数理解析研究所講究録
第 470 巻 1982 年 114-124

Message Passing の理論

米澤明憲(東工大・理)

序

ACTOR理論やSMALLTALK言語の基礎となる，"メッセージのやりとり"（以下，message passingと略す）に基づく計算モデルを，λ一算法（λ-calculus）に類似する形式的体系を用いて理論的に展開しようとする研究が，MITのS.A. Ward, R.H. Halstead Jr. 等によって試られている。本報告は，彼等の研究を文献[1]に沿って概略するものである。

message passingに基づく計算モデルの重要性は，次の2点に要約される。

(1) プログラム言語の意味モデルとして従来用いられてきた作用型計算モデル，すなわち関数とその適用に基づくモデルと比較して，より基本的で一般性が高い。

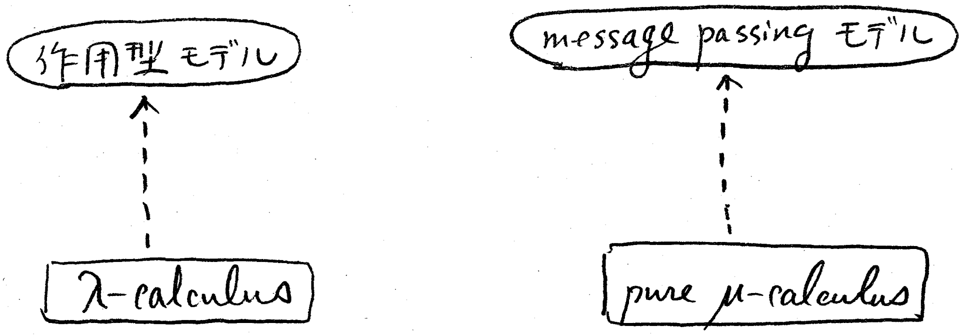
(2) 並列計算，分散処理系に対する極めて自然なモデルを提供する。

作用型計算モデルでは，関数（演算子）のその引数（被演算子）への適用（application）操作 および 適用結果（すなわち式の評価結果）としての値の返却操作（return）という2種類の操作が，常に"対"として現われる。これに対して，message passing に基づく計算モデルでは，メッセージの伝送という唯一の操作に基礎を置くため，作用型モデルのような『二項性』あるいは双方向性がなく，より単純であると同時に，関数の適用操作および 値の返却操作 のどちらも メッセージ伝送として表現できるので，より一般性があると言える。（ごく乱っぽい言い方をすれば，関数の適用は，引数を関数へのメッセージとして伝送すること，値の返却は，関数の合成における，より外側の関数へより内側の関数の適用結果をメッセージとして伝送することと見做すことができる。）

message passing はまた，メッセージ伝送の標的（target）となるモジュールの能動化（activation, invocation）という制御（control）の側面と，その能動化に対するパラメタの伝送という通信（communication）の側面をもち，制御の流れ（control flow）とデータの流れ（data flow）の統一と観ることもできる。一方，作用型モデルは，その本質的な二項性のため，制御，通信の両側面とも二項性に伴う強い制限を余儀なくされる。（しかし，この二項性は，

従来のプログラム言語において，ブロック構造やスタック方式を採用することによって，名前の有効範囲や記憶領域管理に対して効率よい手法を可能にしているが

作用型モデルの基礎となる形式体系はλ-算法であるが，以下に紹介する純μ-算法（pure μ-calculus）は，大雑把に言うとλ-算法に継続（continuation）および事象（event）という概念を系統的に導入することによって得られたもので，message passing に基づく副作用（side effects）のない計算のモデルのための形式体系である．



また本報告の終りの部分で触れるμ-算法（μ-calculus）は，純μ-算法に『conduit』（水路）という並列に進行する計算の間の通信を表現するための構成子を付加した形式的体系で，この体系によって作用型モデルのもつ決定性と簡潔さを保持しながら，多重処理の基礎づけを行うことができる。

純μ-算法

• この体系に現われる領域は，事象(event)，対象(object)および事象間の"因果"関係(causality)である。

[定義1] 対象とは，

(i) あらかじめ決められた定数の集合 $\mathbb{C}$（たとえば自然数）の要素か，

(ii) 変数の集合 $\mathbb{V}$ の要素（英小文字で表記する）か，

(iii) 原始演算（たとえば加算，比較）の有限集合 $\mathbb{P}$ の要素あるいは，

(iv) $(\mu x_1 x_2 \cdots x_n . E_1 E_2 \cdots E_m)$ の形をした μ-式（但し各 x_i は変数，各 E_j は事象）のどれかである。

