オブジェクト指向データモデルの振舞い的側面の形式化

佐 藤 秀 樹

概 要

本論文では,オブジェクト指向データモデルの特徴の1つである振舞い的側面に関連して,属性文法を用いてメソッドに対する形式化を試みる。属性文法によるメソッドの形式化では,メソッドを属性文法で記述し,それを起動するメッセージはメソッドに対応する属性文法で解析されることにより実行される。上位-下位クラススキーマ関係を通したメソッド定義の継承・再定義は,対応する属性文法が定義する言語と変換に関する条件によって定義される。同様に,メソッドの多相性も対応する属性文法が定義する言語と変換に関する条件によって定義される。さらに,メソッド間の演算として,和集合,連接,閉包といった演算を定義する。この形式化により,オブジェクトのメソッドとして,自然言語に近い表現を許す属性文法を与えれば,自然言語に近い多用な文でメッセージを書くことが可能となる。

キーワード オブジェクト指向データベース(Object-Oriented Data Base),オブジェクト指向データモデル(Object-Oriented Data Model),メソッド(Method),属性文法(Attribute Grammar),文脈自由文法(Context Free Grammar)

1 まえがき

1970年,E. F. Coddは,関係代数[1]に基礎付けられた関係データモデルとこのデータモデルに基づく関係データベース[2]を提唱した。これを契機として,1970年代には関係データベースの理論と実装技術に関する研究が各所で活発に行われた。これらの成果は,1980年代に商用の関係データベースシステムを世に送り出すことにつながった。しかし,1980年代中頃より,エンジニアリングアプリケーション,科学技術アプリケーション,先進的オフィスアプリケーションといった非ビジネスデータ処理分野におけるデータ管理に関係データベースを適用するときの問題点が指摘されるようになってきた[3]。こうした問題点を克服するため,近年,オブジェクト指向データベースの研究開発や実用化が盛んに行われてきた。

オブジェクト指向データベースは，プログラミングの実践的方法論としての種々の提案を統合して出来上がったオブジェクト指向プログラミング方法論とそれを具体化したプログラミング言語[4]，[5]を基に発展してきた。こうした出自より，その基礎となるオブジェクト指向データモデルの形式化についての研究は遅れていた。しかし，近年，関係データモデルにならった代数的モデル論の研究[6]が行われ，それに基づくオブジェクト指向データベースの標準化作業[7]も開始され始めた。一方，この代数的モデル化とは異なり，筆者はこれまでに形式文法[8]によるオブジェクト指向データモデルの形式化を行ってきた[9]，[10]。そこでは，オブジェクト指向データモデルの構造的側面について，文脈自由文法[8]を使ってオブジェクトの型を表現することが試みられている。本稿では，この成果を受けて，オブジェクト指向データモデルの特徴の 1 つである振舞い的側面に関連して，属性文法[11]，[12]を用いてメソッドに対する形式化を試みる。

属性文法によるメソッドの形式化では，メソッドを属性文法で記述し，それを起動するメッセージはメソッドに対応する属性文法で解析されることにより実行される。上位／下位クラススキーマ関係を通したメソッド定義の継承・再定義は，対応する属性文法が定義する言語と変換に関する条件によって定義される。同様に，メソッドの多相性も対応する属性文法が定義する言語と変換に関する条件によって定義される。さらに，メソッド間の演算として，和集合，連接，閉包といった演算を定義する。この形式化により，オブジェクトのメソッドとして，自然言語に近い表現を許す属性文法を与えれば，自然言語に近い多用な文でメッセージを書くことが可能となる。

本論文の構成は，以下のとおりである。**2 章**では，本論文で用いる属性文法とその能力ついて述べる。**3 章**では，属性文法の応用によるメソッド表現を例を通して示す。**4 章**ではクラススキーマとそれから導出されるオブジェクト木を定義する。さらに，**5 章**ではメソッド間での定義の継承・再定義および多相性について，**6 章**ではメソッド仕様に対する演算について述べる。**7 章**では関連研究に言及し，**8 章**で本論文のまとめと今後の課題を示す。

2　属性文法

属性文法は，コンパイラ [13]，エディタ [14] などに応用されてきた言語処理のための形式的体系であり，文脈自由文法 [8] により導出される記号列に対して，その意味を対応させる規則を定める。属性文法では，文脈自由文法の各非終端記号に属性が付加される。属性は導出木内での意味計算に使われる変数であり，継承属性と合成属性に分類される。継承属性の値は導出木内を下向きに伝播し，合成属性の値は導出木内を上向きに伝播する。文脈自由文法の各構文規則には，属性値の計算方法を定める意味規則が付加される。以下では，本稿で用いる属性文法を定義する。

定義 1　属性文法 AG は，<G, Y, F>の 3 項組で定義される。

(1) $G=<V_N, V_T, S', P'>$は文脈自由文法であり，AGの基底文脈自由文法とよばれる。ここで，V_Nは非終端記号の有限集合，V_Tは終端記号の有限集合である。$S'(\in V_N)$は開始記号である。P'は構文規則の有限集合である。

(2) Gのすべての非終端次号Xに対して，互いに素な2つの有限集合IA(X)とSA(X)とが付随している。IA(X)の要素を継承属性，SA(X)の要素を合成属性とよぶ。Xの属性全体の集合をATTR(X)で表す。すなわち，ATTR(X)=IA(X)∪SA(X)である。$Y=\cup_{X\in V_N}ATTR(X)$と定義し，YをAGの属性集合とよぶ。また，Xの属性aをa(X)，属性が取りうる値の集合をV(a)と表す。

(3) P'の構文規則 $p: X_0 \rightarrow w_0X_{1w1} \ldots X_nw_n$ $(n\geqq 0, X_i\in V_N, w_i\in V_T^*, 0\leqq k\leqq n)$に対して，$SA(X_0)\cup IA(X_1)\cup \ldots \cup IA(X_n)$に含まれるすべての属性を定義する意味規則の集合$F_p$が付随している。属性$a(X_k)$ $(0\leqq k\leqq n)$を定義する意味規則は，次の形式をとる。

$$a(X_k):=f(a_1(X_{i1}), \ldots , a_m(X_{im}))$$

但し，関数fの引数である属性$a_j(X_{ij})$ $(1\leqq j\leqq m)$は，$IA(X_0)\cup SA(X_1)\cup \ldots \cup SA(X_n)$の要素である。さらに，$F=\cup_{p\in P}F_p$と定義し，FをAGの意味規則集合とよぶ。 □

尚，以降では，属性文法における属性概念とクラススキーマにおける属性概念とを区別するため，前者を g-属性，後者を c-属性と記す。但し，属性文法における継承属性と合成属性，クラススキーマの開始属性はこの限りではない。

本稿では，属性文法に対して，次の仮定を置く。

(1) AGの基底文脈自由文法Gの開始記号S'に関して，$IA(S')\subseteq\{DB_1\}$，$SA(S')\subseteq\{DB_2, ra\}$である。ここで，$DB_1$, DB_2は，それぞれAGによる属性計算の前後におけるデータベースの状態を表すg-属性である。**表1**に，DB_1, DB_2の有無に従って，AGによるデータベース計算の分類を示す。raは属性計算に基づくメソッド実行の戻り値を保持するg-属性であり，非データベース合成属性とよばれる。

