プログラミング－言語・基礎・実践－ 5－10
(1991. 11. 22)

行動ベースエージェントによるマルチエージェントシステム

奥乃 博
日本電信電話 (株) ・ 基礎研究所

大里 延康
NTT グループ事業推進本部

「たくさんの細胞が協力しあって，効果的な情報処理をするこの方法は，論理的というより，何か直観的理解と似ています．これに反して，単細胞原生生物の生き方は論理的理解に通じます．」
清水 博 [13]

あらまし 本稿は，出現的計算 (emergent computation) や直観的応答をマルチエージェントシステムによって記述し，実現するのを目的とする．このために従来のような機能分割によるシステム構築法ではなく，Brooks が提案した行動ベース分割による包摂アーキテクチャを取り上げる．具体的な行動を担うエージェントを積み上げ，総体として出現的計算や直観的応答を行わせることを狙う．いくつかの応用を再構築することによって，マルチエージェントシステムの新しいパラダイムを探るための問題点・課題を指摘する．また，包摂アーキテクチャと「オーバラップする近傍系」を標榜するコネクティクスとの関連についても議論する．

Multi-Agent System with Behavior-Based Agents

Hiroshi G. Okuno
NTT Basic Research Laboratories
okuno@pooh.ntt.jp

Nobuyasu Osato
NTT Affiliated Companies Headquaters
osato@nuesun.ntt.jp

3-9-11, Midori-cho, Musashino, Tokyo 180 JAPAN

Abstract We investigate multi-agent systems to model and implement emergent computations and reactive responses. As a system architecture, we adopt the subsumption architecture proposed by Brooks. The overall computation is carried out by a set of agents each of which does concrete work by communicating with other agents or the world. Based on this approach, we reorganize various systems such as robot control, human behavior in a crowded corridor and natural language processing to investigate the possibility of the subsumption architecture for general purpose framework.

Keywords Multi-agent system, subsumption architecture, Connectics, emergent computation, reactive computation, robot control, natural language processing

1 はじめに

我々は現在**コネクティクス** (*Connectics*) という新しいパラダイムの下で, 複雑で大規模な相互作用をするシステム — 我々は「**社会**」(*Society*) と呼ぶ — の理解と設計を目的に研究をすすめている [16]. コネクティクスでは, 社会をいわば, マクロスコピックとミクロスコピックとの中間に存在するメソスコピック・システムとして捉え, 「**オーバーラップした近傍系**」(*overlapping neighborhoods*) として定義する. コネクティクスが研究の対象とする社会のもつ特徴を以下にまとめる.

- 大規模かつ複雑である.
- 散層構造 (非階層構造) をなしている.
- 要素が非均質である.
- 関係が非晶質であり, メソスコピックな性質が重要である.
- 開放的で動的である.
- 自己組織化され, 自己保持 (成長) 性がある.
- 非決定性があり, 準最適である.

コネクティクスの研究対象としては, 経済学, 社会学, 心理学, 自然科学, 工学等多岐に渡っている. 現在, 特に社会心理学, コンピュータネットワーク, 分散システムなどの研究が進められている. 本稿では, コネクティクスの立場からマルチエージェントシステムの可能性を追求する.

マルチエージェントシステムは, エージェントがお互いにコミュニケーションを進めながら, ある仕事を行うシステムである. エージェントシステムとしては様々なシステムが知られている (例: [14]). 我々は, エージェントを独立した意味のある仕事をする最小単位のものと定義する. これは, Minsky が「心の社会」[8] の中で使用したものと同じである. 本稿では, エージェントのグループを近傍系と捉え, それらのグループがオーバーラップしながら局所的なコミュニケーションによって, 総体としてある仕事が行われるようなシステム構成法を検討する. 個々のエージェントの行動は決定論的であるが, 総体の挙動が陽には記述されていないときに, 総体としての挙動を求めるいう計算メカニズムは, 「出現的計算」(*emergent computation*) [5] と呼ばれており, 分散処理で今後重要な課題となっていこう.

本稿で取り上げたシステム構成法は, 従来の機能別分割法によるシステム構成法ではなく, 行動ベース分割法によるシステム構成法である. すなわち, エージェント間の相互作用 (interaction) を行動ベースから解析し, 行動を担うエージェントを積み上げることによってシステムを構築する手法である. 本稿では, Brooks が自律型移動ロボットの研究で提案した行動ベース分割法によるシステム構成法である「**包摂アーキテクチャ**」(*subsumption architecture*) を新しいパラダイムを検討する出発点としている. 包摂アーキテクチャをプログラミングシステムの観点から論じ, さまざまな問題を包摂アーキテクチャによって再構成し, 新しいパラダイムの可能性を探る.

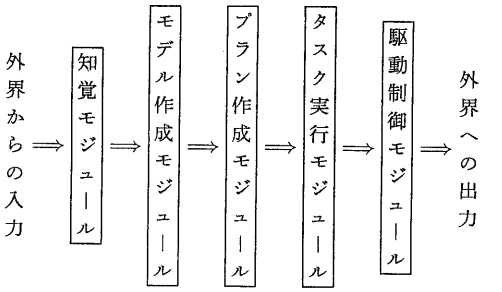


図 1: 従来の機能別分割法

本稿の構成は以下の通りである. 第 2 章で包摂アーキテクチャについて説明し, 第 3 章でプログラミングシステムの観点から考察する. 第 4 章で, マルチエージェントシステムを提案し, 第 5 章でいくつかの例題を説明する. 第 6 章で, まとめを行う.

