# 場と一体化したプロセスの概念に基づく並列協調処理モデル Cellula†

吉田紀彦†† 楢崎修二††☆

協調型問題解決を主な対象とする並列協調処理モデル Cellula を新たに提案する．Cellula は次の2点をその本質とする．すなわち，①実行主体としてのプロセスと通信媒体としての場を一体化し，これを階層的にネストさせる．②パターン・マッチングを通信の基礎とし，通信の形式は通信命令の種別でなく，通信情報の属性で指定する．以上により，従来の並列処理モデルや黒板モデルにはない次のような特長が実現する．すなわち，①プロセス，場（黒板），チャネル，メイルボックスなどがすべて，「プロセス＋場」という唯一のモジュールに統合される．②少数の統一化された命令のみで，直接／間接送信，放送，自律的受信，封鎖／非封鎖通信などの多様な通信が可能になる．通信形式の拡張も，整合性を保ったまま容易に行える．③プログラム内だけでなくプログラム間の通信も，モデルの枠組の中で統一的に記述できる．④一般の問題解決手法だけでなく，分割統治法などの再帰的な手法への直接の対応が可能になる．例として，分枝限定法による巡回セールスマン問題および分割統治法によるマンデルブロ図形生成などについてCellula に基づいて記述を行い，その有効性と適用性を確認した．

## 1. はじめに

協調処理モデル[1]とは，比喩的に言えば，複数の人間が互いに相談したり助け合いながら，全体で1つの処理を実行するような計算モデルをいう．ここで重要な点として，各人は自分に関係することを原則としてすべて自分で処理する自律的な存在であり，管理者や指揮者に類する中央集権的な存在はない．すなわち，このモデルは，全体的な組織や制御の構造が前もってあまり規定できず，処理過程が各人の判断に委ねられるような応用のためのものである．

協調処理モデルは，実行主体の自律性と分散性という点で，オブジェクト指向や並列／分散処理との親和性が高い．このモデルが適する応用として，次のようなものがある．

① シミュレーション[2]

② 自律分散システム (autonomous decentralized system)[3]

③ 知識処理，人工知能

特に知識処理，人工知能に関して言えば，このモデルは，推論機構／知識の構造化や並列／分散処理との融合などについてブレークスルーをもたらすものとして期待されている．そして，協調型問題解決 (cooperative problem solving)[4]，またはより一般的に分散人工知能 (distributed artificial intelligence)[5] という名称で，研究分野が形成されつつある．ここで，問題解決 (problem solving) とは知識処理，人工知能の基礎に位置する技法であり，一言で言えば，アルゴリズミックな解法が（未だに）存在しない問題を，試行錯誤すなわち探索によって解くものである．

数値計算，画像処理，グラフィクスなどにおける従来の並列処理モデルに比較して，問題解決などにおける協調処理モデルは次のような相違を持つ．すなわち，前者が「誰がいつ何をすればよいかが前もって概ねわかっている」という定型的な応用を扱うのに対して，後者は，上で述べたように「誰がいつ何をすればよいかが前もってわからない」という非定型的な応用を扱う．これは，プロシージャルなシステムと，人工知能で言うプロダクション・システムの対立関係にもなぞらえられる．

この定型／非定型の観点から，従来の並列処理モデルの通信機構[6]について，改めて検討を加える．

① 共有メモリ通信：抽象度が低すぎる．同期や相互排除などの制御がプログラマに任されており，プログラマへの負担が大きい．

② メッセージ通信：次のような制約を内包しており，基本的に，定型的な応用にしか適用できない．

- 送信者はメッセージの送信先を知っていなければならない．未知の相手への依頼や，一般公募に相当する通信ができない．
- 受信者はメッセージを到着順に受け取らなければ

† A Parallel Cooperation Model 'Cellula' Composed of 'Process+Field' Amalgams by NORIHIKO YOSHIDA and SHUJI NARAZAKI (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University).

†† 九州大学工学部情報工学科

☆ 現在 NTT ソフトウェア研究所

ならない．受信者の自律的な判断に基づく取捨選択ができない．

したがって，今後大きな発展が期待され，また要求される自律分散システムや協調型問題解決のような応用に対して，従来の通信機構は適当でない．より自由度の高い通信機構が必要とされる．

ここで，人間の知識処理のモデルとして活用されているだけでなく，高度な知識処理システムの基盤として有望視されているモデルに，いわゆる黒板モデル(blackboard model)[7),8)]がある．このモデルは基本的には，1つの黒板とこれをとりまく実行主体から構成される．この黒板とは，情報を一括して溜めておくプールのようなものであり，実行主体は，ここに情報を書き出したり，ここから情報を読み込むことで，自律的に情報の授受を行う．この意味で，黒板は場ないし環境とも呼ばれる．

