LF_003

# 木構造表現を用いたディジタル生命モデルの構築
# Construction of a Tree-Structured Digital Life Model

鈴木 輝彦† 梶田 友貴† 延澤 志保‡ 太原 育夫‡
Teruhiko Suzuki Yuuki Kajita Shiho Nobesawa Ikuo Tahara

## 1. はじめに

人工生命研究のひとつとしてディジタル生命 (ディジタル生物, Digital Organisms) に関する研究がある [3]. これは生命は計算であると考え，計算によって生命の振舞いを再現し，そこから生命の普遍的な性質の発見を目指す研究である.

これまでのディジタル生命の代表的な研究としては Tierra [6] や Avida [1] [2] が挙げられる. これらは計算機のマシン語を模した命令の 1 次元の列によって計算を表現している. これに対し，本稿では遺伝的プログラミング [4] のように木構造によって計算を表現するディジタル生命モデルを構築し，実験によってモデルの有効性を検証する.

## 2. 従来のディジタル生命モデルの問題点

ディジタル生命モデルにおいて，計算はその振舞いを決定するだけのものではなく，生命の複製の対象物となる実体を表現するものであり，遺伝的な作用を受ける要素でもある. このため，計算をどのように表現するかはディジタル生命モデルにおいて重要な問題である. Tierra や Avida に代表される従来のディジタル生命モデルの多くが計算機のマシン語を模した命令の 1 次元の列による表現方法を用いている. しかし，他の表現方法によるモデルとの比較がされておらず，この命令の 1 次元の列による表現方法が必ずしも生命の表現に適切な方法であるとは限らない.

さらに，これまでのディジタル生命モデルでは，モデルに明確に個体といった概念が存在し，細胞，器官，個体，生態系といったような生命の機能的な面での階層的な構造に対する表現力が乏しい. また，生命の重要な特徴である自己相似性を計算の表現方法として持っていない.

以上の問題点を踏まえ，新たなディジタル生命モデルを提案する.

## 3. 木構造表現を用いたディジタル生命モデル

### 3.1 設計方針

現存する生物は，個体は器官の機能の組み合わせによって機能しており，器官は細胞の機能の組み合わせによって機能しているといったように，小規模な機能を組み合わせて大規模な機能を構成している. 本研究では，生命が持つこのような性質に着目し，生命は機能の階層的な組み合わせによって構成されていると考える.

現存する生物において，生態系から見た個体の機能，個体から見た器官の機能，器官から見た細胞の機能といった機能の階層的関係は，互いに類似していることが多く見られる. つまり現存する生物は機能の階層的関係において自己相似性を持つといえる. このことから，モデルにおいても機能の階層的関係が自己相似的特徴を持つべきであると考える.

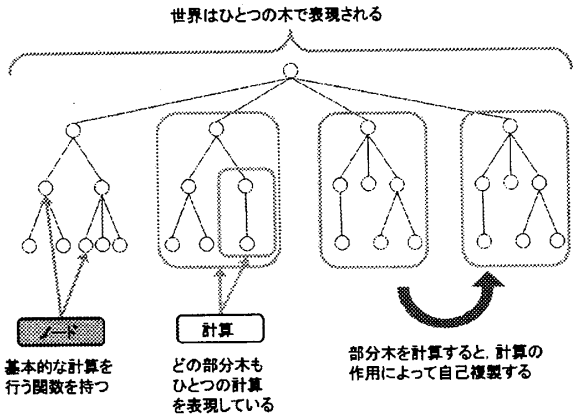


図 1: 本稿提案モデルにおける世界の表現

細胞，器官，個体，生態系といった機能的な階層は，人間が現存の生物を観測した結果作られた概念であり，すべての場合において個体が自己複製の単位であるといえないように，必ずしも生命の普遍的な特徴を分類できている訳ではない. そこで，モデルでは特別な性質をもった階層の機能が生命として特別な役割を担うのではなく，すべての階層が同一の性質を持つことにし，モデルが積極的に個体といった階層的分類を表現するのではなく，生命を再現し，観測した結果，観測者によって階層的分類を行なうものとする.

以上の性質を表現できることから，本モデルでは木構造によって計算を表現する. 計算を木構造で表現することで機能の利用，被利用の関係を直接表現し，計算の記述自体を操作する際に理論的構造が維持され，生命の機能を理論的に表現することができる.