2 包摂アーキテクチャ

2.1 機能別分割によるシステム構築法

自律的移動ロボットの行動には, 徘徊, 探索, 物体の同定, 障害物の回避, 地図作成, 外界への行動プランの作成, 外界への行動プランのモニター, オブジェクトの行動の理由付けなどがある. このようなロボットの行動を実現するためのアプローチとしては, 従来から機能別分割法が使用されてきた. まず, 全体の問題を, 知覚, モデル作成, プラン作成, タスクの実行, そして駆動制御といった一連の機能モジュールに分割する. 次に, 集中型制御システムが分割された各機能モジュールを実行し, その結果を決められた順序で, 緊密に結合された次の機能モジュールへと次々に同期をとって渡していく (図 1). 例えば, 知覚モジュールは世界をセンスし, センサの結果をモデル作成モジュールに渡す. モデル

作成モジュールは，知覚した世界の内部モデルの作成を企てる．このモデル作成で得られた内部モデルがプラン作成モジュールに渡され，そこでプランが計算される．このプランが次にタスク実行モジュールに渡され，タスク実行モジュールはプランを解釈して，駆動制御モジュールにコマンドを与える．駆動制御モジュールは，与えられたコマンドを基に外界に対してロボットを動かす．

機能別分割法によるロボットの行動は，*sense-model-plan-act* という枠組で実行される．この SMPA 方式では，外界の知覚と世界モデルを構築するのに大部分の時間がかかり，プラン作成やタスク実行にはそれほど時間がかからない．したがって，SMPA 方式のロボットは，長考の後，少し動くという緩慢な動作を繰り返すことになる [3]．また，機能拡張のためには各モジュールに手を加えなければならない．さらに，外界と直接やり取りするのはごく小数の機能モジュールだけであるので，ロボットの行動がどう実行されるのかが見えにくいという欠点もある．

2.2 行動ベースによるシステム構築法

従来の集中型制御方式に代わる方法として，行動を担当する非同期型タスクの集合に分解する方法がある．この方法は**包摂アーキテクチャ** (*subsumption architecture*) と呼ばれ，比較的基本的なタスクを担当する行動モジュールが非同期的に局所的な相互作用 (interaction) を行い，その結果，総体のとしてロボット全体の制御が行われる．包摂アーキテクチャでは，各モジュールはお互いあるいは外界と直接コミュニケーションを行う．図 2に，自律型移動ロボットの行動による分割を示す．図から分るように行動は層に分けられている．新たな行動や機能の追加は，上位のレベルを追加し，低位レベルの行動をその上位レベルの行動で置き換えることによって実現する．たとえば，Level 0 である衝突回避レベルでは，障害物に当たったときには止まり，障害物が前方に現われたときには方向を変えることによって，衝突を防ぐようにロボットは制御される．Level 1 の徘徊レベルでは，ロボットは衝突しないで動き回れるように制御される．すなわち，ロボットの進行方向は，Level 0 から衝突回避の情報が得られたときには Level 0 からの進行方向を取り，Level 0 からの情報がないときには Level 1 で決めた進行方向を取るようにする．

包摂アーキテクチャの特徴は，以下の 3 点にある．

(1) 行動の層を積み上げることによって，ロボットを知的にできる．

(2) 直接外界とコミュニケーションすることによって，状況に応じた行動を取ることができる．

(3) 個々の行動がモジュールによって実現されているので，ロボットの行動が具現化し，理解がしやすい．

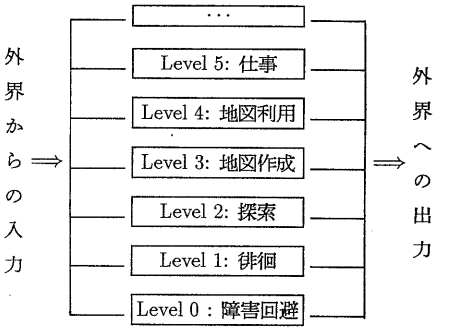


図 2: 自律型移動ロボットの行動による分解

包摂アーキテクチャでは，各行動を担当するタスクは局所的にかつ非同期的に実行され，相互にはきわめて疎にしか結合していない．行動を担当するタスクの各々は，外界あるいは相互に直接コミュニケーションを行う．さらに，包摂アーキテクチャでの行動を担当するタスクは機能別分割法による機能モジュールと比べ，はるかに基本的な機能しか有していない．言い替えると，包摂アーキテクチャでは，行動を担当する個々のタスクは，全体問題を自分が扱える，より単純な部分問題に縮約して，それを解く能力を有していることになる．

機能別分割法は，さまざまなセンサからの情報を統合して世界モデルを構築する**センサフュージョン** (*sensor fusion*) であるのに対して，行動ベース分割法はさまざまなセンサからの情報をそのまま各エージェントに渡す**センサフィッション** (*sensor fission*) である．