黒板モデルは，受信者を特定しない送信，および受信者の自律的な取捨選択を実現する点で，上で述べたメッセージ通信の制約を克服するものであり，協調処理モデルの枠組の基盤としても有効である．ただし，全域的な場が唯一存在するという基本的なモデルは，次のような難点を持つ．

- 大規模なシステムの構築に際して，情報の衝突や不測の書換えなどの回避が難しい．
- すべての情報が唯一の場に含まれるため，情報の動的な検索において，そのオーヴァヘッドが大きい．

本論文では，以上の考察に基づき，協調型問題解決を主な対象とする並列協調処理モデルを提案する．本モデルの本質は次の2点にある[9)]．

① 実行主体としてのプロセスと通信媒体としての場を一体化し，これを階層的にネストさせる(他の協調処理モデルの多くにみられるような[10)]，プロセスと場を別個のオブジェクトとするアプローチはとらない)．

② パターン・マッチングを通信の基礎とし，通信の形式は通信命令の種別でなく，通信情報の属性で指定する（従来の並列処理モデルの多くにみられるような，通信形式ごとに別個の命令を用意するアプローチはとらない)．

これにより，従来の並列処理モデルや黒板モデルにはない，次のような特長が実現する．

- プロセス，場（黒板)，チャネル，メイルボックスなどがすべて，「プロセス＋場」という唯一のモジュールに統合される．
- 少数の統一化された命令のみで，直接／間接送信，放送，自律的受信，封鎖／非封鎖通信などの多様な通信が可能になる．通信形式の拡張も，整合性を保ったまま容易に行える．
- プログラム内だけでなくプログラム間の通信も，モデルの枠組の中で統一的に記述できる．
- 一般の問題解決手法だけでなく，分割統治法などの再帰的な手法への直接の対応が可能になる．
- 場を局所化することにより，上で述べた全域的な場の持つ2つの問題が解決される．

本並列協調処理モデルを，Cellula と名付ける．以下，第2章では，Cellula の基本要素として，プロセスと場を一体化したものとしてのセル，通信情報単位としてのタプルとその属性について順に述べ，多様な通信形態をそれらで実現する方法を示す．第3章では，ワークステーションのネットワーク・システム上に実装した，Cellula に基づく処理系について述べる．第4章では，Cellula による黒板モデルとその階層化の実現について述べ，その適用例として，分枝限定法による巡回セールスマン問題と分割統治法によるマンデルブロ図形生成について論じる．第5章はまとめであり，検討と今後の課題を含む．

## 2. 基本要素

### 2.1 セル＝プロセス＋場

プロセスと場を一体化した存在をセルと呼ぶ．すなわち，セルは実行主体として自律的に処理を進めるとともに，自分の内部に固有の通信情報プール（FIFO型のキューではない）を持ち，この中の任意の情報単位について書出し／読込みを行う．これを図式的に**図1**に示す．セルは Cellula に基づくシステムを構成する唯一の存在である．

セルは新たなセルを動的に生成することができ，子セルはその親セルの「内側」に置かれる．このネスト関係により，次の機構が同時に実現される．

① 親セルは子セルの視野すなわち知人関係の初期値を規定する．子セルは最初は自分の親しか知らない(親は子を知っている)．それらと情報を授受することにより，自身，親，子との直接通信が実現する．ま

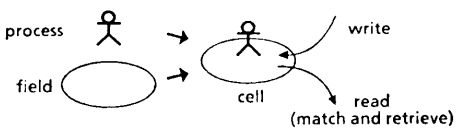


**図1** セル＝プロセス＋場
Fig. 1 Cell=Process+Field.

た，環境としての親を介して，兄弟との間接通信が実現する．さらに，セルそのものを他のセル（通常は親子など）に授受することもできるので，親子関係にないセルの存在を知って，それとの間で通信を行うことも可能である．すなわち，「紹介」してもらうことによって知人関係を広げていくことができる．

② 逆に，親セルは子セルの集団活動を抽象化する．すなわち，子セルの存在やそれらの協調関係を親セルの外側から直接知ることはできない．

③ 親セルは子セルに対して，間接通信や放送の媒体，すなわちいわゆる環境として機能する．セルは環境の中の情報単位についても，自律的な書出し／読込みによって授受を行う．

