プログラミング－言語・基礎・実践 12－8
（1993. 5. 28）

型付確定節プログラムの上昇証明木解釈

山本章博
北海道大学工学部

概要

本稿では確定節プログラムの構成方法を明確にするためにBUD解釈を導入する．BUD解釈とは，ある確定節プログラムの解釈領域として，別の確定節プログラムの上昇証明木全体の集合をとるような解釈である．BUD解釈を用いることにより，関係DBにおける実体関連モデルと確定節プログラムや演繹データベースとの関係を明確にし，実体-関連モデルに基づくプログラムの構成方法を与える．

Bottom-up Derivation Interpretations of Typed Definite Logic Programs

Akihiro YAMAMOTO
Department of Electrical Engineering, N 13 W 8, Kita-ku, Sapporo 060, JAPAN

In this paper we introduce bottom-up derivation interpretations (BUD interpretations, for short) of typed definite logic programs. A BUD interpretation is an iterpretation of a typed definite program, which uses the set of all bottom-up deribvations of another program as its domain. BUD interpretations clarify the relation between the Entity-Relationship models in relational database theory and logic programs. Then it gives a way how to construct logic programs based on Entity-Relationship models.

1 はじめに

本稿では, 論理プログラムの上昇証明木解釈 (bottom-up derivation interpretation, BUD 解釈) について議論する. BUD 解釈とは, ある確定節プログラムの解釈領域として, 別の確定節プログラムの上昇証明木全体の集合をとるような解釈である.

確定節プログラムの解釈領域としてはHerbrand 空間を用いるのが一般的である. Herbrand 空間を用いた意味論は, プログラムの計算過程や結果を記述するのに適しているが, 具体的なプログラムの構成法に関する議論には非力である. Herbrand 解釈を用いると, 基礎項間の関係をプログラム中にで定義される述語によって記述することになるが, どのような述語を用意すべきかという問に対しては全く答えることができない.

論理プログラムを, 特にデータベースのように利用するときには, どのような述語記号を準備すべきかが問題となる. 関係データベース理論においては, 実体 - 関連モデル (Entity-Relationship model) のような概念モデルによる関係設計などが確立されている.(例えば Ullman のテキスト [2] 参照). 論理プログラミングを関係データベースの拡張として扱うのであれば, 論理プログラミングにおいても, このような概念モデルからのプログラム設計が必要となる. 本稿の目的はではBUD 解釈によって, 実体 - 関連モデルからの確定節プログラムに必要な述語を構成する方法を与えることにある.

関係データベースではすべてのデータを属性のタプルの集合である関係 (relation) という形で表現する. 実体 - 関連モデルとは, 現実世界を実体の集合と実体間の関連で表現したモデルである. 実体 - 関連モデルが一つ与えられると, 関係データベース構築に必要な関係は次のようにして求めることができる. まず, 実体の集合を表現する関係を準備する. 各実体はこの関係の要素であるタプルとして表現される. 関連を表現する関係は, 実体集合を表現する関係のキーを属性として定義される.

このような関係設計法を確定節プログラムに拡張する場合には, 最小モデルの要素のキーが必要になる. 本稿ではそのキーとして, 原子論理式が導出された証明木 (の形) を採用する. そして, その証明木を直接プログラム中で扱うために基礎項の BUD 解釈を用いる. また, 実体の属性は, 項の BUD 解釈である証明木中の計算解代入を用いて, 項から項への関数として記述することとする.

2 準備

本稿では型付の一階述語論理と論理プログラムを対象にする.

2.1 型付確定節システム

型付の一階述語論理の言語を構成する要素は, 型, 変数, 関数記号, 述語記号である. 型の集合を T, 変数の集合を X, 関数記号の集合を Σ, 述語記号の集合を Π とするとき, $S = X \cup \Sigma \cup \Pi$ に対して, 二つの写像 $arity : S \longrightarrow T^*$, $range : S \longrightarrow T$ を次の条件を満たすように一つ定めておく.

- 変数 x に対して, $arity(x) = \varepsilon$, $range(x) \in T$.
- 述語記号 p に対して $range(p)$ は定義しない. 必要ならばダミーの型を用意する.

