B-014

表現法と解釈法の視点に立つ写像としての処理モデル構成法

A Processing Model as a Mapping in terms of Representation and Interpretation

古宇田 フミ子* 相田 仁†
Fumiko Kouda Hitoshi Aida

1 はじめに

1.1 背景

計算機上では、自然言語を反映した表現や特定の規則に基づく表現、jpeg 等のデータ圧縮表現、など、多様な表現法があり、それぞれに解釈される。計算機処理は、異なる表現法、例えば、eps 形式の図面と ascii コードの文章とを組み合わせて、「間違いなく」目的のファイルを作成する。計算機上の表現や解釈の多様性の中で、処理の種類も多々あるが、どのような種類の処理であっても、入力データを取り込み、結果をもたらすという共通機能がある。しかし、それ以上の規則性、処理に内在する普遍性は必ずしも明確化されていない。ここでは、現実を簡単化して、処理を写像として捉え (副作用は考えない)、この関係を表現法と解釈法で記述することで、処理における共通機能に内在する規則性を見出すことを目指す。

1.2 問題の捉え方

計算機環境に沿って、モデル要素には以下を考慮する [1][2]。
(s1) 処理ではやり方ではなく、所望の結果が重要である。処理の種類の違いに依らず、処理は入力と結果の対応関係として捉える。また、この対応関係では、どのような表現で表され、どのような機能を表すかが重要なので、この関係を表現法と解釈法を用いて記述する。

ここで、表現法は、表すために用いる記号 (例えば、仮名「あ」、ローマ字「a」等) を用いて、特定の規則に沿った記号列のリストとして表されていると解釈する述語であり、表現は、材料となる記号の表現を基として、この表現法の規則に従い、記号列のリストとして表されたものの説明とする。

解釈法は、表現が「ある表現法の表現を入力とし、これを写像して、その結果を表す」関係として読み取れるか否かの視点から判断する述語とする。解釈値は、入力、写像、結果の関係を表す処理の説明または、その他 (=この解釈では処理を表さない) の二つの場合に制限する。

表現と解釈値に関する基本演算は、解釈法で単位を表す表現の記号列を表現の単位 (単位項) とし、表現列とその解釈値は append によるリストの増減として考える。表現と解釈値を一組とし扱う。
(s2) モデル空間には計算オブジェクトのみが存在すると仮定する。計算オブジェクトは表現法と解釈法で記述される。処理実行の行為者や処理の入力、処理の結果は計算オブジェクトであり、これらの関係も計算オブジェクト間の表現法と解釈法の関係として記述されるものとする。計算オブジェクトを形式化して論理式の形で表す場合は、Z[3][4] の記法で表す。
(s3) 処理の入力には、利用者指定の計算機上のオブジェクトとともに、コマンド自体の更新が外部に影響を与える環境変数や、利用者には見え難いが、コマンドが機能するために必須の共用ライブラリ等も計算オブジェクトとして加える。この理由は、処理では、基幹部分の他に、処理毎に固有なシステム共通の Gliba, xxstd++.so 等の情報や環境変数を引数として用いている。処理を入力と結果の間の関係として捉える場合、これらの情報も処理の入力の一部として扱うと、処理に関わるシステム情報を呈示する手段となり得るためである。

考察する空間に存在するものは計算オブジェクトのみとし、計算オブジェクトが自身の解釈法としての振舞いを表す処理記述に従って、他の (複数種類の) 計算オブジェクトを入力とし、別の計算オブジェクトをもたらす関係を処理とする。このことで処理が処理の種類に依らず表現法と解釈法に基づいて、統一的に記述できることを示すことを目標とする。

コンパイラで用いられる T 図式では、プログラム言語を直接機械語にするのではなく中間言語を導入して、効率化、簡潔化を図る概念である。ここでの処理のモデル化は処理を入力と結果の関係から、処理に内在する普遍性を形式化するものであり、目的が異なる。

*
†東京大学 大学院工学系研究科

2　準備:処理モデル構成要素

2.1　処理で考慮する特徴の抽出

問題解決のために以下の項目に注目する。
1.2 節の (s1) と (s2) については、表現法と解釈法が関わるので、これらを扱う道具として、言語 $L_{\rm G}$ を導入する。