セルのネスト関係は再帰的であり，これにより再帰的な問題解決技法にも直接に対処することが可能になっている．詳しくは後述する．

セルを操作する命令は，次の5つである．

def-cell(Cell-Symbol, Cell-Code)
　セルのプログラム・コードを定義する*

create(Cell-Symbol, {Range})
　子のセルを生成して，その子セルを返す*

suicide() 自殺する

self()　そのセル自身を返す

env()　環境（親）のセルを返す

なお，create の引数の Range は一種のシステム指令であり（省略可能），子のセルを置くネットワーク・ノードを指定できる．具体的には，次のものがある（一部未実装）．

| | |
|---|---|
| domestic | 同じノード |
| same-domain | 同じネットワーク内のいずれかのノード |
| sub-domain | 下位ネットワーク内のいずれかのノード |
| "hostname" | 特定のノード |

この機構により，セルの階層を実際のネットワークの階層に対応付けることなどが可能になる．

### 2.2 タプルとその属性

セル間で通信される情報単位をタプル**と呼ぶ．タプルは名前と，0個以上のデータ要素の順序付けられた組から構成される．各要素は型を持ち，また，値を持つ actual かまだ持たない formal のいずれかである．次章で述べる現在の処理系では，タプル要素の型として，具体的に次のものを用意している．

| | |
|---|---|
| integer | 整数 |
| real | 実数 |
| complex | 複素数 |
| string | 文字列 |
| list | リスト |
| vector | ベクタ |
| cell | セル |

タプルの全要素の型の直積を，そのタプルの型と呼ぶ．

セル間の通信は，セルへのタプルの書出し／読込みによってなされる．1つの通信には，書出しを行う送信者セル，読込みを行う受信者セル，場を提供する媒体セルの3つが関与し，これらは同じ場合もすべて異なる場合もあり得る．ただし，媒体セルは具体的な通信内容について関知しない．タプルの読込みは，指定したタプルとのマッチングによってなされる．この指定したタプルをテンプレートと呼ぶ．受信者セルは媒体セルにテンプレートを書き込むことによって，読み込むべきタプルを探す．

タプルを操作する命令は，次の2つのみである．

out(Cell, Tuple, {Attribute}, {Exception})
　タプルをセルに書き出し，対応するテンプレートを返す（送信）．

in(Cell, Template-Tuple, {Attribute}, {Exception})
　テンプレートにマッチするタプルをセルから読み込み，それを返す（受信）

この2つはいずれもアトミック・アクションとして動作し，相互排斥を保証する．ここで，Attribute はタプルの属性，Exception は例外処理の指定である（いずれも省略時には既定値がとられる）．これらについては，後述する．

セルを介したタプルの授受は，基本的には次の手続きに則ってなされる．すなわち，セルに out 命令で書き出されたタプルとin命令で書き出されたテンプレートの中から，マッチするものを探す．マッチングの条件は，両者が同じ名前と型を持つ，および，同じ位置のデータ要素の値が等しいかいずれか一方が formal である，の2つである．そして，マッチングが成功したならば，テンプレートをそのタプルで具体化して（すなわち，テンプレート内の formal を対応するタプルの actual で置き換えて）in 命令の返り値とし，同

---

* オブジェクト指向で言えば，def-cell はクラス定義に，create はインスタンス生成に相当する．

** Linda[11] の用語を借用している．ただし，名前と属性を持つ点が異なる．

時にタプルをテンプレートで具体化して out 命令の返り値とする．なお，具体的にはタプル属性に応じた処理もなされるが，詳細は割愛する．

上で示したように，この2つのタプル操作命令はいわゆる交換関数にもなっている．従来の交換関数はその用途が必ずしも明確ではないが，ここではタプルの受信者を送信者に伝えるという用途に用いることができる．これは，in を実行したセル自身をテンプレートに埋め込むことにより実現する．

タプルは，通信形式を指定するために，次の3つの属性を持つ*．

① 封鎖／非封鎖：封鎖通信か非封鎖通信かを指定する．封鎖タプルはマッチするテンプレートが現れるまで送信者セルを待たせ，非封鎖タプルは待たせない．封鎖テンプレートはマッチするタプルが現れるまで受信者セルを待たせ，非封鎖テンプレートは待たせない（対応するものがなければ not-found フラグを返す）．

② 排他／非排他：1対1通信か1対多通信かを指定する．タプルとテンプレートのいずれかが排他ならばマッチングの後タプルが消滅するので，同じテンプレートで競合する in 命令の内の1つのみが有効となり，1対1通信が実現される．そうでなければ消滅しないので，競合する in 命令すべてがタプルを読み込み，1対多通信が実現する．なお，テンプレートはマッチングの後必ず消滅する．