### 3.2 本モデルにおける世界の表現

本モデルにおける世界は木によって表現されたひとつの計算で表現される (図 1). この木の中の部分木はそれぞれ独立して同時に計算することができる. この部分木が計算され，木を操作する計算によって自己複製を行ない，変化していくことで進化の再現を行なう. ただし，先述したように本モデルには個体といった概念はなく，階層的に均質に生命を表現する. よって様々な規模の部分木を生命の活動の単位としてとらえることができる.

### 3.3 木構造による計算の表現

本モデルでは遺伝的プログラミングのように計算を木構造で表現するが，ディジタル生命モデルで用いることを考慮して，木の変化に対して致命的な構造の誤りが生じないような方法で計算を表現する.

木の要素であるノードは機能を表現する最小の要素で，子ノードから与えられた計算結果を入力とし基本的な計

†東京理科大学大学院理工学研究科情報科学専攻
‡東京理科大学理工学部

表 1: 計算を構成する主な関数

| 関数名 | 機能 |
|---|---|
| parent | 親のノードを返す |
| children | 子のノードのリストを返す |
| start-node | 計算を開始したノードを返す |
| template | 検索したノードを返す |
| newnode | 新しいノードを作成する |
| connect | ノードとノードを再接続する |
| car | リストの先頭の要素を返す |
| cdr | 第 2 要素以降のリストを返す |
| cons | リストに要素を加える |
| nil | 常に空のリストを返す |
| if | 条件により行う計算を選択する |
| eq | 2 つのノードが等しいか否かを返す |
| not, and, or | 論理演算 |
| call | サブルーチンを呼び出す |
| arg$N$ $(N = 1, 2, 3)$ | サブルーチンの第 $N$ 引数を得る |
| start | 指定したノードに新たなプロセスを作成する |

算を行い，親ノードへ計算結果を出力する機能を持つ．このノードが持つ基本的な計算の手続きを関数と呼ぶことにし，子ノードからの計算結果を引数，親ノードに渡す値を関数値として考える．引数の値と関数値は，あるひとつのノードを示す抽象的な値かこの値のリストかさらにこれらのリストとする．また，ノードは任意の数の子ノードを持つことが可能とする．

関数には木の操作を行うものや，リストの計算を行うもの，計算の流れを制御するもの等を定義する．定義する主な関数を表 1 に示す．関数は任意の数の引数に対して計算の手続きが定義される．

### 3.4 プロセス

ひとつの計算の過程をプロセスと呼ぶ．プロセスは計算しているノードを行き来するノード間の信号と考えることができる．ノードは自身にプロセスが存在してはじめて実際に計算を行なう．あるノード上にプロセスが存在しており，引数の計算が行なわれていない場合，引数の計算を行なうために順に子ノードへプロセスを移す．引数の計算が行なわれていた場合，ノードは自身が持つ関数を計算し，その結果をプロセスに付加し，プロセスを親ノードへ移す．このようにしてプロセスがノード上を信号として移動することで実際に計算が行なわれる．

プロセスは関数 start によって任意のノード上に新たに作成することができる．また，同一のノード上に複数のプロセスが存在することも可能とする．この場合，同時にそれぞれのプロセスの計算が行なわれる．プロセスは計算を開始したノード以下の計算をすべて終えると削除される．

### 3.5 テンプレートによるノードの特定

モデルでの計算においてノードを柔軟に特定する方法として，Tierra が持つテンプレートによるアドレッシングを踏襲した方法を定める．本モデルではテンプレートをノードに対して一致か不一致かを定める概念として定義し，テンプレートに一致するノードを検索し，検索を開始したノードから最も木構造上での距離の小さいノードに注目することでノードの特定を行なう．テンプレートは関数 template の第 1 引数以下の部分木を計算として評価するのではなく，部分木として比較するための情報として扱うことで実現し，テンプレートと相補的に一致した部分木の根のノードに着目することでノードの特定を行う (図 2)．