(2) AGは任意の導出木において，各ノードに付随したg-属性の値が循環定義に陥らない非循環属性文法に制限する。

次に，属性文法が与える言語と変換を定義する。

定義2 属性文法 $AG=<G, Y, F>$に関して，

(1)言語：AGの言語B(AG)は，基底文脈自由文法 $G=<V_N, V_T, S', P'>$の言語L(AG)である。すなわち，B(AG)はP'の構文規則を有限回適用してS'から書換え可能な$w(\in V_T^*)$の集合である。

表1 属性文法AGによるデータベース計算の分類

IA(S') \ SA(S')	DB_2を含む	DB_2を含まない
DB_1を含む	データベース更新	データベース検索
DB_1を含まない	データーベース初期化	非データベース計算

$AG<G,Y,F>, G=<V_N,V_T,S',P'>$

DB_1, DB_2:AGにおける属性計算の前後におけるデータベース状態を表すg-属性

(2)変換：tをAGの任意の導出木とする。$m1_{AG}(t, d)$は，tの根ノードに非データベース合成属性raが付随する場合，データベース状態dのもとでのtのraの値を表す。そうでなければ，未定義値⊥を表す。また，$m2_{AG}(t, d)$はtの根ノードにデータベース合成属性DB_2が付随する場合には，データベース状態dのもとでのt のDB_2の値を表す。そうでなければ，dを表す。このもとで，AGによる変換T(AG)は，以下に定義される。但し，DBSはデータベース状態の集合を表す。

$$T(AG)=\{<w, d_1, v, d_2>|w \text{ is a string of terminal symbols derived by } t,$$
$$d_1 \in DBS, v=m1_{AG}(t, d_1), d_2=m2_{AG}(t, d_1) \in DBS\} \quad \square$$

従来，属性文法の変換はT(AG)は，<w, v>の2項組を要素とする集合として定義されている[11]。本稿では，データベースへの適用を図るために**定義2**のように拡張を行った。次に，**定義2**をもとに属性文法間の能力の関係を定義する。

定義3　AG_1, AG_2を属性文法であるとして，

(1) $B(AG_1)=B(AG_2)$かつ$T(AG_1)=T(AG_2)$であれば，AG_1とAG_2とは同じ能力を持つ。

(2) $B(AG_1) \supseteq B(AG_2)$かつ$T(AG_1) \supset T(AG_2)$であれば，$AG_1$は$AG_2$よりも強い能力をもつ。　$\square$

定義2，**定義3**は，クラススキーマ間でのメソッド定義の継承・再定義および多相性の定義に用いられる。

3　属性文法によるメソッド表現

本稿では，属性文法を使ってメソッドを定義する。**表2**は，メソッド概念と属性文法概念との対応を示す。メソッドのインタフェース仕様は，セレクタ，パラメータリスト，戻り値の方からなる。この中で，セレクタ，パラメータリストの仕様は，属性文法による解析対象文の集合に対する基底文脈自由文法として与えられる。この点から，セレクタとパラメータといった区別は陽には現れない。また，メソッドの戻り値の型は，属性文法に従う導出木の根ノードに付随する非データベース属性が取りうる値の集合となる。メソッドのインプリメンテーション仕様は，属性文法におけるg-属性集合／意味規則集合を使って記述される。このメソッドの仕様記述のもとで，メッセージは属性文法による解析対象となる文であり，メソッド実行はメッセージに対する文解析プロセスとなる。

属性文法によるメソッド仕様の記述例として，オブジェクトの年齢を計算するメソッドに対する属性文法を示す。**表3**は，メッセージの文集合に対する構文規則と意味規則を示す。また，**表4**は基底文脈自由文法の非終端記号とそれに付随するg-属性集合を示す。**表3**と**表4**における非終端記号ISは，開始記号である。メッセージの戻り値は，属性計算の結果として，開始記号ISに付随する合成属性ageに与えられる。**図1**は，メッセージに対する導出木上でのg-属性計算の様子を示す。**図1(a)**は，現在のオブジェクトの年齢を計算する。また，**図1(b)**は，1969年におけるオブジェクトの年齢を計算する。

表２　メソッド概念と属性文法概との対応

メソッド概念	属性文法概念
メソッド仕様	属性文法記述
インターフェース仕様	
セレクタ，パラメータリスト	基底文脈自由文法
戻り値の型	導出木の根ノードの非データベース合成属性の値の集合
インプリメンテーション仕様（メソッド本体）	g-属性集合，意味規則集合
メッセージ	文
メソッド実行	文解析プロセス

表３　属性文法の規則集合

構文規則	意味規則
IS → IP BV NP QM	age(IS)=age(NP) year(NP)=this year( )
IS → IP BV NP AP QM	age(IS)=age(NP) year(NP)=year(AP)
NP → PN N1	age(NP)=age(N1) year(N1)=year(NP) birthday(N1)=obj(PN).birthday
PN → “your”	obj(PN)=self( )
N1 → “age”	age(N1)=year(N1)−birthday(N1)
AP → PP N2	year(AP)=year(N2)
N2 →...	“1975”	...	year(N2)=value( )

（注）a (N)：非終端記号 N の g-属性 a
obj. attr：オブジェクト obj の c-属性 attr
self ( )：メッセージのレシーバオブジェクトの識別子を返す関数
value ( )：終端記号の値を返す関数
this year ( )：現在の年を返す関数

表４　非終端記号に付随する g-属性集合

非終端記号	合成属性	継承属性
IS	age	—
NP	age	year
PN	obj	—
N1	age	year, birthday
AP	year	—
N2	year	—