③ 優先順位：同じ名前と型を持つタプル／テンプレートの間で，マッチングの最優先順位を指定する（優先順位の等しいタプルについては，古いものが優先される）．タプルの優先順位は競合する out 命令の順位を，テンプレートの優先順位は競合する in 命令の順位を意味する．

なお，既定値は各々，非排他，非封鎖，優先順位0（中間）である．

また，この2つのタプル操作命令では，指定したセルがなかった場合（すでに自殺している場合など）の処理を，例外処理として指定することができる．その既定値は「セルの自殺(suicide)」である．この機構により，あるセルの子孫すべてを自殺するようにしむけることなどが可能になる．

* タプルとテンプレートの区別も，出力／入力という属性と考えることができる．

### 2.3 様々な通信形態の実現

上で述べたような単純かつ統一的な機構の上で，セルの階層とタプルの属性を組み合わせることにより，次のような様々な形態の通信を実現することができる．なお，これらの通信形態の概略を，図式的に**図2**に示す．

（1） 直接通信

特定のセルへの直接通信は，相手のセルにタプルを書き出すことによって実現する．そのセルがタプルを（値の指定なしに）順に受け入れて読み込むならば，それはメッセージ通信に等しい．例えば，

送信：
out(Destination-Cell, Tuple)
相手セルでの封鎖受信：
in(self(), Template-Tuple, blocking)

（2） 間接通信

未知のセルへの間接通信は，相手を含んでいる環境（親）セルに排他属性のタプルを書き出すことによって実現する．この場合，その環境内のいずれか1つのセルがそのタプルを読み込む．例えば，

送信：
out(env(), Tuple, exclusive)
いずれかの兄弟セルでの受信：
in(env(), Template-Tuple, blocking)

（3） 放送

上の間接通信において非排他属性のタプルを用いるならば，マッチするすべてのセルがそれを読み込むため，放送が実現する．例えば，

送信：
out(Destination-Environment, Tuple)
対象セル（複数）での受信：
in(env(), Template-Tuple, blocking)

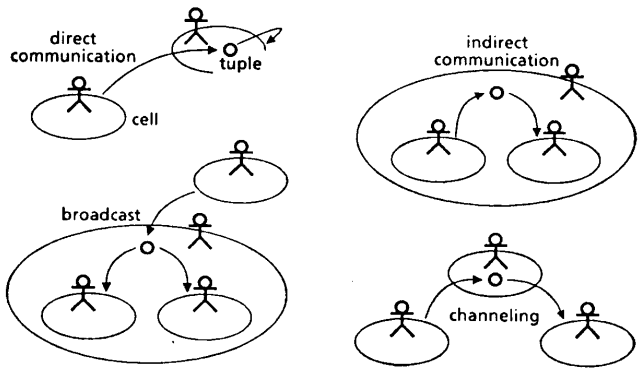


**図2** 様々な通信形態
Fig. 2 Various styles of communication.

(4) チャネル通信

あるセルを中間に介して通信を行うことによって，そのセルをチャネルと見なした通信が実現する．例えば，

送信：

out(Channel-Cell, Tuple)

受信：

in(Channel-Cell, Template-Tuple, blocking)

(5) 自律的受信

タプルを読み込む際に要素の型や値を指定することによって，それにマッチするタプルのみを自律的に選択する受信が実現する．

## 3. 処理系の実装

Cellula の実現性を検証するために，Unix ワークステーションのネットワーク上に処理系の実装を行った．これは具体的には，Sun-3 上で Hokkaido Common Lisp (HCL) を用いている．各プロセッサ上の処理系の構成を，**図3**に示す．

(1) 基本方針

ここでは，前章で示した Cellula 命令（セル操作5つ，タプル操作2つ）を，いわばシステム関数として HCL に追加する方式をとっている．タプルはリストで表現し，その属性の指定にはキーワード引数を用いて省略可能なようにしている．

(2) 並行処理機能

HCL は並行処理機能を持たない．そこで，マルチタスキング機能を持つ簡単なインタプリタを HCL 上に構築し，その上で Cellula のプログラムを解釈実行するようにしている．

(3) タプルの管理

タプルのマッチングの効率化のために，多段階ハッシュを用いている．すなわち，タプルをそのセル番号と名前に従ってハッシュする．さらに，タプルの型に応じてキューを用意する．マッチングはこのキューの上で行う．

