F-009

機能の利用関係によるディジタル生命モデルの構築
Construction of a Digital Life Model Based on Functional Relation

鈴木 輝彦† 太原 育夫‡
Teruhiko Suzuki 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) に関する研究が行なわれている [3]. これは生命は計算で表現可能であるという立場から, 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モデルとしては Tierra[4] や Avida[1][2] が挙げらる. また, 著者らは計算を木構造で表現したディジタル生命モデルを構築し, 検証を行なってきた [5][6]. 本稿では従来のモデルの特徴を考察した上で, 新たに機能の利用関係によるディジタル生命モデルを提案する.

2. 従来のディジタル生命モデル

生命を計算で表現するディジタル生命モデルにおいて, 計算の記述は生命の振舞いを決定するものであり, 生命だけでなく生命が存在する世界の振舞いを定義付けることも多い. さらに, 多くのモデルで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を構築する上で重要な問題である. 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機械語を模した命令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.

また, Tierra や Avida におけるメモリの値の変更のように, 生命が計算の記述に対して何らかの操作を行うことを考えた場合, 命令の列による表現方法ではあるひとつの理論的にまとまった計算に対して操作を行うことが困難である. すなわち, 生命が高次な機能に対して何らか作用を及ぼすことが困難な仕様になっている. これは, 現存する生物においてあるまとまった機能はその実体も物理的にまとまっており, 機能のまとまりとして扱うことができることから, 適切な仕様とはいえず, 進化の再現においては機能の組み合わせよってより大きな機能が構成されるような創発を妨げているといえる.

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定義されている. Tierra や Avida においては CPU と書き込み可能なメモリ領域 (またはセル) によって個体が定義されている. つまり, モデルの世界における最小の構成要素である命令を用いて高次な計算を構成する枠組みがモデルによって用意されている. しかし, 個体に限らず, 細胞, 器官, 生態系といった機能的な階層の分類は人間が現存する生物を観測した結果作られた概念であり, 生物学において個体といった概念を明確に定義できないように, 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. すなわち, ディジタル生命モデルにおいて個体のような概念をあらかじめ定義することは機能の柔軟な階層的構造が創発することを妨げているといえる.

著者らが提案した木構造表現を用いたモデルは, 生命を原始的な機能を要素とする木構造としてとらえ, 機能の利用関係のみに着目してモデル化した. このため進化の再現において空間的な局所性がなく, 適応的な計算は即座に全体に広がるため, 大きな多様性が生じなかった. また, 同モデルでは木構造の親子関係によって機能の利用関係を構造化して表現したが, サブルーチンの特殊な表現方法により, 機能として利用されているはずのサブルーチンが, 部分木として存在するためだけに親子関係になっており, 完全には機能の利用関係を表現しているとはいえなかった.

そこで, 本稿ではこれらの問題点を解決し, 厳密に機能の利用関係を構造化したモデルを構築する.

3. 機能の利用関係によるモデル

3.1 基本的思想

本稿では, 生命を観測者が振舞いを予測できないほど複雑になった機能であると考え, ディジタル生命を, 生命を抽象化した上で進化を再現し, 進化によって生命の条件である振舞いの予想の困難さを増加させ, 生命を観測させるための手段と考える. 観測によって生命と判断された場合, この判断の基準を, 完全に振舞いが予想できる単なる計算としての立場から探ることで, 生命の性質を発見し, 本研究の生命の定義が生命科学において有効なものであることを示すことが本研究の基本的思想であり, 最終的な目標である.

現存する生物の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, 現存する生物は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. つまり生命の機能の複雑さは機能が組み合わされることによって成立している. 本モデルでは, 生命が持つこの性質に着目し, 生命は機能とその利用関係のネットワークによって表現できると考える.

機能の利用関係を抽象化するという立場から現存の生物を考察すると, 個体があるふたつの異なる器官を持っていたとして, これらふたつの器官を利用するには, 利用する方法に加え, 利用するために必要な資源も異なる. そこで, 本モデルでは機能の利用にかかるコストの概念を定義する.