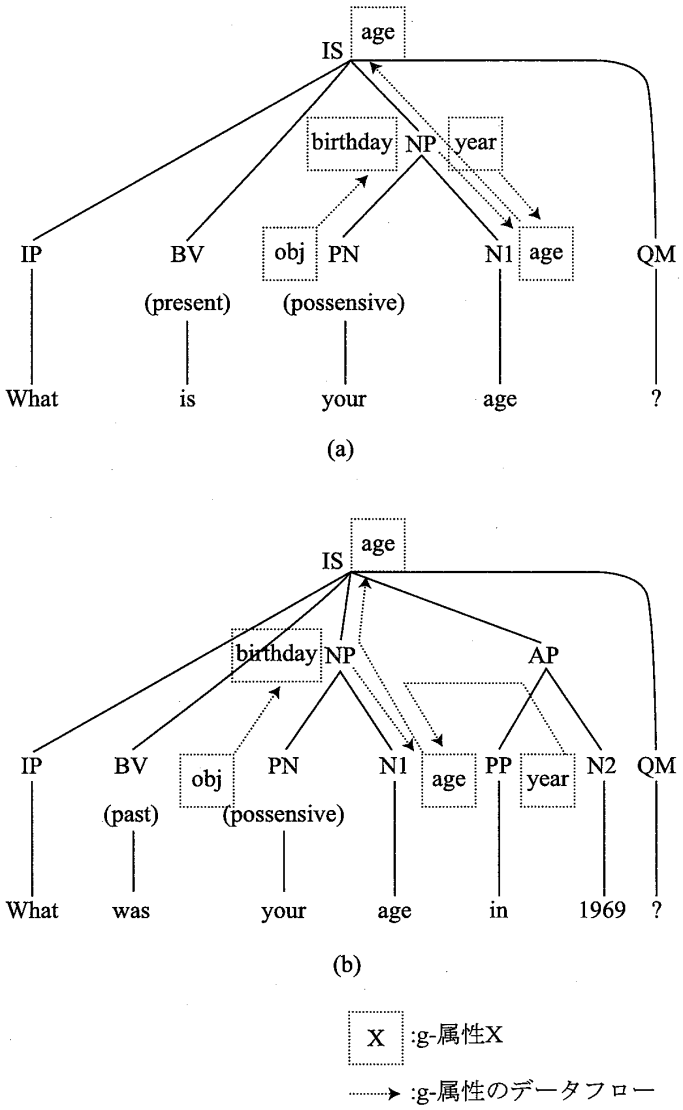


図1　メッセージに対する導出木上での属性計算

4　クラススキーマとオブジェクト木

クラススキーマは，オブジェクトの定義を与えるデータ構造である。クラススキーマ上のオブジェクトは<i, t>の対である。i はオブジェクト識別子であり，その一意性はオブジェクトが実世界の実体をモデル化可能であることに通じる。また，t はオブジェクト木である。オブジェクト木はオブジェクトに対して貯えられる情報であり，実世界の実体の性質をとらえる。本章では，文献[9]における定義を拡張し，メソッド概念を取り込んだクラススキーマを再定義する。

定義4　クラススキーマ CS は，<N, A, O, C, M, S, P>の7項組で定義される。ここで，N, A, O, C は，それぞれ c-属性の有限集合，基本ドメインの有限集合，オブジェクトドメインの有

限集合，定数の有限集合であり，これらは互いに素な集合である。Mは，<MA, AG>の有限集合である。MAはメソッドドメインであり，AGはMAを定義する属性文法である。また，mas(M)は，Mにおけるメソッドドメインの集合を表す。すなわち，mas(M)={MA|<MA, AG>∈M}である。さらに，S(∈N)は開始属性である。Pは構文規則の有限集合であり，各c-属性X(∈N)に対してXを左辺にもつ構文規則を1つ要素としてもつ。構文規則の形式には，以下の4つがある。

（形式1）$X \rightarrow Y_1 | \ldots | Y_n$，ここで$Y_i \in N, X \neq Y_i (1 \leqq i \leqq n)$である。この形式は，Xが異なるc-属性$Y_i$のいずれかにより構成されることを示す。

（形式2）$X \rightarrow w$，ここで$X \in N, w \in N^+$であり，wにおいて各c-属性はたかだか1回しか現れない。

（形式3）$X \rightarrow Z_q^r$，ここでX, Z∈Nである。XはZのq回以上，r回以下の繰返しである。また，この形式の変形として，$X \rightarrow Z^*$（Zの0回以上の繰返し），$X \rightarrow Z^+$（Zの1回以上の繰返し）を含む。

（形式4）$X \rightarrow b$，ここで$X \in N, b \in (A \cup O \cup C \cup mas(M))$である。 □

定義4では，メソッドドメインが各c-属性構造の定義に組み込まれている。これにより，クラススキーマにおいて導出属性が自然に実現されることになる。

クラススキーマにおける各c-属性，基本ドメイン，オブジェクトドメイン，定数，メソッドドメインの定義域は，以下のとおりである。

定義5 CS=<N, A, O, C, M, S, P>クラススキーマとする。

(1)基本ドメイン a(∈A)の定義域 DOM(a)は a である。DOM(a)は，未定義値を示す下限要素$\perp_a$を含む。

(2)オブジェクトドメインo(∈O)の定義域DOM(o)は，オブジェクト識別子の集合IDである。DOM(o)は，未定義値を示す下限要素$\perp_o$を含む。

(3)定数 c(∈C)の定義域 DOM(c)は{c}である。

(4)メソッドドメイン m(∈mas(M))の定義域 DOM(m)は<m, ag>∈M, ag=<G,Y, F>, $G=<V_N, V_T, S', P'>$のとき，SA(S')が非データベース合成属性 ra を含めば V(ra)である。そうでなければ，DOM(m)は$\{\perp_m\}$である。ここで，$\perp_m$は未定義値を示す下限要素である。

(5) c-属性 X(∈N)の定義域 DOM(X)は，Xを左辺にもつ構文規則の形式に応じて，以下の(a)~(d)に定義される。

(a) $X \rightarrow Y_1 | \ldots | Y_n (\in P), 1 \leqq i \leqq n$ の場合，$DOM(X)=DOM(Y_1) \cup \ldots \cup DOM(Y_n)$である。

(b) $X \rightarrow w (\in P), w = X_1 \ldots X_k (\in N^+)$であり，k≧1 の場合，$DOM(X)=DOM(X_1) \cup \ldots \cup DOM(X_k)$である。但し，DOM(X)は未定義値を示す下限要素$\perp_m=<\perp_{X1}, \ldots, \perp_{Xk}>$を含む。

(c) $X \rightarrow Z^* (\in P)$の場合，$DOM(X)=2^{DOM(Z)}$である。$X \rightarrow Z^+ (\in P)$の場合，$DOM(X)=^{2DOM(Z)}—\{\phi\}$である。$X \rightarrow Z_q^r (\in P)$の場合，$DOM(X)=\{e | q \leqq |e| \leqq r, e \in 2^{DOM(Z)}\}$である。但し，DOM(X)は，未定義値を示す下限要素$\perp_X$を含む。

(d) ここでX, Z∈Nである。XはZのq回以上，r回以下の繰返しである。また，この形式の変形として，Zの0回以上の繰返し），X→Z^+（Zの1回以上の繰返し）を含む。

(e) X→b(∈P), b∈(A∪O∪C∪mas(M))の場合，DOM(X)=DOM(b)である。　□

以下に，オブジェクト木の定義を示す。

定義6　クラススキーマCS=<N, A, O, C, M, S, P>上のオブジェクト木Tは，以下の条件を満たす導出木である。

(1)Tの各ノードは，$(N\cup(\cup_{a_i\in A}DOM(a_i))\cup(\cup_{o_j\in O}DOM(o_j))\cup C\cup mas(M))$の要素のいずれか1つをラベルとしてもつ。

(2) Tの根ノードのラベルはSである。

(3) Tの内部ノードのラベルは，Nの要素に限られる。

(4)ラベルがXであるノードがk個の子ノードを持ち，かつ，それらのラベルが左から順にX_1 ... X_kであるとする。この場合，Pの要素であり，X(∈N)を左辺にもつ構文規則について，次の(a)~(d)のいずれかが成立する。

(a) 構文規則が$X\rightarrow Y_1|\ ...\ |Y_n$，但し$Y_i\in N$, $1\leqq i\leqq n$であるとき，k=1かつ$X_1=Y_i$こであるようなiが存在する。