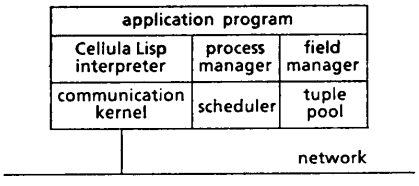


**図 3** Cellula 処理系の構成
Fig. 3 Configuration of the Cellula system.

(4) ネットワーク化

以上で述べた処理系を基礎に．ネットワーク上に分散処理系を実装している．分散化にあたって，セルを各々ノードに割り当て，ノード間の通信には TCP/IP のソケットを用いている．そして，システムコール・インタフェイスをCで記述して，ソケットを KCL のストリームとして扱えるようにしている．

## 4. 応　　用

### 4.1 黒板モデルとその階層化

Cellula は黒板モデルを基礎にしている．黒板モデル（および Cellula）では，実行主体の間の情報の流れは前もって静的に決定されるのではなく，各々が黒板上で自律的に読書きを行うことにより，動的に決定される．これにより，この1つのモデルの上で様々な処理形態が実現される．そのうちには，従来の並列処理における次のような処理形態も含まれる．

- パイプライン処理：ある実行主体の処理結果を他の実行主体がデータとして処理する．
- SIMD 処理：データが均質な集合の場合に，その要素を同一の実行主体のコピーが各々処理する．

この SIMD 処理において，各実行主体を別個のノードに置くことで負荷分散が実現される．各実行主体は，データ要素が黒板上にある限りそれを処理するという動作を繰り返す．これにより，データ数が実行主体数を越える場合にも，自動的に対処できる．さらに，データ集合の大きさが動的に変化するような場合にも，動的負荷分散が自動的に実現される．

黒板モデルに基づく最も単純なシステムは，1つの黒板，および1人の管理者と何人かの作業者として機能する実行主体から構成される．管理者は解くべき問題を黒板上に提示して，作業者の処理結果が揃うのを待つ．一方，作業者は黒板上の問題の全体ないし一部を読み込んでこれを処理し，結果を黒板に書き出す．より高度なシステムでは，管理者が作業者間の情報の流れに介入したり，ある作業者が他の作業者に対して管理者として振る舞うこともある．しかし，管理者と作業者は黒板に対して対等な関係にあり，黒板モデルに基づくシステムは平板な構成を持つ．

以前に提案した階層的黒板モデル (hierarchical blackboard model)[12]* は，管理者と作業者の関係に階層を導入したものである．すなわち，このモデルで

* 再帰的主体黒板モデル (recursive agent-blackboard model) とも呼んでいる．

は，ある作業者がより下位の作業者に対して管理者として振る舞い，ある管理者がより上位の管理者に対して作業者として振る舞うことができる．この黒板モデルの階層化は，分割統治法のような，問題を再帰的に分割していく手法への直接の対応を可能にする．平板な黒板モデルでは，これは不可能である．階層的黒板モデルについては，これに基づいてオブジェクト指向型言語の上にパターン認識システムを構築し，知識処理への適用性を確認している[12]．

上で述べたような黒板モデルに基づくシステムをCellula で構築しようとする場合，黒板と管理者は1つのセルに一体化され，その子セルが作業者として機能する．したがって，管理者／作業者関係がセルの親子階層に反映する．すなわち，Cellula は階層的黒板モデルを直接に具体化するものである．管理者／作業者の階層化を実現する Cellula のプログラム骨格の例を，**図4**に示す．なお，ここでは Algol 風の表記法を用いている．

Cellula の有用性と適用性を確認すべく，食卓の哲学者の問題や行列乗算などについて，先に述べた処理系の上でプログラムを作成して実験を行った．ここでは Cellula に基づく記述例として，分枝限定法による巡回セールスマン問題，および分割統治法によるマンデルブロ図形生成を取り上げる．前者は協調型問題解決において問題集合が動的に変化する例であり，後者は階層的黒板モデルにおける再帰的問題分割の例である．以下に節を改めて，この2つについて述べる．

### 4.2 分枝限定法による巡回セールスマン問題

分枝限定法[13]は，問題解決における解の探索を効率化するための手法の1つであり，その本質は，何らかの評価関数と閾値に基づく探索木の枝刈にある．

分枝限定法の並列化については，密結合（共有メモリ結合）マシンおよび疎結合（バス結合）マシンの上で様々な提案がなされている[14]．代表的な並列化法の1つとして，探索の初期段階で探索木をプロセッサ台数分に分割して，それ以降は各プロセッサで独立に探索を進めるというものがある[15]．しかし，この方法は，負荷の不均衡が避けられない，閾値の更新を全プロセッサに放送する必要が生じる，という難点を持つ．ここで，黒板モデルを用いることにより，動的負荷分散と閾値更新の効率化が実現する．

