北海道大学工学部研究報告
第151号（平成2年）

Bulletin of the Faculty of Engineering,
Hokkaido University. No. 151 (1990)

# メッセージ到着順序制御機構を導入した並列オブジェクト指向モデル

渡辺　慎哉　　宮本　衛市*

（平成2年4月6日受理）

# An Object-Based Concurrent Model with Message-Order Control Mechanism

Shin-ya WATANABE and Eiichi MIYAMOTO

(Received April 6, 1990)

**Abstract**

In distributed computation, agents (objects) act cooperatively exchanging information with each other. In this case, there is some temporal delay of the information propagation because objects are widely distributed. The delay time is not constant but varies according to situations. Thus, when we construct applications based on distributed computation, we must add instructions for synchronization in the programs. In this paper, we propose an object-based concurrent model which controls arrival orders of messages according to the intention of sender object, and also discuss a mechanism for implementing the model. Using our model makes it possible to represent order of message arrival declaratively.

Thus, it enables us to construct high-level debuggers and verification systems for distributed programs.

## 1. はじめに

ここでは，本研究の動機として，現在分散計算において問題となっている同期操作の必要性について述べ，また，それが並列オブジェクト指向モデルにおいても重要な問題となっていることを述べる。

### 1.1 分散計算における同期操作の必要性

分散計算においては，それぞれの計算実体（オブジェクト）が互いに情報を交換し合いながら協調的に処理を進めて行く。このとき，情報の伝達には，オブジェクトが分散されていることによる時間的遅延が生じる。この時間的遅延は一般に一定ではなく，状況に応じて様々に変化する。従って，分散計算に基づいてアプリケーションを構築する場合には，時間的遅延を考慮し，同期のための命令をプログラム中に埋め込む必要がある。例えば，図1において *object A* から *object B* に対して2つのメッセージ $m_1$, $m_2$ を送る場合を考えてみると，$m_1$ と $m_2$ の送信の間にいかな

* 北海道大学工学部　情報工学科

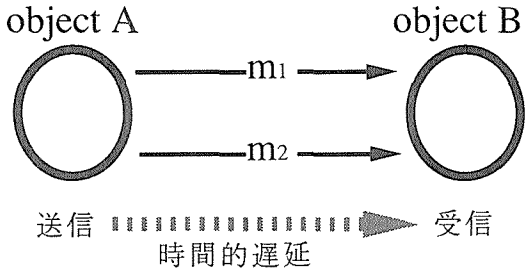


図1 メッセージ送信の時間的遅延

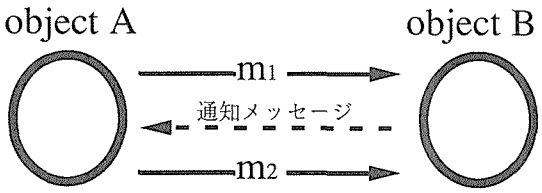


図2 順序づけのための同期操作

る時間的隔たりが存在していても，*object B* への到着順序は非決定的となる。従って，$m_1$が $m_2$ より先行して *object B* へ到着することを保証するためには，図2のように $m_1$ が到着した後に，$m_2$ の到着を通知するメッセージを *object A* に送信し，その通知メッセージを受理した後に $m_2$ を送信しなければならない。これは，プログラムの可読性を妨げ，また，大規模な分散アプリケーションの構築には大きな障害となっている。さらに，通知メッセージの到着を待つ（すなわち，同期をとる）ことは並列性の低下を招く要因ともなる。

### 1.2 並列オブジェクト指向モデルにおける同期操作

並列オブジェクト指向モデル[4]は，メッセージ通信をオブジェクト間相互作用に用いているため，分散計算を表現するのに適している。その大半はアクターモデル[2]に基礎を置いている。すなわち，オブジェクトが並列化の単位であり，それぞれのオブジェクトがメッセージを送って他のオブジェクトを活性化することにより計算が進行する。