項, **原子論理式**, **節**, **確定節**は通常の一階述語論理と同様の手法で定義する. **確定節プログラム**とは確定節の有限集合である. BUD 解釈を定義するためには, 2 つの異なる言語上のプログラムを扱わなければならないので, プログラムとそれを記述する言語をあわせた概念を導入する.

定義 1 言語 L 上の確定節プログラム P に対して, L を構成するために用いた型の集合 T, 関数記号の集合 Σ, 述語記号の集合 Π をあわせた 4 つ組 (T, Σ, Π, P) を**型つき確定節システム** (TDS) とよぶ.

例 1 次のプログラムを TDS として扱うことを考える.

$$P = \left\{ \begin{array}{l} append(nil, y, y) \leftarrow \\ append(cons(x, y), z, cons(x, w)) \leftarrow append(y, z, w) \end{array} \right\}$$

そのためには, T, Σ, Π, および $aritty$, $range$ を明示しなければならない. 例えば, $T = \{symbol, list\}$, $\Sigma = \{a, b, nil, cons\}$, $\Pi = \{append\}$ とし,

$$\begin{array}{ll} arity(a) = arity(b) = arity(nil) = \varepsilon, & arity(cons) = symbol \cdot list, \\ arity(append) = symbol \cdot list \cdot list, & \\ range(a) = range(b) = symbol, & range(nil) = range(cons) = list \end{array}$$

と定義すると, (T, Σ, Π, P) が TDS となる.

例 2 Lloyd[1] にある 雇用者 - 部品 - 仕事を表す演繹データベースの一部は, 以下のような TDS となる.

$$\begin{array}{rcl} T & = & \{sno, sname, city, pno, pname, color, weight, jno, jname, number\} \\ \Sigma & = & \left\{ \begin{array}{l} S1, S2, S3, Smith, Jones, James, Adelede, Sydney, Perth, Melbourne, \\ P1, P2, Screw, Nut, White, Black, 10, 20, 100, 200, \\ J1, J2, Build, Repair, \end{array} \right\} \\ \Pi & = & \{supplier, part, job, quantity\} \\ P & = & \left\{ \begin{array}{lcl} C_{s_1} & : & supplier(S1, Smith, Adelade) \leftarrow \\ C_{s_2} & : & supplier(S2, Jones, Sydney) \leftarrow \\ C_{s_3} & : & supplier(S3, James, Perth) \leftarrow \\ C_{p_1} & : & part(P1, Screw, White, 10) \leftarrow \\ C_{p_2} & : & part(P2, Nut, Black, 10) \leftarrow \\ C_{j_1} & : & job(J1, Build, Melbourne) \leftarrow \\ C_{j_2} & : & job(J2, Repair, Sydney) \leftarrow \\ C_{q_{100}} & : & quantity(100) \leftarrow \\ C_{q_{200}} & : & quantity(200) \leftarrow \end{array} \right\} \end{array}$$

例 3 横田 [3] にある巡回路をもった経路を TDS で表現すると以下のようになる.

$$T = \{point\},\ \Sigma = \{a, b, c, d, e\},\ \Pi = \{arc, path\}$$

$$P = \left\{ \begin{array}{lllllll} C_{arc1} & : & arc(a, b) \leftarrow & C_{arc2} & : & arc(b, c) \leftarrow & C_{arc3} : arc(b, d) \leftarrow \\ C_{arc4} & : & arc(d, e) \leftarrow & C_{arc5} & : & arc(e, b) \leftarrow & \\ C_{path1} & : & path(x, y) \leftarrow arc(x, y) & C_{path2} & : & path(x, y) \leftarrow arc(x, z), path(z, y) & \end{array} \right\}$$

定義 2 TDS $S_1 = (T_1, \Sigma_1, \Pi_1, P_1)$, $S_2 = (T_2, \Sigma_2, \Pi_2, P_2)$ に対して, $T_1 \subset T_2$, $\Sigma_1 \subset \Sigma_2$, $\Pi_1 \subset \Pi_2$, $P_1 \subset P_2$ のとき, S_1 は S_2 の**部分システム**であるといい, $S_1 \subset S_2$ とかく.