言語 $L_{\rm G}$ では、

(i) 記号の種類と並びの規則を表す表現法の述語 $p_{\rm SY}$ と、この $p_{\rm SY}$ で真 (true) となる表現を説明する関数 $f_{\rm SY}$、

(ii) 処理を表すか、その他を表すかを解釈する解釈法の述語 $p_{\rm MN}$ と、この $p_{\rm MN}$ で真 (true) となる解釈値を表す関数 $f_{\rm MN}$ を持つ (式 (1))。

$$L_{\rm G} = (p_{\rm SY} : f_{\rm SY}, p_{\rm MN} : f_{\rm MN}). \quad (1)$$

(iii)　解釈法で単位を表す表現の記号列を表現の単位 (unitIntp($p_{\rm SY}$) の形で表す: 単位項) とする。例えば、表現の記号列が'1001' の場合、バイナリとして理解すると unitIntp($p_{\rm Bin}$) = {'1', '0'} 16 進として理解すると、unitIntp($p_{\rm Hex}$) = {'9'} となる。

表現法と解釈法の間には、

$$sem_p = f_{\rm MN}(syn_j). \quad (2)$$

が成り立つ。

理解に関しては、それを説明する概念間の関係を利用して関係づける。

(iv)　ここでは、簡単のため、単位項間の組合せは append 演算で行なう。$f_{\rm SY}$ は入力記号列の append 演算と準同型の関係にあるものとし、$f_{\rm MN}$ は表現の append 演算に対して準同型の関係にあるものとする。

例えば、言語 L_{pltx} に基づく表現法 $p_{\rm SYpltx}$ と解釈法 $p_{\rm MNpltx}$ では、ある存在する記号列に対して、表現法で真となる表現は f_{SYpltx} =「漢字や仮名の記号を用いて日本語の規則に従った記号列に、ローマ字の記号で、整形に関する規則を埋め込んだ構造」と説明され。解釈法の解釈値は (「その他」であり)、f_{MNpltx} =「platex に基づく整形構造の指示がある日本語の文章」となる。

この言語で、記号列が漢字や仮名と理解できない場合、表現法と解釈法は偽 (false) となる。

定義 1 (言語間の関係 (共通性)) 二つの言語が
$L_{\rm GS}$ = ($p_{\rm SYS}$: $f_{\rm SYS}$, $p_{\rm MNS}$: $f_{\rm MNS}$)
　と
$L_{\rm GB}$ = ($p_{\rm SYB}$: $f_{\rm SYB}$, $p_{\rm MNB}$: $f_{\rm MNB}$)
であると仮定する。

言語間に共通性があるとは、関数 commL($L_{\rm GS}$, $L_{\rm GB}$) が以下のいずれかの値を取ることとする。

X = commL($L_{\rm GS}$, $L_{\rm GB}$) $\Longleftrightarrow$
∀ (($p_{\rm SYS}$, $p_{\rm MNS}$): $L_{\rm GS}$, ($p_{\rm SYS}$, $p_{\rm MNS}$): $L_{\rm GS}$ •
　{ [I: 同等] ((Equal)　$L_{\rm GS} = L_{\rm GB} \Leftrightarrow$
$p_{\rm SYS} = p_{\rm SYB} \wedge p_{\rm MNS} = p_{\rm MNB}$)
　∨
((Subsumed)　$L_{\rm GS} \subset L_{\rm GB} \Leftrightarrow$
　(∀ $list_X$: {f_{SYS}} | $p_{\rm SYS}(list_X)$ == true
$\Rightarrow$ ($p_{\rm SYB}(list_X)$ == true $\wedge$ ($f_{\rm MNS}(list_X) \subset f_{\rm MNB}(list_X)$)))) $\Rightarrow X = L_{\rm GS}$ }
　∨
{[II: 解釈法が同一]　$p_{\rm MNS}$ == $p_{\rm MNB}$ •
　(∃ n_S, n_R : N • $n_S \times$ unitIntp($p_{\rm SYS}$) == $n_R \times$ unitIntp($p_{\rm SYB}$) $\wedge$ ($p_{\rm SYS} \subset p_{\rm SYB} \vee p_{\rm SYB} \subset p_{\rm SYS}$))
$\Rightarrow X = p_{\rm MNS} \wedge n_S \times$ unitIntp($p_{\rm SYS}$) }
　∨
{ [III: 表現法が同一]　$p_{\rm SYS}$ == $p_{\rm SYB}$ •
　(∃ n_S, n_R : N • $n_S \times$ unitIntp($p_{\rm SYS}$) == $n_R \times$ unitIntp($p_{\rm SYB}$))
$\Rightarrow X = p_{\rm SYS} \wedge n_S \times$ unitIntp($p_{\rm SYS}$) } [1] □