(b) 構文規則がX→w，但しw∈N^+, k≧1であるとき，w=X_1 ... X_kである。

(c) 構文規則がX→Z^*, X→Z^+, X→Z_q^rのいずれかであり，X_1= ... =X_k=Zかつkが構文規則におけるZの繰返し回数の条件を満足する。

(d) 構文規則がX→b, b∈(A∪O∪C∪mas(M))であるとき，k=1かつX1∈DOM(b)である。

□

定義6により，オブジェクト木はメソッド概念を取り入れるように拡張された。オブジェクト木の同形，オブジェクトの定義域，インスタンスの定義については文献[9]の定義に同じである。

5　継承・再定義と多相性

本章では，**定義3**の属性文法間の能力を用いて，メソッド定義の継承・再定義および多相性を定義する。

5.1　クラススキーマ間でのc-属性定義の継承・再定義

クラススキーマ間には，上位-下位クラススキーマ関係が存在する。クラススキーマの集合はこの関係のもとでの半順序集合であり，クラススキーマ束を導出する。クラススキーマ束において，c-属性の定義の継承・再定義，クラスの包含関係といった概念を提供する。

上位-下位クラススキーマ関係を定義するため，先ず，関数substitute(CS, V, W)を示す。関数substitute(CS_1, V, W)は，クラススキーマCS_1=<N_1, A_1, O_1, C_1, M_1, S_1, P_1>における属性Vを属

性 W で置換したクラススキーマ CS_2=<N_2, A_2, O_2, C_2, M_2, S_2, P_2>を以下のように作成する。但し，s-attr(X, V, W)は X が V と同一であれば W を，そうでなければ X を返す。s-rule(p, V, W)は構文規則 p における属性 V の出現を属性 W で置換した構文規則を返す。

(1) N_2={s-attr(X, V, W)|X∈N_1}

(2) A_2=A_1

(3) O_2=O_1

(4) C_2=C_1

(5) M_2=M_1

(6) S_2=s-attr(S_1, V, W)

(7) P_2={s-rule(p, V, W)|p∈P □

以下に，上位-下位クラススキーマ関係を定義する。

定義 7 クラススキーマ CS_1=<N_1, A_1, O_1, C_1, M_1, S_1, P_1>がクラススキーマ CS_2=<N_2, A_2, O_2, C_2, M_2, S_2, P_2>の上位クラススキーマ（等価的には，CS_2 は CS_1 の下位クラススキーマ）であるための必要十分条件は，以下のとおりである。

(1) $S_1 \neq S_2$, $S_1 \notin N_2$, $S_2 \notin N_1$

(2) CS_3=substitute(CS_2, S_2, S_1)=<N_3, A_3, O_3, C_3, M_3, S_3, P_3>とする。このとき，すべての X∈(N_1∪N_3)について，X を左辺にもつ構文規則の形式に応じて，次の(a)~(f)のいずれかが成立する。

(a) X→$Y_{1,1}$| ... |$Y_{1,n1}$∈P_1, $Y_{1,i}$∈N_1, 1≦i≦n1 の場合，X→$Y_{3,1}$| ... |$Y_{3,n3}$∈P_3, $Y_{3,j}$∈N_3, 1≦j≦n3, かつ，{$Y_{1,1}$, ... ,$Y_{1,n1}$}⊇{$Y_{3,1}$, ... ,$Y_{3,n3}$}である。

(b) X→w_1∈P_1, w= $X_{1,1}$ ... $X_{1,n1}$, $X_{1,i}$∈N_1 の場合，X→w_3∈P_3, w= $X_{3,1}$ ... $X_{3,n3}$, $X_{3,j}$∈N_3，かつ，{$X_{1,1}$, ... ,$X_{1,n1}$}⊆{$X_{3,1}$, ... ,$X_{3,n3}$}である。

(c) X→Z^*∈P_1, X→Z^+∈P_1, X→Z_{q1}^{r1}∈P_1，のいずれかであり，Z の繰返し回数が i_1 回以上，j_1 回以下の場合，かつ，X→Z^*∈P_3, X→Z^+∈P_3, X→Z_{q3}^{r3}∈P_3，のいずれかであり，Z の繰返し回数が i_3 回以上，j_3 回以下であり，$i_1 \leqq i_3 \leqq j_3 \leqq j_1$ である。

(d) X→o_1∈P_1, o_1∈O_1 の場合，X→o_3∈P_3, o_3∈O_3 であり，かつ，o_1, o_3 に対応するクラススキーマが同一であるか，o_1 に対応するクラススキーマが o_3 に対応するクラススキーマの上位クラススキーマである。

(e) X→b_1∈P_1, b_1∈(A_1∪C_1)の場合，X→b_3∈P_3, b_3∈(A_3∪C_3)，かつ，DOM(b_1)⊇DOM(b_3)である。

(f) X→m∈P_1, <m, ag_1>∈M_1 の場合，X→m∈P_3, <m, ag_3>，かつ，①ag_1 と ag_3 は同じ能力をもつ，あるいは②ag_1 は ag_3 よりも強い能力をもつ。 □

定義 7 の(2)の(f)は，クラススキーマ CS_1, CS_2 のメソッドドメイン間の条件に関する。これが，文献[9]の上位-下位クラススキーマ関係の定義に対して，本稿で追加を行った箇所である。①は CS_2 が CS_1 のメソッド定義を継承する場合である。②は CS_1 のメソッド定義を特殊化して，CS_2 でメソッドを再定義する場合である。

5.2　メソッドの多相性

多相性はオブジェクトのクラススキーマを正確に指定することなく，そのオブジェクトのメソッドを呼び出す機能である。**図2**のクラススキーマの階層構造において，Employeeオブジェクトの給与計算メソッドは多相性の例である。すなわち，WageEmployee オブジェクトは時間給であり，SalesPersonオブジェクトは時間給分と歩合給分を足した給与であり，Manager オブジェクトは固定給となっている。このように，多相性は給与計算の同じメッセージ構文に対して，複数の異なる形態のメソッド実行を仮定する能力を意味する。

以下に，メソッドの多相性の定義を対応する属性文法間の関係を通して行う。

定義 8　メソッドの集合 $MS=\{m_1, \ldots, m_n\}$が多相性であるための必要十分条件は，各メソッドを定義する属性文法の集合 $AGS=\{AG_1, \ldots, AG_n\}$に関して，以下が成立することである。但し，$AG_i=<G_i, Y_i, F_i>$, $G_i=<V_{Ni}, V_{Ti}, S_i', P_i'>$ $(1 \leqq I \leqq n)$である。

(1) AGS の要素である各属性文法が定義する言語は同じである。すなわち，$B(AG_1)= \ldots =B(AG_n)$が成立する。

(2) AGS に関して，次の(a), (b)のいずれかが成り立つ。

(a) すべての $SA(S_i')$が非データベース合成属性 ra_i を含み，かつ $V(ra_1), \ldots, V(ra_n)$が互換性のある集合である。