2.2 型つき木

項と証明図を同一方法で扱うために, 最初に型付木を定義しておく.

定義 3 T, S を集合とし, 二つの写像 $arity : S \longrightarrow T^*$, $range : S \longrightarrow T$ が与えられているものとする. このとき, T **によって型付けされた** S **上の木** t とは, $\mathrm{N}_+{}^*$ の有限部分集合 $dom(t)$ から S への写像で以下の条件を満たすものである:

1. $dom(t)$ は前綴に関して閉じている, すなわち, $u \cdot w \in dom(t)$ ならば $u \in dom(t)$ である.

2. ある $k \geq 1$ に対して $w \cdot k \in dom(t)$ であれば, $i = 1, \ldots, k$ に対して $w \cdot i \in dom(t)$ である.

3. $arity(t(w)) = \tau_1 \cdots \tau_n$ かつ $\{w \cdot 1, \ldots, w \cdot n\} \subset dom(D)$ であれば, $range(t(w_i)) = \tau_i$ かつ $w \cdot (n+1) \notin dom(D)$ である.

なお, $range(t(\varepsilon))$ を t の型といい, $range(t)$ とかく.

例 4 $T = \{symbol, list\}$, $S = \{a, b, nil, cons\}$ とし,

$$arity(a) = arity(b) = arity(nil) = \varepsilon, \qquad arity(cons) = symbol \cdot list$$
$$range(a) = range(b) = symbol, \qquad range(nil) = range(cons) = list$$

とするとき, $dom(t)$ と t を

$$dom(t) = \{\varepsilon, 1, 2, 2 \cdot 1, 2 \cdot 2\}$$
$$t(\varepsilon) = cons,\ t(1) = a,\ t(2) = cons,\ t(2 \cdot 1) = b,\ t(2 \cdot 2) = nil$$

と定義すれば, t は T によって型付けられた S 上の木である.

T によって型付けされた S 上の木全体の集合を, $\mathcal{T}_T(S)$ と書く. 添字の T は明らかな場合には省略する. TDS(T, Σ, Π, P) における項全体の集合は, $\mathcal{T}(\Sigma \cup X)$ と同一視することができるので, 以下においては両者を区別しない.

次に, 導出原理による証明木における変数の付けかえを統一的に行なうために, 変数全体の集合に次のような構造を仮定する.

[仮定] $X^\tau = \{x\ ;\ range(x) = \tau\}$ とするとき, $X^\tau = \sum_{w \in \mathbf{N}^*} X^\tau_w$ であり, しかも任意の $w, u \in \mathbf{N}^*$ に対して, $\phi_{w,u} : X^\tau_w \longrightarrow X^\tau_u$ なる全単射が存在する. X^τ_ε 中の変数 x に対して, $\phi_{\varepsilon,w}(x)$ を x_w と書く.

TDS の記述には X_ε 中の変数を用いることとし, $w \neq \varepsilon$ なる X_w 中の変数は BUD を構成するときに用いるものとする. 相異なる $w, u \in \mathbf{N}^*$ に対して, 変数付け換え代入 $\sigma_{w,u}$ を $\theta_{w,u} = \{x_w := x_u\ ;\ x \in X_\varepsilon\}$ と定義する.

3 BUD 解釈

3.1 上昇証明木

確定節プログラムにおける導出原理による上昇証明は, 節の選択と単一化からなる. そこで, 節の選択だけを先に行なって証明木の骨格 (skelton) を構成し, 次に, その骨格が実際に証明木になっているかどうかを単一化によって判定することにする.

定義 4 TDS(T, Σ, Π, P) における**証明骨格**とは, $\mathcal{T}(P)$ の要素となる木 σ で,

$$\sigma(w) = A \leftarrow B_1, \ldots, B_n \text{ かつ } \sigma(w \cdot k) = A_k \leftarrow B_{k,1}, \ldots, B_{k,n_k} \text{ ならば } pred(B_k) = pred(A_k)$$

であるものをいう.