ここで取り上げる巡回セールスマンの問題[16]とは，簡単に言えば，「距離の定義された辺からなるグラフの上で，すべての節点を一巡する経路のうち，総距離が最小のものを求める」という問題であり，代表的なNP 完全問題である．

分枝限定法で巡回セールスマン問題を解く Cellula のプログラムは，次のタプルを扱う．

- candidate：探索木の枝のうちでまだ有効な候補を表す．この個数は実行に伴って変化する．
- threshold：枝刈の基準となる閾値を表す．この値はすべての作業者から参照され，適宜更新される．これをタプルとして場の中に置くことで，放送は自動的に実現される．
- result：探索の結果として解があったか枝刈がな

```
def-cell staff:
    function manage: begin                          % 管理者としての機能
        loop create(staff) end;                         % (必要なだけの) 子セルの生成
        out(self(), Global-Data-Tuple);                 % (必要ならば) 大域データの初期化
        loop out(self(), Data-Tuple, exclusive) end;    % データの分配
        loop in(self(), Result-Tuple, blocking) end;    % 結果の収集
        in(self(), Global-Data-Tuple, exclusive)        % ゴミの消去
    end;
    function work: begin                            % 作業者としての機能
        in(env(), Global-Data-Tuple, blocking);         % 大域データの読込み
        loop
            in(env(), Data-Tuple, blocking);            % データの読込み
                ......                                  % 何らかの処理
            call manage;                                % (必要ならば) さらに下位への委譲
            out(env(), Data-Tuple, exclusive);          % (必要ならば) データの更新
            out(env(), Result-Tuple, exclusive)         % 結果の書出し
        end
    end;
begin call work end.
```

**図4** 階層的黒板モデルのプログラム骨格
Fig. 4 Program skeleton for the hierachical blackboard model.

されたかを表す．これは作業者から管理者への報告のためのものである．

そして，次の2種類のセルを持つ．

• Manager：初期設定および後始末を行う管理者．タプルはこのセルの中に置かれる．

• Searcher：探索木の枝を伸ばす，ないし刈ることで探索を行う作業者．このセルは同一のもののコピーが複数個用意される．そして各々は，candidate を自律的に1つ読み込み，探索を進めて0個以上の新たな枝を生成し，それらを candidate として書き出すという動作を，candidate がある限り繰り返す．すなわち，問題集合の大きさは動的に変化するが，負荷の不均衡が生じることはなく，動的負荷分散が自動的に実現される．

このプログラムの構造を，図式的に**図5**に示す．

### 4.3 分割統治法によるマンデルブロ図形生成

マンデルブロ (Mandelbrot) 図形ないし集合[17]とは，次の条件を満たす複素数 $c$ の集合である．

$z_0=0$, $z_{n+1}=z_n^2+c$ の時

$|\lim_{n\to\infty} z_n|<\infty$

この集合を複素平面上にプロットしたものは，自己再帰的すなわちフラクタルな図形になる．この図形を生成する際には $n\to1000$ 程度とするのが普通であるが，それでも多量の計算を要する．ここで，分割統治法を応用して長方形領域を再帰的に分割していく手法が，近似解法として有効である．これは，領域の境界上の点について値を計算し，すべて等しいならば領域全体が集合内／外にあるとみなし，そうでないならばその領域を4分割して同じ手続きを再帰的に適用する，というものである．

分割統治法でマンデルブロ図形生成を行う Cellula のプログラムは，次のセルを持つ．

• Rectangle：上で述べた処理を行う作業者．このセルは，領域分割の度に，自身のコピーを4つの子セルとして動的かつ再帰的に生成し，これらの子セルが親セルの内側で並列に動作する．親セルは管理者として機能する．そして，未処理領域の情報と処理結果が，タプルとして親子セル間で授受される．このプログラムの構造を，図式的に**図6**に示す．

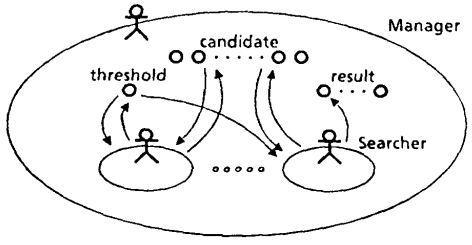


**図5** 分枝限定法を実行するプログラムの構造
Fig. 5 Program structure for a branch-and-bound method.

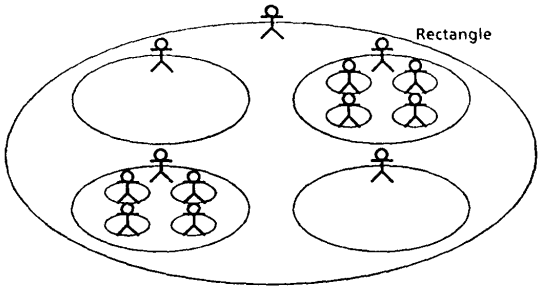


**図6** Mandelbrot 図形を生成するプログラムの構造
Fig. 6 Program structure for generating a Mandelbrot chart.

## 5. おわりに

本論文では，協調型問題解決法を主な対象とする並列協調処理モデル Cellula を新たに提案し，その有効性や適用性について論じた．以下に，Cellula に関して幾つかの検討を加える．