また, 現存する生物において細胞同士が機能を利用するコストよりも器官同士が機能を利用するコストの方が大きいように, 規模の小さい機能の間での利用にかかるコストより規模の大きい機能の間での利用にかかるコストの方が大きいといえる. これにより, モデルに個体や細胞といった固定的な階層の概念を定義することなく生

†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

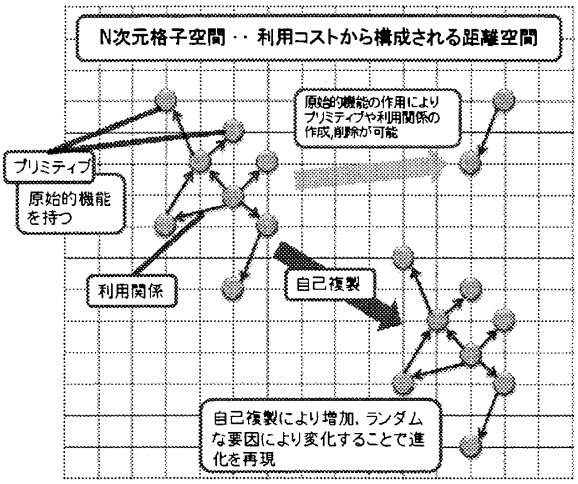

図 1: 本モデルの概要

命の機能の規模を利用のコストの大きさによって説明することができると考える．

3.2 本モデルの概要

本モデルの概要を図 1 に示す．本モデルでは原始的な機能を持つプリミティブと呼ぶ要素の有向ネットワークにより生命の機能とその利用関係を表現する．有向ネットワークのアークは利用関係にあることを表現し，アークの方向は利用するプリミティブと利用されるプリミティブを表す．任意のふたつのプリミティブ間で利用関係の有無に関わらず利用コストが定義されており，利用コストによってプリミティブは任意の次元の格子空間上に配置されていると考えることができる．この利用コストによる距離空間はあくまで利用コストによって構成されたものであり，現実の物理的空間を抽象化したものではない．

プリミティブが持つ原始的機能にはプリミティブの作成や利用関係の操作をする作用を持つものがある．これらを組み合わせることで自分自身のネットワークを複製する，すなわち自己複製を行なうネットワークを構成することができる．この自己複製を行なうネットワークをはじめに用意し，自己複製が行なわれ，ネットワークが増加し，モデルのランダムな作用によって変化し，淘汰されていくことで進化を再現し，本稿の生命の定義である複雑な機能の再現を目指す．

3.3 プリミティブ

本モデルでは機能の記述ではなく，機能として実際に動作するものが生命を表わす概念になる．そこでモデルに機能として動作するものの基本的な概念としてプリミティブと呼ぶ概念を定義する．プリミティブは原始的な機能として動作する機械のような概念である．原始的な機能とは記号で表現された入力によって出力となる記号を決定し，同時にモデルにおける世界の状態を変化させる作用を伴うものとする．この原始的な機能を説明の便宜上，関数と呼ぶことにする．

プリミティブを利用するとはプリミティブが持つ関数によって出力として決定された記号を得て，さらにその関数が持つ世界の状態への作用を実際に引き起こすことである．また，プリミティブがプリミティブを利用するとは，あるプリミティブが利用の対象となる他のプリミティブを動作させ，その出力を自身の入力のひとつとして得ることとする．プリミティブが複数のプリミティブを利用することで得た複数の入力はそれぞれを識別することができ，この識別の種類のことを利用方法または引数と呼ぶ．

関数にはプリミティブやその関係を操作する作用を持つものがあり，プリミティブやその関係は機能をモデル化しただけのものでなく，機能を変化させるという作用の対象となる．

先述したように，生物が持つ様々な機能の利用関係について，それぞれの機能の利用にかかるコストは同一でない．そのため，本モデルにおいてもこれに従い，プリミティブ間の利用にかかるコストを，あるプリミティブの出力があるプリミティブに伝わるのに必要な時間として各プリミティブ間に定義する．

