機械の自己組立てと自己修復†

村田 智*・富田 康治*・黒河 治久*・小鍛治 繁*

Self-Assembly and Self-Repair Algorithm for a Distributed Mechanical System

Satoshi MURATA*, Kohji TOMITA*,
Haruhisa KUROKAWA* and Shigeru KOKAJI*

Self-assembly and self-repair algorithm for homogeneous distributed mechanical systems is proposed. We focused on a category of a distributed system composed of numbers of identical units which can dynamically change connections among them. Each unit has an onboard micro-processor and local communication between neighboring units is assumed to be available. A group of the unit with random configuration at the initial time metamorphoses into desired shape by cooperation of the units. This process called "self-assembly" is realized by homogeneous software. An extension of this process, "self-repair" is also examined. In self-repair, cut off of an arbitrary part of the system is considered. When some part of the system detects a damage the whole system degenerates and reconstructs itself.

Key Words: self-assembly, self-repair, distributed mechanical system

1. はじめに

本論文では機械システムを基本構成要素(以下ユニットと呼ぶ)の集合体によって構成する手法を提案する。この機械システムは、ユニットの協調によって組立てられ(自己組立て)、故障が発生した場合も外部の助けを借りることなくユニットの協調によって修理する(自己修復)という、従来の機械システムでは本質的に実現が困難であるような機能を備えている。また、徹底した汎用ユニットの使用により、システムの設計・生産・保守の労力を大幅に軽減し、信頼性の高い大規模システムを柔軟に設計することを意図している。

このような発想は生物のからだの基本構成に触発されたもので、von Neumann の自己増殖機械¹⁾に遡る歴史をもっており、「人工生命」²⁾等の研究として現在にまで引き継がれている。ただ、von Neumann はじめ多くの研究は論理的空間での構築を前提として行われており、物理的にこれを実現する試みはごく少数の例にとどまっている。われわれは物理的な実体をもった機械ユニットを試作し、これを用いた実験を行うことにより、ユニットがもつべき要件や、各ユニットが行う処理の詳細について具体的な検討をすすめ、分散構造をもつ機械システムの実現に近づこうとする立場をとっている。

われわれはすでに小規模なユニット数(十程度)のシステムの自己組立て手法を開発してきているが、本論文ではさらに大規模なユニット数(数百程度)のシステムにも適用可能な手法について述べる。ここでは、現実の機械システムがもつ多くの機能を捨象し、形態を実現することだけを当面の目標として扱う。すなわち、「自己組立て」ではユニット群がランダムに結合している初期状態から出発して、相互の接続を変化させることによりある目標形状(2次元)を組立て、また「自己修復」では完成した目標形状からユニットが欠落した場合に、残ったユニット群が接続を変更して再び目標形状に戻るといふ問題を考える。

2. 機械ユニット

システムを組立てる材料として数種類のユニットの存在を仮定することも可能であるが、ここでは1種類だけのユニットを考える。1種類のユニットだけでできた完全に均質なシステムは理論的な扱いが容易で、以降で説明するような自己修復などの手順を設計する場合にも有利だからである。また、各ユニットはある程度の情報処理ができ、周囲のユニットと情報交換ができるものとする。さらに、外部の助けなしでユニットの置き換えを行うために、ユニット間の物理的な結合を変更する機能も必要である。これによって故障ユニットを取り替えたり全体の形状を組立てたり変更したりすることができるようになる。

われわれは実際に結合状態を変更できるユニットを試作してその基本機能の実現性を検討してきた(Fig. 1)³⁾。このユニットは「フラクタム」と呼ばれ、2次元機械を構成するため

† 第6回, 7回, 8回自律分散システムシンポジウムで発表(1995・1)(第6回)(1995・11)(第7回)(1996・1)(第8回)

* 機械技術研究所 つくば市並木1-2

* Mechanical Engineering Laboratory, Tsukuba
(Received June 10, 1996)
(Revised December 5, 1996)

Fig. 1 Mechanical unit "fractum"

(a) change connection (c) transportation

Fig. 2 Basic functions of fracta

の基本ユニットである。各ユニットは6本の電磁力による結合腕をもつ。これらの結合腕はユニット上のCPUによってそれぞれ独立に結合・解放の制御ができ、また隣接ユニットのCPU間ではシリアル通信による情報交換が行えるようになっている。腕の結合・解放動作をうまく組み合わせると、ユニット間の位置関係の変更、ユニットの切り離し、ユニットの輸送などさまざまな機能が実現できる (Fig. 2)。