(b) すべての $SA(S_i')$が非データベース合成属性 ra_i を含まない。

(3) AGS の要素である各属性文法は，異なる変換を定義する。すなわち，すべての異なる AG_i, $AG_j(\in AGS)$について，$T(AG_i) \neq T(AG_j)$が成立する。　□

定義 8 の(1)は，多相性の機能を実現するメソッドの集合の各要素が同一のメッセージを受けることができることを保証する。(2)は各メソッドが実行結果の戻り値をもてば，それらが互換性のある定義域をもつことを保証する。また，(3)は各メソッドが異なる形態の実行を行うことを示す。

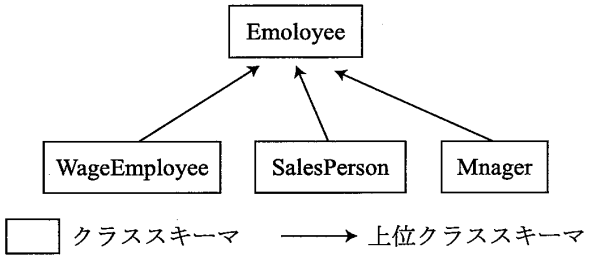


図2　Employee クラススキーマ階層

6 メソッド仕様に対する演算

本章では，メソッド仕様に対して拡張性および再利用性を図るため，属性文法の変換に対する演算を示す。これは，メソッドによる計算はメソッド仕様となる属性文法 AG の変換 T(AG)によって与えられることに基づく（**定義 3** 参照）。すなわち，T(AG)の各要素<w, d_1, v, d_2>は，メソッドがデータベース状態 d_1 のもとでメッセージ w を受信すると，データベース状態 d_2 への遷移を起こし，戻り値 v を返すことを示すものであり，T(AG)は属性文法 AG によるメソッド計算の内容を表す。以下では，属性文法の変換に対する和集合，連接，閉包といった演算を対象とする。各演算毎に，先ずその定義を示す。次に，属性文法の変換がその演算について閉じていることを示す。これらの性質により，和集合，連接，閉包といった演算を繰返し適用することにより，既存のメソッド仕様をもとに新たなメソッド仕様が構成可能であることになる。

6.1 和集合演算

オブジェクト指向データモデルにおけるメソッドのインタフェース仕様は，セレクタ，パラメータ，戻り値の型により定義される[7]。このインタフェース仕様には自由度はなく，曖昧性は排除されている。このため，類似の機能を持つメソッドであっても，それらは異なるインタフェース仕様を持つメソッドとして明確に区別される。本稿では，属性文法の変換に対する和集合演算を定義し，類似の機能を持つ複数メソッドを 1 つのメソッドに統合可能とすることにより，メソッドの実現・管理面で柔軟性をもたらす。

定義 9 属性文法 AG_1, AG_2 の変換に対する和集合（∪）演算は，次のように定義される。

$T(AG_1) \cup T(AG_2)=\{<w, d_1, v, d_2> | <w, d_1, v, d_2> \in T(AG_1)$ または $<w, d_1, v, d_2> \in T(AG_2)\}$ □

図 3(a)では，類似の機能を持つ 2 つのメソッド（“今年の年齢を求めるメソッド”と“ある年の年齢を求めるメソッド”）およびこれらのメソッドに対するメッセージの送信例が示されている。これら 2 つのメソッドは類似の機能を持つ。しかし，前者はパラメータを有しないが，後者は「ある年」を表すパラメータを有するといった違いが存在する。このため，従来のオブジェクト指向データモデル[7]では，これらは異なるメソッドとして定義される。一方，**図 3(b)**では，メソッド仕様に対する和集合演算を適用することにより，上記の 2 つのメソッドを統合する新たなメソッドが構成され，そのメソッドに対してメッセージが送信されるようになっている。

次に，和集合演算に関する閉包定理を示す。

定理 1 属性文法の変換は，和集合の演算について閉じている。 □

（証明）AG_1, AG_2 を次の属性文法とする。

$AG_1=<G_1, Y_1, F_1>$, $G_1=<V_{N1}, V_{T1}, S_1, P_1>$, $IA(S_1)=\{d_{11}\}$, $SA(S_1)=\{d_{12}, ra_1\}$,

$AG_2=<G_2, Y_2, F_2>$, $G_2=<V_{N2}, V_{T2}, S_2, P_2>$, $IA(S_2)=\{d_{21}\}$, $SA(S_2)=\{d_{22}, ra_2\}$

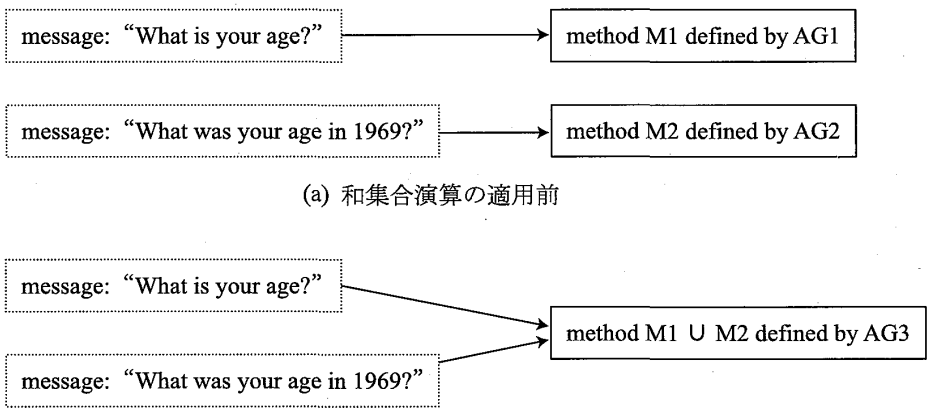


図3　メソッド仕様に対する和集合演算

但し，$d_{11}, d_{12}, d_{21}, d_{22}$ はデータベース状態を表すg-属性，ra_1, ra_2 はメソッド実行の戻り値を保持するg-属性である。ここに，一般性を失うことなく，V_{N1}, V_{N2} は互いに素であるとする。

属性文法 AG_3 を $AG_3=<G_3, Y_3, F_3>$, $G_3=<V_{N1} \cup V_{N2} \cup \{S_3\}, V_{T1} \cup V_{T2}, S_3, P_1 \cup P_2 \cup \{p_{31}:S_3 \rightarrow S_1, p_{32}:S_3 \rightarrow S_2\}>$, $Y_3=Y_1 \cup Y_2 \cup \{d_{31}, d_{32}, ra_3\}$, $F_3=F_1 \cup F_2 \cup F_{p31} \cup F_{p32}$, $F_{p31}=\{d_{11}=d_{31}, d_{32}=d_{12}, ra_3=ra_1\}$, $F_{p32}=\{d_{21}=d_{31}, d_{32}=d_{22}, ra_3=ra_2\}$のように構成する。$S_3$ は，V_{N1}, V_{N2} のいずれにも含まれない非終端記号であるとする。いま，$<w, d_1, v, d_2> \in T(AG_1)$とすると，$AG_1$ の構文規則，意味規則は AG_3 の構文規則，意味規則でもあるから，$<w, d_1, v, d_2> \in T(AG_3)$である。$<w, d_1, v, d_2> \in T(AG_2)$のときは，同様にして$<w, d_1, v, d_2> \in T(AG_3)$である。よって，$T(AG_1) \cup T(AG_2) \subseteq T(AG_3)$がなりたつ。