待ちのない非同期メッセージ通信のみを備えたモデルを考えた場合，メッセージの到着順序を保証する必要が生じた時には，前節で述べたような同期操作を強いられる。特に，並列オブジェクト指向モデルの場合には，メッセージの受け口がメソッドとなっているため，前節で述べた到着通知メッセージを受けるためのメソッドを用意する必要がある。このように一連の処理を複数のメソッドに分割するのは，問題を表現する上で好ましいこととは言えない。そこで *ABCM/1*[3]など現在までに提案されているモデルでは，同期的メッセージ通信を備えたものが多数存在する。この同期的メッセージ通信を用いることにより，メッセージ送信の時間的順序が暗黙的に受信順序を決定するので，順序制御は可能となるが，同期待ちによる処理効率の低下は免れないことになる。

並列オブジェクト指向モデルは，従来のオブジェクト指向モデルをそのまま並列化したものであるため，オブジェクト間相互作用の手段としてもメッセージをそのまま用いている。しかし，従来から使用されているメッセージ通信は，上に挙げたような理由により比較的低レベルな通信手段といえる。本研究では，メッセージを送信側オブジェクトの意図通りの順序で受信側オブジェクトに到達するものとして捉えることにより，並列オブジェクト指向モデルにおける相互作用の抽象化を行なった。これによって，従来，メッセージの到着順序を保証するために余分なメッセージの授受を行なっていたものを，メッセージの受信順序の宣言に置き換えることが可能となった。この手続から宣言への移行は，プログラムの可読性の向上に寄与し，また，並行プログラムに対する高水準デバッギングや正当性の検証を容易にする。

以下，2節では，本研究で提案する並列計算モデルについて述べる。また，3節では，到着順序を保証するためのメカニズムに関して述べる。そこで述べられているメカニズムは，同期のた

めの余分なメッセージを必要としないため，同期待ちによる並列性の低下を抑制することができる。4節では，簡単な例題を用いて，従来のモデルとの比較を行なう。

## 2．メッセージ到着順序を保存する並列計算モデル

ここでは，本研究で提案する並列計算モデル[6]について解説を行なう。まず最初に，最も単純な場合として，あるオブジェクトから他のオブジェクトへ直接メッセージを送信する場合の通信モデル（direct preserving モデル）に関して議論する。次に，それを一般化した場合として，他のオブジェクト群を介した場合の通信モデル（indirect preserving モデル）に関して議論する。実際には，direct preserving モデルは indirect preserving モデルに完全に包含される。

### 2．1　direct preserving モデル

この節で議論する並列計算モデルは，次のような性質を持つ。

**あるオブジェクト $s$ から別のオブジェクト $d$ に2つ以上のメッセージ $m_1, \ldots, m_n$ を送信した場合，これらのメッセージは $s$ が何らかの方法で指定した順序関係を保ったままで $d$ に到着することが保証される。**

このモデルは，1節で述べた分散計算におけるメッセージ到着順序の非決定性の問題を解決するためのもので，本研究で行なう議論の最も基本的な部分に相当する。この direct preserving モデルを並列オブジェクト指向モデルにおけるオブジェクト間相互作用に用いることにより，メッセージの到着順序を制御するための余分なメッセージのやりとりをプログラム中に埋め込むことが不要となり，プログラマの負担を軽減することができる。

メッセージの到着順制御を行なうモデルには，上で挙げたもの以外に，メッセージを受ける側が，自分の中にあるメソッド群の実行順序に対する制約に応じて順序を自分自身で制御するものも考えられる。この考え方は，Campbell らによる path expression モデル[1]に相当するものである。この path expression モデルでは，共有資源へのアクセスの順序が共有資源の中で宣言的に与えられ，他のプロセス群からのアクセスのための手続き呼出しは，その順序にしたがって処理される。しかし，この順序は，共有資源の性質でのみ決定され，送信側オブジェクトの思惑を反映できないために，定型的な処理（例えばデータベースオブジェクトにおける read, write の順序制御等）にしか適用できない。

### 2．2　indirect preserving モデル

ここでは，direct preserving モデルを拡張したモデルを説明する。このモデルでは，基本的には，順序を指定する送信側オブジェクトとその指定された順序通りにメッセージが到着すべき受信側オブジェクトを対象とし，さらに，その間に複雑に絡み合ったオブジェクト群が介在することが前提となる。順序を指定するオブジェクトからこの系全体を見ると図3のようになる。この図において，オブジェクト $s$ が関心があるのは，自分が発したメッセージ群が原因となって起こる他のオブジェクト群のオブジェクト $d$ へのメッセージ群の到着順序である。