これとよく似た機械ユニットをPamechaとChirikjian^{4),5)}も提案している (Fig. 3)。そのユニットは6本のリンクからなる機構で、リンク間の角度を3個のサーボモータで制御することにより、ユニットを自由に変形させることができる。また、自分のリンクと隣のユニットのリンクを機械的なフックで接合できる。リンクの変形と隣接ユニットとの結合離反をうまく組み合わせれば、ユニット間の位置関係を変更できる。このユニットはサーボモータなどを用いているので単体の構成は相当複雑になるが、基本的にはわれわれのユニットと同様な動作が可能である。剛性の高い構造体を構成できることや、大きな力を発生できるなどの利点もある。

福田らは6角形の機械セルを基本要素とするロボットを構成しており、CEBOTと呼んでいる⁶⁾。このシステムではセル間の結合・解放の機能はもつが、結合位置を直接変更することはできない。セル間の位置関係を変更するときは、一度結合を切り離し、別の位置に自力で移動して再結合する。した

Fig. 3 Link unit by Chirikjian et al.

がって、前述の2例とは基本的動作が異なる。また、各ユニットは異なる機能をもつことを前提としている。

このように機械ユニットを製作する場合、いろいろなデザインが可能であるが、本論文では、ユニットの具体的な機構設計には立ち入らず、次のような抽象化したユニットのモデルを扱うことにする。

〔ユニットのモデル〕 ユニットは正6角形をしており、6本の結合腕をもつ。その結合腕は結合力および反発力を発生し、ユニットを回転させることができる。各ユニットは情報処理機構をもち、隣接ユニットと情報を交換しながら結合腕を制御する。 ■

結合腕は6本でなくともよいが、6角形は幾何学的に見て豊富な構造をもっていること、また、実際につくられたユニットが存在することなどからこれを選んだ。以下ではこのようなユニットを前提として、ユニット群が分散的に全体の構造を組立てる「自己組立て」のアルゴリズムや、故障が起こった場合に自分自身でそれを修復する「自己修復」のアルゴリズムを導く。

3. 小規模な自己組立て

ここではこれまでに開発した小規模システムのためのアルゴリズムを簡単に説明する。最初ランダムな形状をもっていたユニット群が、あらかじめ定められた目標の形状に自分自身を変形させることを自己組立てと呼ぶ。ただし、各ユニットに搭載するプログラムがすべて同一であることと、各ユニットが隣接ユニットとしか情報交換できないことを制約条件とする。すなわち、自己組立ての過程では、初期状態においては互いに区別できなかったユニット群が、互いに情報交換をしながらそれぞれの役割を認識し、分化しつつ協調動作を行う。

このアルゴリズムでは、形状を記述するプリミティブとして結合の様式を表す「型」という記号を導入し、この記号をもちいて全体の形状を記述する。(型については後述。)たとえば、10ユニットからなる三角形は、

$o(K, K)$,

$K(o, K, K, s)$,

s(K, K, K, K, K)

のようにいくつか(この場合3行)の文によって記述される。o, K, sなどが結合の型を表す記号で、たとえば文 o(K, K) は、o型のユニットはK型のユニット2個と結合せよという意味である。各ユニットは隣と通信することで自分の近傍の結合状態を調べ、それがこの記述のうちのどれかの文と完全に一致するまで勝手な方向に動き回る。記述との一致度に応じて動く頻度を調節することによって、全体が次第に目標の形状に近づいていくようにすることができる。シミュレーションでは、10個のユニットが三角形を形成する問題で97%の成功率が得られている。

この方法は、システムが小規模で、目標の形状の対称性が高い場合はうまく働くが、ユニットの数が増えたり形状が複雑になったりすると、その効率は急激に悪化する。例えば同じ三角形を作る実験でも、ユニットを1列増やして15個としただけで成功率は73%におちる。また、三角形以外の形状を作る実験では10個に満たない極めて小規模な場合しか成功しなかった。その原因はすべてのユニットが記述のどれかの文と一致するまで運動をやめないというところにある。あるユニットが記述を満たせずに動くと、それがすでに記述を満たしている別のユニットの状態をくずしてしまう。全体の形態形成が収束するためにはすべてのユニットが同時に記述を満たす必要があるが、ユニットの数が増えるとそのようなことはきわめて起こりにくくなるのである。