2.3 拡張型有限オートマトン (AFSM)

ロボットを制御するためのアーキテクチャは，**拡張型有限オートマトン** (*augmented finite state machine, AFSM*) の集合で規定される [2]．各拡張型有限オートマトンは，

(1) 入力メッセージが入るレジスタ類 — 入力があると古いメッセージはオーバライトされる．

(2) 時計，タイマ，等

(3) 出力線

(4) 有限オートマトン

から構成される．入力線には，センサか他の拡張型有限オートマトンの出力が接続され，出力線は，ロボット駆動器か他の拡張型有限オートマトンの入力に接続される．接続には，特別なゲートを経由させることができる．入力線には，入力抑制ゲートが付加でき，上位レベルからのメッセージによって，入力が入らないようにできる．出力線には，出力削除ゲートが付加でき上位レベルからのメッセージによって，出力を削除し，上位レベルの出力で置換することができる．前節で説明した自律型移動ロボットでは，探索モジュールによる進行方向の決定がその下位レベルの徘徊モジュールによる進行方向の決定よりも優先するのは，出力線に付加した出力削除ゲートによって実現される．また，何かに衝突したときにすぐに停止できるのは，最下位レベルの衝突回避モジュールからの停止信号が出力削除ゲートを介さないようにして衝突回避モジュールからロボット駆動器に接続されているからである．

3 プログラミングとしての包摂アーキテクチャ

包摂アーキテクチャをプログラミングシステムとして見たときの特徴を以下にまとめる．

- **コンカレントプログラミング**

  各々のレベルのタスクが複数のゴールをコンカレントに追求するので，並列性がある．また，同じレベル内での複数エージェントをコンカレントに実行出来るので，さらに並列性を引き出すことが出来る．最終的にどのタスクが実行されるかは，それぞれのタスクの結果を持ち寄って，調整されるので，事前にどのタスクを選択するのか決定する必要はない．

- **実時間プログラミング**

  複数のゴールが同時追求され，そのときどきの状況によって，取るべき行動が決定される．つまり，ゴールを追求する複数のタスクが状況に相応しい結果を出すように競争する．このようなシステムは，梅村氏の言葉を借りれば，*Computer Supported Competitive Work* (CSCW) と言える．このように動的な判断が下せるという能力は，実時間処理，特に実時間 AI にとって重要な機能である．判断を下すのに時間がかかり応用分野が限定されていたエキスパートシステムが活用出来る範囲が大幅に拡大出来ると予想される．

- **拡張可能性**

  新たなレベルを追加すれば，新たな機能を追加することができる．さらに，レベル間のコミュニケーションはそれほど大きくないので，新たなレベルを新たなプロセッサに割り当てることによって，それまでの処理に影響を与えないようにすることができる．すなわち，このアーキテクチャは疎結合型のマルチプロセッサにも適用可能である．

- **段階的プログラミング**

  包摂アーキテクチャでは，各レベルでのタスクはそれ自身である行動を行うことができるので，新たなタスク (行動) を追加するときには，既存のレベルのタスクを変更する必要が全くない．

- **フォールトトレラントプログラミング**

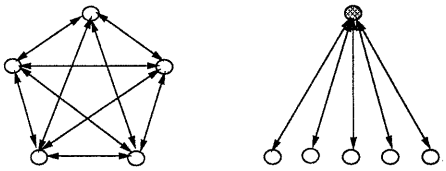
  ある行動を担当するタスクが失敗しても，時々あるいは完全にシステムが停止したりせず，全体としての実行速度が遅くなったり，機能が低下したるするだけである．

- **複数の入力 (センサ) の同時的処理**

  従来のシステムでは，センサからの入力を内部表現に変換し，さらに，複数のセンサからの入力を統合するような処理が必要であった (センサフュージョン)．しかし，必要な情報はタスクによって異なるので — 非常に詳しい情報が必要な場合もあれば，信号が変化したかどうかの情報だけで十分な場合もあろう —，包摂アーキテクチャでは，各タスクが必要に応じてセンサ情報を使用することができる (センサフィッション)．

- **頑健性 (ロバストネス)**

  複数のセンサを使用し，各タスクが必要に応じてその情報を利用するセンサフィッションの結果がうまく活用されるとシステムの頑健性は向上する．さらに，包摂アーキテクチャによってもシステムの頑健性は向上する．というのは，下位レベルのタスクが上位レベルが加わっても常時働き続け，その出力が上位レベルの出力で置き換えられて抑圧されるだけである．したがって，上位レベルがタイムリな結果を出すことができなければ，下位レベルのよくデバッグされ信頼性の高い出力が使用される．もちろん，上位レベルの出力ほど適切ではない十分にありうるが，不作為による手遅れよりははるかによい．



(a) 完全結合　　(b) 等価な場

図 3: 相互作用と場との同値関係

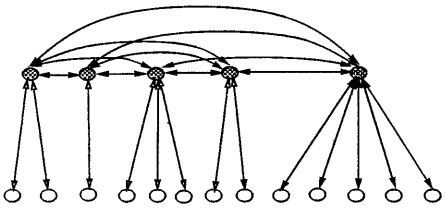


図 4: クラスター化された場

4　マルチエージェントプログラミング

Minsky の「心の社会」[8] では, エージェントは「心の一部分とか, 心の中の一プロセスであって, それ自体だけで理解できるような, 十分に単純なもの」と定義されている. また, エージェントのグループは「エージェンシー」と呼ばれ, その構成部分であるエージェントそれぞれが何をするかではなく, 全体として何ができることだけを問題にする. エージェントあるいはエージェンシが集まって, 心の社会を構成している.

