北海道大学工学部研究報告
第156号（平成3年）

Bulletin of the Faculty of Engineering
Hokkaido University No. 156 (1991)

分散協調計算の動作的意味論

川口雄一[1]　宮本衛市

（平成3年3月29日受理）

An Operational Semantics of Cooperative Distributed Calculus

Yuuichi KAWAGUCHI and Eiichi MIYAMOTO

(Received March 29, 1991)

Abstract

This study was intended to proviele a formal semastic analysis for the cooperative calculus by some processes in a distributed environment. The asynchronization is one of the characteristics of the distributed cooperative calculus, therefore, when some processes provide a cooperative effort on a distributed environment, communications are conducted with messages. This communication is basically asynchronous. So the means to treat it theoretically are required. And one more characteristic of distibuted cooperative calculus is the time required by communication. This time is no smaller than the time which is required by a real calculation.

In this paper we extend R. Milner's process calculus CCS and construct a new process calculus which gives the formal semantics for cooperative distributed calculi with asynchronous communication that takes some time.

1. はじめに

計算機資源を分散して配置する計算方法は，負荷分散や処理効率，耐故障性等の面から見て集中計算方式に対して優れている。しかし，分散協調計算の場合，他の計算プロセス[2]が保持するデータを用いて計算を行うためには，他プロセスと通信を行い，同期をとりつつ計算を進める必要があり，この協調のための通信アルゴリズムは一般的に非常に難しい。従って，分散協調計算のアルゴリズムが正しいことを検証するための理論的考察が重要になる。

本研究の目的は，分散環境上でプロセス群により行われる協調計算に対し，形式的な意味を与えることにある。与えられた意味を用いることによって，その計算の検証を行うことが可能になる。具体的な検証項目として，停止性（正当性），非デッドロック性（健全性）等が挙げられる。

分散協調計算の特徴は非同期性にある。すなわち，分散配置された各々のプロセス毎に時計が存在し，それぞれの時計はそれぞれ固有の時を刻む。プロセス群全体で共通な時計は存在しない。個々のプロセスは個々の時計と同期し，従って全体としては非同期的に計算が進む。

---

[1] 情報工学科言語情報工学講座

[2] 本論文で言うプロセスとは，計算を行う任意の主体を指す。

それ故に，複数のプロセスが同期をとって分散協調計算を行うためには，メッセージを介した通信を行う必要がある。分散協調計算においてプロセス間の協調手段は通信のみである。通信の種類は，同期通信と非同期通信とに分けられる。それぞれの特徴を示すと，非同期通信を用いる協調方式では各プロセスの並列度が高く，処理効率の良い分散協調計算が可能である。しかし，通信経路については一般に不定であるので，メッセージの追越しが起こる可能性がある。結果として，送信されたメッセージが送信された順番で受信側に到着するかどうかは不明である。同期通信を用いた場合は，プロセス群の並列度が低い分散協調計算になってしまう。しかしながら，メッセージの到達順序については送信された順序で受信されることが保証される。

分散環境では，上述のとおり非同期性が本質であるので，プロセス間の協調手段としては基本的に非同期通信を用いる。同期通信は非同期通信を基にして実現される。

ところで，集中環境つまり密結合環境に比べて，分散環境での通信にかかる時間は実際の処理時間に対して無視できないほどの大きさをもち，通信動作をアトミックアクションとすることができない。従って，同期通信であろうが非同期通信であろうがメッセージを送信してから受信されるまでの時間で，プロセス群の中のいくつかのプロセスの処理が進み，結果としてプロセス群の状態が変化してしまう。故に通信時間はプロセス間の協調動作に重大な影響を及ぼす。

本論文では，同期通信を用いた並行協調計算に対して形式的意味を与える手段である R. Milner のプロセス計算 CCS (a Calculus of Communicating Systems) を基礎とし，CCS では扱うことのできない分散協調計算，すなわち，非同期通信及び通信時間の概念に関しての拡張を施す。以下，2 章では CCS の説明をし，3，4 章でこれを分散環境を表現すべく拡張する。5 章ではその具体例を示す。