（1） 関連する他のモデルとの比較

• Linda[11]：これは場を介した通信に基づく並列処理モデルであり，アーキテクチャ独立を標榜している．しかし，通信形式についての本質的な整理がなされておらず，通信形式ごとに個別の命令を用意しているために煩雑である．また，階層性を持たないために再帰的な手法に適さない．

• Kamui[18]：これは並行オブジェクト指向モデルに場を導入したものである．しかし，受信者によるメッセージの自律的な取捨選択を許さない．また，階層性を持たないために再帰的な手法に適さない．これは Ether[19] も同様である．

• 分散オブジェクト指向言語上のタプル空間[20]：Linda で言うタプル空間をオブジェクトで実現した多重タプル空間モデルであるが，集団活動の抽象化のような概念がない．

他の並列処理モデルや協調処理モデルと比較した場合の Cellula の特長は，以下のとおりである．

• (Hearsay-Ⅱ[21] に見られるような) 黒板の論理的な階層化や分割を，モデルの上で直接に支援する．

• 一般の問題解決手法だけでなく，分割統治法などの再帰的な手法をも直接に支援する．

●集団活動の抽象化を支援する．

（2） 通信形式の指定

Cellula では，従来の並列処理モデルと異なり，1対1／1対多通信や封鎖／非封鎖通信といった通信形式を，命令の種別でなく通信情報であるタプルの属性で指定する．これにより，通信命令を単純かつ対称な形で構成できるとともに，通信形式の拡張を，属性の追加により，単純性や整合性を保ったまま行うことができる．拡張の例としては，一定時間が経過すると消失するタイムアウト・タプルなどが考えられる．

（3） プログラム間の通信

実際の並列処理システムの構築に際して，従来の並列処理モデルや言語の多くでは1つのプログラムの中での通信しか記述できず，他のプログラムやシステムとの通信を実現するには，モデルや言語の枠組とは独立にオペレーティング・システムの機能を借用するしかない．

一方，Cellula では，セルは親を介して自身の兄弟と間接的に通信することができる．したがって，プログラムの最上位セルのさらに親として仮想的なセルを用意することにより，プログラムが他のプログラムと通信する機能を，その仮想的な親セルを介するという形で，モデルの枠組の中で整合性を崩すことなく提供することができる．現在の処理系では，プログラムのこのような仮想親として root というシステム・セルを用意している．ライブラリなどは，この root の中で提供される．

（4） 場に含まれる情報の検索

Cellula の処理系では，セル番号およびタプルの名前と型に基づく検索空間の分割を行っている．したがって，全域的な場を持つ基本的な黒板モデルに比較して，実行時の検索オーヴァヘッドの低減が実現している．これは大規模なプログラムにおいて顕著になるはずのものである．

なお，現在の処理系はインタプリタ方式のため，コンパイラによる最適化が活用できず，マッチングそのもののオーヴァヘッドは低減していない．さらには，ネットワーク通信のオーヴァヘッドも大きい．したがって，セルおよびタプルの粒度は粗い方が有利である．マッチングの高速化については，プロダクション・システムにおける Rete ネットワークの技法[22]などが有効と考えられる．

（5） 分散処理系でのセルへのアクセス

現在の処理系では，セルを各々ネットワーク・ノードに割り当てている．したがって，作業者セルを持つノードの間では動的負荷分散が実現するが，管理者セルを持つノードへのアクセスの集中が生じる．これの回避については，仮想共有メモリ[23]などと同様な技法が有効と考えられる．

（6） 今後の課題

現在は，インスペクタなどのシステム開発／実行環境の整備を進めるとともに，Cellula への継承機構の導入について検討中である．今後は，Cellula の適用性について分析を続け，マルチエイジェント・システムや分散意思決定支援システムなど，より実際的な問題への応用を試みていく予定である．さらに，このモデルに基づく処理系をマルチプロセッサ・システム上に実装して，実行効率の評価などを行うことも考えている．