例えば、postscript 構造として理解する言語 L_{gs} で、三角形の図に見えるものは、表現は ps 形式、解釈値は「三つの座標とそれを結ぶ線分からなる」、となり、バイナリ構造を表す言語 L_{Bin} として理解すると、表現は二種類の異なる値の並びの列、解釈値は「数 1 と 0 の並びの列」、となる。$L_{Bin} \subset L_{gs}$ であるので、unitIntp($p_{\rm gs}$) = $n_B \times$ unitIntp($p_{\rm Bin}$) となる。

性質 2 (言語と計算オブジェクト) ある言語で理解された計算オブジェクトが別の言語でも理解される条件は、この二つの言語の間に共通性があることである。□

説明: ⇒ 二つ目の言語で理解できるということは、この言語で単位項があり、単位項を基とした対応づけがあることを表す。このことは解釈法が同一になるか表現法が同一になる場合であるので、言語に共通性であることを示す。

⇐ 言語間に共通性があれば、それぞれの単位項が決められる。この関係を利用して、計算オブジェクトから新たな表現と解釈値が作られる。□

定義 3 (計算オブジェクト) [Base-Case:] 最小の計算オブジェクトを現実の計算機上のバイナリ構造と対応させる。二進構造を前提とすると、二つの異なるものの識別のための表現 (△ と ▽ で表す) を 0 と 1 に対応づける言語 $L_{\rm Bin0}$ があれば、二つの計算オブジェクトは言語 $L_{\rm Bin0}$ で
　<△, 0> と <▽, 1>
として理解され、表現 △ と ▽ をそれぞれ 1 と 0 に対応づける言語 $L_{\rm Bin1}$ があれば、

[1] ここで、(subsumed) ($f_{\rm MNS}(list_X) \subset f_{\rm MNB}(list_X)$) の関係があるとは、それぞれの言語で理解される表現 $f_{\rm MNS}(list_X)$ と $f_{\rm MNB}(list_X)$ を表す概念間で refinement 関係があるとする。

<△, 1> と <▽, 0>
と理解される。

[Repetition-Step:](i)　言語 $L_{G0} = (p_{SY0}\colon f_{SY0}, p_{MN0}\colon f_{MN0})$ で理解された計算オブジェクト $co(0)$ の表現が記号列 $list_0$ であるとする。

これが言語 $L_H = (p_{SY_H}\colon f_{SY_H}, p_{MN_H}\colon f_{MN_H})$ で理解された場合、その言語 L_H による表現 syn_j は $syn_j = f_{SY_H}(list_X)$ であり、解釈値は $sem_p = f_{MN_H}(syn_j) = f_{MN_H}(f_{SY_H}(list_X))$ となる。このように理解された計算オブジェクトを

$$co(i) =< syn_j, sem_p : L_H > . \quad (3)$$

または

$$co(i) =< f_{SY_H}(list_X)), f_{MN_H}(f_{SY_H}(list_X)) > . \quad (4)$$

の形で表す。

言語 L_H が $co(i)$ を理解することを述語 $ustd$ で表す。

$$ustd(co(i) : L_H) \quad (5)$$

(ii)　言語 L_G で理解されない場合は、記号列 $list_X$ はそのまま残る。

(iii)　$co(i)$ に対して、別の言語で理解される場合は (i) を繰り返す。□

例えば、latex 言語の予約語と日本語の文章からなる記号列は、L_{pltx} で理解すると、表現は「tex の注釈付きの日本語の文章」、解釈値は「日本語の要素で表され、tex の注釈に従った整形構造を表す」、となる。

この計算オブジェクトは
co(i) = <(sequences-japanese-characters,
　　separated-by-english-words-and-symbols),
(formatted-sequences-japanese-characters-by-latex-annotation)>
で表される。