我々は, エージェントは, 独立した役割を持つ包摂アーキテクチャを構成する基本単位と定義する. これは, Minsky の定義と同じく, 極めてミクロな立場にある. エージェントを個々の人, ロボット, あるいは, プロセス (スレッドやメソッドのような計算単位ではなく, 保護単位としてのプロセス) と定義する立場とは異なっている.

マルチエージェントプログラミングでの研究課題は,

- エージェントの表現方法.
- 出現的計算を記述方法.
- オーバラップするエージェントグループの表現.

オーバラップするエージェントグループの表現を考える前に相互作用 (コミュニケーション) について考察する.

4.1　相互作用としての場

外界との直接のコミュニケーションあるいはエージェント間のコミュニケーションは, 別の概念で置き換えることも可能である. 我々は場 (field) という考えをそのために導入する.

エージェント間の相互作用は, 場との相互作用と置き替えることができる.

図 3 に示した (a) はノード数が 5 の完全結合のグラフである. 完全結合では, どの 2 つのノード間でも相互に情報をやりとりすることができるから, 情報の場を仮想的なノードと考えると, 同じ図の (b) と等価であることが分る.

外界との直接の相互作用は, 図 3 に示した (b) に相当し, 外界を場として捉えることは自然である. しかし, エージェント間の相互作用はどのようなエージェントの組合せに対しても可能であるという大域的である必然性はない. 通常は, ごく限られたエージェントのグループだけでコミュニケーション (相互作用) が行なえれば十分なことが多い. このようなエージェントのグループは固定しているのではなく, 計算が進むにつれて, あるいは状況に応じて動的に変化していく. したがって, グループを仮想的なノードと捉え, グループ間の相互作用を仮想ノード間の相互作用として捉える (図 4). 場を表現する仮想ノード間の相互作用は, 同じように仮想ノードを仮想化した「仮想仮想ノード」で表現することもできよう. しかし, 多段の仮想化は, 包摂アーキテクチャの具現化 (embodiment) という性質を破壊するので, 我々は多段の仮想化はとらない.

場の概念に加えて, 場に熱という概念を導入し, それがエージェント間相互作用の強さを規定させるようにする. 場の温度が高いとは, 相互作用が強いことを意味し, 温度が低いと相互作用が弱いことを意味する.

エージェントと場との相互作用, 場の間での相互作用という考えは, コネクティクスの主張するオーバラップする近傍系という考えそのものである. このようなフレームワークを実現する計算モデルとして我々は NueLinda モデル [1, 9] を提案している. NueLinda は, Linda モデルをコネクティクスの考えを取り入れて拡張した分散処理のための計算モデルである. Linda モデルでは, 複数のプロセスが共有する論理的なデータ空間であるタプル空間は単一である. それに対して, NueLinda ではタプル空間をクラスター化し, クラスタータプル空間上での演算を提供している. さらに, クラスター

タブル空間のためのリフレクティブインタプリタを定義いている. 本稿で提案するマルチエージェントシステムは, 場の温度の除いては素直に実現できる.

4.2 エージェントの表現法

エージェントは, 以下の情報を持つ.

- 入力バッファ — センサからの入力, 他のエージェントからの入力, 場からの入力
- 出力線 — 外界への行動を行うエージェント, 他のエージェントへの出力
- 内部変数 — 内部状態
- アラームクロック — 時間の割込み
- 状態遷移 — 入力ディスパッチ, 割込みディスパッチを状態遷移表で与える.

さらに, エージェントへの入力および出力の接続関係を規定する情報が付加される. Brooks が使用している接続関係,

(1) 削除 (Inhibitor) — 上位レベルからトリガーがかかるとここから先には情報が流れず, 入力が削除される.

(2) 置換 (Suppressor) — 下位レベルの出力が削除され, 上位レベルでの出力で置きかえれる.

(3) 暗黙値 (Defaultor) — 上位レベルからトリガーがかかると, 暗黙値が出力される.

をそのまま使用する.

場の表現は, 特に設けていない. 場はシステムインタプリタによって, 各エージェントの入力として与えられる. したがって, 複数の場を提供することが出来る. 場の温度は, 場からの情報を入力に与えるときのパラメータとして使用される.

5 包摂アーキテクチャによるシステムの再構成の例題

5.1 ロボットプログラミング

ハノイの塔の問題を実際に解くロボット [11] を包摂アーキテクチャで再構成してみよう (図5). 3本のポールがあり, あるポールに積み重ねられた3枚のディスクを別のポールに移す作業を行う. ディスクはロボットが持ちやすいようにシリンダ状をしており, その穴はポールよりわずかに広いだけである.