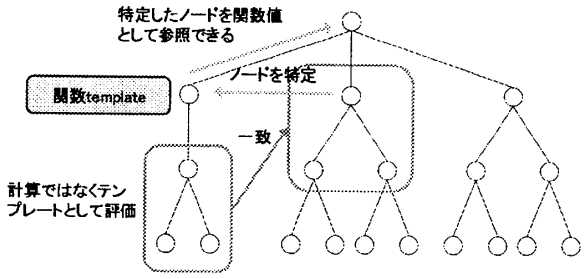


図 2: テンプレートによるノードの特定

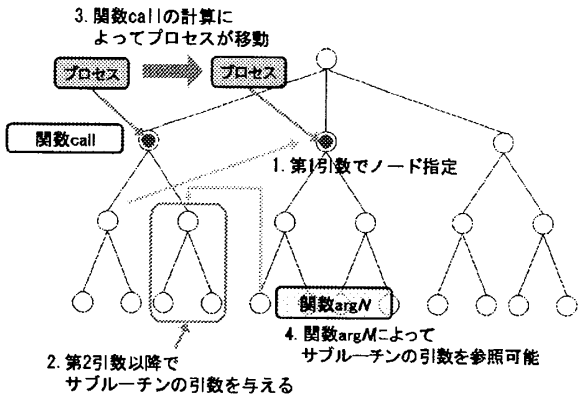


図 3: サブルーチンの実現方法

### 3.6 サブルーチン

計算機のプログラミング言語と同様に，繰り返し行なわれる計算を一箇所で記述し再利用するような計算の記述方法を実現するためにモデルでのサブルーチンよる計算方法を定める．進化的に計算が変化する際のサブルーチンの実現方法として遺伝的プログラミングにおける ADFs がある [5]．しかし，ADFs はサブルーチンが個体が持つ木と独立して存在しており，本モデルが目指す階層的に均質で自己相似性をもったモデルには適さない．

そのため，本モデルでは任意の部分木をサブルーチンとして計算することができる記述方法を考案した．サブルーチンの呼出しを行なう関数 call が計算されることで第 2 引数以降をサブルーチンの引数として，第 1 引数で示されたノードをサブルーチンとして計算する (図 3)．ここで他の関数と同様に任意の位置に存在できる関数 arg$N(N = 1, 2, 3)$ がサブルーチンとして計算されている際に計算されることでサブルーチンの引数を得ることができる．このようにすることでディジタル生命モデルに有効な柔軟な記述方法でサブルーチンを実現している．ノードの親子関係が機能の階層的構造を表現しているのに対し，サブルーチンは機能間の相互作用であるといえる．

### 3.7 不確実性

生命を構成する機能という視点から見た環境は，現存する生物を見ても分かるように，確定的な振舞いをする

とはいえない．本モデルでは木に対して無作為な変更を行なうことで環境の不確実性を再現する．具体的には，プロセスが作成された際にプロセスを作成したノード以下の部分木に対して一定の確率で変更を行なう．変更は部分木の削除，ノードの削除，ノードの挿入，ランダムに作成した新しい木の挿入，ふたつの部分木の交換，ノードの関数のランダムな変更をそれぞれ独立に一定の確率で行なう．変更をプロセス作成時に限定しているのは，変更を行なった時点で作成したプロセスに解析用の情報を付加することができ，任意の時点で変更を行なうことと実質的には差がないためである．

このようにモデルの世界の不確実性を再現することで結果として進化を再現する上での突然変異の役割を果たすことが期待できる．

### 3.8 資源

計算時間とノードをモデルの世界における資源として考え，これらに資源としての性質を定める．

計算時間については，各関数にその関数を計算するのに必要な時間を定義する．計算に必要な時間は木構造上での空間的な作用が小さいものほど短く，大きいものほど長く定義される．空間的な作用がない関数の計算に必要な時間を 1 としてモデルの世界での時間の単位とする．ここで空間的な作用が大きいとは，関数の何らか操作の対象となるノードと現在計算しているノードとの木構造上での距離が大きいことをいう．

このように空間的作用と時間的な資源の関係を定義することで空間的作用の大きさに対するコストが生じるようになっている．

また，ノードはモデルの世界を構成する最小の要素であることから，ノードを物質的な資源として考え，世界に存在できるノードの数に制限を設定する．具体的には，単位時間あたりに全体で作成できるノードの数を制限し，この制限を超えるとノードの作成を不可能にする．また，親子の接続の状態が一定時間変化しなければ削除する．