つぎに，$<w, d_1, v, d_2> \in T(AG_3)$とすると，w の導出の第1段階に用いられる構文規則および付随する意味規則によって，$<w, d_1, v, d_2> \in T(AG_1)$であるか，あるいは，$<w, d_1, v, d_2> \in T(AG_2)$であるかのいずれかである。よって，$T(AG_3) \subseteq T(AG_1) \cup T(AG_2)$である。

以上より，$T(AG_3)=T(AG_1) \cup T(AG_2)$が結論される。　□

6.2　連接演算

オブジェクトに送信されたメッセージは，対応するメソッドの実行を引き起こす。このメソッド実行は，同様に，メッセージを受信したオブジェクトをも含む他オブジェクトへのメッセージ送信により実現される。換言すれば，メソッドは関数的な合成により実現されている。通常，メソッドの関数的な合成の制御仕様は，メソッドの実装者によりメソッドの本体内に暗黙的に記述される。本稿では，属性文法の変換に対する連接演算を定義し，既存のメソッド仕様を基にメソッドの関数的な合成を実現する。連接演算によれば，メソッドの関数的な合成の制御仕様はメソッドの本体内から取り出され，明示的に示される。このことは，新たなメソッド仕様が合成可能となると同時に，合成に関する誤りの出現を低下させることが期待できる。

定義 10　属性文法 AG1, AG2 の変換に対する連接（・）演算は，次のように定義される。

$T(AG_1)\cdot T(AG_2)=\{<w_1w_2, d_{11}, v_2, d_{22}>|$

$<w_1, d_{11}, v_1, d_{12}>\in T(AG_1), <w_1, d_{21}, v_1, d_{22}>\in T(AG_2), d_{12}= d_{21}\}$　□

図 4(a)では，2つのメソッド（“空間内での移動命令メソッド”と“空間内での位置の問合せメソッド”）およびこれらのメソッドに対するメッセージの逐次的な送信例が示されている。一方，**図 4(b)**では，メソッド仕様に対する連接演算を適用することにより，上記の2つのメソッドを合成する新たなメソッドが構成され，そのメソッドに対して前述のメッセージ列が1つのメッセージとして送信されるようになっている。

次に，連接演算に関する閉包定理を示す。

定理 2　属性文法の変換は，連接の演算について閉じている。　□

（証明）AG_1, AG_2 を次の属性文法とする。

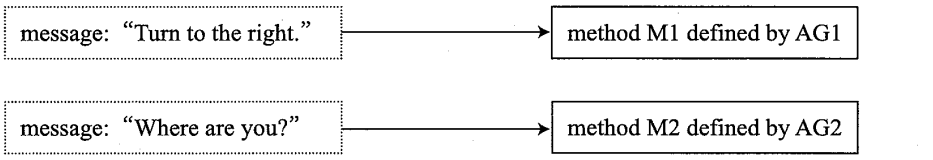
$AG_1=<G_1, Y_1, F_1>$, $G_1=<V_{N1}, V_{T1}, S_1, P_1>$, $IA(S_1)=\{d_{11}\}$, $SA(S_1)=\{d_{12}, ra_1\}$,

$AG_2=<G_2, Y_2, F_2>$, $G_2=<V_{N2}, VT_2, S_2, P_2>$, $IA(S_2)=\{d_{21}\}$, $SA(S_2)=\{d_{22}, ra_2\}$

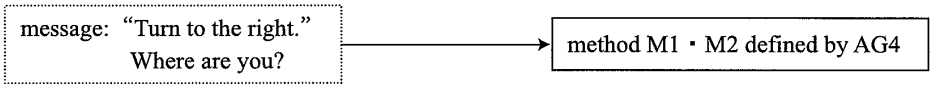
但し，$d_{11}, d_{12}, d_{21}, d_{22}$ はデータベース状態を表すg-属性，ra_1, ra_2 はメソッド実行の戻り値を保持するg-属性である。ここに，一般性を失うことなく，V_{N1}, V_{N2} は互いに素であるとする。

属性文法 AG_4 を $AG_4=<G_4, Y_4, F_4>$, $G_4=<V_{N1}\cup V_{N2}\cup\{S_4\}, V_{T1}\cup V_{T2}, S_4, P_1\cup P_2\cup\{p_4:S_4\rightarrow S_1S_2\}>$, $Y_4=Y_1\cup Y_2\cup\{d_{41}, d_{42}, ra_4\}$, $F_4=F_1\cup F_2\cup F_{p4}$, $F_{p4}=\{d_{11}=d_{41}, d_{42}=d_{22}, d_{21}=d_{12}, ra_3=ra_2\}$のように構成する。$S_4$ は，V_{N1}, V_{N2} のいずれにも含まれない非終端記号であるとする。いま，$<w_1, d_{11}, v_1, d_{12}>\in T(AG_1)$, $<w_2, d_{21}, v_2, d_{22}>\in T(AG_2)$, $d_{12}=d_{21}$ とすると，AG_1, AG_{21} の構文規則，意味規則は AG_3 の構文規則，意味規則でもあるから，$<w_1w_2, d_{11}, v_2, d_{22}>\in T(AG_4)$である。よって，$T(AG_1)\cdot T(AG_2)\subseteq T(AG_4)$がなりたつ。

つぎに，$<w, d_1, v, d_2>\in T(AG_4)$とすると，w の導出の第1段階に用いられる構文規則および付随する意味規則によって，$<w, d_1, v, d_2>$は S_1, S_2 を導出木の根ノードとする2個の変換要素である$<w_1, d_1, v_1, d_m>$, $<w_2, d_m, v, d_2>$に分割される。但し，$w_1w_2=w$ である。V_{N1} と V_{N2} とは互い



(a) 連接集合演算の適用前



(b) 連接集合演算の適用後

図 4　メソッド仕様に対する連接演算

に素であるから，w_1 の導出には P_1 の要素が使われて$<w_1, d_1, v_1, d_m>\in T(AG_1)$，$w_2$ の導出には P_2 の要素が使われて$<w_2, d_m, v, d_2>\in T(AG_2)$となる。よって，$T(AG_4)\subseteq T(AG_1)\cdot T(AG_2)$である。

以上より，$T(AG_4)=T(AG_1)\cdot T(AG_2)$が結論される。 □

6.3　閉包演算

メソッドの実行に際して，自身をも含む他オブジェクトへの同一のメッセージを任意回繰り返し送信することがある。メソッドの関数的な合成の場合と同様に，このための繰り返しの制御仕様は，メソッドの実装者によりメソッドの本体内に暗黙的に記述される。本稿では，属性文法の変換に対する閉包演算を定義し，既存のメソッド仕様を基にメソッドの繰り返し実行を実現する。閉包演算によれば，メソッドの繰り返し実行の制御仕様はメソッドの本体内から取り出され，明示的に示される。このことは，メソッドの繰り返し実行が可能であると同時に，このための制御に関する誤りの出現を低下させることが期待できる。