性質 4 (計算オブジェクト間の演算) 同一の言語で理解される計算オブジェクト間の演算は append により、add, subtract, substitute が表せる。□

説明: 言語の性質 (iv). □

処理を写像として扱う場合、1.2 節の (s3) より、その入力は計算オブジェクトとなる。入力としての役割の区別を、(i) その要素が欠けると処理が実行できない、必須 (mnd:mandatory)、(ii) その要素がが欠けても処理は実行されるが、結果はその要素が不足したものとなる、環境 (env:environmental)、(iii) 表現変換や解釈法の処理に関わる、任意 (ilv:involving) に分ける。

(mnd) にはシステムで用いる Gliba, xx.so 等の共用ライブラリや、リンカで引き出される特定のファイルがある。

(env) はコマンドの基幹ではなく、版を変える場合などの状況に応じて変わり得るもので、例えば、結果の計算オブジェクトの表現のための記号の種類、文字データベースへのポインタとしてのディレクトリ、configure の引数として示され、環境変数としても表されるもの等がある。

これらを入力として仮定し、不足や冗長があれば、変更する、という方針で臨む。

性質 2 より、計算オブジェクトは言語が異なると表現と解釈値が異なる。このことは、一つの計算オブジェクトを別の計算オブジェクトに変換していると見ることもできる。このことは特定の表現法を持つ計算オブジェクトを入力として、他の表現法で表される計算オブジェクトをもたらすことを表す. そこで、処理は (幾つかの計算オブジェクトを入力とした) 言語間の対応づけで表されるものとして以下のように定める。

定義 5 (処理の写像 (Gmap)) 処理の写像を入力側の (複数の) 計算オブジェクトを理解する (複数の) 言語から、結果側の (一つの) 計算オブジェクトを理解する (一つの) 言語への対応づけの記述とする。□

この場合、言語を考慮した処理では表現法と解釈法が重視される。処理に用いられる言語間では、その共通性を軸 (pivot) として変換しているように見える。

例えば、処理コマンド: C_{platex} が、入力として日本語 latex 形式の文書と eps 形式の図があるとき解釈法は言語 L_{pltx} の解釈法で固定し、表現法を注釈付日本語文章列から dvi 形式に変換している。実際、入力列の解釈値は「日本語文書と図を注釈としての指示で整形する」結果の解釈値は「日本語文書と図が整形された」であり、処理では同一の解釈法で異なる表現法に変換している。

同じ解釈値で表現法を変える他の例として、「一つであること」の表し方は、数式という言語では'1', 英語では'one' となり、異なる表現となる。

同様に、計算オブジェクトの解釈法を変える場合、それぞれの解釈法を持つ言語間で表現法を共通なものとした変換と捉えることができる。同じ表現法で解釈法を変える例として、nkf コマンドで代表されるコード間変換がある。

性質 6 (処理の写像が成り立つ条件) n 個の入力側の言語が

$L_{G_i} = (p_{SY_i}\colon f_{SY_i}, p_{MN_i}\colon f_{MN_i})$ $(1 \le i \le n)$
で表され、結果側の言語が
$L_R = (p_{SYR}\colon f_{SYR}, p_{MNR}\colon f_{MNR})$
であるとする。

処理の写像が言語間で成り立つ条件は、

$$\exists L_{\mathrm{G_j}} \bullet (commL(L_{\mathrm{G_1}}, .., L_{\mathrm{G_n}}) \cap L_{\mathrm{G_j}} \neq \emptyset) \\ \wedge \ (commL(L_{\mathrm{G_j}}, L_{\mathrm{R}}) \subset L_{\mathrm{R}}). \quad (6)$$

が成り立つことである。 □

説明: 入力側の言語の数 n(と計算オブジェクトの数) による帰納法 (省略)

誰が処理実行の行為者になるかの問題は、コマンド名と処理の実際との対応関係を参考として、計算オブジェクトの解釈法が処理の写像を表す場合の計算オブジェクトを処理の行為者として対応づける。この対応づけは計算オブジェクトの定義 3 とも合致する。