このモデルの性質は，以下の通りである。

**あるオブジェクト $s$ から他のオブジェクト $o_1, \ldots o_n$ にそれぞれメッセージ $m_1, \ldots m_n$ を送信した場合，$o_1, \ldots o_n$ が活性化され，さらに他のオブジェクト群へメッセージが送られる。ここで，メッセージ $m$ が原因となって発生する，全てのメッセージの集合を $Res(m)$ とし，その中で $d$ に到着するメッセージの集合を $Res_d(m)$ と定義す**

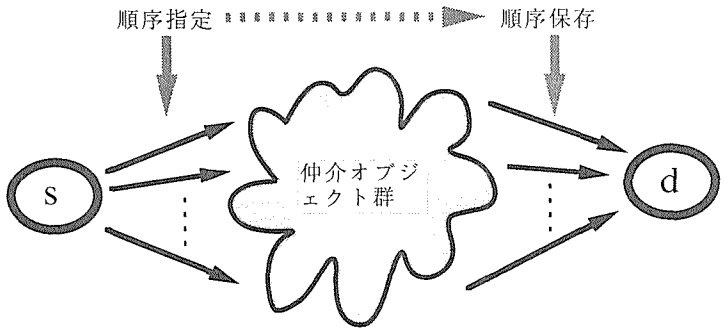


図3 indirect preserving モデル

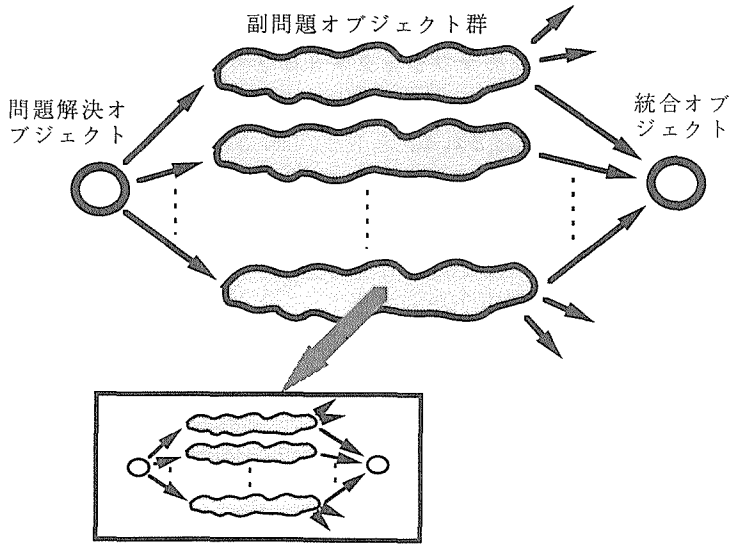


図4 indirect preserving モデルによる問題の抽象化

**る。このとき，$s$ が $d$ に対して $m_1, \ldots . m_n$ の到着順序を指定するならば，$d$ において，その順序に相当した順序で，$Res_d(m_1), \ldots ., Res_d(m_n)$ が到着することが保証される。**

このモデルは，分散協調問題を表現する上での抽象化の手段を提供する。問題解決を副問題への分割とその部分的統合という立場で捉えた場合，図4に示すように，問題解決オブジェクトは，自分の直属の副問題オブジェクト群と統合オブジェクト群のみを考慮に入れた抽象化のレベルで問題を表現する。従って，統合オブジェクト群へのメッセージの到着順序の指定も，このレベルで行なうことができる。また，それぞれの副問題オブジェクトは，また，自分の下に直属の副問題オブジェクト群と統合オブジェクト群を持っており，そのレベルで到着順序を指定すると考えることができる。