[定義2]

事象とは対象の列 $(A_1 A_2 \cdots A_n)$ である。（但し $n > 1$）

• 事象 E が直接事象 E' を引き起すならば $E \rightarrow E'$ と表わすことにする。$\rightarrow$ は半順序であることに注意。

[公理1]（μ-簡約）

事象 E が $((\mu x_1 x_2 \cdots x_n . E_1 E_2 \cdots E_m) A_1 A_2 \cdots A_n)$ のとき，全ての E_j' に対して $E \rightarrow E_j'$ が成立する。（但し，E_j' は E_j の中の $x_1, \cdots, x_n$ の自由な出現に対して $A_1, \cdots, A_n$ をそれぞれ代入して得られた形 $S[A_1, A_2, \cdots; A_n; x_1, x_2, \cdots, x_n : E_j]$。）

[公理2]（原始演算）

事象Eが（$p c_1 c_2 \cdots c_n A$）のとき，E →（$A \hat{p}[c_1; \cdots c_n]$）が成立する．但し，$p \in \mathbb{P}$，$c_i \in \mathbb{C}$，$\hat{p}$は$p$が表わす$n$変数の関数である．

純μ-算法の事象（すなわち対象の列）$A_1 A_2 \cdots A_n$は，$A_2 \cdots A_n$という対象の$n-1$個の列の，対象A_1に対する伝送と解釈される．（ $A_1 \Leftarrow [A_2 \cdots A_n]^{伝送}$ ）対象はデータを，定数は数やその他のデータを表わすと解釈される．公理1は，μ-式（$\mu x_1 x_2 \cdots x_n . E_1 E_2 \cdots E_m$）がメッセージ$A_1 A_2 \cdots A_n$を受け取ると，$m$個の新しい事象が引き起される．たとえば

（（$\mu x c. + x x c$）3 R）は　（+ 3 3 R）を引き起す

（（$\mu c.$（+ 3 3 c）（c 4））R）は　（+ 3 3 R）と（R 4）とを互に並列に引き起す．

（（$\mu x c.$）7 R）は何も事象を引き起さない

公理2はあらかじめ指定された記号を特定の関数として解釈することをいっている．たとえば，σを後者関数，+を加算，>を通常の大小関係と考えれば

（σ S C）→（C S+1）

（+ S T C）→（C S+T）

$$(> S\,T\,C) \longrightarrow (C\ \mu abc.ca) \quad \text{if } S > T$$

$$(> S\,T\,C) \longrightarrow (C\ \mu abc.cb) \quad \text{if } S \leq T$$

この公理で重要なもう1つの点は，原始関数 p の適用結果 $p[c_1; c_2; \cdots c_n]$ が必ず A に送られることである．すなわち A は $(p\,c_1\,c_2 \cdots c_n\,A)$ の事象に対する継続である．(上の例では C が継続を表わしている。) 上の例の S, T を具体的な自然数をあてはめると，その結果は継続 C に伝送されることになる。

$$(\sigma\,8\,R) \longrightarrow (R\,9)$$

$$(+\,3\,4\,R) \longrightarrow (R\,7)$$

$$(>\,6\,7\,R) \longrightarrow (R(\mu abc.cb))$$

$$(>\,5\,2(\mu x.x\,10\,R)) \longrightarrow ((\mu x.x\,10\,R)(\mu abc.ca))$$

message passing による計算モデルでは引き起こされる事象によって意味が記述されるところが，従来の作用型モデルにおける値による意味記述と大きく異なる点である。このため作用型モデルでは $p(f(3))$ と表わされる式は，事象 $(f\ 3\ p)$ という形になる．この場合，f は3に対する値を計算して，継続である p にその値を送ることになる．作用型モデルでは，$f(3)$ という値は $f(3)$ が現われる文脈に非明示的に指定されるのに対して，message passing による

計算モデルでは，継続 p が f に送られるメッセージの中に含まれていることになる。文献[1]には，λ-算法に基づく作用式(applicative expression)を純μ-算のμ-式に翻訳するための規則が与えられ，それらの規則が標準的な解釈のもとで正しい，漏れがないことが証明されている。