2. プロセス計算

この章ではプロセス計算 CCS [4] の基本方針について概説する。本論文でも CCS のこの基本方針を採用している。

CCS は以下の二つのことを行う。

(1) プロセス（群）の挙動を代数法則を用いて表現する。

(2) プロセスの或る部分が或るプロセスと同値であるかどうかを判定する。

CCS では，プロセスの外側に「観測者」を置く。この観測者から見たプロセスの挙動をもってプロセスの意味とする。この意味は計算機言語意味論の中での分類では動作的意味 (operational semantics) に相当する。この意味が等しい部分プロセス群は CCS における同値なプロセスである。

プロセスの動作をイベントと呼ぶ。イベントには 2 種類あり，それぞれ「送信イベント」と「受信イベント」と呼ばれる。送信イベントと受信イベントが同期することによって通信が起こる。観測者はこれ以外のイベントを観測しない。すなわち，CCS ではプロセスの動作のうち，通信動作のみについて注目する。

送信プロセスは送信すべきメッセージ名とそれに付随する引数を指定して送信を行う。受信プロセスは受信メッセージ名を指定して受信動作を行う。引数を受信するためには変数を使用する。送信側は受信プロセスを指定しないので，メッセージ名が合致した任意の受信プロセスの中のどれか一つと通信を行う。この選択は非決定的になされる。

通信の開始（送信開始）から終了（受信完了）までがプロセス群全体でアトミックアクション

として扱われる。この意味するところは，通信には常に時間がかからないと見なせるということである。プロセス内部で起こった通信は外部にいる観測者には観測できず，「τ」という特別なイベントとして観測される。プロセス内部で起こる状態変化は全てτとしてしか観測できない。

2. 1 プロセス表記の具体例

CCSにおいて送信イベントはメッセージ名に上線を付けて表す。受信イベントはメッセージ名で表す。プロセスの動作記述を動作式（behaviour expression）と呼ぶ。以下は動作式の例であり，メッセージaかdを送信し，次にbかcを送信するプロセスpを示している。各メッセージには引数として1，4，2，3が付いている。又，「.」によってイベントの逐次実行を表し，「+」によってイベントの選択的実行を表現する。

$$p=(\bar{a}1+\bar{d}4).(\bar{b}2+\bar{c}3) \tag{1}$$

このプロセスpは外部との通信で$\bar{a}1$という送信イベントを起こす可能性があるが，これを二項関係「→」を用いて，

$$p \xrightarrow{\bar{a}1} \bar{b}2+\bar{c}3 \tag{2}$$

と表す。この二項関係を導出（derivation）という。導出可能かどうかは別に定義された推論規則によるが，この推論規則についての説明は省略する［4］。

次に，メッセージaを受信し，続いてb又はcを受信するプロセスqは以下で示され，次の導出が可能である。ここでx，y，zはメッセージの引数を受信するための変数である。

$$q=ax.(by+cz) \tag{3}$$

$$q \xrightarrow{ax} by+cz \tag{4}$$

そして，pとqが並行に動作するプロセス群は，

$$p|q=(\bar{a}1+\bar{d}4).(\bar{b}2+\bar{c}3)\,|\,ax.(by+cz) \tag{5}$$

で示される。この時，送信イベント$\bar{a}$と受信イベントaが同期して通信が起きる可能性があるが，これは観測者から見てプロセス内部の通信なので，τイベントとして観測者に観測される。すなわち，導出を使った表現では，以下のようになる：

$$p|q \xrightarrow{\tau} (\bar{b}2+\bar{c}3)\,|\,(by+cz) \tag{6}$$

このように動作を記述されたプロセスが，導出によって観測者に観測される可能性のあるイベント列は一般的に木構造を成すが，この木をもってそのプロセスの意味とする。意味が等しい二つのプロセスの関係を「観測同値」という。観測同値性は部分的にプロセスが交換可能であるかどうかの判定に用いられる。