ロボットは, 2種類のセンサを持つ.

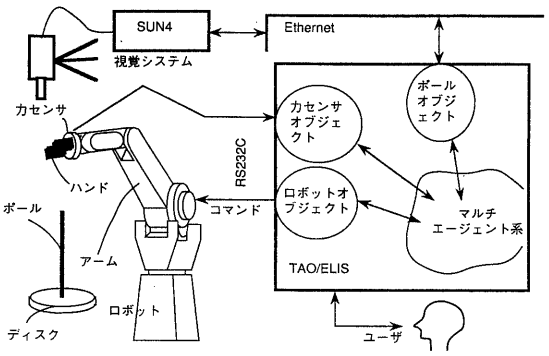


図 5: ロボットの構成図

(1) 視覚センサ — カメラからボールのおおまかな2次元の位置を得る.

(2) 力 / トルクセンサ (force-torque sensor) — ロボットの手首に装備された力センサによってハンドが何かにぶつかったことを知る.

ロボットは TAO/ELIS システム [15] と RS232C で接続されており, これを介してロボットに運動コマンドを送るとともに, ハンド位置などの状態の情報を取得する.

包摂アーキテクチャによるいくつかの動作の構成を詳しく見てみよう.

ディスクの穴をポールに入れるときの動作 :

- **ハンドの移動 − 層**: 視覚センサから得られた2次元座標をもとにハンドをポールの位置まで移動し, ディスクの穴にポールを入れるためにハンドを下げる. 反力センサとハンドの高さからポールの先頭が穴にはいったことが分ると, ディスクをポールに押しさげる動作へ移る.
- **横方向へのゆらし − 層**: ポールがディスクの穴の位置に合っていないときには, 押し下げるとディスクに上向きの力がかかり, それがハンドの反力センサから感じられるので, この層が起動される. ポールの位置に穴が来るように, 水平方向に小さな同心円を書くように軽くディスクをゆする. ロボットの制御を簡単にするために, 実際には, 同心円ではなく, 正方形を書くように水平方向に順々に動きを繰り返す. ハンドからの反力がなくなると, 終了する. その結果, 中断されていた押し下げ動作が再開される.

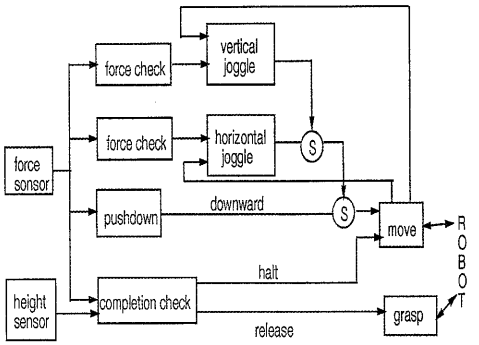


図 6: 包摂アーキテクチャによるディスクを押しさげるときの動作

ディスクをポールに押しさげるときの動作 ：(図 6 参照)

- **押し込む－層**：ハンドに持ったディスクの穴がポールに入ると，起動される．ハンドを下に降ろし，ディスクが下まで着いたらディスクを放す．ディスクが下まで着いたのは，ハンドの反力センサとハンドの高さから判定される．

- **横方向のゆらし－層**：押し下げている最中にディスクがポールに引っかかるとハンドの反力センサから反力が感じられるので，起動される．ディスクを引っかかったポールから戻すために，水平方向に小さな同心円を書くように軽く揺る．ロボットの制御を簡単にするために，実際には，同心円ではなく，正方形を書くように水平方向に順々に動きを繰り返す．ハンドからの反力がなくなると，終了する．その結果，中断されていた押し込み動作が再開される．

- **縦方向のゆらし－層**：横方向のゆらしと同時に起動される．ディスクを引っかかったポールから戻すために，上下に軽く揺る．すなわち，上向きにディスクを軽く動かし，すぐに下向きにディスクを動かすことを繰り返す．ハンドからの反力がなくなると，終了する．その結果，中断されていた押し込み動作が再開される．

  実際の問題では，横方向のゆらしだけで十分であるので，このレベルの実装は行っていない．

5.2 人の流れにおける自己組織化・出現的計算

通路を通過する人の動きを模倣するシステムを包摂アーキテクチャで構成してみよう．このシステムでの目的は 2 つの課題を検討することにある．

(1) 通路を通過する人の数が多くなるとそれまで乱雑であった人の流れがまとまった大きな流れに変わっていくという現象を，個々の人の動きと近傍との相互作用を記述するだけで実現できるか．

(2) 初期状態が少し異なると現われる現象が大きき変わるように，プログラミングできるか．

上記の問題の糸口を得るために，熱の流れを検討しよう．熱源と外部との熱の出入りがない場合には，システムと熱源とが同じ温度になるとそれ以上変化しなくなる．すなわち，熱的な平衡状態に達する．熱源が外部と熱の交換をするような (供給あるいは吸収が生じる) 場合には，1 つの熱源と接するシステムでは，熱的な平衡状態にはいつまでも達しない．