μ-算法

純μ-算法は，公理1によって，1つの事象から複数の事象が並列的に引き起されることを許すが，逆に，幾つかの独立した事象によって1つの事象が引き起されるような現象を許していない。このような現象は，少し複雑な並列計算を行う場合，並列に走るプロセス間の通信，同期という形で必ず現われるものである。そこで純μ-算法にconduit(水路)という概念を導入拡張して，より一般的なmessage passingのモデルを，形式的体系μ-算法として構成する。

conduitはUNIXのpipeの概念を近いもので次のように定義される。

[定義3] conduitとは次のような集合の要素をいう。

$$\mathbb{K}=\{(\mathbb{r}_X,\mathbb{W}_X)\mid X\text{ は }\mu\text{-算法の対象}\}$$

ここで $\mathbb{R}_x$, $\mathbb{W}_x$ は新しいクラスの対象で，任意の conduit $\langle \mathbb{R}_x, \mathbb{W}_x \rangle$ に対して，$\mathbb{R}_x$ をこの conduit の読み側，$\mathbb{W}_x$ を書き側と呼ぶ．このような conduit を特徴づける公理を次に述べる．このために新しい因果関係 $\Rightarrow$ の導入が必要となる．既に導入した因果関係 $\rightarrow$ は "直接的" な因果関係であるのに対して $\Rightarrow$ は "遠隔的" な因果関係である．

[公理3] (conduit)

(1) $E \rightarrow E'$ ならば $E \Rightarrow E'$

(2) $E \Rightarrow E'$ および $E' \rightarrow E''$ ならば $E \Rightarrow E''$

(3) $E \Rightarrow (\mathbb{R}_x A)$ および $E \Rightarrow (\mathbb{W}_x B)$ 但し $\langle \mathbb{R}_x, \mathbb{W}_x \rangle \in \mathbb{K}$ ならば，$E \Rightarrow (A B)$

(4) $E \Rightarrow E'$ は (1)〜(3) の規則の有限回の適用だけから得られるものである．

[公理4] (conduit の創生)

創生作用素 ξ は次のような性質をもつ．

$$\xi C \longrightarrow (C \mathbb{R}_c \mathbb{W}_c)$$ 但し C は μ-算法の対象．

C は作用素の働きによって新しく作り出される conduit を受け取る継続の役割をもつと同時に，その読み側と

書き側の対応(対)関係を指定する名前の役をもはたす。

公理3と4の働きを例を使ってみてみよう。新しいconduitの受け取り手として次のような対象Xを考える。

$$
\begin{aligned}
X \equiv (\mu r w.\ &(r(\mu x.+xxR))\\
&(rR)\\
&(w3)\\
&(w4)\\
&(w5))
\end{aligned}
$$

事象($\equiv X$)は公理4によって($X\ \mathbf{r}_x\ \mathbf{W}_x$)を引き起す。この事象は$\mu$-簡約の公理1によって次の5つの事象を並列的に引き起すことになる。

$$
\left.
\begin{array}{l}
(\mathbf{r}_x\ \mu x.+xxR)\\
(\mathbf{r}_x\ R)\\
(\mathbf{W}_x\ 3)\\
(\mathbf{W}_x\ 4)\\
(\mathbf{W}_x\ 5)
\end{array}
\right\}
\begin{array}{l}
X\text{中の}r\text{に}\mathbf{r}_x\text{を}\\
w\text{に}\mathbf{W}_x\text{を}\\
\text{代入.}
\end{array}
$$

さらに公理3の(3)によって、"そのうちに"次の6つの事象が引き起される。(これらの事象がどのような順序で起るかは規定されない。)

$$
\begin{array}{ll}
((\mu x.+xxR)3) & (R3)\\
((\mu x.+xxR)4) & (R4)\\
((\mu x.+xxR)5) & (R5)
\end{array}
$$

すなわち、書き側に送られた対象(3, 4, 5)は、"そのうちに"必ず、読み側に送られた対象($\mu x. + x x R$, R)に送られることになる。

以下、詳説はしないが、このようなconduitを用いることにより、

(i) 作用式の引数の並列評価

(ii) μ-算法の対象の並列評価

(iii) リスト構造の表現

(iv) 自己参照をもつデータ構造

(v) 再帰表現、

などが、統一的な枠組の中できっちりと表現できることになる。これらの点の詳細は文献[1]を参照されたい。conduitは"cell"という記憶、状態に対応する概念を導入せずに、上の(i)〜(v)などを表現可能にする構成子で、作用的構造と命令的構造(imperative structure)の間に位置する中間的な興味深い算法構造をつくり出すものであることは、注目に値する。

λ-算法、ACTOR理論の初等的なもの、および、作用的構造と命令的構造に関する議論については、文献[2]を参照されたい。

文献

[1] Ward, S.A., Halstead, R.H., "A Syntactic Theory of Message Passing", JACM. Vol. 27 No. 2. (1980).

[2] 木村泉, 米澤明憲, 「算法表現論」岩波講座 情報科学 12. (1982).