定義 11　属性文法 AG_1 の変換に対する閉包（＊）演算は，次のように定義される。

$T(AG_1)^*=\{\lambda\}\cup T(AG_1)\cup T(AG_1)\cdot T(AG_1)\cup T(AG_1)\cdot T(AG_1)\cdot T(AG_1)\cup .....$

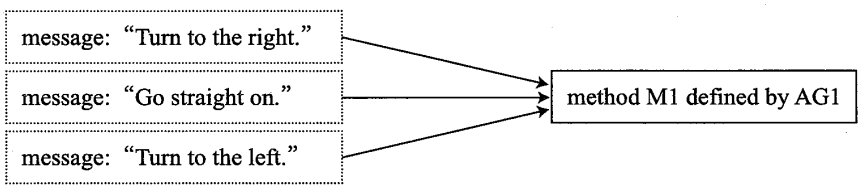
但し，λ は$<\varepsilon, d, \perp, d>$を，$\varepsilon$ は空記号列を，$\perp$ は未定義値を表す。 □

図 5(a)では，“空間内での移動命令メソッド”およびこのメソッドに対するメッセージの任意回の繰り返し送信の例が示されている。一方，**図 5(b)**では，メソッド仕様に対する閉包演算を適用することにより，新たなメソッドが構成され，そのメソッドに対して前述のメッセージ列が 1 つのメッセージとして送信されるようになっている。

次に，閉包演算に関する閉包定理を示す。

定理 3　属性文法の変換は，閉包の演算について閉じている。 □

（証明）AG_1 を次の属性文法とする。



(a) 閉包集合演算の適用前

(b) 閉包集合演算の適用後

図 5　メソッド仕様に対する閉包演算

$AG_1=<G_1, Y_1, F_1>$, $G_1=<V_{N1}, V_{T1}, S_1, P_1>$, $IA(S_1)=\{d_{11}\}$, $SA(S_1)=\{d_{12}, ra_1\}$

但し，d_{11}, d_{12}はデータベース状態を表すg-属性，ra_1はメソッド実行の戻り値を保持するg-属性である。

属性文法AG_5を$AG_5=<G_5, Y_5, F_5>$, $G_5=<V_{N1} \cup \{S_5\}, V_{T1}, S_5, P_1 \cup \{p_{51}:S_5 \rightarrow S_1S_5, p_2:S_5 \rightarrow \varepsilon\}>$, $Y_5=Y_1 \cup \{d_{51}, d_{52}, ra_5\}$, $F_4=F_1 \cup F_{p51} \cup F_{p52}$, $F_{p51}=\{d_{11}=d_{5(1)1}, d_{5(2)1}=d_{12}, d_{5(1)1}=d_{5(2)2}, ra_{5(1)}=ra_{5(2)}\}$, $F_{p52}=\{d_{52}=d_{51}, ra_5=\perp\}$のように構成する。$S_5$は，$V_{N1}$に含まれない非終端記号であるとする。いま，$<w_1, d_{1,1}, v_1, d_{1,2}>$, $<w_2, d_{2,1}, v_2, d_{2,2}>$, $<w_3, d_{3,1}, v_3, d_{3,2}>$, ...$\in T(AG_1)$, $d_{i,2}=d_{i+1,1}(i \geqq 1)$すると，

(ｉ) $<\varepsilon, d_0, \perp, d_0> \in T(AG_5)$である。

(ii) $<w_1w_2 \ldots w_k, d_{1,1}, v_k, d_{k,2}> \in T(AG_5)$であると仮定すると，$d_{k,2}=d_{k+1,1}$であるから，$<w_1w_2 \ldots w_kw_{k+1}, d_{1,1}, v_{k+1}, d_{k+1,2}> \in T(AG_5)$である。

(ｉ), (ii)から数学的帰納法により，$T(AG_1)^* \subseteq T(AG_5)$がなりたつ。

つぎに，$<w, d_1, v, d_2> \in T(AG_5)$とすると，wの導出に用いられる構文規則および付随する意味規則によって，$<w, d_1, v, d_2>$はS_1を導出木の根ノードとするk($\geqq 0$)個の変換要素である$<w_1, d_{m0}, v_1, d_{m1}>$, $<w_2, d_{m1}, v_2, d_{m2}>$, ... , $<w_k, d_{mk-1}, v_k, d_{mk}>$と$S_5$を導出木の根ノードとする変換要素である$<\varepsilon, d_2, \perp, d_2>$に分割される。但し，$d_{m0}=d_1$, $d_{mk}=d_2$, $v_k=v$, $w_1w_2 \ldots w_k=w$である。ここで，$<w_i, d_{mi-1}, v_i, d_{mi}> \in T(AG_1)$ $(1 \leqq i \leqq k)$である。よって，$T(AG_5) \subseteq T(AG_1)^*$である。

以上より，$T(AG_5)=T(AG_1)^*$が結論される。 □

7 関連研究

オブジェクト指向データモデルの形式化については，複合オブジェクトの操作システム[15], [16], [17]，継承（インヘリタンス）[18], [19]，オブジェクト識別性[20]などの研究が行われてきた。これらの研究では，形式化の枠組として，代数的モデル論，述語論理に基づく論理型計算，ラムダ計算に基づく関数型計算が有力なパラダイムになっている。

本研究では，オブジェクト指向データモデルの形式化のために形式文法[8]を用いている。同様に，形式文法に基づくデータモデルの研究も行われている[21], [22], [23]。これらの研究は，大きく2つに分類される。文献[21]の研究は階層的な情報構造に対して形式文法による記述を図っており，本研究と目的を同じにする。しかし，このデータモデルは値指向であり，メソッドといった振舞い的側面も考慮されておらず，オブジェクト指向データモデルを完全には扱っていない。これに対して，文献[22], [23]の研究は，データベースによるテキストデータの管理のために，形式文法の適用を図っている。近年，SGML[24], XML[25]などの構造化文書に対する標準化仕様が提示されており，電子化文書が急速に普及してきている。これに伴い，大量のテキストデータの管理に対するニーズは高まってきている。こうした背景のもと，これらの研究はその基礎となっている。

GROOVY[26]は超グラフにより形式化されたオブジェクト指向データモデルであり，オブ

ジェクト指向データモデルの構造的側面を扱っている。しかし，このデータモデルもメソッド概念を取り入れてはいない。本研究では，メソッドの形式化のために属性文法[11], [12]を用いている。文献[27], [28], [29], [30]の研究は，いずれもオブジェクト指向パラダイムを使って属性文法の能力を拡張している。これに対して，本研究はオブジェクト指向データモデルのメソッド概念を属性文法により形式化するものであり，前者の研究とは目的を異にしている。すなわち，研究目的とそれを達成するための手段・道具立てが互いに逆の関係となっている。