このようにノードの数を制限することで世界に存在することのできる木の数が制限され，結果として進化を再現する上での淘汰の圧力になることが期待できる．

### 3.9 種

あるふたつの同じ内容の部分木を計算したときの振舞いは，その部分木以外の木の状態など，外的要因を除けばほぼ同じであるといえる．そのため計算の振舞いが同じものを識別するため，計算が開始された，つまりプロセスが作成された時点での部分木の内容を種と呼ぶことにする．

種はあくまで同等の振舞いをする計算を識別するもので，必ずしも複製によって誕生した同じ振舞いをする生命の集まりを識別するとは限らない．便宜上，種と呼んでいるが現存の生物の種にのみ対応しているわけではなく，タンパク質や酵素，細胞の種類といったものにも対応させて考えることができる．

## 4. 本モデルの検証実験

### 4.1 実験の目的

本モデルが生命を再現する能力を持っているかの検証を行うため実験を行った．検証の方法は進化を再現できているか，生命の多様性を再現できているか，その他に現存する生命と類似した何らかの現象が見られるかに着目して行なう．

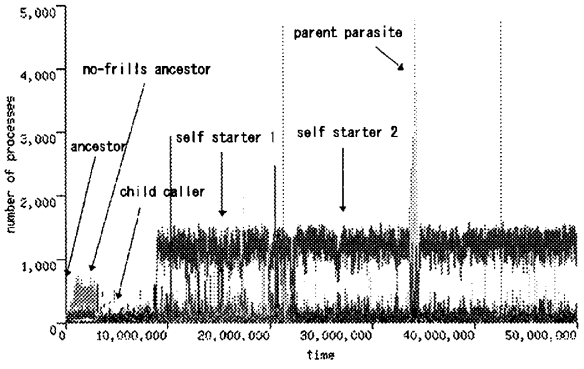


図 4: 種ごとのプロセス数

### 4.2 実験方法

実験の初期設定として，ancestor と呼ぶ自己複製を行なう種とこれを計算するプロセスを用意する．ancestor が持つ木は自分の兄弟に当たる位置に自身と同じ木を複製し，複製した木に対してプロセスを作成するといったことを 2 度行う計算になっている．

ancestor が自己複製によって増加した後，ノードの数の制限によって存在できる部分木の数が制限され，これが進化における淘汰の圧力になる．またプロセスの作成の際に起きる木のランダムな変化が突然変異の役割を果たす．これらによって進化を再現し，生命の振舞いの再現を行なう．

プロセスが作成された際に木に変更を行なう確率を $10^{-3}$ に設定し，モデルにおける時刻 $50 \times 10^6$ まで実験を行った．

### 4.3 結果

複数の乱数の系列で実験したところ，多くの場合で同様の結果が出たため代表的な結果を挙げる．

種ごとのプロセスの数を図 4 に示す．この図は同じ種を計算するプロセスの数を種ごとの線で表し，縦軸がプロセスの数，横軸が時刻を表している．

この図からプロセスの数の多い種が時刻とともに変化しており，種が次第に変化していく様子が読みとれる．また，多くの時刻でひとつの種のプロセスの数が多く，ひとつの種が支配的であることが観測できた．

種ごとのプロセスの数に着目した際の木の変化によって誕生した代表的な種を挙げると以下のようになる．

- no-frills ancestor
  ancestor が持つ計算のうち，省略しても計算としては等価になる引数など，自己複製をする上で冗長な部分木が無くなった種．ancestor と比較してノードの数が少ないため，木の複製に必要な時間が短くなっており，適応的になっている．

- child caller
  no-frills ancestor から派生し，複製した部分木をサブルーチンとして計算するようになった種．複製した木をさらに計算することで自己複製を繰りかえ

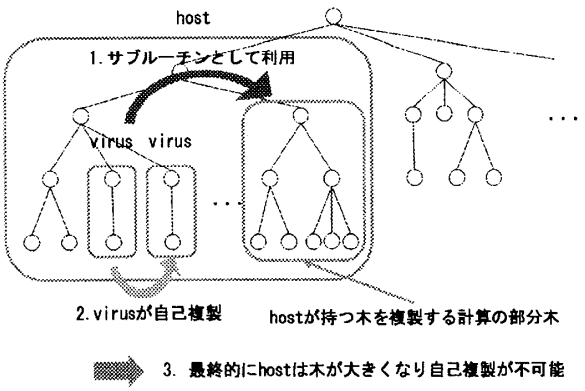


図 5: host と virus

すようになり，no-frills ancestor よりもより多くの木とプロセスを作成でき，より適応的になっている．

- self starter 1，2
  self starter 1 は自身の根のノードにプロセスを作成する計算が追加された種．このため，自身の木を計算するプロセスが増えて行き，プロセスの数だけ同時に自己複製が行なわれるので，より多く自己複製を行なう．self starter 2 は self starter 1 と本質的に同様の計算だが，これまでの種が縦型検索で木を複製していたのに対し，横型検索になっている点が異なる．