3.4 プリミティブ間の関係

本モデルでは機能の構成をプリミティブとそれらの関係で表現する．プリミティブ間の関係は以下によって構成される．

- 利用コスト (任意のプリミティブ間で定義)
- 利用状態 (利用する，利用していないのいずれか)
- 利用方法 (利用している場合のみ定義)

そこでこれらの詳細を，プリミティブ自身の操作を行なう関数がプリミティブ間の関係を決定する方法とともに定義する．

利用コストは非負，同一，対象，三角不等式の距離の性質を満たすのが自然と考え，これに従うものとする．またモデル上で表現する機能を簡単にするため利用コストは非負の整数とする．ここで利用コストをマンハッタン距離として考えると，プリミティブは N 次元の格子空間上に配置されていると考えることができる．ただし N は任意の非負の整数である．関数の作用によってプリミティブを新たに作成する等のプリミティブ間の利用コストを決定する必要がある場合は，N 次元の格子空間上での点を示すことで決定する．このような方法を用いることで任意のプリミティブ間で利用コストを距離の性質を満たしたまま決定することができる．

利用状態については，プリミティブが新たに作成された際には，他のどのプリミティブとも利用する関係にも利用される関係にもなっていないものとする．利用状態を設定する作用を持つ関数を動作させることで利用状態を設定する．

利用方法は，各プリミティブに N 次元の格子空間上での方向の概念を定義し，これを基準に利用するプリミティブから見た利用されるプリミティブの方向によって定義する．以降，利用方法をすべてのプリミティブに共通で第 1 引数，第 2 引数... と呼ぶことにする．2 次元格子空間の場合の利用方法の定義例を図 2 に示す．ここで n 引数関数とは n 種類の引数を参照する関数とする．同じ引数の方向に複数のプリミティブが存在する場合は利用コストが最も小さいプリミティブを利用するものとする．プリミティブが持つ方向は関数の作用によって変更できるので，利用コストの空間のある方向がプリミティブの向きに対して特別な意味を持つことはない．

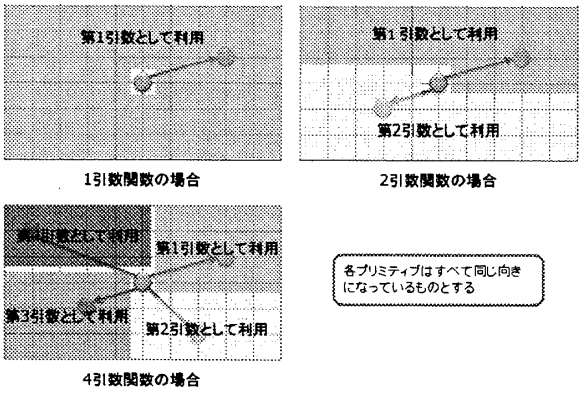

図 2: 2 次元格子空間の利用方法の定義例

利用コストは距離の同一の性質を満たすので空間上のひとつの点に異なるプリミティブが存在することはできない．よってあるプリミティブに対して，ある利用方法，ある利用コストとして存在できるプリミティブの数は制限され，利用コストが小さいほど制限されるプリミティブの数も小さくなる．

このように空間上にプリミティブが配置されていると考えることで，容易にプリミティブ間の関係を決定することができる．また，機能間の明確な関係が定義された空間上での操作を定義することで操作の作用自体に機能としての意味を持たせることができる．

3.5 モデルにおける資源

プリミティブはプリミティブが存在しない空間上の点であれば，関数の作用により作成することができる．作成されたプリミティブは一定の時間が経過することで削除される．よって本モデルにおけるプリミティブは生命から見た活動を行なうための資源であるといえる．また，プリミティブが存在しない空間上の点は新たにプリミティブを作成できる，自己複製を行なう上での資源であるといえる．

3.6 不確実性

本モデルでは生命の機能に着目した抽象化をしているが，現存する生物を見ても分かるように，生命には生命以外からの作用もあたえられる．そこで生命の機能を構成する要素に対する生命以外からの作用として，無作為にプリミティブの作成と削除，プリミティブの関数の種類，向き，利用関係の変更を行なう．このような非確定的な作用を再現することで，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．