2. 2 CCSの問題点

CCSで分散協調計算を表現する場合に問題となる点は「CCSで扱える通信は同期通信のみであり，しかもアトミックアクションである。」という点である。前述のとおり，分散環境での通信は無視できない程の通信時間がかかり，通信時間が実際の計算に大きな影響を及ぼす。よって，通信時間を表現できないならば分散環境下での通信を表現し得ないといえる。

又，同期型の通信のみしか表現できないということは，やはり前に示したとおり，各プロセスの並列度が低い分散協調計算しか表現できないことになる。

よって，以降ではCCSで非同期通信を扱うように拡張する。それと同時に，通信時間を表現で

きるようCCSに拡張を施す。

3. 分散協調計算への拡張

本章では分散協調計算を表現すべく，プロセス計算CCSの拡張を行う。具体的な拡張点は前述のとおり，通信時間の表現と非同期通信の扱いである。又，計算モデルとして，並行オブジェクト指向計算モデルKamui 88［7］を採用する。並行オブジェクト指向計算モデルを用いた場合，各プロセスを並行オブジェクトと見なすことにより，分散協調計算を自然に表現できるからである。

Kamui 88モデルはオブジェクト間の通信手段として，制限された放送を提供しているので，今回のCCSの拡張に際して，CCSが放送を扱えるようにも拡張する。

3. 1 Kamui 88

本節ではKamui 88についての説明を行う。Kamui 88モデルはオブジェクトと場により構成される。オブジェクト間の協調手段として，前述のとおり放送，すなわち一対多の通信が提供される。

オブジェクトは一般的なオブジェクト指向モデルの場合と同様，内部状態と受信したメッセージに対するメソッド（動作記述）を持つ。内部状態を他のオブジェクトが直接参照・変更することはできない。又，非同期に送信されてくるメッセージを受信するためのメッセージキューを持ち，ここにメッセージを到着順に格納する。

オブジェクトはメッセージキューからメッセージを一つ取りだし(受信)，そのメッセージに対応するメソッドを実行する。メソッド内ではメッセージの送信か内部状態の変更のみを行い，メッセージを受信することはない。メソッドの実行はオブジェクト内でアトミックアクションとして扱われるので，複数のメソッドが同時に実行されることはない。場はオブジェクトをグループ化して扱うための手段である。オブジェクトは複数の場に動的に属することができる。オブジェクトが通信手段として放送を用いる場合は，相手オブジェクトではなく場を指定する。放送はその場に属しているオブジェクト群にのみ送られる。すなわち，場は放送のスコープの役目を果たしている。

場に属しているオブジェクト群が実行時に変化することにより，動的な通信相手（群）の決定が可能である。即ち，従来の一対一通信を用いた場合には表現できなかった，受信相手を群として見なしたオブジェクト間の協調をKamui 88モデルでは提供している。

3. 2 放送の扱い

Kamui 88モデルでは放送をオブジェクト間の協調手段として提供しているが，相手が一つのオブジェクトに制限された放送と一対一の通信は等しい効果をもたらすので，オブジェクトは通信手段として放送のみを用いることとする。この場合，オブジェクトにはそれと同名な場が必ず存在していて，その場にはそれと同名なオブジェクトが必ず属していることを仮定する。

放送の始まりから終わりまでは次のようになっている：「送信オブジェクトが放送を開始した後，非決定的な時間が経った或時刻において，放送を聞くことのできるオブジェクト群が決定される。この決定が為された後の各オブジェクトの内部状態の変化はこの決定に影響を与えない。この決定の後，放送されるメッセージは各オブジェクトへと送られる。」

放送は送り手と受け手の間で非同期的に行われる。同期的に行われる放送では複数の受け手全ての受信態勢が整うまで送り手が待つことになり，送り手オブジェクトの並列性を著しく損なう