## 2．3 Kamui88 との融合

前節で提案したモデルを具体化する場合，次の2つの実現方法が考えられる。

1．各オブジェクトが送信側オブジェクトの意図を理解し，その意図に沿って，受信したメッセージの選択的実行を行なう。

2．オブジェクト系の外側に，送信側オブジェクトの意図に沿った順序で受信側オブジェクトにメッセージを到着させるような機構を設ける。

このどちらの方法も，メッセージの抽象化という観点から見て相違はない。しかし，実際の分散環境での使用を考えた場合，1の方法は，計算モデルに関して閉じた系，すなわち，系全体がこの計算モデルに基づくオブジェクトによって構成されていなくてはならない。それに対して，2の方法では，将来重要となると考えられる，非一様なオブジェクト群で構成された分散系(様々な計算モデルに基づくオブジェクトが入り交じった系。何らかの方法で，通信プロトコルは統一されなければならない）においても適用可能である。この観点から，我々は，2の方法を具体化の手段として用いることにした。

kamui88[5]は，我々が以前提案した並列オブジェクト指向計算モデルであり，場を用いたオブジェクト群のグループ化と，そのグループへのメッセージのブロードキャストを行なうことに特徴がある。また，場を通してメッセージを伝えることによって，オブジェクト同士の結合力が従来のモデルに比べて疎になったため，オブジェクト同士のより柔軟な関係を表現することができる。

我々は，kamui88における場がオブジェクト同士の疎な結合を実現していることに着目し，場

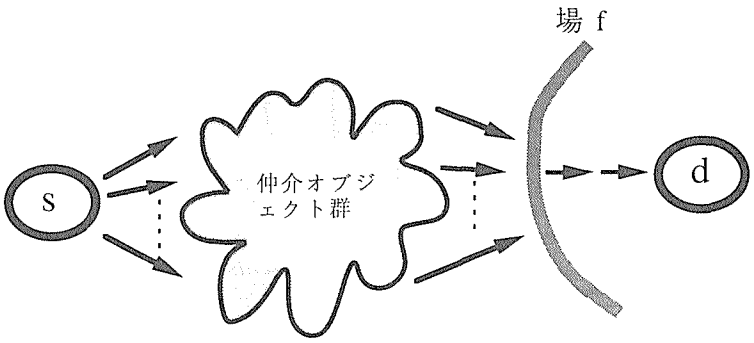


図5　Kamui88による実現

を用いてメッセージの到着順序を制御するメカニズムを考案した。これを示したのが図5である。図5において，オブジェクト $s$ から送信されたメッセージは，場 $f$ に非決定的な順序で到着する。ここで場 $f$ は，$s$ の意図に沿った順序でメッセージを通過させるように，通過順序の制御を行なう。従って，$s$ の意図通りにメッセージを受ける必要のあるオブジェクトは，場 $f$ に入ることにより $s$ の意図が満たされることになる。

## 3．メッセージ到着順序の制御

ここでは，前節で述べた2つのモデルに対して，メッセージの到着順序を制御する方法について述べる。ここで述べる方法は，順序情報の指定とその情報を利用した到着順制御の2つに分割できる。

### 3．1　direct preserving モデルにおける順序制御

この節では，あるオブジェクト中の1つのメソッドから発せられるメッセージ群に順序情報を付加するメカニズム，および，場がそれらの順序情報を基にメッセージの通過制御を行なうメカニズムについて述べる。

### 3．1．1 順序情報付加のメカニズム

まず，順序情報が付加されたメッセージの構造を次のように定義する。

**Definition 3.1 (Message Structure 1)** 場の集合を$\mathscr{F}$，順序情報を除いた正味のメッセージの集合を$\mathscr{M}$，自然数の集合を$\mathscr{N}$とするとき，順序情報が付加されたメッセージの構造を，

$$\langle f,\ n,\ m\rangle$$

と定義する。ここで，$f\in\mathscr{F}$, $n\in\mathscr{N}$, $m\in\mathscr{M}$ である。

例えば，オブジェクト $s$ が場 $F$ に対して直接，$k$ 個のメッセージ $m_1, m_2, .., m_n\in\mathscr{M}$ を送信する場合を考える。このとき，sの意図として，場Fの中に存在するオブジェクト群に対して $m_1 > m_2 > .. > m_k$ という順序でメッセージを到着させる必要があるとする。この場合，sから発せられるk個のメッセージの構造は次のようになる。

$$\langle F,\ 1,\ m_1\rangle,\ \langle F,\ 2,\ m_2\rangle,\ ...,\ \langle F,\ k,\ m_k\rangle$$

