの宣言的意味と定義する。これを$M(P)$と表わす。

参考文献

[1]赤間清，野村祐士，宮本衛市：プログラム変換
野村祐士，宮本衛市：プログラム変換による自然言語の意味解釈，コンピュータソフトウェア，Vol.12，No.5，1995.

[2]Hassan Aït-Kaci, Roger Nasr: *LOGIN: A Logic Programming Language with Built-In Inheritance,* The Journal of Logic Programming,3,1986.

[3]W. Chen, M. Kifer, D. S. Warren: *Hilog : A first-order semantics for higher-order logic programming constructs,* Proceedings of the North American Conference on Logic Programming, Cleveland, Ohio, 1989.

[4]塚本昌彦，西尾章治郎，長谷川利治：DOT記法とIS－A関係にもとづく項表現を用いた演繹・オブジェクト指向データベースモデル，コンピュータソフトウェア，Vol. 9, No.1, 1992.

[5]Gopalan Nadathur, Dale Miller:
An Overview of λProlog, Proceedings of the Logic Programming : Fifth International Conference on Logic Programming, R. Kowalski, K. Bowen, eds. , MIT PRESS, 1988.

[6]川口雄一，赤間清，宮本衛市：クラスと部分構造を持つ対象の表現と計算，日本ソフトウェア科学会第12回大会論文集，1995.

[7]Michael Hanus:
Logic Programming with Type Specifications, Types in Logic Programming, MIT PRESS, 1992.

[8]Bob Carpenter: *The Logic of Typed Feature Structures,* Cambridge Tracts in Theoretical Computer Science 32, Cambridge University Press, 1992.

[9]繁田良則，赤間清，宮本衛市：論理オブジェクトに基づく知識表現言語UL/α，人工知能学会誌，Vol.10，No.6，PP971-979，1995.

(平成7年11月30日受理)