からである。

3. 3 メッセージの到達順序保持機構

CCS では同期通信のみを扱うので，あるプロセスから別のプロセスへ複数のメッセージが送信された場合，メッセージは送信された順番に到着する。上述のとおり，本モデルでは通信が非同期に行なわれ，しかも環境は分散環境なので，送り手から見ればメッセージの通信路に関しては全く不定である場合が多い。故に送り手のオブジェクトから送られたメッセージが，受け手のオブジェクトにどの順序で到達するかが不定となる。そこで例えば[8]のような方法を用い，メッセージの到着順序を送り手の意図どおりにすることが必要である。本節では ACTOR[3]によって示された，送り手オブジェクトが順序指定を行うアイディアを CCS に適用する。

オブジェクトはメッセージを送信する場合に以下を指定する。

(a) メッセージ名

(b) メッセージ番号

(c) 場の名前

(d) このメッセージよりも先に到着すべきメッセージの集合（順序指定）

(e) メッセージの引数

オブジェクトは内部状態として一つの自然数を持っていて，これをメッセージ番号と呼ぶ。オブジェクトはメッセージ番号の値を設定・参照できる。送信時にはメッセージ名に合わせてこれを指定する。従ってメッセージ名が同じでも異なるメッセージ番号が指定されたメッセージは区別される。メッセージ名のみを区別すべき場合は番号として 0 を指定する。

受信オブジェクトは既に受信した番号付きのメッセージの集合（履歴）を保持し，これと指定された順序とを照合することにより，送信側の指定した順序通りにメッセージを受信する。メッセージの履歴はオブジェクトの内部状態ではあるが，オブジェクトはこれを操作できず，自動的に更新される。

この仕組により，一つのオブジェクトから放送された一連のメッセージがその指定通りの順序で受信オブジェクト群に到着することが保証される。

以上のように拡張された CCS を CCS-A と名付ける。

4. CCS-A

本節では CCS-A におけるプロセスの表記法について説明する。CCS を拡張した点は放送の表現とメッセージの順序保持に関するプロセスの導出である。推論規則は，CCS のものをそのまま適用できる。

4. 1 動作式

オブジェクトは各メソッドの先頭でのみメッセージを受け取るので，オブジェクトの動作式は以下の B のような形に制限される。

$$B=(a_1.B_1+\cdots+a_n.B_n).B \tag{7}$$

各 a_i はメッセージの受信を表わすアトミックアクションであり，各 B_i は受信したメッセージに対するオブジェクトの行動（メソッド）を表わす動作式である。

オブジェクトは以下の 3 つの行動を起こすことが可能である。

(a) メッセージの放送

(b) 内部状態の変更

(c) メソッドの先頭でキューよりメッセージを取り出す

(a)は例えば メッセージ m をメッセージ番号 n で場 c に順序指定 A で放送する場合,

$$\boxed{\tilde{m}(n)[c,\ A](x,\ y,\ z,\cdots)} \tag{8}$$

と書く。x, y, z, ...は引数である。

(b)は, 変数の単一代入規則を満足するために動作定義式(behaviour identifier) [4]を用いる。例えばxという内部状態を持つオブジェクトの動作定義式作 $b(x)$ として,

$$\boxed{b(x)=(my.b(y+x)+m'z.b(z-x))} \tag{9}$$

と定義されていたとすると, これは受信イベントによって内部状態が変化するオブジェクトを表している。

(c)はオブジェクトが メッセージ m に対応するメソッドを持ち, そのオブジェクトの所属する場の集合が C で履歴が H の場合メソッドの先頭 (上述の a_i) は,

$$\boxed{m(x)[C,\ H](p,\ q,\ r,\cdots)} \tag{10}$$

と書く。x は送信時に指定されたメッセージ番号を受けるための変数, $p,q,r,\ldots$はそれ以外の引数を受け取るための変数である。

これら3種類の動作を"."と"+"で連結することにより, CCS と同様にオブジェクトの動作式が表現される。

4. 2 放送を含むプロセスの導出

放送されたメッセージをどのオブジェクト群に送るのかが決定されるまでの時間は不定である。その時間を表現するために「放送 (通信) の途中」という概念を導入し, 放送を3段階の導出によって表現する。図1にその概念図を示す。