### 3．1．2 場によるメッセージ通過制御のメカニズム

ここでは，前節で示したような順序情報を伴ったメッセージが場に到着した時に，場がその情報を基に行なうメッセージ通過制御の方法について述べる。

まず，場は次のような2つの状態を持っている。

*lastNum*: その場を既に通過したメッセージの順序数のうち，最も大きなものを保持する。

*mesBuf*: その場を通過することができずに待っているメッセージの集合を保持する。

これらの初期値は，

$lastNum = 0$

$mesBuf = \{\phi\}$

にそれぞれセットされている。場 $F$ に次のような順序情報を伴ったメッセージ $m$ が到着した場合を考える。

$$m = \langle field,\ number,\ contents\rangle$$

このとき，場 $F$ は次のアルゴリズムにより，そのメッセージの通過・待機を決定する。

$number = lastNum + 1$ のとき

$m$ を通過させ，$lastNum > number$ にセットする。

*mesBuf* の各要素に対して，このアルゴリズムを再帰的に適用する。

$number > lastNum + 1$ のとき

*mesBuf* に $m$ を加える。

## 3．2 indirect preserving モデルにおける順序制御

この節では，オブジェクト群を仲介して場に到達するメッセージ群の順序制御の方法について述べる。

### 3．2．1 順序情報付加のメカニズム

indirect preserving モデルにおいては，メッセージは次のような構造をとる。

**Definition 3.2 (Message Structure 2)** 場の集合を$\mathscr{F}$，順序情報を除いた正味のメッセージの集合を$\mathscr{M}$とするとき，順序情報が付加されたメッセージの構造を，

$$\langle f,\ orderList,\ m\rangle$$

と定義する。ここで，$f\in\mathscr{F}$, $m\in\mathscr{M}$とし，*orderList* は$\mathscr{N}$のリストである。

図6のように，オブジェクト $s$ の中の1つのメソッドから，順序情報が付加されたメッセージ $m^s_1$, .., $m^s_n$ がそれぞれオブジェクト $o_1, .., o_{ns}$に送られるとする。このとき，これらのメッセージに

よって活性化されるメッセージ群が，場 $f$ で $s$ の意図通りに順序づけられるためには，メッセージ $m_1^s, .., m_n^s$ の構造は，それぞれ，

$m_1^s = \langle f, (1), contents_1^s \rangle$
$m_2^s = \langle f, (2), contents_2^s \rangle$
...
$m_n^s = \langle f, (n_s), contents_n^s \rangle$

となる。一般的に，これらのメッセージによって起動されたオブジェクトがそのメソッド中でまた複数のメッセージを発生し，これらの1部は場 $f$ に送られ，残りは他のオブジェクトへ送られることになる。オブジェクト $o_i$ を例にとると，メッセージ $m_i^s$ によって起動されたオブジェクト $o_i$ のメソッドから発せられたメッセージ $m_1^i, .., m_n^i$ の構造は，それぞれ，

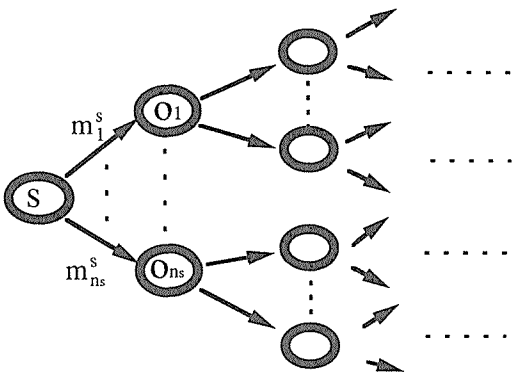


図6　メッセージ送信の連鎖

$m_1^i = \langle f, (i\ 1), contents_1^i \rangle$
$m_1^i = \langle f, (i\ 2), contents_2^i \rangle$
...
$m_n^i = \langle f, (i\ n_i), contents_n^i \rangle$

というように，受け取ったメッセージの *orderList* に新たな順序情報が append された形となる。この中の1部は，また他のオブジェクトへ送られ，そこからまた新しい順序情報が append されたメッセージが発せられることになる。

この因果関係の連鎖は，動的に変化するため，最終的に場 $f$ に到達するメッセージの数を $f$ は知ることができない。従って，この因果関係の連鎖に関わる全オブジェクトは，それぞれ，自分が発生したメッセージ列の最後を $f$ に伝える必要がある。このための特別なメッセージを終了メッセージと呼び，次のような構造をしている。