一方，外部と熱の交換をする熱源が 2 つあるようなシステム (非平衡システム) の場合には，厳密な意味 (ミクロな立場) では平衡状態ではないが，マクロな立場での擬似的な平衡状態になることがある．例えば，下方の熱源が熱を供給しつづけ，上方の熱源が熱を吸収しつづけるとシステムを考えよう．エネルギーは温度の高いほうから低いほうに流れ，温度の低いほうの熱源からは散逸する．熱の流れがあるレベルを越えると，熱伝導ではエネルギーが運び切れないこと，上方にある水の比重が下方の水よりも大きくなり，情報の水が支えきれなくなることのために，ある時点で突如対流が生じる．これは，熱の伝わり方が，ミクロな機構である伝導からマクロな機構である対流へと変化したためである．伝導から対流への変化は一種の相転移であり，ミクロからマクロへの平衡状態の相転移を散逸現象と呼び，非平衡システムでの重要な研究課題となっている．

我々は，上記の熱の問題のアナロジーから，通路を 2 種類のペアの熱源に接したシステムとして捉える．2 種類の熱源の組は熱の流れの方向が逆になっている．通路を通過する人をエージェントとみなし，エージェントの動きをプログラムする．また，通路の両方の口から入ってくる人は，適当な分布で入ってくる時間と入口での位置が与えられるものとする．

- **第 1 レベル** 直進する．前の同じ方向を歩いている人にぶつかったときには止まる．障害物あるいは前方から進んで来た物体にぶつかりそうなときには，左右どちらかによる．前が空けばまた前進する．後から押されると，前に動かない障害物があるときには，左右どちらかに動き，それ以外のときには前に動く．

- **第 2 レベル** 前の人の後に続く．全体的な状況は気にせず，ただひたすら前の人の動きをフォローする．

- **第3レベル** 空いているところを探して動く. できるだけぶつからない様に動く道筋を探して, 移動する.
- **第4レベル** 面白そうなものを探しながらぶつからない様に動く.

各レベルの動きは, 下位のレベルの情報と外界を直接センスして得られる情報を下に, 取るべき行動を決定する. そして決定された行動が下位の行動を包摂することによって個々のエージェントの実際の動きが取られることになる. このような個々のエージェントの記述で, 通路を通過するエージェントの動きが, その個数が増えるに従って, マクロな動きが生じてくるかということである. 第2レベルでの処理のポイントは他のエージェントの行動の複製を作る機能にあり, 第3レベルでの処理のポイントはエントロピーを増加させるように行動を取ることである.

このように個々のエージェントの動きを記述するだけで, エージェント (通路を通る人) の数が増加すると, ある時点でミクロからマクロな動きが生じるエージェント群の大きな流れが出現するようにするのが, ここでの課題である. すなわち, 各エージェントは場との相互作用によって, 自己組織化が生じ, マクロな動きが発生する, ということが簡単にプログラムできるのであろうか. さらに, 初期状態がほんの少しだけ異なるだけで, 全く似合わないような現象が出現するという初期値敏感性が達成できるであろうか.

出現的計算 (emergent computation) とは, 次の3つの性質をもった現象である: [5]

- エージェントの集合体 — 各エージェントの挙動は陽に規定されている. すなわち, ミクロスコピックな挙動は陽に記述されている.
- エージェント間の相互作用が規定 — エージェントの集合体総体としての挙動, マクロスコピックな挙動は陰にしか記述されていない.
- 総体としての挙動を計算するインタプリタが存在.

出現的計算は, 決定論的カオスに相当する. カオスには, 3つの定義がある [17]:

(1) 天地創造以前の状態 — 秩序のない状態.

(2) 完全に秩序のない状態 — 大混乱のこと.

(3) 決定論的カオス — 個々の挙動は決定論で記述されるが, 総体として不規則で乱雑な挙動を示す.

決定論的カオスには, テイラー渦流, ベルナール対流, 気象, 化学反応など様々なものが知られている.

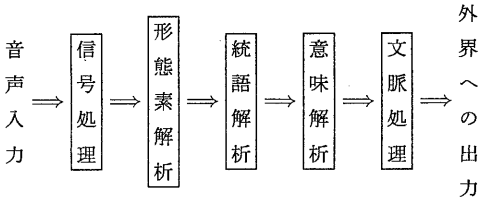


図 7: 従来の機能分割法による自然言語処理

5.3 自然言語処理への適用

自然言語処理もロボットと同様に従来は機能分割によって処理されてきた (図7).

しかし, 一般には上記のような形で機能分割することは極めて難しい. たとえば, 音声処理が信号処理だけで解決できることは極めて希であり, 構文解析や意味解析が必要となる. 岡田 [10] は, 文脈自由文法あるいは一般の句構造文法などの制約の下で連続音声のパージングや意味解析を行うときに, 音声フレームの特徴系列から直接文の候補や意味解釈を得る処理方式を提案している. 岡田の方法は, 機能別分割による階層的なアプローチではなく, 処理ベースでの散層的 (非階層的) なアプローチである. このアプローチを発展させると, 包摂アーキテクチャによる自然言語処理が可能であると期待される.