4. 大規模な自己組立てータマネギ法

前章で述べたような自己組立ての難しさは、ユニット群の上に何らかの座標を導入することで軽減できる。ユニット群上に原点を設定し、組立てを行う位置や対称軸の方向をあらかじめ指定することができれば、ユニット群のむだな動き(探索の手間)を大幅に減らすことが可能である⁷⁾。ここでいう座標とは、デカルト座標のように明示的な座標軸をもつものである必要はなく、ユニット間のつながり方をうまく記述できるものであればなんでもよい。

ところで、生物の形態形成、すなわち発生過程は、単純な形態から複雑な形態へと順を追って変化する段階的な過程である。そこでは受精卵から始まって、自分がどの細胞から分裂したかという情報(系譜)の局所的な受け渡し、重要な鍵になっているといわれている⁸⁾。

そこで、これら「組立ての原点の導入」と、「局所的な情報の受け渡しによる段階的な形態形成」という2つのアイデアをモチーフとして、大規模で複雑なシステムの組立てアルゴリズムを構築することを考える。

アルゴリズムの大筋は以下のようなものである。まず、ユニット群のなかから一個を選んでこれを組立ての原点とする。このユニットを「核」と呼ぶ。核は生物における受精卵に相当する。次に、核に選ばれたユニットは、ある種の組立て設計図にしたがって、隣接するユニットの中から適当なものを選んで仮想的な結合のネットワークを形成する。これを

第1段階とする。その次は、第1段階のネットワークに含まれるユニットがそのネットワークの外側のユニットを取り込んで第2段階のネットワークを形成する。この手順を繰り返すことによって次々に新しい段階に移り、全体は次第に大きく複雑に成長してゆく。ネットワークが、一個の核から一枚ずつ層を重ねるように段階的に成長して行くので、このアルゴリズムをタマネギ法(onion method)とよぶ。以降では、順を追ってその詳細を説明する。

4.1 物理型と論理型

われわれは6本の結合腕をもった機械ユニットをモデルとして採用したが、この腕による機械的な結合を物理結合とよぶ。物理結合の様式は、回転や鏡像を無視すると Fig. 4 に示したように12通りあり、これらを「物理型」と呼ぶ。(3章の同時組立てではこれを用いた。)

さて、ユニット間の結合の種類として、物理結合のうえに別の種類の結合を定義する。これを論理結合とよぶ。論理結合はユニット上のCPUが(あとに述べるアルゴリズムにしたがって)設定するもので、物理結合中のユニット間でも論理結合はしていない場合がある。論理結合の型の種類とその名称も物理結合と同様であるが、全く結合がないという型“n”が加わる。これらを「論理型」とよぶ。タマネギ法では主にこの論理型をもちいてシステムの組立てを行う。

4.2 核の選出

すべてのユニットが完全に同一なソフトウェアで駆動されているときに、ユニット群の中のひとつを選び出す問題が核の選出問題である。これにはソフトウェアの均質性の制約に抵触しないいくつかの方法が考えられるが、以下ではマイクロプロセッサによる実行が容易な、乱数をもちいる方法を説明する。

[核の選出法] 初期状態において、すべてのユニットは全く同じ状態にあるとする。まず、各ユニットは適当な方法で乱数を振り、その値を変数 $m(i)$ に保存する ($m(i) \leftarrow \text{random}(i)$) (i はユニット番号、以下同じ)。次に自分の隣接ユニットすべてと通信して、それぞれの変数を調べ、もしも自分より大きな値をもつものがあれば、その値で $m(i)$ を書き換える。この通信と書き換えの操作を決められた回数 N だけ繰り返す。その結果、もし変数 $m(i)$ が最初に振った乱数 $\text{random}(i)$ に一致すれば、自分の N 近傍には自分より大きな値をもったものがいなかったことが分かる。したがって、 N を十分大きく取って(たとえばユニット総数の平方根)ユニット群全体を