**謝辞** 日頃からご支援を頂く九州大学牛島和夫教授に深謝する．

## 参考文献

1) 「分散と協調」特集号，計測と制御（計測自動制御学会誌），Vol. 26, No. 1 (1987).
2) 内木，丸一，所：行動シミュレーションに基づいたアニメーション・システム Paradise，コンピュータ・ソフトウェア，Vol. 4, No. 2, pp. 24-38 (1987).
3) Ihara, H. and Mori, K.: Autonomous Decentralized Computer Control Systems, *IEEE Comput.*, Vol. 17, No. 8, pp. 57-66 (1984).
4) Decker, K. S.: Distributed Problem-Solving Techniques: A Survey, *IEEE Trans. Sys. Man and Cyber.*, Vol. SMC-17, No. 5, pp. 729-740 (1987).
5) Huhns, M. N. (ed.): *Distributed Artificial Intelligence*, Morgan Kaufmann (1987).
6) Andrews, G. R. and Schneider, F. B.: Concepts and Notations for Concurrent Programming, *ACM Comput. Surv.*, Vol. 15, No. 1, pp. 3-43 (1983).
7) Lindsay, P. H. and Norman, D. A.: *Human Information Processing*, Academic Press (1977).
8) Engelmore, R. and Morgan, T. (eds.): *Blackboard Systems*, Addison-Wesley (1988).
9) 吉田，檜崎：場と一体化したプロセスの概念に基づく並列協調処理モデル，電子情報通信学会技術研究報告，CPSY 89-19 (1989).
10) Filman, R. E. and Friedman, D. P.: *Coordinated Computing*, McGraw-Hill (1984). (雨宮，尾内，高橋訳，協調型計算システム，マグロウヒル (1986).)

11) Carriero, N. and Gelernter, D.: Linda in Context, *Comm. ACM*, Vol. 32, No. 4, pp. 444-459 (1989).
12) Yoshida, N. and Hino, K.: An Object-Oriented Framework of Pattern Recognition Systems, *Proc. OOPSLA '88*, San Diego, pp. 259-299 (1988).
13) Lawler, E. L. and Wood, D. E.: Branch-and-Bound Methods: a Survey, *Oper. Res.*, Vol. 14, No. 4, 699-719 (1966).
14) Wah, B. W. and Ma, Y. W. E.: NANIP—a Multi-computer Architecture for Solving Combinatorial Extreme-Search Problems, *IEEE Trans. Comput.*, Vol. C-33, No. 5, pp. 377-390 (1984).
15) 吉田：並列処理記述言語の研究，東京大学大学院工学系研究科修士論文 (1981).
16) Bellmore, M. and Nemhauser, G. L.: The Traveling Salesman Problem: a Survey, *Oper. Res.*, Vol. 16, No. 3, pp. 538-558 (1968).
17) Dewdney, A. K.: Computer Recreations, *Sci. Am.*, Vol. 253, No. 8, pp. 8-14 (1985).
18) 渡辺，原田，三谷，宮本：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, pp. 41-55 (1989).
19) Kornfeld, W. A. and Hewitt, C. E.: The Scientific Community Metaphor, *IEEE Trans. Sys. Man and Cyber.*, Vol. SMC-11, No. 1, pp. 24-33 (1981).
20) Matsuoka, S. and Kawai, S.: Using Tuple Space Communication in Distributed Object-Oriented Languages, *Proc. OOPSLA '88*, San Diego, pp. 276-284 (1988).
21) Erman, L. D. et al.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty, *ACM Comp. Surv.*, Vol. 12, No. 2, pp. 213-253 (1980).
22) Forgy, C. L.: Rete: a Fast Algorithm for Many Pattern/Many Object Pattern Match Problem, *Artif. Intell.*, Vol. 19, No. 1, pp. 17-37 (1982).
23) Li, K.: Shared Virtual Memory on Loosely Coupled Multiprocessors, Ph. D. Thesis, Yale Univ. (1986).

(平成元年8月31日受付)
(平成2年4月17日採録)

**吉田 紀彦** (正会員)

1957年生．1979年東京大学工学部計数工学科卒業．1981年同大学大学院修士課程修了．(株)三菱総合研究所勤務，東京大学工学部助手を経て，現在，九州大学工学部情報工学科助手．工学博士．プログラミング方法論，並列／分散／協調処理，人工知能などに興味を持つ．ソフトウェア科学会，計測自動制御学会，ACM, IEEE, AAAI各会員．

**楢崎 修二** (正会員)

1965年生．1988年九州大学工学部情報工学科卒業．1990年同大学大学院修士課程修了．現在，NTTソフトウェア研究所勤務．並列／分散／協調処理，記号処理などに興味を持つ．ソフトウェア科学会会員．