定義 5 TDS $S = (T, \Sigma, \Pi, P)$ に対して, $EQ(S) = \{t = s\ ;\ t, s \in \mathcal{T}(\Sigma \cup X) \text{ and } range(t) = range(s)\}$ と定義する.

定義 6 TDS S における証明骨格 σ に基づく**上昇証明木** $D(\sigma)$ とは, 任意の $w \in dom(\sigma)$ に対して $D(\sigma)(w) = (\sigma(w), E(w))$ なる $\mathcal{T}(P \times EQ(S))$ の木である. ここで, 任意の k について $w \cdot k \in dom(\sigma)$ であれば, $E(w) = \phi$ とし, $\sigma(w) = (A \leftarrow B_1, \ldots, B_n)$ かつ $\sigma(w \cdot k) = (A_k \leftarrow B_{k,1}, \ldots, B_{k,n_k})$ であれば $E(w) = \bigcup_{k=1}^{n} \{B_k\theta_{\varepsilon,w} = A_k\theta_{\varepsilon,w\cdot k}\}$ とする.

TDS S 上の上昇証明木全体の集合を $D(S)$ と表す.

定義 7 証明骨格 σ に基づく上昇証明木 $D(\sigma)$ に対して, 等式集合 $E(\sigma) = \bigcup_{w \in dom(\sigma)} E(w)$ を $D(\sigma)$(または σ) の**計算等式**という. $E(\sigma)$ が解形式 (solved from)θ をもてば, θ を代入とみなしたものを $D(\sigma)$(または σ) の**解代入**という.

定義 8 証明骨格 σ に対して, $pred(head(\sigma((\varepsilon))))$ を σ の**根述語記号**といい, $pred(\sigma)$ と表す. 上昇証明木 $D(\sigma)$ に対しても $pred(\sigma)$ をその**根述語記号**という.

例 5 例 3 の TDS S における証明骨格の例として以下のように定義された σ がある.

$$
\begin{aligned}
dom(\sigma) &= \{\varepsilon, 1, 2, 2\cdot 1, 2\cdot 2, 2\cdot 2\cdot 1, 2\cdot 2\cdot 2, 2\cdot 2\cdot 2\cdot 1, 2\cdot 2\cdot 2\cdot 2, 2\cdot 2\cdot 2\cdot 2\cdot 1\} \\
\sigma(\epsilon) &= path(x,y) \leftarrow arc(x,z), path(z,y) \\
\sigma(1) &= arc(a,b) \leftarrow \\
\sigma(2) &= path(x_2,y_2) \leftarrow arc(x_2,z_2), path(z_2,y_2) \\
\sigma(2\cdot 1) &= arc(b,d) \leftarrow \\
\sigma(2\cdot 2) &= path(x_{2\cdot 2},y_{2\cdot 2}) \leftarrow arc(x_{2\cdot 2},z_{2\cdot 2}), path(z_{2\cdot 2},y_{2\cdot 2}) \\
\sigma(2\cdot 2\cdot 1) &= arc(d,e) \leftarrow \\
\sigma(2\cdot 2\cdot 2) &= path(x_{2\cdot 2\cdot 2},y_{2\cdot 2\cdot 2}) \leftarrow arc(x_{2\cdot 2\cdot 2},z_{2\cdot 2\cdot 2}), path(z_{2\cdot 2\cdot 2},y_{2\cdot 2\cdot 2}) \\
\sigma(2\cdot 2\cdot 2\cdot 1) &= arc(e,b) \leftarrow \\
\sigma(2\cdot 2\cdot 2\cdot 2) &= path(x_{2\cdot 2\cdot 2\cdot 2},y_{2\cdot 2\cdot 2\cdot 2}) \leftarrow arc(x_{2\cdot 2\cdot 2\cdot 2},z_{2\cdot 2\cdot 2\cdot 2}) \\
\sigma(2\cdot 2\cdot 2\cdot 1) &= arc(b,c) \leftarrow
\end{aligned}
$$

3.2 メタシステムと BUD 解釈

TDS S_1 の解釈の領域として, 別の TDS S_2 の BUD 全体 $D(S_2)$ をとることを考える.