3 処理の形式化の提案

3.1 形式的処理記述

定義 7 (MapUnit) 処理の入力側の n 個の言語を
$L_{\mathrm{G_i}} = (p_{\mathrm{SY_i}}\colon f_{\mathrm{SY_i}}, p_{\mathrm{MN_i}}\colon f_{\mathrm{MN_i}})\ (1 \leq \mathrm{i} \leq \mathrm{n})$
結果の言語を
$L_{\mathrm{R}} = (p_{\mathrm{SYR}}\colon f_{\mathrm{SYR}}, p_{\mathrm{MNR}}\colon f_{\mathrm{MNR}})$
であるとする。

処理記述の枠組を表す MapUnit を

(i) 生成関数: < 入力の種類の判定 ($DesideIn$) と処理の写像 (G-map)>: 入力の役割により、処理が成立するか否かを判断し、成立しない場合は結果を示さず、成立すれば、入力を役割により、結果の表現の記法に加えるか、定義 5 の処理の写像で扱うか、に分類し、入力の (複数の) 計算オブジェクトと結果の計算オブジェクトを対応づける。

(ii) 生成関数で変換され、結果側の言語 L_{R} の表現法と解釈法で表された結果の計算オブジェクト (muP) の集合

(iii) 入力の計算オブジェクト (他の Mapunit の要素 ((muP))): 処理の写像 (G-map) の引数としての役割 (必須 mnd(∗), 環境 env(∗)(直接結果側に渡される表現の記法は結果側の言語の引数となる), 任意 ilv(∗) かの区別) と、注目要素の種類 (表現 syn(∗) または解釈値 sem(∗)) と、これらの演算の関係 (入力要素を加える _add_, 後者の要素を前者の要素から除く _sub_, 後者の要素を前者の要素の対応部分で入れ替える _stt_) (図 1). □

例えば、処理コマンド C_{platex} では日本語 latex 形式の文書と理解する言語 L_{pltx} と eps 形式の図を理解する言語 L_{Eps} を入力とし、結果側では dvi 形式を理解する言語 L_{DVI} を持つ。

muP はこの MapUnit と入力の計算オブジェクトで規定されるので

$$\exists L_{\mathrm{G_1}}, , ., L_{\mathrm{G_n}}, L_{\mathrm{R}} \mid Gmap(L_{\mathrm{G_1}}, , ., L_{\mathrm{G_n}}) = L_{\mathrm{R}} \ \bullet \\ \exists ustd(co_{MD(1)}, .., co_{MD(k1)} : L_{\mathrm{G_{MD}}}), \\ ustd(co_{EI(1)}, .., co_{EI(k2)} : L_{\mathrm{G_{Ei}}}), \\ ustd(co_{ER(1)}, .., co_{ER(k3)} : L_{\mathrm{G_{ER}}}), \\ ustd(co_{IN_{aj}} : L_{\mathrm{G_j}}), , .., ustd(co_{IN_{an}} : L_{\mathrm{G_n}}) \mid \\ DecideIn(role(mnd(co_{MD(1)}, .., co_{MD(k1)}), \\ env((co_{EI(1)}, .., co_{EI(k2)}), (co_{ER(1)}, .., co_{ER(k3)})), \\ ilv(co_{IN_{aj}}, .., co_{IN_{an}}))) \ \bullet muP : MapUnit \\ \Longleftrightarrow \quad (7) \\ f_{SYR}(muP) = (ope(\{add \mid sub \mid stt\}, \{syn \mid sem\}, \\ (co_{IN_{aj}}, .., co_{IN_{ar}})), (co_{ER(1)}, .., co_{ER(k3)})) \\ \wedge \\ muP =< f_{SYR}(muP), f_{MNR}(f_{SYR}(muP)) : L_{\mathrm{R}} > .$$

muP の表現はそれが属する MapUnit への入力で決まる、解釈値は MapUnit の結果側の言語の解釈で決まる。ここでは別の MapUnit を表すか、状態かのいずれかになる。