- parent parasite
  自身の木の根の親のノードをサブルーチンとして呼び出すのみで，このサブルーチン中に木の複製を行なう計算が含まれていれば自己複製を行なうことができる種．自身が持つノードの数が他の種と比べて極端に少ないため，少ない資源で自己複製を行なうことができる．

また，乱数の系列の異なる他の実験では，host と呼ぶ種の木の中で自己複製を行なう virus と呼ぶ種が誕生した．virus は host の木の中の小さい木として存在し，この木には自己複製を行なうための計算は含まれておらず，host が持つ計算をサブルーチンとして呼び出して自己複製を行なう．virus は複製する木が小さいので高速に自己複製を行ない，host は大量に自己複製された virus を含んだ自身の木を自己複製することができなくなる (図 5)．この種は構造が単純であること，自身だけでは自己複製ができないこと，host に害をあたえるという点で現実に存在するウィルスと共通の特徴を持つ．この host と virus のような現象はいくつかの実験の複数の時点で観測することができた．

### 4.4 考察

初期に用意した計算と比べ，より適応的な計算を行なう種が発生し，進化が再現できているといえる．特に，任意の部分木をサブルーチンとして呼び出すことのできる仕様と，任意のノードから計算を開始させることができる仕様のために，計算の流れを変化させてより適応的な計算に変化している例が多く見られた．

しかし，多くの時刻においてひとつの種が支配的で，多様性を再現できているとはいえない．この理由として自己複製を成功させるにはノードを得ることのみが必要で，資源であるノードをすべてのプロセスが同じ条件で奪い合っており，プロセスを多く作成しノードを得る確率を高くするか，自身が持つ木を小さくし，すべてのノードを得ることができる確率を高くするという戦略が適応的であるという点がある．つまり資源を有効に得る方法が限られており，適応的な戦略も限られているため同様の計算を持つ種が支配的になっている．さらに，種が持つ木は小さいほど自己複製に必要な時間が短く，適応的になるため，冗長な部分が変化し新たな機能を持つといった構成的な変化による進化が起こりにくいことも理由として挙げられる．

virus が host の木の中で増加することで host の自己複製を不可能にするといった振舞いは，virus の自己複製するという機能を host が自身の木を大きくするという非適応的な機能として利用するという，機能の階層的構造の作用である．つまり，これは本モデルのように計算を木構造によって表現したことによって現存する生物にあるような機能の階層的構造を表現できた一例である．

これまでの説明では自己複製する部分木の計算を個体のようにとらえてきたが，全体の木をひとつの個体とみなし，実験全体を発生の過程であるととらえることもできる．これは本モデルの大きな特徴である．

## 5. 結論と課題

階層的構造を持った計算の表現方法による新たなモデルを提案することができた．また，このモデルの有効性を実験により検証した．その結果，階層的構造の表現が可能で，生命の機能の組み合わせに着目して進化を分析できること，計算を理論的な構造で表現しているため，計算の振舞いが予測し易く，分類が容易になるということが示された．課題としては，他のモデルで再現できている多様性にはまだ及ばず，現時点では計算の表現方法の違いによる比較ができていないといったことが挙げられる．

## 参考文献

[1] C. Adami and C. T. Brown, "Evolutionary Learning in the 2D Artificial Life System 'Avida'," *Artificial Life IV*, pp.377–381, 1994.

[2] C. Adami, *Introduction to Artificial Life*, Springer, pp.225–247, 1998.

[3] 有田 隆也, 人工生命, 医学出版, 2002.

[4] John R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[5] John R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994.

[6] T. S. Ray, "An Approach to the Synthesis of Life," *Artificial Life II*, pp. 371–408, 1992.