定義 9 二つの TDS$S_1 = (T_1, \Sigma_1, \Pi_1, P_1)$ と $S_2 = (T_2, \Sigma_2, \Pi_2, P_2)$ が以下の 2 条件 (M1), (M2) を満たすとき, S_1 は S_2 の**メタシステム**であるという.

(M1) $T_1 = \Pi_2$.

(M2) 写像 $\gamma : \Sigma_1 \longrightarrow P_2$ が存在して, 任意の $f \in \Sigma_1$ に対して,

$$
\begin{aligned}
&arity(f) = \tau_1 \cdots \tau_n, range(f) = \tau, \gamma(f) = A \leftarrow B_2, \ldots, B_n \text{ ならば} \\
&pred(A) = \tau \text{ and } pred(B_k) = \tau_k \ (k = 1, \ldots, n).
\end{aligned}
$$

任意の S_2 に対して, それのメタシステムであるような S_1 の存在は容易に示すことができる. その逆も次のようにして示すことができる。

補題 1 *TDS* $S_1 = (T_1, \Sigma_1, \Pi_1, P_1)$ が $\Sigma_1 \neq \phi$ を満たせば, S_1 がメタシステムであるような *TDS* S_2 が存在する.

証明 $\Sigma_1 = \{f_1, \ldots, f_n\}$ とし, $arity(f_i) = \tau_{i,1} \cdots \tau_{i,n_i}$, $range(f_i) = \tau_i$. このとき, TDS $S_2 = (T_2, \Sigma_2, \Pi_2, P_2)$ を, $T_2 = \{bit\}$, $\Sigma_2 = \{s_0, s_1\}$, $\Pi_2 = T_1$, $P_2 = \{\gamma(f_i)\ ;\ f_i \in \Sigma_1\}$ と定義すれば, S_1 は S_2 のメタシステムとなる. ここで, $i = 1, \ldots, n$ に対して

$$\gamma(f_i) = \tau_i(\underbrace{s_1(\cdots(s_1}_{i \text{ times}}(s_0))\cdots))) \leftarrow \tau_{i,1}(x_1), \ldots, \tau_{i,n_i}(x_{n_i})$$

とする.

定理 1 *TDS* $S_1 = (T_1, \Sigma_1, \Pi_1, P_1)$ が $S_2 = (T_2, \Sigma_2, \Pi_2, P_2)$ のメタシステムであり, $T_1 \cap T_2 = \phi$, $\Sigma_1 \cap \Sigma_2 = \phi$ であるとする. *TDS* $S_1' = (T_1 \cup T_2, \Sigma_1 \cup \Sigma_2, \Pi_1, P_1)$ がメタシステムとなるような *TDS* $S_2' = (T_2', \Sigma_2', \Pi_2', P_2')$ で, $S_2 \supset S_2'$ なるものを構成できる.

例 6 $T_1 = \{atom, list\}$, $\Sigma_1 = \{a_1, \ldots, a_n, nil, cons\}$ とすると, 上の証明で得られる TDS$S_2 = \{\{bit\}, \{s_0, s_1\}, \Pi_2 = T_1, P_2\}$ については

$$P_2 = \left\{ \begin{array}{lcl} C_1 & : & atom(s_0) \leftarrow \\ \quad \cdots & & \\ C_n & : & atom(s_1(s_1(\cdots(s_1(s_0))\cdots))) \leftarrow \\ C_{nil} & : & alist(s_0) \leftarrow \\ C_{cons} & : & \forall x/bit\, y/bit\, alist(s_1(s_0)) \leftarrow atom(x), alist(y) \end{array} \right\}$$

となる. 写像 γ を

$$\gamma(a_1) = C_1, \ldots, \gamma(a_n) = C_n, \gamma(nil) = C_{nil},\ \gamma(cons) = C_{cons}$$

と定義すれば, S_1 は S_2 のメタシステムである.

例 7 $T_1 = \{num\}$ かつ $\Sigma_1 = \{0, s\}$ の場合には, TDS $S_2 = (\{bit\}, \{s_0, s_1\}, \{num\}, P_2)$ を

$$P_2 = \left\{ \begin{array}{lcl} C_0 & : & num(s_0) \leftarrow \\ C_s & : & num(s_0) \leftarrow num(s_0) \end{array} \right\},$$

と定義し, さらに $\gamma(0) = C_0$, $\gamma(s) = C_s$ と γ を定義すれば, S_1 は S_2 のメタシステムとなる.