導出 (*ED* 1) で送信オブジェクトが送信を行い, 導出 (*ED* 2) で受信オブジェクト群が決定され, それぞれのオブジェクトのメッセージキューにメッセージが到着する。キューへの到着の順番は導出の順番ではなく, 送り手に指定されたとおりになる。導出 (*ED* 3) で, キューからメッセージが取り出される。取り出される (導出 (*ED* 3) の起きる) 順番は履歴との比較により送信時に指定された制約による。導出 (*ED* 3) の後 $b(\ldots)$ 中の履歴は更新される。

$\tilde{m}$, $\bar{m}$ メッセージが存在している間にプロセス群中のいくつかのプロセスがいくつかのイベントを起こし, プロセス群の内部状態が変化する。$\tilde{m}$, $\bar{m}$ の存在している時間をもって, 通信時間を表現している。CCS ではこの通信時間を表現できなかった。ところで, 後で述べるがこれら2つのイベントは仮想的なものなので, メソッド中には存在しない。

それぞれの導出を式で書くと以下のようになる。B, B_x を動作式とする。メッセージ m を場 c に順序指定 A で送信する動作式は以下になる。メッセージを送信する場合, $\hat{m}$ のように書く。

$$\hat{m}(n)[c,\ A].B \xrightarrow{\hat{m}(n)[c,\ A]} \tilde{m}(n)[c,\ A]|B \quad (ED\ 1)$$

この導出の後, $\hat{m}$ が送信されたメッセージ $\tilde{m}$ に変化する。この $\tilde{m}$ は送信オブジェクトの動作式 B とは並行に存在する。この導出によって非同期通信を表している。

$\bar{m}(n)[c,A]$ を含むプロセスは以下の導出が可能である。$\#i$ はオブジェクト名である。

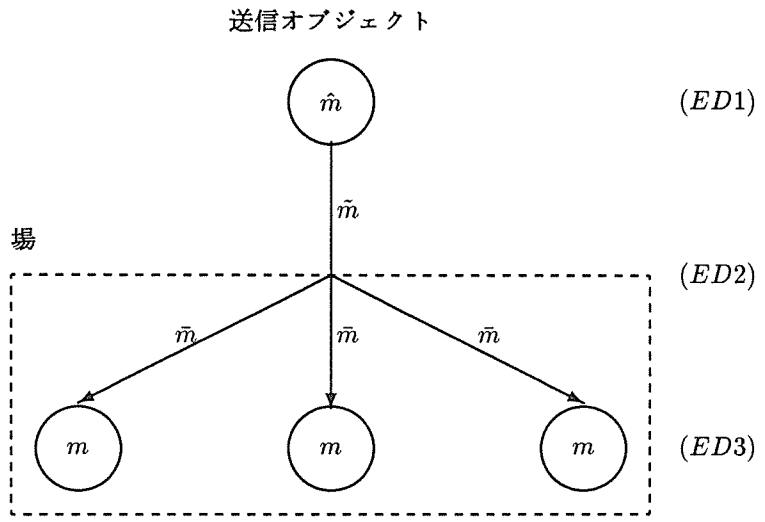

図 1 放送の概念図

$\tilde{m}(n)[c,\ A]$
$|m(a_1)[\{\#\,1\}\cup C_1,\ H_1].B_1$
$|\cdots$
$|m(a_x)[\{\#\,x\}\cup C_x,\ H_x].B_x$
$\xrightarrow{\tilde{m}(n)[c,\ A]}$
$\bar{m}(n)[\#\,i_1,\ A]$ $(ED\ 2)$
$|\cdots$
$|\bar{m}(n)[\#\,i_k,\ A]$
$|m(a_1)[\{\#\,1\}\cup C_1,\ H_1].B_1$
$|\cdots$
$|m(a_x)[\{\#\,x\}\cup C_x,\ H_x].B_x$