次に，本研究におけるメソッド表現と従来のオブジェクト指向データモデルにおけるメソッド表現との違いについて言及する。ODMG（Object Database Management Group）標準に準拠するオブジェクト指向データモデル[7]におけるメソッドのインタフェース仕様は，セレクタ，パラメータ，戻り値の型により定義される。このインタフェース仕様には自由度はなく，曖昧性は排除されている。このため，類似の機能を持つメソッドであっても，それらは異なるインタフェース仕様を持つメソッドとして明確に区別される。これに対して，属性文法によるメソッド表現に従えば，メソッドは基底文脈自由文法が定めるメッセージを解析し，意味計算規則が定めるメソッド計算を実現する。これは，前者と比して，メソッドの能力を強力なものとし，その記述を柔軟なものとしている。

最後に，属性文法の変換に対する和集合演算，連接演算，閉包演算について述べる。正規言語，文脈自由言語，文脈依存言語といった形式言語に関しては，前記の演算およびそれらの閉包性についての研究は既に確立されている[8]。しかし，本研究のように，属性文法に関して同様な演算および閉包性について議論を行っているものはない。また，属性文法の変換に対する和集合演算，連接演算，閉包演算は，既存のメソッド仕様についての拡張性・再利用性を高め，新たなメソッド仕様を構成可能とする点で有用である。

8　むすび

本論文では，オブジェクト指向データモデルの特徴の1つである振舞い的側面について，属性文法を用いてメソッドの形式化を行った。本研究は，次の特徴を備える。

- 属性文法によるメソッド表現に従えば，メソッドは基底文脈自由文法が定めるメッセージを解析し，意味計算規則が定めるメソッド計算を実現する。従来のモデルに比して，メソッドの能力を強力なものとし，その記述を柔軟なものとしている。
- メソッド仕様は導出木上の属性値の純関数的な計算モデルに基づく属性文法の性質である宣言的な記述をそのまま継承する。
- 属性文法は構文規則集合により構成されるが，これらの構文規則に対して属性集合／意味規則集合が付随する。この点より，メソッド記述は構文規則レベルでのモジュール性を有する。

・属性文法の変換に関して和集合演算，連接演算，閉包演算およびそれらの閉包性を明らかにしている。その結果，既存のメソッド仕様を基に新たなメソッド仕様を構成可能であることを示した。

今後の課題としては，本論文で提案した形式化に従う処理系の実装に関連して，特に形式言語に対する既存のアルゴリズムやツールなどが適用可能であるか否か，あるいは適用するために必要となる改良点を明らかにすることが急務となる。また，属性文法に関する他の演算について，例えば埋め込み演算などの高次計算の検討，属性文法の差分的記述[30]の検討などが挙げられる。

参 考 文 献

[1] E. F. Codd, "Relational Comleteness of Data Base Sublanguages", in *Data Base Systems*, Prentice-Hall, 1972, pp. 33-64

[2] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communication of ACM*, Vol. 13, No. 2, Feburaury 1970, pp. 377-387

[3] 北川博之，“データベースシステム”，昭晃堂，1996 年 7 月

[4] A. Goldberg and D. Robinson, *"Smalltalk80 : The Language and Its Implementation"*, Addison-Wesley, 1983

[5] B. Stroustrup, *"The C++ Programming Language"*, Addison-Wesley, 1986

[6] L. Fegaras and D. Maier, "Towards an Effective Calculus for Object Query Languages", *Proceedings of ACM SIGMOD International Conference on Management of Data*, May 1995, pp. 47-58

[7] R. G. G. Cattel and D. K. Barry, *"The Object Database Standard : ODMG 2.0"*, San Francisco, California, Morgan Kaufmann Publishers, Inc., 1997

[8] 福村晃夫，稲垣康善，“オートマトン・形式言語理論と計算論”，岩波書店，1982 年 3 月

[9] 佐藤秀樹，伊藤暢浩，林達也，“文法表現によるオブジェクト指向データモデル”，情報処理学会データベースシステム研究報告，95-DBS-102，社団法人情報処理学会，1995 年 3 月，pp. 65-74

[10] 佐藤秀樹，“オブジェクト指向データモデルの拡張に関する研究”，九州大学学位論文，1998 年 7 月

[11] D. E. Knuth, "Semantics of context-free languages", *Mathematical Systems Theory*, Vol. 2, No. 2,1968, pp. 127-145

[12] 西野哲郎，片山卓也，佐々正孝他，“属性文法入門”，共立出版，1996 年 2 月

[13] 佐々正孝，“属性文法によるコンパイラの記述例”，情報処理，Vol. 35，No. 4，社団法人情報処理学会，1994 年 4 月，pp. 358-369

[14] 今泉貴史，篠田陽一，“ソフトウェア環境への属性文法の応用”，情報処理，Vol. 35，No. 7，1994 年 7 月，pp. 651-657

[15] H. Ait-Kaci, "An Algebric Semantics Approach to the Effective Resolution on Type Equations", *Theoretical Computer Science*, Vol. 45,1986, pp. 293-351

[16] F. Bancilhon and S. Khoshafian, "A Calclus for Complex Objects", Proceedings of Symposium on Principles of Database Systems, 1986, pp. 53-59

[17] 大堀淳，“オブジェクト指向データベースの形式化”，情報処理，Vol. 32，No. 5，1991 年 5 月，pp. 550-558

[18] L. Cardelli, "A Semantics of Multiple Inheritance", *Information Comupting*, Vol. 76,1988, pp. 138-164

[19] M. Kifer and G. Lausen, "F-Logic : A Higher Order Language for Reasoning about Objects, Inheritance, and Schema", *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1989, pp. 134-146

[20] S. Abiteboul and P. Kanellakis, "Object Identity as a Query Language Primitive", *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1989, pp. 159-173

[21] L. S. Colby and D. V. Gucht, "A grammar model for databases", *Technical Report No. 282, Computer Science*

Department, Indiana University, 1989

[22] G. H. Gonnet and F. W. Tompa, "Mind your grammar : a new approach to modeling text", *Proceedings of International Conference on Very Large Data Base*, 1987, pp. 339-346

[23] M. Gyssens, J. Paredaens, and D. V. Gucht, "A grammar-based approach towards unifying hierarchical data models", *Proceedings of International Conference on Very Large Data Base*, 1989, pp. 263-272

[24] C. F. Goldfard, *"The SGML Handbook"*, Oxford University Press, 1990

[25] 村田真，“XML 入門”，日本経済新聞社，1998 年 1 月

[26] M. Levene and A. Poulovassilis, "An object-oriented data model formalized through hypergraphs", *Data & Knowledge Engineering*, Vol. 6,1991, pp. 205-224

[27] G. Hedin, "An object-oriented notation for attribute grammars", Proceedings of ECOOP'89,1989, pp. 329-345

[28] Y. Shinoda and T. Katayama, "Object oriented extension of attribute grammars and its implementation using distributed attribute evaluation algorithm",*Lecture Notes in Computer Science 461*,Springer-Verlag,1990,pp.177-191

[29] K. Koskimies, "Object-orientation in attribute grammars", *Lecture Notes in Computer Science 545*, Springer-Verlag, 1991, pp. 297-329

[30] 永尾制一，“属性文法のモジュール化および差分的記述に関する研究”，東京工業大学修士論文，1998 年 2 月