定義 10 S_1 が S_2 のメタシステムであるような S_2 に対して, S_1 の次のように定義された前解釈 J に基づく解釈を S_1 の S_2 による **BUD 解釈**という.

- 任意の $\tau \in T_1$ に対して $J(\tau) = \{D \in D(S_1)\ ;\ pred(D) = \tau\}$.
- $f \in \Sigma_1$ が $arity(f) = \tau_1 \times \cdots \tau_n$, $range(f) = \pi$ であるとき, 任意の $D_1 \in J_{\tau_1}$, $\cdots$, $D_n \in J_{\tau_n}$ に対して, $J(f)(D_1, \ldots, D_n)$ を次のように定義する
  1. $dom(J(f)(D_1, \ldots, D_n)) = \bigcup_{i=1}^{k} \{k \cdot w\ ;\ w \in dom(D_k)\}$.
  2. $k = 1, \ldots, n$ に対して, $D(k \cdot w) = D_k(w)\sigma(w, k \cdot w)$ とする. ここで $\sigma(w, w \cdot k) = \{x_w := x_{k \cdot w}\ ;\ x_w \in X_w\}$.
  3. $D(\varepsilon) = (\gamma(f), E)$ かつ E は D が上昇証明木の定義を満たすように定める.

S_1 の S_2 による BUD 解釈で S_1 のモデルとなるのものを, S_1 の S_2 による **BUD モデル**という.

4 BUD 解釈を用いた実体 - 関連モデルからのプログラミング

この節では与えられた実体関連モデルを実現する確定節プログラムに必要な述語の構成法を与える. 最初に実体関連モデルを形式的に定義する.

定義 11 実体 - 関連モデルとは 8 項組 $(T, A, C, R, \tau, \alpha, \rho, H)$ である. ここに,

- T, A, C, R は相異なる集合であり, $is\text{-}a \notin R$ である. T, A, S, R の要素をそれぞれ **型**, **属性名**, **実体集合名**, **関連名**とよぶ.
- τ は A から T への写像である. $\tau(a)$ を属性 a の**型**という.
- α は C から A^* への写像である. $\alpha(c)$ を c の**属性型**という.
- ρ は R から S^* への写像である. $\rho(r)$ を r の**引数型**という.
- H は, C 上の半順序関係である. $(c, c') \in H$ のとき, c は c' と is-a の関係にあるという.

定義 12 $(c, c') \in H$ なる c' が存在しない実体集合名 c を C の H に関する**基底**という. C の H に関する基底全体の集合を C_H と表す.

例 8 雇用者 - 部品 - 仕事を表す演繹データベースは, T, A, C, R は,

$$
\begin{aligned}
T &= \{term\} \\
A &= \{sno, sname, city, pno, pname, color, weight, jno, jname, quantity\} \\
C &= \{supplier, part, job, number, local\text{-}supplier, major\text{-}supplier\} \\
R &= \{spj, is\text{-}a\}
\end{aligned}
$$

とし, τ, α, ρ, H を以下のように定義した実体関連モデル M に基づいている.

$$
\begin{aligned}
&\tau(x) = term \text{ for all } a \in A \\
&\alpha(supplier) = sno \cdot sname \cdot city \\
&\alpha(part) = pno \cdot pname \cdot color \cdot weight \\
&\alpha(job) = jno \cdot jname \\
&\alpha(number) = quantity \\
&\alpha(local\text{-}supplier) = sno \\
&\alpha(major\text{-}supplier) = sno \\
&\rho(spj) = supplier \cdot part \cdot job \cdot number \\
&H = \{(local\text{-}supplier, supplier), (major\text{-}supplier, supplier)\}
\end{aligned}
$$

このとき, $C_H = \{supplier, part, job, number\}$ である.

さて, 実体 - 関連モデル $M = (T, A, C, R, \tau, \alpha, \rho, H)$ が一つ与えられたとき, これに基づいて, TDS を構成する方法を与える.