送信時に送信オブジェクトは受信相手として場 c を指定したが, 導出(ED 2)を経ることによってこのメッセージの送信相手が, この導出を起こした時点でその場に含まれていたオブジェクト一つ一つ ($\#\,1,\ldots,\#\,k$) に変化している。この時点で受信相手群が決定されたわけである。

導出 (ED 2) を終了した時点で受信オブジェクトそれぞれのメッセージキューにメッセージが格納されたことを示す。

(ED 2) の導出を起こすためには, 以下の前提条件を満足しなければならない。

$$(\text{if } c\in\{\#\,i_j\}\cup C_{ij},\ 0\le i_1\le i_j\le i_k\le x) \tag{11}$$

$i_1=i_k=0$ の場合には, $\tilde{m}(n)[c,A]$ は誰にも受信されず世界から消えてしまう。しかし, 送信オブジェクトはメッセージを導出 (ED 1) によって送出できる。

$\bar{m}(n)$ $[\#\,i,\ A]$ を含むプロセスは以下の導出ができる。

$$\bar{m}(n)[\#\,i,\ A].B|b(C,\ H,\ R)\xrightarrow{\tau} B|b(C,\ H\cup\{m(n)\},\ R) \qquad (ED\ 3)$$
但し, $b(C,\ H,\ R)=m(x)[C,\ H].b(C,\ H\cup\{m(x)\},\ R),\#\,i\in C$

(ED 3) の導出を起こすことによってキューに格納されたメッセージをオブジェクトが受信し

たことを示している。この導出が起こる順番はオブジェクトの内部状態である履歴と比較して決められる。すなわち，導出（*ED* 3）を起こすには，以下の条件を満足していなければならない。この条件を満たして導出を起こすことにより，メッセージの到達順序は送信オブジェクトの指定どおりになる。

$$\begin{pmatrix}\text{if } n\neq 0\wedge A\subset H\\ \qquad\qquad \text{又は}\\ \text{if } n=0\wedge(\forall m(n)\in A\Rightarrow\exists\, x \text{ s.t. } m(x)\in H)\end{pmatrix} \tag{12}$$

この導出の後，受信オブジェクトの履歴 H は自動的に受信メッセージを格納して更新される。

5. 具体例（会議の召集）

簡単な応用例として，会議を召集するオブジェクトを考える。このオブジェクトは，議員オブジェクト群に対して会議が召集されたことを知らせる。その後，召集オブジェクトは会議の日程を知らせるか，又は，会議が中止になったことを知らせる。

召集のメッセージを a，日程のメッセージを b，中止のメッセージを c とする。議員オブジェクトはあらかじめ場 G に入っている。召集オブジェクト $C(\ )$ は以下のように表される。

$$C(n)=\hat{a}(n)[G,\{\ \}].(\hat{b}(n)[G,\{a(n)\}]+\hat{c}(n)[G,\{a(n)\}]) \tag{13}$$

b メッセージと c メッセージには到着順序指定が付いているので，a メッセージの後に受信オブジェクトに到着する。

議員オブジェクトの動作式を以下に示す。

$$\begin{aligned}G_{\#i}(H)=a(x)[\{\#\,i\}\cup\{G\},\ H].(&\\ &b(y)[\{\#\,i\}\cup\{G\},\ H\cup\{a(x)\}]+\\ &c(z)[\{\#\,i\}\cup\{G\},\ H\cup\{a(x)\}])\end{aligned} \tag{14}$$