Fig. 4 Physical types and logical types

覆うようにすれば、これで核の選出ができる^(注1)。

4.3 記述行列—大規模構築の設計図

すべてのユニットは記述行列とよばれる特殊な設計図をもっている。記述行列には各段階においてシステム内の位置に応じてユニットがどの論理型をとればよいか書かれている。各ユニットは段階 $stage(i)$ と位置指標 $location_index(i)$ という2つの引数をつかってこの行列を参照し、自分の論理型を決定する。記述行列は Fig. 5 のような下三角行列で、横軸が位置指標、縦軸が段階を表す^(注2)。

ユニットは初期状態では、段階=0, 位置指標=-1をもっているが(位置指標=-1のときは、論理型は“n”とする)、核になったユニットだけは段階=0, 位置指標=0という値をもつことにする。

この例では最終段階で28個のユニットが三角形を構成す

		location index						
		0	1	2	3	4	5	6
stage	0	(s)						
	1	s	ε					
	2	-	K	o				
	3	-	f	-	o			
	4	-	s	K	K	ε		
	5	-	-	s	-	K	ε	
	6	-	-	-	-	-	K	o

Fig. 5 Description matrix

Fig. 6 Stages of self-assembly

(注1) 乱数の解像度が低い場合は、同じ最大値を取るユニットが複数個存在する可能性が高いが、これを排除するためには、最大値を取ったと思ったユニットが乱数をもう一度振りなおして同じプロセスを繰り返せばよい。

る手順を示している。第0段階(第0行)は、核になるユニットだけが孤立した状態を表す。第1段階(第1行)では、核は新しい6つのε型のユニットに囲まれて、論理結合のネットワークが完成する(ネットワークの完成についてはあとで定義する)。第2段階では、前の段階でε型だったユニットはK型に変化し、新たにo型のユニットが加わってネットワークをひとまわり大きく拡張する。記号“-”は、前の論理型を保持するという意味である。以下同様にこの手順を繰り返したときの組立ての過程を Fig. 6 に示す。

この記述にはある論理型をとるべきユニットが何個あるかという情報は陽に含まれないが、後に述べるネットワークの整合性によって間接的に決まるようになっている。また、論理型の組み合わせや出現の順序を工夫することで、複雑な形状を表現することができる。また、対称性の高い形状では記述をコンパクトにまとめることができる。さらに、複数の記述行列をポイントで組み合わせて使うこともでき、繰り返し構造などの表現も可能である⁹⁾。Fig. 7 は2つの行列を用いた記述の例で、一つの行列が中央の大きな三角形を表し、もう一つが小さな三角形を表す。前者の記述の最後にはポイントがあり、後者にジャンプする。そのおわりにもポイントがあり、自分を再び読み出すようになっている。この繰り返りで図のような形状がつけられる。

4.4 段階の更新

ユニット群は記述行列にしたがって段階ごとに形状を完成して行く。ある段階が完成するとは、そのときの論理ネットワークに含まれるすべてのユニットにおいて相手の論理結合と結ばれていない論理結合がひとつもないことと定義する。そこで、これをユニット群が局所的に判定する手順が必要になる。このために各ユニットに完成度 *complete* なる s 次元のベクトル変数をもたせる。ただし、 s は組立ての最終段階数、ベクトルの初期値は $(0, 0, \dots, 0)$ とおく。このベクトルの各要素には各段階においてネットワークがどのくらいでき

Fig. 7 Example of recursive structure

(注2) ただし、この記述法には一意性はない。なぜなら、ある形状を与えたとき、その形状に至るルートはいろいろ考えられ、それぞれが違った記述行列をもつからである。

あがったかという(局所的な)評価値はいる。その評価値が判定スレシホールド TH より大きくなったとき、(そのユニットにおいては)その段階が完成したと判定される。核ユニットだけは特別に初期状態で完成度 $= (TH + 1, 0, \dots, 0)$ を設定する。つまり、核だけは第0段階をすでに完成したものとみなすわけである。したがって、核は選出されるとただちに第1段階に移行する。第1段階以降のネットワーク完成の確認は次に述べる方法で行う。

〔最小完成度伝播法¹⁰⁾〕あるユニットの論理結合腕が、隣のユニットの論理結合腕とちょうど対向しているとき、この腕は充足しているという。さらに、そのユニットのすべての論理結合腕が充足していることを完全充足と定義する。各ユニットが完全充足しているとき、隣接するすべてのユニットの同段階の完成度を調べ、そのうちの最小の値に1を加えたもので自分の完成度を置き換える。そうでないときはその段階の完成度を0とおく。すべてのユニットはスレシホールドを越えるまでこのプロセスを繰り返す。