定義 13 TDS $S_E^0 = (T_E, \Sigma_E, \Pi_E, P_E)$ は, $T_E = T$, $\Pi_E = C_H$ であり, 任意の $p \in \Pi_E$ に対して $arity(p) = \alpha(p)$ であるとき, M **の実体集合の基底** C_H **を表現している**という.

例 9 例 2 の TDS S は例 9 の実体関連モデル M の実体集合の基底 C_H を表現している.

定義 14 TDS $S_R = (T_R, \Sigma_R, \Pi_R, P_R)$ は, $T_R = T \cup C$, $\Pi_R \supset R \cup (C \times A)$ であり, 任意の $p \in \Pi_R$ に対して

$$\begin{aligned} &p \in R\text{ならば} && arity(p) = \rho(p) \\ &p \in C \times A\text{ならば} && arity(p) = c \cdot \tau(p) \end{aligned}$$

を満たし, しかも $C \times A$ に含まれる述語記号は P_R の頭部に出現しないとき, S_R は M **の関連** R **を表現している**という.

命題 1 *TDS* S_E^0 が M の実体の基底を表現し, S_R が M の関連を表現しているとき, $S_E^0 \subset S_E$ かつ S_R が S_E のメタシステムであるような S_E を構成することができる.

定義 15 TDS $S_R = (T_R, \Sigma_R, \Pi_R, P_R)$ が M の関連 R を表現しているとき, $\Pi_R \supset C - C_H$ であり,

$$p \in C - C_H\text{ならば} arity(p) = base(p)$$

であり, さらに Π_R には $h(c) > h(c')$ なる階層関係があり, P_R がその階層関係で階層プログラムになるとき, S_R は**階層関係** H **を表現している**という.

属性 a の解釈 $I(a)$ を与えることにより, P_R の解釈を次のような繰り返しで定義する.

$$\begin{aligned} T_{P_R} \uparrow 0 &= \bigcup_{a \in A} I(a) \\ T_{P_R} \uparrow (k+1) &= T_{P_R} \uparrow k \\ T_{P_R} \uparrow \omega &= \bigcup_{k \in \omega} T_{P_R} \end{aligned}$$

例 10 Lloyd[1] 雇用者 - 部品 - 仕事という関連を, 例 2 の TDS のメタシステム S_R として定義する. メタシステムの定義から, $S_R = (\{supplier, par, job, quality\}, \{s_1, s_2, p_1, p_2, j_1, j_2, q_{100}, q_{200}\}, \Pi_R, P_R)$ という形である. ここで,

$$\begin{aligned} &\gamma(s_1) = C_{s_1}, && \gamma(s_2) = C_{s_2} \\ &\gamma(p_1) = C_{p_1}, && \gamma(p_2) = C_{p_2} \\ &\gamma(j_1) = C_{j_1}, && \gamma(j_2) = C_{j_2} \\ &\gamma(q_{100}) = C_{q_{100}}, && \gamma(q_{200}) = C_{q_{200}} \end{aligned}$$

とする. ここで, $arity(spj) = supply \cdot par \cdot job \cdot quality$ と定義すれば, 雇用者 - 部品 - 仕事の関連をプログラムで表すと, 次のようになる.

$$P_R = \left\{ \begin{array}{l} spj(s_1, p_1, j_1, q_{100}) \leftarrow \\ spj(s_2, p_2, j_2, q_{200}) \leftarrow \\ local\text{-}supplier(s_1) \leftarrow \\ local\text{-}supplier(x) \leftarrow city(x, Melborne) \\ major\text{-}supplier(x) \leftarrow quantity(x, q), q \geq q_{100} \end{array} \right\}$$

参考文献

[1] Lloyd, J. W. , *Foundations of Logic Programming : Second, Extended Edition*, Springer - Verlag, 1987.

[2] Ullman, J. D. , *Princeples of Dadabase and Knowledge-base Systems, VolumeI, II*, Computer Science Press, 1988.

[3] 横田一正, 演繹オブジェクト指向データベースについて, コンピュータソフトウエア 9(4):3–18 (1992).