$\langle field, orderList, \perp \rangle$

ここで，$\perp$ は，そのオブジェクトがそのメソッドの実行において送信したメッセージの最後を示すマークである。メッセージに順序情報が付加される様子を図7に示す。図7において，オブジェクト $o_1$ から送信される2つのメッセージ $m_1^1, m_1^2$ の間に $m_1^1 > m_1^2$ なる順序が課せられているとき，それぞれ次のようなメッセージの構造が作られる。

$m_1^1$ : $\langle f, (1), contents_1^1 \rangle$
$m_2^1$ : $\langle f, (2), contents_1^2 \rangle$

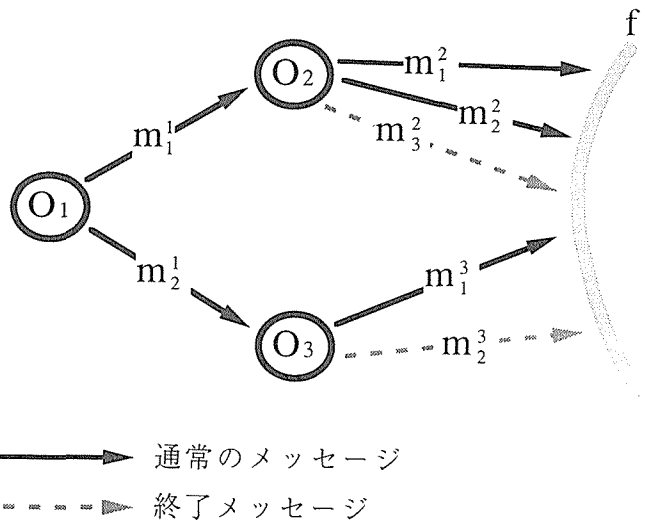


図7　indirect preserving モデルの実現

また，$m_1^2, m_2^2, m_3^2$ は $m_1^1$ の情報を受け継いでいるので，$m_1^2 > m_2^2$ の関係があり，かつ，$m_3^2$ が終了メッセージであるとすると，

$m_1^2$： $\langle f, (1\ 1), contents_1^2 \rangle$

$m_2^2$： $\langle f, (1\ 2), contents_2^2 \rangle$

$m_3^2$： $\langle f, (1\ 3), \perp \rangle$

となる。同様にして $m_1^3$, $m_2^3$ に対しても，$m_2^3$ が終了メッセージであるとすると，

$m_1^3$： $\langle f, (2\ 1), contents_1^3 \rangle$

$m_2^3$： $\langle f, (2\ 2), \perp \rangle$

となる。

### 3．2．2　場によるメッセージ通過制御のメカニズム

direct preserving モデルの場合には，順序数の比較のみで通過制御を行なうことができたが，indirect preserving モデルの場合には，リスト全体にわたっての比較になるため，通過制御は多少複雑になる。

まず，場は次のような2つの状態を持っている。

*lastList*: 既にその場を通過したメッセージの順序情報のうち，最も新しく場を通過したものを保持する。

*mesBuf*: その場を通過することができずに待っているメッセージの集合を保持する。

これらの初期値は，

*lastList*$=nil$

*mesBuf*$=\{\phi\}$

にそれぞれセットされている。場 $F$ に次のような順序情報を伴ったメッセージ $m$ が到着した場合を考える。

$$m=\langle field,\ orderList,\ contents \rangle$$

このとき，場 $F$ は次のアルゴリズムにより，そのメッセージの通過・待機を決定する。

*orderList* の**要素がすべて1**のとき

$m$ を無条件で通過させ，*lastList* に $m$ の順序情報をセットする。

*mesBuf* の全ての要素に対して，このアルゴリズムを再帰的に適用する。

*orderList* が$(n_1\ n_2\ \ldots\ n_k)$のとき

1．$n_k > 1$ のときには，*lastList* が$(n_1\ n_2\ \ldots\ n_{k-1}\ (n_k-1))$ の通常メッセージまたは終了メッセージであるとき $m$ を通過させ，*lastList* に $m$ の順序情報をセットする。
*mesBuf* の全ての要素に対して，このアルゴリズムを再帰的に適用する。

2．$n_k = 1$ のときには，*lastList* が $(n_1\ n_2 \ldots n_{k-1})$ の通常メッセージまたは終了メッセージであるとき $m$ を通過させ，*lastList* に $m$ の順序情報をセットする。
*mesBuf* の全ての要素に対して，このアルゴリズムを再帰的に適用する。

