対象の属性の変化として捉えた処理のモデル化

3J－9

古宇田フミ子　近山 隆
{fumiko, chik}@logos.t.u-tokyo.ac.jp
東京大学工学部*

1　はじめに

計算機に処理を指示する時にはその計算機に特有なコマンドが必要になる。計算機向きの指示方法をそのまま人間にも適用しているために、利用者には不便である。この問題の一つの解決法は、人間が計算機向きに翻訳されたコマンドを直接使うのではなく、利用者は人間向きの言葉で説明し、計算機側で人間の言語世界で説明されたことを理解可能とするような知的システムを構成すること、即ち、利用者には計算機にやらせたいことを主体的に比較的自由な表現法で記述でき、システム側では、この記述を正しく解釈し、対応するコマンド群を導けるようにすることである。利用者の自由な記述は知識や表現法の違いにより多様性が生じ得るが、計算機の処理の説明なので、ある動作を説明する、という点では共通性がある。この点に注目して動作を説明する方法の違いにより、既に、三層構造の処理モデルを基本構想として提案した: これらは、利用者の記述から、同じ意図を表す標準的な表現に変換する「表現層」: 処理を形式化した「論理層」: 論理層での形式的説明を実在するコマンドに写像する「実現層」である。

利用者の多様な記述とOS毎に異なり得るコマンド間を直接対応つけるのは容易でない。これらの中間に処理を概念化、形式化した論理層を設けた。論理層では、処理を属性を変化させることとして捉えた。今回は、論理層のモデル化を属性と実体の面から考察し、処理の意味の明確化を図る。

2　論理層のモデル化の考慮点

(1) 処理概念の形式化。
処理の形式化の方針として、処理の主体を「システムが」とし、処理されるものを「実体（ entity ）」と捉え、処理は実体を構成する属性（ attribute ）の一部を変化させると見る。形式的には $entity = \{attributes\}$ となる。

*Modelling of computer processings based on the view of changing of entities and their own attributes
Funiko Kouda and Takashi Chikayama
Graduate School of Engineering, the University of Tokyo
3-1 Hongo 7-chome, Bunkyo-ku, Tokyo 113-8656, Japan

属性の変化はどのような手順で進めるか（ how to do ）ではなく、何をやるか（ what to do ）という視点で見る。処理における変化前と変化後の結果に焦点を置く。
(2) 利用者の計算機理解を反映した(システムでの)処理の見せ方。
利用者が処理をどう見ているか、に重点を置く。システムの処理に忠実に属性の変化を表すのではない。利用者が意識していないと処理が進まない場合は、利用者に強く意識（ user awareness ）させ、システム処理としては重要であっても利用者にはさほど関係ない属性は見せない（ user transparent ）。
(3) 処理を行なうための前提条件の明記。
筆者らが1997年12月に行なったアンケート結果によると、計算機を使う時の利用者の悩みでは、一連の処理の繋がりのうち一部のみに注目しているために、前提条件が分からなくて使えない、という事実があった。また、通常の計算機マニュアルではコマンドそのものの詳しい説明はあるものの関係する前提条件などは分かりにくい面もある。このような不便を解消するために、利用者には処理に必要な前提条件を見せる必要がある。

3　論理層のモデルの構成法

3.1　属性のみに注目した処理

属性の変化として動作の説明を行なう。利用者の視点では少なくとも以下の候補が可能である。
(1) 一つの処理過程は、開始、途中、終了結果に分けられる。
(2) 処理には変化する属性がある。以下の場合がある。
(i) 処理開始時点に処理のために着目された属性（ 着目属性（ ca ） ）が処理によって変化する。処理の結果、変化した属性を更新属性（ ma ）と呼ぶ。この場合は更に、(a) 処理後は着目属性は消滅する、または、(b) 処理後も着目属性が残る、の二通りがある。
(ii) 変化する属性は処理開始時点では存在せず、処理中に以下のいずれかで属性を生成する。(a) 動的に採り入れた情報（ (3)(ii) ）を利用して、または、(b) システムに存在する属性（ (3)(i) ）を参照して、または、(c) シ

$$\left\{ \begin{array}{c} 1 \\ ca \end{array} \right\} \times \left\{ \begin{array}{c} ra \\ 1 \end{array} \right\} \times \left\{ \begin{array}{c} inf \\ 1 \end{array} \right\} \rightarrow$$

$$\left\{ \begin{array}{c} ga \\ \left\{ \begin{array}{c} 1 \\ ca \end{array} \right\} \times \left\{ \begin{array}{c} ma \end{array} \right\} \end{array} \right\} \times \left\{ \begin{array}{c} ra \\ 1 \end{array} \right\} \times \left\{ \begin{array}{c} inf \\ 1 \end{array} \right\}$$