この方法によれば、ある段階のネットワークが完成すると(論理結合が過不足なく閉じると)、どのユニットの完成度もこのプロセスを繰り返すたびに1ずつ上がって行き、いずれはどのユニットでも完成度が判定スレシホールドを越える。つまり、あるユニットを中心とし、判定スレシホールドを半径とする円内のユニットがすべて完全充足したとき、そのユニットは次の段階に移行する。ただし、すべてのユニットが一斉に移行するわけではなく、完成したものから順に移行してゆく。

4.5 段階および位置指標の決定

ある段階のネットワークが完成すると、論理ネットワークはただちに次の段階の組立てを開始する。つまりすでに組立てられた部分に周りの未分化ユニット(段階=-1)をつけ加えてそれを一回り大きく拡大しようとする。このとき、新たに加わるユニットに適切な論理型を取らせる必要がある。これにはそれらのユニットに段階と位置指標を教えてやればよい。(この手続きは生物の発生において、その細胞がどの細胞から由来するかという情報の受け渡しを行うことに相当する。)それは次のようにして行われる。

完成したネットワークに属するユニットは、新しい段階に移行すると、記述行列にしたがって次の論理型になり、新たな論理腕を周囲に延ばす。ネットワークの周りにいたユニットはこの論理腕を向けられると、相手の段階に自分の段階を合わせてその段階に加わろうとする。(ただし、論理腕を出している相手が既に完成したネットワークに含まれていないときは無視する。)このとき位置指標は段階と同じ値でよい。なぜなら、その段階に新たに加わるユニットの論理型は、つねに記述行列の対角部分に書かれているからである。

こうして加わった新たなユニットがネットワークに組み込まれ、この段階が完成すると、以後、そのユニットの位置指標はこのときの値に固定される。

4.6 論理型の方角探索

すべてのユニットが適切な論理型を適切な向きに置いて初めて論理ネットワークが完成するわけだが、記述行列には論理型の方角は書かれていないので、各ユニットは自分で正しい方向を見つけださなければならない。方向の探索は簡単なマッチング演算でできるが、効率の良い探索を行うために、次のような工夫を施す。

〔マッチング〕各ユニットは自分の論理型の向きをひとつずつ回転させながら、どの方向が最も完全充足に近いかを判定する。具体的には、自分の論理型を表す配列と相手からの結合要求を表す配列をひとつずつシフトさせながら積和を計算し、最大値を取ったところを自分の方向とする。(もちろんこれらの配列は循環している。)

結合要求: [0 0 1 1 0 0]

新たに加わるユニット: [1 1 0 0 0 0]

の論理型

一般には、最大値を取る場所は複数あるが、そういった場合はそのうちの一つをランダム¹¹⁾に選ぶ。

〔強結合と弱結合〕すでに完成した論理ネットワークを次の段階を組立てるときの方向探索によって壊してしまわないために、論理結合に強弱の区別を導入する。すべての論理結合ははじめ弱結合として設定されるが、ある段階のネットワークが完成した時点でそのネットワークに含まれる論理結合を弱結合から強結合に変更する。すなわち、論理ネットワークはまず弱結合によって構築され、これが完成すると強結合に変更される。

4.7 ユニット供給のメカニズム

ここまで、論理ネットワークの段階的な組立ての方法について述べてきたが、段階が進むにつれ、いつかは論理ネットワークがユニット群の縁を越えて先へ延びなければならなくなる。もしもそのような場所にいつでも未分化のユニットを供給することができれば、ネットワークはユニットがある限り成長し続けることができ、ついには目標とする物理的な形状に論理ネットワークの形状が一致するであろう。

2次元機械では、システムがどのような形状になってもその周りは一本の閉曲線であるという性質がある。この性質を利用すると、簡単にそのようなユニットの供給を行うことができる。

〔巡回によるユニット供給〕ユニットはその物理型にしたがって、可動型と非可動型に分けられる。ここでは、“e”、“o”、

(注3) しかし、全くのランダムでは、計算するたびに方向が変わることになり、その結果、ネットワークが頻繁に途切れて収束が遅くなったり、結合をさぐりあう一種のリミットサイクルに落ちこんだりする。そこで、いつもランダムに書き換えるのではなく、ある一定の確率 P_{match} で書き換えることにする。