3．上記1.および2.で通過が許可されない時には $m$ は *mesBuf* に加えられる。

## 4．例題及び応用

ここでは，本モデルを応用する簡単な例題を示す。例として，図8のように幾つかの図形をウィンドウに重ね合わせて表示する処理を考えてみる。このとき，この処理には，

1．ウィンドウの中を消去する

2．四角形を描く

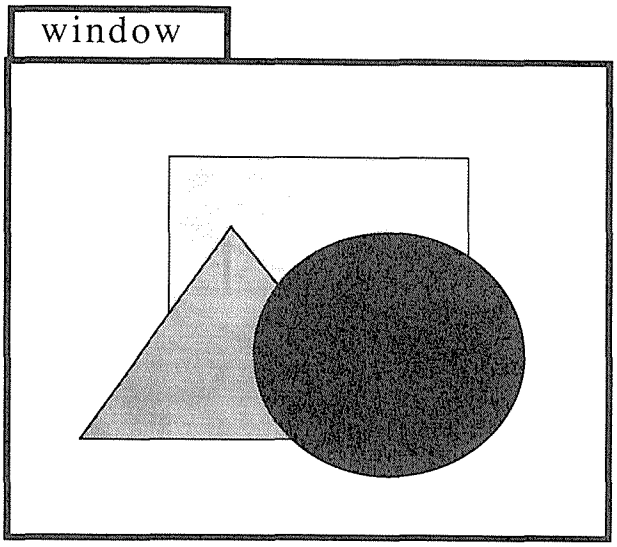


図8　例題

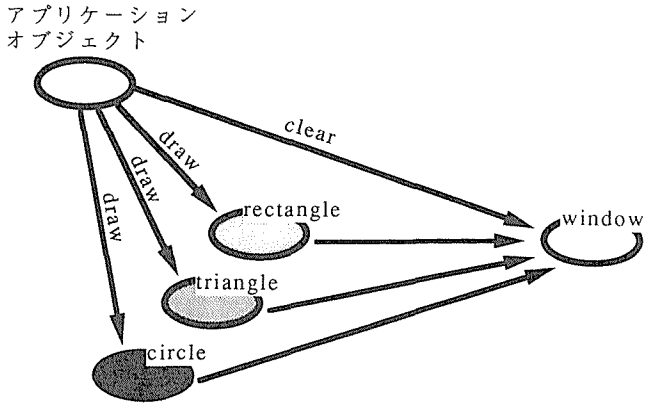


図9　従来のモデルによる例題の実現

3．三角形を描く

4．円を描く

という厳密な順序が必要である。これを並列オブジェクト指向モデルを用いて自然な形で表現すると図9に示すようなオブジェクト系になる。アプリケーションオブジェクトは，まずwindowオブジェクトに画面を消去するメッセージclearを送信する。次に，各図形の位置や大きさを計算し，その情報を引数として各図形オブジェクトに描画開始メッセージdrawを送信する。また，rectangle, triangle, circleの各オブジェクトは，位置と大きさの情報を基に自分の描画データを計算し，それぞれがwindowオブジェクトに描画の指令を送る。このとき，これら4つのメッセージは，非決定的な順序でwindowオブジェクトに到着するが，これらの間には，上で示したような順序関係が存在するため，その順序を保証する手続きが必要となる。

従来の並列オブジェクト指向モデルを用いてこの問題を表現する場合，アプリケーションオブジェクトとウィンドウオブジェクトとの間で，同期信号をやりとりして順序を保証することが考えられるが1．2節で述べたように，アプリケーションオブジェクト側に同期信号を受けるための