音声の感情情報は従来の音声情報処理ではあまり扱われてこなかった. 音声における感情情報のうち, ある種の音声は特定の感情的な性質を表すものとして認識されている. さらに身振やスキンシップでも同じように感情を伝えると観察されている [8, 513ページ]. 音声からの感情情報をもとに適切な単語列の検索, あるいは構文処理, 意味処理, 談話処理に活用することが出来よう. 岡田の方法を包摂アーキテクチャで再構成し, さらに感情情報の処理を加えた方式を図8 に示す. 図には4つの層が示され, それぞれ単語列検索, 極小構文ネットワーク作成, プラン推論, 感情情報処理のタスクを担当している. 図からは分らないが, 各層には, 単語列の検索, 構文処理, 意味処理, 談話処理などが含まれるが, いずれの処理も機能別分割構成法で使用される処理のサブセットとなっている. 感情情報処理は他の処理が作成されてから後に加えられるが, 包摂アーキテクチャによって, 自然に機能追加ができる.

図に示した連続音声認識をデータベース検索システムと結合すると, 単語が決定していくと同時にデータベース検索を開始する直観的な処理が可能となる. たとえば, システムが「Which book of Joyce」まで聞けば, 「Joyce が書いた

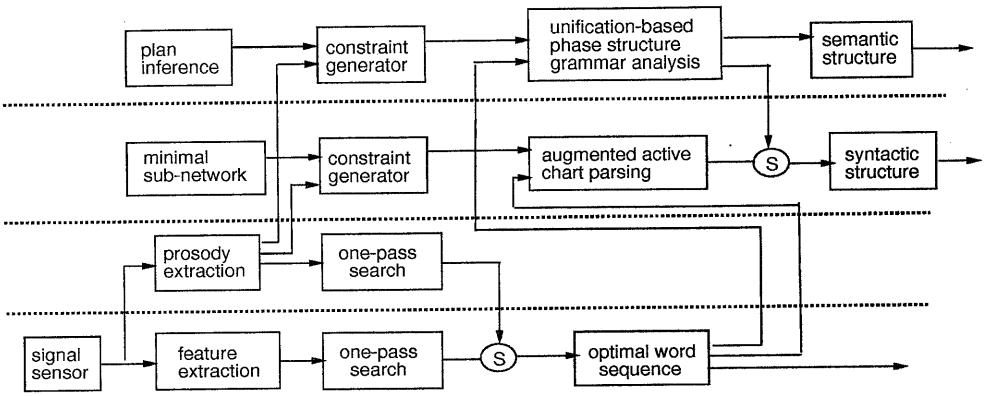


図 8: 行動ベースよる自然言語処理 ([8] の再構成)

本」の検索と「Joyce の持っている本」の検索を同時に開始することができる. データベース検索に時間がかかるとすると, 従来のように音声認識, 言語処理が終了してからデータベース検索を行うよりも, より迅速な応答を返すことができよう.

6 おわりに

本稿では, マルチエージェントシステムの新しいパラダイムを, Brooks の包摂アーキテクチャ, コネクティクスをその出発点として検討を行った. 今後本格的なシステムを構築し, 提案したアプローチの評価を行っていく. 最後に幾つかの知見と今後の課題についてまとめる.

(1) 行動ベースによる問題の分割と包摂アーキテクチャは, 並列プログラミング, 動的な判断 (直観的な判断), 拡張可能性, 段階的プログラミング, フォールトトレラント, 頑健性などを支援すると期待される.

(2) エキスパート・システムを実世界へ応用するためには実時間処理が不可欠な課題である. 差し迫った緊急の事態に対処するために, 少々不正確でもよいから即断することが要求される. さらに, 応急措置的, 直観的な行動の後で, より適切な行動をするためにフォローも必要となる.

例えば, 救急車で患者が運び込まれてきた時の処置を考えよう. 患者は血圧が高く, 危険なので, 降圧剤を使用し, 直ちに血圧を下げるようにした (「**直観的な行動**」). この患者は高血圧であろうと取敢えず仮定をし判断を下した. しかし, 降圧剤使用後, 急速に血圧が低下し始めた. 急いで, 患者のカルテを調べたところ, 低血圧であることが分った. そのまま降圧剤の使用を続ければ, 血圧低下のために心臓ショックを引き起こす恐れがあるので, 直ちに降圧剤の使用を中止し, 事なきを得た (「**フォロー**」).

このようなシステムは, 包摂アーキテクチャによって, たとえば

- **レベル 0:** 即効的な治療法を立案
- **レベル 1:** カルテをもとに治療法を立案
- **レベル 2:** 精密検査をもとに治療法を立案

という層構成で実現することが出来る.

(3) 場の考えかたは, 従来のセンサーフュージョン (sensor fusion) ではなく, センサフィッション (sensor fission) のための枠組を与える.

(4) 出現的プログラミングでは, できるだけ文脈に依存しないように記述することが問題である. 包摂アーキテクチャは, 各エージェントの決定論的な動きを与えるだけであるので, 出現的計算のプログラミングシステムと期待できる.