また、詳しくは述べないが、論理結合要求を一方だけから受けているユニットは、デッドロックを引き起こしやすいので、そういうユニットの論理結合を切り離す手続きも必要になる。そのようなユニットは確率 P_{cut} でリセットされ、未分化な状態(位置指標=-1)に戻される。

“ε”を可動型とする。ユニットは自分の物理型が可動型で、かつ未分化である場合に、ある確率 P_move で右に1ステップころがって移動する。このようにすると、ユニット群の周上にある未分化のユニットは、群のまわりを反時計周りに巡回する。

このルールだけでは未分化部分の形状が突起を持ちやすく、ユニットの巡回が効率的に行われない。とくにユニット群に穴ができると、その内部にユニットを供給することが難しくなる。(したがって、目標形状そのものも穴を持っていないものに制限される。)そこで、いくつかの補助ルールをくわえて、ちょうど表面張力によって液滴が収縮するように、未分化部分がなめらかな輪郭を保つようにした。そのルールの内容については細くなるのでここでは省略する。

4.8 自己組立てのシミュレーション

実際に上述のアルゴリズムを適用する場合は、各ユニットで、

- 0) 核の選出
- 1) 段階と位置指標の決定、論理型の決定
- 2) 論理型の向きの探索
- 3) 完成度の伝播と段階の移行判定
- 4) 未分化ユニットの移動

のように手順を進めることになる。各ユニットは0)を終えた後は1)から4)を繰り返すだけであり、完全に分散的、並列的なアルゴリズムとなっている。また、初期状態において、各ユニットのプログラムおよびデータは全く同一である。ユニット間でやり取りされる情報は、核選出用の乱数、物理型、段階、論理結合要求、完成度などで、これらはすべて整数型である。

アルゴリズムのなかでチューニングを必要とするパラメータは P_match , P_cut , P_move と完成度判定の TH だけであるが、これらは互いに関係し合っているので設定には若干の注意が必要である。たとえば、 TH を大きく取って広い領域で完成することを要求するときは、 P_match や P_move を小さく設定して論理結合の保持時間や未分化ユニットの滞留時間を延ばしてやらないと、いつまでもその段階を完成できなくなる。 P_cut と P_match の間にも同様の関係がある。

以上のアルゴリズムを評価するために、いろいろな形状を目標として与え、計算機実験を行った。まず、Fig. 6 の28ユニットからなる三角形を目標形状として同じ初期状態から1000回の試行を行ったところ、そのすべてのケースで組立てに成功した。(アルゴリズム中に乱数を用いているので、同じ初期状態からでも毎回違った経過をたどる。)このとき、最終段階に到達するまでに要した時間の平均は480ステップ(シミュレーションのサイクル数)であった。

次に、80個(うち5個は予備)のユニットからなる比較的複雑な形状を目標として実験を行った。このときの構築の過程を Fig. 8 に示す。図中の数字は段階を表し、ひとつの数字がひとつのユニットに対応する。“0”は未分化のユニットを表す。初期状態において中央にあるのが核ユニットである。核

Fig. 8 Simulated sequence of self-assembly

選出の過程は省略し、最初から中央に配置してある。ユニット間にある線は論理結合を表している。この例題についても1000回の試行を行ったところ、そのうちの979回で5000ステップ以内の組立てに成功した。この場合、最終段階(第25段階)には平均2600ステップで到着している。失敗したケースでは、5000ステップ以内に組立てができなかったか、あるいは、システム内に穴ができたことによるデッドロックに陥っていた。これ以外の形状についても実験を行ったが、ほとんどの場合、高率で組立てに成功した。このとき目標形状ごとにアルゴリズム中のパラメータをチューニングし直す必要はなかった。