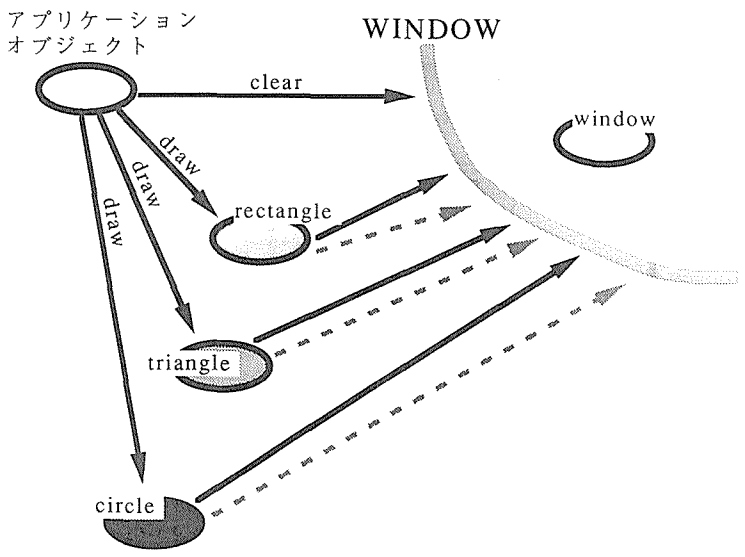


図10 本モデルによる例題の実現

メソッドを用意する必要があり，不自然な表現を強いられることになる。また，同期信号を受けてから次のメッセージを送信するために，図9の rectangle, triangle, circle の各オブジェクトの処理は完全に逐次的に行なわれることになる。

本研究における indirect preserving モデルを用いた場合，まず，図10に示すように，window オブジェクトは場 WINDOW に置かれる。アプリケーションオブジェクトは，順序保存の対象となる場として WINDOW を指定し，また，

画面消去
⇩
四角形の描画開始
⇩
三角形の描画開始
⇩
円の描画開始

という順序関係を指定して4つの非同期的メッセージを送信する。それによって，各メッセージには，順序情報が加えられ，場 WINDOW には最終的に，終了メッセージを含む7つのメッセージが到着することになる。そこで，場 WINDOW は，メッセージの順序情報を基に4つのメッセージを指定された順序で通過させる。この方法を用いることにより，window オブジェクトの画面消去処理，rectangle, triangle, circle の各オブジェクトの描画データ計算という4つの処理が並列に処理可能となる。

## 5．おわりに

本研究では，まず，分散計算においては，メッセージの送受信の間の時間的遅延により，メッセージの到着順を制御するためには同期操作をプログラム中に埋め込む必要があり，それがプログラムの可読性や実行時の並列性を阻害することを述べた。そして，それを解決するために，メッ

セージの到着順序を制御できる新しい並列オブジェクト指向モデルを提案した。そこでは，あるオブジェクトから他のオブジェクトへ直接メッセージを送る場合を対象としたdirect preservingモデル，途中に仲介するオブジェクト群が存在するindirect preservingモデルの2つを提案した。また，Kamui88における場を用いたそれらの実現方法を示した。次に，簡単な例題を基に，従来の並列オブジェクト指向モデルと本モデルとの比較を行なった。そこでは，本モデルを用いることによるプログラムの可読性及び実行時の並列性の向上が期待できることを示した。

現在，このモデルに基づいた並列オブジェクト指向言語をCommon LISP上で作成中であるが，今後の課題としては，この言語を用いて複雑なアプリケーションを構築し，プログラミング上での優位性の確認及び実行時の並列性の向上を確認することが挙げられる。また，メッセージ到着順序の指定の宣言性を活用したデバッグシステムの考案も重要な課題である。

## 参考文献


1) Campbell, R.H. and Habermann, A.N.: The Specification of Process Synchronization by Path Expressions, *Lecture Notes in Computer Science,* Vol.16(1974), pp.89－102.

2) Hewitt, C.E.:Viewing Control Structures as Patterns of Passing Messages, *Artif. Intell.,* Vol.8, No.3(1977), pp.323－364.

3) 米澤明憲 他：オブジェクト指向に基づく並列情報処理モデルABCM/1とその記述言語ABCL/1，コンピュータソフトウェア，Vol.3，No.3（1986），pp.9－23.

4) Yonezawa, A. and Tokoro, M.: *Object-Oriented Concurrent Programming,* The MIT Press, 1987.

5) 渡辺慎哉 他：場とイベントによる並列計算モデル－Kamui88，コンピュータソフトウェア，Vol.6，No.1（1989），pp.41－55.

6) 渡辺慎哉 他：並列オブジェクト指向モデルにおけるイベント処理順序保存機構，日本ソフトウェア科学会第6回大会論文集，（1989），pp.249－252.