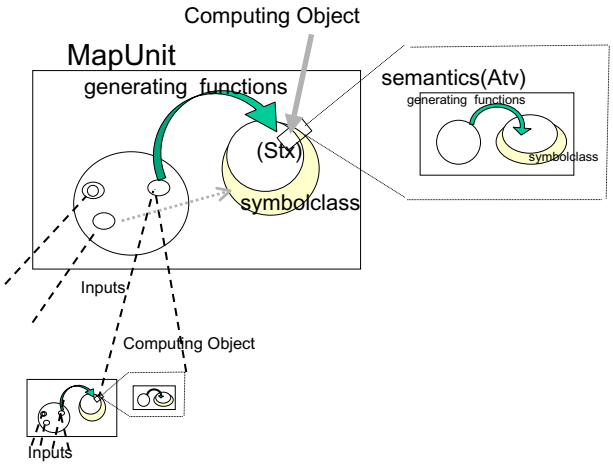

図 1: 写像としての処理

Algorithm 8 (処理の写像のアルゴリズム) [I] $n = 1$ の場合、
[I-e] $L_{\mathrm{G_1}} = L_{\mathrm{R}}$ または $L_{\mathrm{G_1}} \subset L_{\mathrm{R}}$ の場合、写像は同一表現法、同一解釈法の下での対応づけであり、計算オブジェクトの組合せは append 演算で add, subtract, substitute がある。
[I-(crim)] 定義 1 の言語間の関係で、[I: 同等] または、

[II: 解釈法が同一] の場合、変換は、(i) 共通の解釈法を (軸 (pivot) として) 用いて、入力計算オブジェクトの表現を解釈し、結果側の表現の単位項と対応させ、これを結果側に用いる記号の種類の要素で表す、(ii) 入力計算オブジェクトの表現列をを append 演算でリストとして表し (iii) これを append 演算の準同型性を利用して結果側の記号で表された単位項のリストとして結果の表現とする。(iv) この表現から f_{MNR} で解釈値を得る (図 2)。

[I-(ircm)] [I: 同等] または [III: 表現法が同一] の場合、入力のそれぞれの計算オブジェクトの表現の単位項と同じ計算オブジェクトを結果側の解釈法で理解した単位項の (n 倍または $1/n$ 倍とした) 間で対応づけ、結果側の単位項を append 演算でリストとして表すことで得られる (図 3)。

[II] $1 \leq n$ の場合、性質 6 より、入力側の言語 $L_{\mathrm{G_i}}$(1 ≤ i ≤ n) では、少なくともその一つ $L_{\mathrm{G_j}}$ に対して、

(1) commL($L_{\mathrm{G_j}}$, L_{R}) = 解釈法 ($p_{\mathrm{MNG_j}}$) の場合、式 (6) の関係から、入力側の言語の組は $L_{\mathrm{G_j}}$ の解釈法 ($p_{\mathrm{MNG_j}}$) と等しいか、$L_{\mathrm{G_j}}$ の表現法 $p_{\mathrm{SYG_j}}$ と等しいかのどちらかになる。

[II-(im-im)] 入力側の言語の解釈法がすべて等しいか、($p_{\mathrm{MNG_j}}$) に含まれる場合は、各言語の表現法による理解を $p_{\mathrm{SYG_j}}$ に統一するため、[I-(crim)] により、前処理として解釈法を言語 $L_{\mathrm{G_j}}$ の $p_{\mathrm{MNG_j}}$ で固定して、各 i について、表現法を $p_{\mathrm{SYG_i}}$ から、表現法 $p_{\mathrm{SYG_j}}$ に変える。続いて、[I-e] より、一つの入力計算オブジェクトとして [I-(crim)] を用いて L_{R} で理解する計算オブジェクトを得る。

[II-(ir-im)] 入力側の言語の組の表現法がすべて等しいか、$p_{\mathrm{SYG_j}}$ に含まれる場合、入力側の言語ではその表現法を固定して、解釈法を $p_{\mathrm{MNG_j}}$ に変換し、append により一つの入力計算オブジェクトにしてから、$L_{\mathrm{G_j}}$ と L_{R} の共通解釈法 $p_{\mathrm{MNG_j}}$ を軸として表現変換する (図 4)。

(2) commL($L_{\mathrm{G_j}}$, L_{R}) = 表現法 ($p_{\mathrm{SYG_j}}$) の場合、

[II-(im-ir)] 入力側の言語の解釈法がすべて等しいか、$p_{\mathrm{MNG_j}}$ に含まれる場合は、入力側の計算オブジェクトを $L_{\mathrm{G_j}}$ で理解できる表現にして append で一つの計算オブジェクトとし、これについて [I-(ircm)] により結果の計算オブジェクトを得る (図 5)。