5. 自己修復のアルゴリズム

前章まで述べてきた自己組立てアルゴリズムは、その階層性を利用すると、ごく自然に自己修復アルゴリズムに拡張することができる¹¹⁾。以下では、システムの自己修復、すなわちシステムにおいて任意のユニットが故障した場合にこれを自己修復する方法について考察する。ただし、故障を一般的に扱うことはきわめて困難であるので、システムに起こりうる故障として、任意の複数のユニットがシステムから欠落するというカテゴリの故障のみを考察する。欠落は任意の時間に任意の場所で何度起こってもよいとするが、残されたユニットはすべて正常に機能すると仮定する。システムはユニットの欠落を検出し、残った部品によって欠落部分を代替する。(すでに組立てた部分をばらして、でてきた未分化部品を修理に充てることもある。)この過程は組立ての段階を後戻りすることにより自然に実現できる。つまり、途中でいった形状の組立てを停止し、確実に完成している段階まで後退し、それ以降の組立てをもう一度やり直せばよいのである。

5.1 欠落の検出

ユニットの欠落の検出はすでに分化を終えて論理ネットワークに組み込まれたユニットが行う。すなわち、これらのユニットの間で論理結合ができていないはずの結合腕において相手からの通信がとだえた場合に、相手ユニットが欠落したとみなすことにする。

5.2 退化信号

システムの段階の後戻りは、欠落を検出したユニットが退化信号をまわりのユニットに送出することにより開始される。退化信号にはどの段階まで退化するかという退化の深さ(退化レベル)が書き込まれている。退化の深さとしては欠落したユニットの位置指標をもちいる。これは残ったシステムに含まれる論理ネットワーク構造のうち、最大の段階のものを残すためである。

5.3 段階の後戻り

欠落を検出したユニットからレベル n の退化信号が送信された場合、システム全体を少なくとも段階 n まで後戻りさせる必要がある。(アルゴリズムを単純にするため、実際には段階 $n-1$ まで退化するが、本質的には段階 n までの退化と考えてよい)以下、Fig. 9 の記述行列を使って説明する。後戻

Fig. 9 Self-repair by degeneration

りのためにはシステム内のユニットのうち段階が n を越えるもの(領域 A と B)が、 n まで段階を戻せばよい。このとき、段階 n より後に形状構成に参加したユニット、つまり領域 A は未分化の状態に戻る。領域 B ではそれぞれ段階 n の論理型にもどる。また領域 C は欠落箇所が 1 カ所だけのときは使用されない領域である。この領域は、たとえばあるユニットがあるレベルの退化信号を受け取って退化したあと、システムの別の部分が破損し、それよりも浅いレベルの信号を送信してきた場合、その退化信号を無視して不要な伝播を終止させるために必要となる。

具体的には、この退化手続きは各ユニットが次のような処理を行うことにより実現できる。各ユニットに対して周囲から受けとった最小の退化信号のレベルを n 、自分の位置指標

Fig. 10 Simulated sequence of self-repair

を l , 段階を s とする. $n < s$ であれば, 論理的に結合されている周囲のユニットにレベル n の退化信号を送信する. さらに, n と l , s の大小関係に応じて以下を実行する.

- $n < l$ のとき (領域 A): ユニットの初期化され, 未分化の状態に戻る. 位置指標と段階も初期値に戻る.

- $l \leq n < s$ のとき (領域 B): ユニットの段階を n に戻す. したがって, 論理型も段階 n のものに戻る.

- $s \leq n$ のとき (領域 C): 退化信号を無視する.

以上の手続きにより, システム全体が適当な段階まで退化することができる. また, 退化が終わると退化信号の伝播もとまり, 信号だけがいつまでも残ることはない.

この方法では, 退化信号の送信を行ったユニットはすぐに形状の組立てを再開できるので, 組立てと並行して修復を行うことができる. また, 複数のユニットが立て続けに欠落した場合, いくつかのレベルの退化信号がシステム内を飛び交うことになるが, システムはそのうちの最も深いレベル (最小のレベル) に後戻りする. ただし, 核を取り除いてしまった場合は, 一番最初の核選出からやり直さなければならない.

5.4 自己修復のシミュレーション

自己修復手続きを実際にユニットにインプリメントするには, 欠落の検出および退化信号に関わる処理を 4.8 節のフローの 2) と 3) の間で行うようにすればよい.

以下にシミュレーションを実行した例を示す. ここでは, Fig. 8 の形状からある部分を取り除いて自己修復を行った (Fig. 10). まず, 一度完成した形状からユニットを取り除く. “0” のユニットは未分化の予備ユニットであり, システムのまわりを巡回している. 次の図から 4 番目の図までは退化信号の伝播の様子を表す. 取り除かれたユニットの位置指標 19 にシステム全体が段階を戻そうとしている. 退化終了後の形状構成は通常の場合と全く同様に行われ, もとどおりの形状が再構築される.