そして，例えば議員が g 人いるプロセス群 q の協調動作の様子は以下のように3段階の通信を伴って観測される。この例では召集メッセージ a が送信されてから受信されるまでを示す。

$$\begin{aligned}&q=C(0)|G_{\#1}(\{\ \})|\ldots|G_{\#g}(\{\ \})\\ &\ =\hat{a}(0)[G,\{\ \}].(\hat{b}(0)[G,\{a(0)\}]+\hat{c}(0)[G,\{a(0)\}])\\ &\quad |G_{\#1}(\{\ \})|\ldots|G_{\#g}(\{\ \})\\ &\quad\ \vdots\\ &\underline{\hat{a}(0)[G,\{\ \}]}\qquad\qquad\qquad\qquad\qquad\qquad;(\text{送信開始})\\ &\quad(\hat{b}(0)[G,\{a(0)\}]+\hat{c}(0)[G,\{a(0)\}])\\ &\quad|\tilde{a}(0)[G,\{\ \}]\\ &\quad|G_{\#1}(\{\ \})|\ldots|G_{\#g}(\{\ \})\\ &\quad\ \vdots\\ &\underline{\tilde{a}(0)[G,\{\ \}]}\qquad\qquad\qquad\qquad\qquad\qquad;(\text{放送})\\ &\quad(\hat{b}(0)[G,\{a(0)\}]+\hat{c}(0)[G,\{a(0)\}])\\ &\quad|\bar{a}(0)[\#\,1,\{\ \}]|\ldots|\bar{a}(0)[\#g,\{\ \}]\\ &\quad|G_{\#1}(\{\ \})|\ldots|G_{\#g}(\{\ \})\\ &\quad\ \vdots\\ &\underline{\tau}\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad;(\text{受信})\\ &\quad\ldots|G_{\#i}(\{a(0)\})|\ldots\end{aligned} \tag{15}$$

（放送開始）の導出で非同期に送信されたメッセージ a は場 G を指定されている。a より先に受信すべきメッセージはないので，順序指定は付いていない。(放送)の導出が起きた時点で，場 G に属しているオブジェクト群に対してそれぞれにメッセージが送信され，キューに格納される。この後，(受信)の導出によってメッセージ a は受信されるが，(受信)の前に例えば中止メッセージ c が送信されていたとしても，c の順序指定である $\{a\,(0)\}$ によって a が受信される前に c が受信されることはない。

CCS と比較して，通信に放送を用いているので，召集オブジェクトが召集される議員オブジェクト群の数をあらかじめ知る必要がない。又，メッセージの到達順序を指定できるので，召集する前に中止のメッセージが先に到着することのないことが保証される。更には，議員オブジェクトが別の仕事で忙しく，召集を受信可能な状態でなくとも，非同期通信を用いることによって召集オブジェクトは，メッセージの送信が可能になる。その場合でも，場 G に属してさえいれば議員オブジェクトは召集メッセージを聞き漏らさない。このように CCS-A を用いることによって並列性が高く，しかもメッセージの到達に関して安全な計算モデルを表現可能である。

6. CCS-A に関する考察

本節で示したプロセス計算は，「通信の途中」の概念を用いることによって，分散環境でのメッセージ通信に本質的な影響を及ぼす通信時間を表現した。又，非同期メッセージ通信を行う場合に送信オブジェクトが到着順序を指定可能な枠組みを提供することによって，メッセージの順序制御を含めたプロセス群の動作的な意味を表現可能となった。

ECCS [6] でも CCS で非同期通信を扱う拡張を行っているが，ECCS はプログラムの実行時には送られるべき全てのメッセージが式中に並行に存在しているのに対し，CCS-A では導出を行わなければ，メッセージが送信されないという点で異なっている。更に，ECCS でも到着順序を保証する順序メッセージ通信を扱うが，上記の方法とは異なり，順序を保存するためのプロセスを特別に用意し，受け手側で順序制御を行うものである。又，ECCS では順序制御のための手続きを書く必要があるが，本論文による方法では，到着順序を指定するだけで到着順序が保証され，いかなる実現手段によるかは問わない。到着順序制御の方法まで含めたオブジェクトの動作を扱う場合は，ECCS の方法を使う必要があるが，本論文では到着順序制御はオブジェクトの役目ではないと考える。