(5) エージェント間の相互作用と場との相互作用とが等価であると主張した. この場が唯一のものとするか, あるいは, オーバーラップする近傍系をとらえるかは, 今後の研究の課題である. はじめにで述べたように, 我々は「オーバーラップする近傍系」であるコネクティクスという考えかたで分散システムの研究を進めており,

Linda 計算モデルをコネクティクスの観点から拡張した NueLinda モデルを提案している [9]. 本研究と NueLinda との融合を図っていく予定である.

(6) 出現的計算の実験での重要なポイントは, 初期値を少し違えると, 出現する現象が全く似ていないものなるという初期値敏感性が達成できるかという点である.

(7) 場との相互作用, 場の温度による相互作用に制御を正方格子整列問題, ライフゲームへ適用し, 更に検討をすすめる.「運動場に集まった小学生に正方形に並びなさい」と命令をしたときに個々の小学生がとるべき行動をどのように記述するか, という問題である [16]. Conway が発明したライフゲームで, 8-近傍 の状態から次の状態が定まるが, この生死の条件を N-近傍まで拡張し, かつ, 場での情報の伝達に有限時間がかかると仮定したときに, 個々の点 (エージェントととらえる) をどう記述するかという問題である. 場での情報伝達あるいはどの範囲の近傍まで扱うかを指定するのに, 場の温度を使用することができよう.

(8) エージェントの記述ではオブジェクト指向プログラミングを使用する. オブジェクト指向プログラミングの拡張として, AOP (Agent-Oriented Programming) [14] が提案されている. AOP の主な主眼点は, エージェントの心的状態 (mental state) — 信念, 選択, 能力, 委任等 — であり; 論理的なフレームワークでエージェントの行動を記述しようと試みている. 一方, 我々は論理的なフレームワークで記述されたエージェントが集まって, 直観的な行動がどのように生じうるかに興味がある.

(9) 最後に本稿で提案したシステムを具体的なハードウェア上に実装し, 並列処理・分散処理に応用し, 実装上の問題点を明らかにし, その解決法を考案するとともに, 性能を測定し, 評価することが今後の最重要な課題である.

謝辞

コネクティクスの考え方をまとめ, 本研究に多大な影響を与えるとともに日頃ご指導を頂く NTT 基礎研究所竹内郁雄リーダ, NueLinda の共同研究者である NTT ソフトウェア研究所村上健一郎主任研究員, TAO-Linda の開発を担当している明石修社員, 自然言語処理で御教授頂いた竹内研究グループの皆さま, および, ロボット関係でご協力頂いた NTT ヒューマンインターフェース研究所大原秀一主任研究員, 酒井リーダに感謝をいたします.

参考文献

[1] Akashi, O., H.G. Okuno, K. Murakami, and Y. Amagai: NueLinda: Reorganizing the Linda Model with Connectics. *submitted for publication*, 1991.

[2] Brooks, R.A.: A Robust Layered Control System for a Mobile Robot, *IEEE J. Robotics and Automation, RA-2*, Apr (1986) 14–23.

[3] Brooks, R.A.: Intelligence without Reason, *Proc. of IJCAI-91*, Sydney, 1991, 569–595.

[4] Huberman, B.A. (*ed.*): *The Ecology of Computation*, North-Holland, 1988.

[5] Forrest, S. (*ed.*): *Emergent Computation*, special issue of *Physica D*, The MIT Press/North-Holland, 1991.

[6] Koza, J.R.: Evolution of Subsumption, *in Genetic Programming*, Draft, Sep. 1991.

[7] Maes, P. (ed.): *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, special issue of *Robot and Autonomous System*, The MIT Press/Elsevier, 1991.

[8] Minsky, M.: *Society of Minds.* Simon & Schuster, Inc., 1986. 邦訳「心の社会」, 安西訳, 産業図書, 1990.

[9] 村上, 明石, 天海, 奥乃: 計算モデル Linda の TAO への導入. 記号処理 57 - 4, 情報処理学会, Nov. 1990.

[10] 岡田 美智男: 音声言語のバージングにおける最適な単語列の探索について. 自然言語研究会, 情報処理学会, 1991.

[11] 大里延康: エージェントをベースとしたロボット・プログラミング, 信学会春季全国大会予稿集, D-231, 1991.

[12] 大里, 奥乃, 大原: マルチエージェントシステムとその行動ベースロボット制御への適用, 日本ソフトウェア科学会第 8 回大会論文集, D1-6, Sep. 1991.

[13] 清水 博: 「生命を捉えなおす」増補版, 中公新書 503, 中央公論, 1991.

[14] Shoham, Y.: Agent-Oriented Programming, (Revised), Technical Report STAN-CS-1335-90, Stanford, 1990.

[15] Takeuchi, I., H.G. Okuno and N. Osato: A List Processing Language TAO with Multiple Programming Paradigm, *New Generation Computing*, Ohmsha/Springer-Verlag, vol.4, no.4 (1986) 401–444.

[16] 竹内, 後藤, 尾内, 斎藤, 奥乃: コネクティクス構想, 日本ソフトウェア科学会第 8 回大会論文集, B3-1, Sep. 1991.

[17] 戸田 盛和: 「カオス — 混沌のなかの法則」, New Science Age - 46, 岩波書店, 1991.