図 1: Processing model based on attributes

ステムのアルゴリズム等により決まった法則を用いる。これを生成属性 (ga) と呼ぶ。
(iii) 着目属性が処理によって消滅 (ϕ と表す) する。

ここで、処理による変化を $\rightarrow$ で表すと、(i)(a) $ca \rightarrow ma$ (i)(b) $ca \rightarrow ma, ca$ (ii)(a) $inf \rightarrow ga$ (ii)(b) $ra \rightarrow ga$ (ii)(c) $\phi \rightarrow ga$ (iii) $ca \rightarrow \phi$ となる。
(3) 処理時に他の属性情報を利用して着目属性を変化させる場合がある。利用される情報を関連属性 (la) と呼ぶ。情報源としては、(i) システムに既存で 処理開始時点から存在するもの (参照属性 ra) と、(ii) 処理中に動的に利用者等や他所から由来するもの (inf) がある。この情報は、(a) 処理時に参照され、処理後もそのまま残る場合と (b) 処理に取り込まれて使われてしまう場合がある。

処理による変化は (i)(a) $ra \rightarrow ra$ (i)(b) $ra \rightarrow \phi$ (ii)(a) $inf \rightarrow inf$ (ii)(b) $inf \rightarrow \phi$ となる。

処理では、(2) の変化する属性だけが関係する場合と、(2) と (3) が組み合わされる場合がある。

後者は、(3)(i) と (3)(ii) を処理開始側の属性として同一視すると $ca, la \rightarrow ma$ または $ca, la \rightarrow ma, ca$ または $ca, la \rightarrow ma, la$ または $ca, la \rightarrow ma, la, ca$ または $ca, la \rightarrow \phi$ または $ca, la \rightarrow la$ となる。

ここで、属性の変化を組合せる表現記号を導入する。処理で、$a_i \rightarrow b_i$ と $c_k \rightarrow d_k$ を組合せる場合は、

$\{a_i \rightarrow b_i\} \times \{c_k \rightarrow d_k\} = \{a_i, c_k \rightarrow b_i, d_k\}$

と表す。但し、右辺は $i \times j$ の要素の組合せで行毎に矢印がある場合は矢印の左側には左辺同士、右には右辺同士を並べ、ない場合はそのまま並べる。$1 \times M = M$, $M \times 1 = 1$, $1 \times 1 = \phi$ とする。

この記法を用いると、(2)(i) と (iii) は以下になる。

$$\left\{ ca \rightarrow \phi \right\} \times \left\{ \begin{array}{c} \phi \rightarrow ma \\ \phi \rightarrow ma, ca \\ 1 \end{array} \right\} = \left\{ \begin{array}{c} ca \rightarrow ma \\ ca \rightarrow ma, ca \\ ca \rightarrow \phi \end{array} \right\}$$

属性の変化のみに注目した処理全体 ((2) だけと (2) と (3) を組み合わせたもの) は図 1のように表される。

表 1: Examples

	開始	処理中	結果
(1)	ϕ	$inf \rightarrow ga, db$	$ga + db$
(2)	$ca + cb$	$ca, ra_i \rightarrow ma$	$ma + db, ca + cb$
	$ra_i + rb_i$	db	$ra_i + rb_i$

3.2 実体と処理

属性は実体の構成要素であり、属性と実体は独立には存在し得ない (*)。しかも、利用者には実体が見えている。そこで、処理に関係する属性を a, その他の属性を b で表し、実体を $e = a + b$ で表すことにする。

処理の開始点では、3.1節の (2)(i), (3) に対応する実体はは各々 $ca + cb$, $ra + rb$ となる。

処理中は開始点で取り込んだ実体は見かけ上属性に分解 (再構成) される。3.1節の (2)(ii) や (*) から処理中に変化した属性 (a とする) との組合せにより新たな実体が生じることもある。これを $a + db$ とする。

処理結果では、処理に関係した属性とその他の属性が再構成される。3.1節の記号を用いると、処理結果で生じ得る個々の実体は (i) 属性が変化したものについては $ma + cb, ma + db, ma + rb, ga + db, ga + rb$ (ii) 処理前から存在するものが処理後も残った場合は、$ca + cb, ra + rb$ (iii) 処理前から存在するものが処理後に再構成される場合は、$ca + db, ca + rb, ra + cb, ra + db, inf + db, inf + cb, inf + rb$ (iv) $ca + cb, ra + rb$ 等の形の実体が複数ある場合は、お互いの間で「組み替え」があり得るから、$ca_i + cb_i, ra_i + rb_i$ の他に $ca_i + cb_k, ra_i + rb_j, ma_i + cb_j$ 等がある。

処理開始、処理中、結果各々の実体を組み合わせることで一つの処理ができる。これを Pea と呼ぶ。Pea を組み合わせることでより複雑な処理の骨格が示される。

3.3 実体モデルの表現と解釈例 − 表 1参照

(1) 外部情報を基に管理テーブルを作る。(2) ライブラリ ($ra_i + rb_i$) を見ながら、ソースプログラム ($ca + cb$) をコンパイルし結果として ($ma + mb$) を生成する。他のファイルも残っている。

4 考察

アトミックな処理を Pea として表した。形式化の表示ではすべての場合を数え上げるのではなく、必要要素との組合せで表現可能である。利用者の認識を考慮したので、無から有が生じたり、処理中に消滅したりすることが起こる。今後の課題は、2章 (3) より Pea 間を繋ぐ時の属性の条件の明確化や利用者の意識調査等によりモデルの精密化を行なうことである。