CCS ではプロセス外部に存在する観測者が観測可能なイベントのみを扱った。すなわち，送信と受信のみを観測可能とした。これに対して，CCS-A では送信されたメッセージがどのように受信プロセス群に到達するかまでを外部の観測者に示している。しかし，これは観測者がプロセス内部を観測可能になったことを意味しない。通信路と場の仕様についてはプロセス計算では関知しておらず，従って，それらはプロセスの外にあることになり，仮想的に $\tilde{m}$ とメッセージ $\hat{m}$ が，$\tilde{m}$ と $\hat{m}$ がそれぞれ通信を行った結果外部に観測されたのである。実際に $\tilde{m}$，$\tilde{m}$ が存在することはないので，これらが「τ」として観測されることはない。

ところで，受信オブジェクト群が決定されるまでの時間が通信途中の時間であるが，この決定が本モデルでは時間軸上のある一点でなされている。Kamui 88 モデルでは場が通信（放送）を司っている，つまり集中管理しており，このような決定がなされるわけであるが，これは厳密な意味で分散協調計算モデルを表してはいないとも考えられる。

厳密に分散環境である場合，メッセージを放送するオブジェクトは Kamui 88 モデルの送信方

法を用いても，受信オブジェクト群を決定するために場のような集中管理機構を使うことは不可能であろう。この場合，放送されたメッセージは全オブジェクトに送られ，それぞれのオブジェクトが個々に，そのメッセージが自分の受信すべきメッセージかどうかを判断することになる。しかしこの場合は，例えば，同じメッセージが複数回同一オブジェクトに到着することを防がなければならないという問題が現れる。

7. 将来の課題

CCS-A では時間や距離を具体的な数量として表現する枠組みを提供していないので，通信に「時間がかかる」ということしか示すことができない。将来的にはプロセス間の距離のようなものを導入することによって，通信時間を表せるプロセス計算を実現したい（例えば［2］）。

本論文の場合は通信される可能性のあるメッセージ名が実行前に全てわかっていると仮定していたが，この仮定を取り払った場合（例えば［1］），受信オブジェクト側は全てのメッセージを受信するために，メッセージ名を変数で受ける必要がある。現在は，この枠組みを提供していない。更には，メッセージの引数にオブジェクトや動作式そのものをつけることの可能な計算モデルも存在するが，これも表現し得ない。これらは全て現在の CCS-A に対して高階な扱いを要求するものである［5］。

高階な扱いが可能になれば，例えばメッセージキューというプロセスの意味も論じられるようになる。今後は，CCS-A をこの方針で拡張して行きたい。

謝　辞

本研究を行うにあたり，御指導頂きました北海道大学工学部情報工学科渡辺慎哉助手に深く感謝致します。又，種々の面で御助言を頂きました赤間清助教授，三谷和史助手，言語情報工学講座の皆様に感謝致します。

参考文献

(1) C. Hewitt, *et. al*. *Open Systems*. On Conceptual Modelling, Springer-Verlag, 1984.
(2) Chris Tofts. *Timed Concurrent Processes*. Semantics for Concurrency, Springer-Verlag, 1990.
(3) Gul A. Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems*. The MIT Press, 1986
(4) R. Milner. *A Calculus of Communicating Systems*. LNCS 92, Springer-Verlag, 1980.
(5) R. Milner. *Functions as Processes*. LNCS 443, Springer-Verlag, 1990.
(6) 戸村 哲, *et.al*. プロセス代数モデルに基づく並行オブジェクト指向言語の意味定義. コンピュータソフトウェア, vol. 7 No. 2, 1990
(7) 渡辺 慎哉, *et.al*. 場とイベントによる並列計算モデル - *Kamui* 88. コンピュータソフトウェア, vol. 6 No. 1, 1989.
(8) 渡辺 慎哉, *et.al*. 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, D 4-2, 1989.