[II-(ir-ir)] 入力側の言語の組の表現法がすべて等しいか、$p_{\mathrm{SYG_j}}$ に含まれる場合、入力側の言語ではその表現法を固定して、解釈法を $p_{\mathrm{MNG_j}}$ に変換し、append により一つの入力計算オブジェクトにしてから、この表現法を基として解釈法を L_{R} で理解する計算オブジェクトにする。□

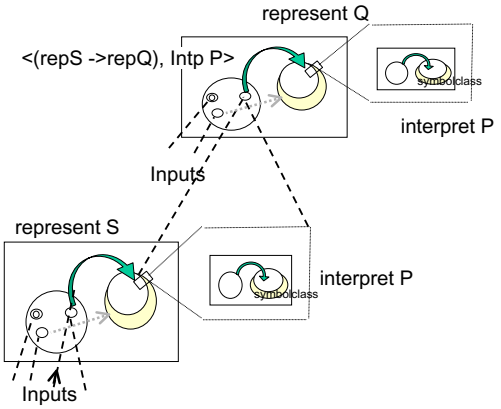

図 2: 解釈法を同一とした、一入力の表現間変換

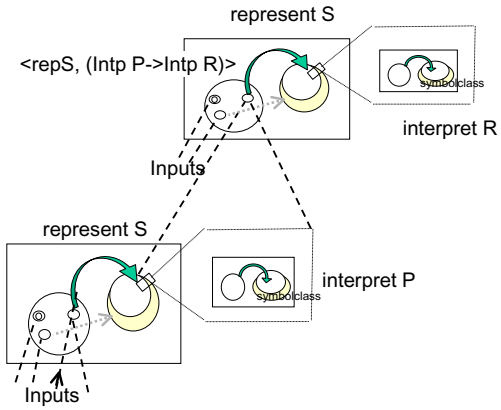

図 3: 表現法を同一とした、一入力の解釈法変換

3.2 計算オブジェクトによる個別処理

[基本的な場合:] 処理の行為者となる計算オブジェクト co_{EX} とこの処理の入力となる計算オブジェクト co_{I1},.., co_{In} が与えられたとする。

co_{EX} の解釈値の MapUnit に基づき、処理が進む。入力の計算オブジェクト co_{I1},.., co_{In} をこの MapUnit の $DecideIn$ で比較する。

[I] 必須の入力 (mnd) が無い場合は処理は実行されない。

[II] 環境の入力 (env) や任意の入力 (ilv) があり入力側の言語で理解できる場合、アルゴリズム 8 に従う。

例えば、計算オブジェクト co_{EXA} が C-ソースプログラムと理解する言語 L_{Csrc} と環境変数と理解する言語 L_{ENV} を入力側の言語とし、実行可能と理解する L_{Exec} を結果側の言語として、