3.7 プリミティブ動作時の文脈

利用関係によって動作する一連のプリミティブが同じ情報を共有することを可能にするため，プリミティブ動作時の文脈として，関数が扱う値を要素としたスタックを用意する．これにより，スタックによって与えられた値を作用の対象とするような機能があった場合，スタックの値を変更し再度この機能を利用することで，機能を再利用するような機能の表現が可能になる．

3.8 プリミティブが持つ関数

プリミティブが持つ関数が入力，出力とする値は各プリミティブを示す抽象的な値，整数，真偽値，これらのリスト (リストのリストも含む) とする．Tierra や Avida では計算で扱う整数値，アドレス，命令をすべて整数で表現し，同じ値で異なる概念を根拠なく対応づけていた．本モデルでは，異なる概念は異なる値として表現する．

プリミティブが持つ関数の定義を付録表 1 に示す．ここで，ベクトルとは利用コストの空間の次元だけの要素を持つ整数のリストである．

これらの関数は意図していたものとは異なる引数があたえられた場合でも必ず何らかの値は出力し，無作為な変更が行なわれても動作が停止するようなことはない．

4. モデルの検証

本モデルが生命を再現する能力を持っているかを検証するため，実験を行う必要がある．実験は進化を再現し，複雑な機能を発生させることで行なう．実験において実際に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，創発的な現象が起きているか，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，これらの現象が本モデルの抽象化によるものかを考察することでモデルの検証を行なう．

実験は以下の方法で行なう．初期状態として，自己複製を 2 度行なうネットワークを用意し，起点となるプリミティブを動作させる．自己複製は自身の基準となるプリミティブから一定の利用コスト以下となるすべてのプリミティブを取得し，これらのプリミティブから一定の量だけ移動した位置にプリミティブを新たに作成し，自身と同じように利用関係を設定することでプリミティブとそれらの利用関係を複製し，最後に動作の起点となるプリミティブを動作させることによって行なう．複製の対象となる自分自身という概念を利用コストによって認識している点は本モデルの特徴によるものである．2 次元格子空間の場合の実験の初期状態の例を図 3 に示す．各ますが空間上の点を表し，ます内にプリミティブが持つ関数の関数名が表記されている．矢印は始点が利用するプリミティブを，終点が利用されるプリミティブを表している．利用方法の定義は図 2 に示したものを使用しており，プリミティブの向きはすべて図 2 と同じものとする．

自己複製によってネットワークが増加した後，空間上の制限によって存在できるネットワークの数が制限され，これが進化における淘汰の圧力になる．また非確定的な作用によるプリミティブの状態の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目指す．

5. おわりに

本稿では機能の利用関係によるディジタル生命モデルの構築を行なった．本モデルではすべての機能の間に，利用コストという距離の性質を満たす関係が定義されている．様々な大きさの利用コストによってネットワークを分割したものを様々な機能の規模として考えることによって，生命が持つ細胞，器官，個体といったような機能の階層的構造と高次な利用関係を柔軟に説明できることが期待できる．

また，利用コストで隔てられた空間によって局所性が再現され，生命が持つ多様性が再現されることが予想できる．すなわち多様性についても利用コストによって説

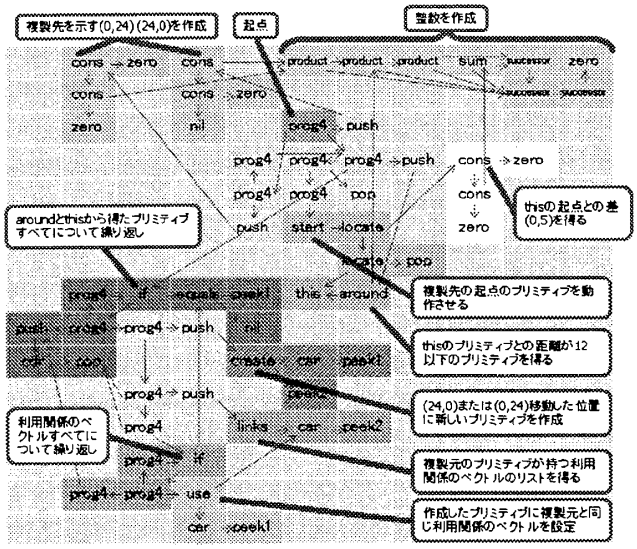

図 3: 2 次元格子空間の実験の初期状態の例

明できることが期待できる.