6. おわりに

本論文は, 同一の機械ユニットで大規模な 2 次元機械システムを組立てる方法, および, そのシステムに故障 (欠落) が生じた場合にシステム自身がそれを修復する手法について述べたものである.

ユニットのモデルとして, 情報処理能力のある結合ユニットを考え, これらすべてに同一のプログラムをいれること, および, 隣接ユニット間でしか情報交換をしないことの 2 つの制約条件の下でアルゴリズムを設計した.

システムの組立てに関する情報は, 記述行列と呼ばれる特殊な設計図にコンパクトに表現されており, 各ユニットはこの設計図に基づいて段階的にシステムの組立てを行う. 記述行列は生物の遺伝子のようなもので, これを書き換えることにより様々な形状が表現できる. さらに, システムの一部が欠落した場合に, 全体が適当な段階に後戻りし, もう一度システムの組立てをやり直す手順を導入することによって, 自己修復の機能を実現した.

われわれは, 実在の機械ユニットを前提としたことで, 自己組立て・修復のアルゴリズムを深いレベルまで詳細に検討することができたと考えている. 今後は実機を用いた検証とともに, ユニットの改良をすすめ, 機械システム本来の機能を実現してゆきたい.

参考文献

- 1) J. von Neumann: Theory of Self-Reproducing Automata, Univ. Illinois Press (1966)
- 2) C. G. Langton, editor: Artificial Life I-IV, Addison Wesley (1990-1993)
- 3) 村田, 黒河, 小鍛治: 自己修復する機械, 計測自動制御学会論文集 31-2, 254/262 (1995)
- 4) G. S. Chirikjian: Kinematics of Metamorphic Robotic System, Proc. IEEE Intl. Conf. Robotics and Automation, 449/455 (1994)
- 5) A. Pamecha and G. S. Chirikjian: A Metamorphic Robotic System, IEEE Intl. Conf. Robotics and Automation, Video Proceedings (1995)
- 6) 川内, 稲葉, 福田: セル構造化ロボットシステムに関する研究, 日本ロボット学会誌, 12-1, 116/132 (1994)
- 7) 村田, ほか: 分散機械システムの設計 (第 3 報), 第 6 回自律分散システムシンポジウム資料, 265/270 (1994)
- 8) 塩川光一郎: 分子発生学, 東京大学出版会 (1990)
- 9) 村田, ほか: 分散機械システムの設計 (第 5 報), 第 8 回自律分散システムシンポジウム資料, 307/312 (1996)
- 10) 小鍛治繁: 多自由度機構と分散制御, 精密工学会誌, 54-10, 1921/1926 (1988)
- 11) 富田, ほか: 分散機械システムの設計 (第 4 報), システム情報関連合同シンポジウム資料, 15/20 (1995)

[著者紹介]

村田 智 (正会員)

1985 年, 名古屋大学工学部航空学科卒業, 87 年同大学院修士課程修了. 同年工業技術院機械技術研究所入所. 分散機械システムの研究に従事. 92 年 IEEE-IE 論文賞, 96 年本会論文賞受賞. 日本機械学会, 日本ロボット学会, 電気学会などの会員.

富田 康治 (正会員)

1988 年, 筑波大学第三学群情報学類卒業, 90 年同大学院修士課程理工学研究科修了. 同年工業技術院機械技術研究所に入所し現在に至る. 物理情報部知識工学研究室にてマシンインテリジェンスの研究に従事. ACM, IEEE-CS, 情報処理学会などの会員.

黒河 治久 (正会員)

1978 年, 東京大学工学部精密機械工学科卒業, 81 年同大学院工学系修士課程修了. 同年工業技術院機械技術研究所入所. 宇宙における機構の制御および自律分散システムの研究に従事. 精密工学会, 日本機械学会, 日本航空宇宙学会, AIAA の会員.

繁（正会員）

1970年、東京大学工学部精密機械工学科卒業、72年同大学院工学系修士課程修了。同年工業技術院機械技術研究所入所。機械システムの分散制御を研究(工学博士)。精密工学会、情報処理学会、日本ロボット学会などの会員。

|||||