$$\mathrm{Gmap}(L_{\mathrm{Csrc}}, L_{\mathrm{ENV}}) = L_{Exec}$$

を持つ MapUnit の要素 (=muP) とする。

$$\mathrm{ustd}(co_{EXA} : L_{Exec})$$

であり、その表現を L_{Exec} で理解すると解釈値が

$$\mathrm{Gmap}(L_{pltx}, L_{Eps}, L_{Glib}) = L_{DVI}$$

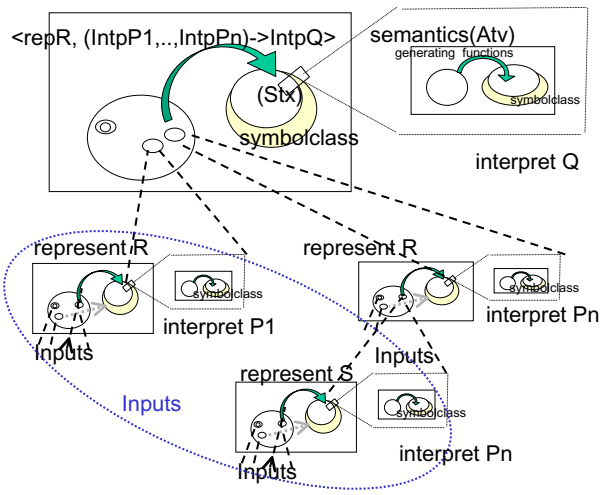

図 4: 前処理で $n(=2)$ 表現法を同一にした解釈法変換

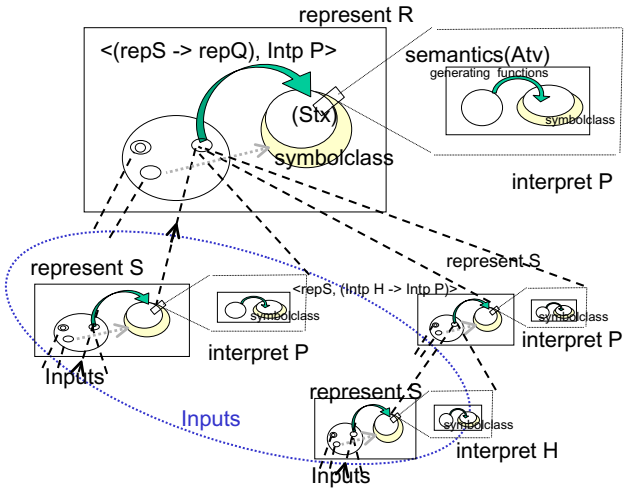

図 5: 前処理で入口側の $n(=2)$ 種類の解釈法を同一にした表現間変換

を持つ MapUnit となり処理コマンド C_{platex} として機能するものとする。

計算オブジェクト co_{EXA} を処理の行為者として、日本語 latex 形式の文書と eps 形式の図を入力の計算オブジェクトとすると、C_{platex} は dvi 形式の計算オブジェクトを出力する処理を行なう。

[処理実行可能な計算オブジェクト co_{EX} 自身が他の計算オブジェクト co_{OTEX} から処理を受ける場合] 処理の行為者となる計算オブジェクト co_{OTEX} から見れば計算オブジェクト $co_{EX}(=<syn_{EX}, sem_{EX}>)$ が処理の行為者になるか否かに拘らず、co_{OTEX} の解釈値の MapUnit に基づき [基本的な場合] として、co_{EX} を入力の計算オブジェクトとして扱える。この場合、co_{EX} の表現 syn_{EX} を変えることも co_{EX} の解釈値 sem_{EX} を変えることもできる。このことは、処理自体の更新が処理により行なえることを示す。

4 おわりに

多様な形態の処理に共通な性質を見出すため、処理を入力と結果の関係と捉え、計算オブジェクトを導入し、これらの関係を表現法と解釈法を表す言語を用いて記述することで、処理が成立する条件を導くことができた。この共通性は処理では変換時の軸 (pivot) となることを示している。言語間の表現法が軸ならば、処理では解釈法を変え、解釈法が軸ならば表現法を変える。

この関係を利用すると、どのような場合に処理が可能かが示され、また、複数の異なる言語を入力とした処理が説明できる。

処理を入力、写像、結果のオブジェクトの関係として単位 (MapUnit) 化して表すことで、計算オブジェクトはそれ自身の表現が MapUnit に入力を作用させた結果として、解釈値が MapUnit そのものとして、表すことができたので、処理実行は、自身が処理の行為者になるか、ならないかの区別なく説明できた。また、入力にシステム情報を加えたことで、処理に関する扱いの幅が広がると考えられるが、ここでは、扱わなかった。

今後の課題は、MapUnit に基づく処理間の関係づけ、言語に基づく計算オブジェクト間の関係づけ等、の明確化や精密化、また、提案手法を具体化し、検証実験をすることなどである。

謝辞

この研究の基本枠組は近山隆東京大学名誉教授のとの議論を基にしました。

参考文献

[1] Wand, Y. and Weber, R., An ontological evaluation of systems analysis and design methods, pp.79 – 107, Information Systems Concepts: An Indepth analysis, IFIP, 1989.

[2] Yair Wand and Ron Weber, An Ontological Model of an Information System, pp. 1282 –1292, IEEE Transactions on Software Engineering, Vol.16, No.11, November, 1990.

[3] J.M. SPIVEY, UNDERSTNAGING Z, Cambridge University Press, p. 131, 1988.

[4] Diller, A. Z An Introduction to FormalMethods, 2nd Eds. p.374, 1994.