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] T. S. Ray, "An Approach to the Synthesis of Life," *Artificial Life II*, pp. 371–408, 1991.

[5] 鈴木 輝彦, 延澤 志保, 太原 育夫, "木構造表現を用いた人工生命," 電子情報通信学会 2004 年 総合大会, vol.D-I, no.D-8-18, p.105, 東京工業大学, Mar. 2004.

[6] 鈴木 輝彦, 梶田 友貴, 延澤 志保, 太原 育夫, "木構造表現を用いたディジタル生命モデルの構築," 第 5 回 情報科学技術フォーラム (FIT2006), no.LF-003, pp.101–104, 福岡大学, Sep. 2006.

A 付録

表 1: 関数の定義

関数名	引数の数	機能の内容
zero	0	整数値 0 を返す.
successor	1	引数の値に 1 加えた値を返す.
predecessor	1	引数の値に-1 加えた値を返す.
sum	2	2 つの引数の和を返す.
difference	2	2 つの引数の差を返す.
product	2	2 つの引数の積を返す.
quotient	2	2 つの引数の商を返す.
nil	0	空のリストを返す.
car	1	引数で与えられたリストの先頭の要素を返す.
cdr	1	引数で与えられたリストの第 2 要素以降のリストを返す.
cons	2	第 2 引数で与えられたリストに第 1 引数の値を先頭の要素として追加したリストを返す.
equals	2	第 1 引数の値と第 2 引数の値が等しい場合は真, そうでない場合は偽を返す.
not	1	第 1 引数の値の論理否定を返す.
and	2	第 1 引数の値と第 2 引数の値の論理積を返す.
or	2	第 1 引数の値と第 2 引数の値の論理和を返す.
push	1	引数の値をスタックに追加する.
pop	0	スタックから値を取り出す.
peek$N$	0	スタックの上から N 番目の値を関数値として返す. スタック自体は変化しない. $(N = 1, 2, ..., 5)$
this	0	自身のプリミティブを示す値を返す.
around	2	第 2 引数で与えられたプリミティブから第 1 引数で与えられた距離内にあるプリミティブのリストを返す. リストは距離の順になる.
locate	2	第 2 引数で与えられたプリミティブから第 1 引数で与えられたベクトルだけ移動した位置にあるプリミティブを示す値を返す.
search	2	第 2 引数で与えられたプリミティブから第 1 引数で与えられたベクトルだけ移動した位置から最も距離の小さい位置にあるプリミティブを示す値を返す.
create	2	第 1 引数で与えられたプリミティブと同じ関数, 同じ向きのプリミティブを作成し, そのプリミティブを示す値を返す. 作成する位置は第 1 引数で与えられたプリミティブの位置に第 2 引数で与えられたベクトルを加えた位置になる.
move	2	第 1 引数で与えられたプリミティブを第 2 引数で与えられたベクトルの分だけ移動する.
rotate	1	第 1 引数で与えられたプリミティブの引数の分割の方向を変更する.
prog4	4	それぞれの引数のプリミティブを動作させ, 最後の引数の値を返す. すべての引数が与えられていない場合も, 与えられた引数の動作は行われる.
if	4	第 1 引数が真のとき, 第 2 引数として利用しているプリミティブを, そうでない場合第 3 引数として利用しているプリミティブを動作させ, その値を返す. 第 4 引数は動作させず, 無視される.
use	2	第 1 引数で与えられたプリミティブが第 2 引数で与えられたベクトルで表された利用関係を持つようにし, 第 1 引数の値を返す.
links	1	引数で与えられたプリミティブが持つ利用関係をベクトルのリストとして返す.
start	1	引数で与えられたプリミティブを動作させる.