数理解析研究所講究録 1381 巻 2004 年 65-79

ラムダゲーム：メタゲームへのアプローチ

The Lambda Game System : an approach to a meta-game

舛本 現 (MASUMOTO Gen)
京都産業大学 経済学部 (Faculty of Economics, Kyoto Sangyo University)

1 はじめに

人工生命 (Artificial Life) の研究においてはゲームにおける戦略の open-end な進化は重要なテーマであり，これまでも数多くの研究がなされてきた．しかしルールが rigid に決まり，そのなかでの最適な行動が存在するのなら open-end な時間発展など不可能であり，あとに残るのは，本当は最適な戦略があるのだが人間の合理性や計算能力が限られているからそこに到達しえない，という問題だけになってしまう．そこで，多くの open-end な時間発展を扱う研究では，戦略の空間に openness を (非常に限定されたかたちで) 導入した．それはある時は記憶戦略における記憶の長さであり [11][14], また各個体のつくる相手のモデルの決まらなさであったりした [12][13][16].

しかし他方にはルールにこそ戦略を open-end に発展させる源があるという見方はできないだろうか．これはつまりルールが open であることこそが戦略の open-endedness を生み出すという立場である．ルールの openness を導入した過去のモデルでは，例えば Howard の meta-game theory のように囚人のジレンマにおいて「あなたが協調するなら私も協調し，あなたが裏切るなら私も裏切る」といったルールに言及するメタ戦略を (one-shot でも) 許すような枠組を用いたり [8][9][10], あるいは Hofstadter が紹介した Nomic というゲームのようにすべてのルールを明示的に自然言語で書き，それを更新していくゲームを考えたりすることが行われる [7][17][18].

ここで考えているルールが開かれたゲームとは，言わば遊び (プレイ) とでも呼ぶべきものであり，そこでは，ルールは一応決まっているはずなのだが，ルールでは想定されていない新しい戦略が次々と生み出され，それに対してルールを強引に解釈することによってそれでもなんとか遊びは続いていくような状況のことである．たとえば，子供はゲームのような行動をとることは少なく，ほとんどすべての状況においてここで遊びと呼んでいるような interaction をしているのでないだろうか．例えばじゃんけんはゲーム論的に考えると 3×3 の利得行列ですべてが表わされるゼロサムゲームであるが，子供のじゃんけんにおいては，グーチョキパーを同時に表現してるから無敵だと称する奇妙な手の握りや，繰り返しのじゃんけんでは後出

し何回で一回分の負けなどといったローカルルールがどんどん生み出される.

ではルールの openness というのはどのようにして可能になるのであろうか. ルールは戦略の定義域を規定してそれらの組に対する利得を決定し, 各プレイヤーはルールの定義域の範囲内の戦略で利得の最適化を図る. このような点において, ここで考えているゲームにおけるルールと戦略というのは, 関数と変数のような関係にあるといえよう. 変数 (戦略) は決して関数 (ルール) の規定する定義域の外に出ることはできないし, 関数 (ルール) も定義域の外の入力 (戦略) に対しては出力 (得点) を与えることはできない. 一方, プレイでは定義域が開かれているので, ルールは定義域の外の入力に対しても出力を返さなくてはいけない. ここで要求されているのは, 定義域を限定せずにどんな入力に対してもとりあえずは出力を返す体系である. そのためには定義域のクラスが関数自体のクラスと同じくらい広く, 関数と変数, すなわちルールと戦略, が同じフォーマットで書かれている type-free なものでなければいけない.

本研究ではゲームを記述するために (型のない)λ 計算を導入する. この λ 計算は type-free な計算体系であり, その要素である λ 式には関数と変数の区別がない. つまりどんなふたつの λ 式 M と N をとってきても N を M に代入することが出来る. さらに λ 計算ではゲームの利得表を記述するのに必要な真偽値や if 文, 自然数を表わすことができる. また, ここで論じたのと同じように λ 式の関数/変数の duality に注目してそれを化学反応における分子の duality に適用した有名な研究に Fontana と Buss により提案された Algorithmic Chemistry(Alchemy) がある [4][5][6][15].

2 λ ゲーム

λ 計算では, 真偽値, 条件文や自然数をコードするには様々な方法が知られているが, ここでは以下の Barendregt による表現 [2] を使ってゲームを記述することを考える.

- Boolean values

$$\mathbf{T} \equiv \lambda xy.x \qquad \mathbf{F} \equiv \lambda xy.y \tag{1}$$

- Conditional

$$\text{“if } x \text{ then } P \text{ else } Q\text{”} \equiv \lambda x.xPQ \tag{2}$$

$$\begin{aligned}(\lambda x.xPQ)\,\mathbf{T} &\to \mathbf{T}\,P\,Q \to (\lambda xy.x)\,P\,Q \to (\lambda y.P)Q \;\to\; P\\ (\lambda x.xPQ)\,\mathbf{F} &\to \mathbf{F}\,P\,Q \to (\lambda xy.y)\,P\,Q \to (\lambda y.y)Q \;\to\; Q\end{aligned}$$

- Numerals

$$\mathbf{0} \equiv \lambda x.x \;\; (=\mathbf{I}), \tag{3}$$

$$\mathbf{n+1} \equiv \lambda x\,.\,x\mathbf{Fn}, \tag{4}$$

$$\mathbf{succ} \equiv \lambda xy.y(\lambda xy.y)x \quad (= \lambda xy.y\mathbf{F}x), \tag{5}$$

$$\mathbf{pred} \equiv \lambda x.x(\lambda xy.y) \quad (= \lambda x.x\mathbf{F}), \tag{6}$$

$$\mathbf{iszero} \equiv \lambda x.x(\lambda xy.x) \quad (= \lambda x.x\mathbf{T}). \tag{7}$$

図1の左図のような協調(C)と裏切り(D)のふたつの選択肢をもつ一回限りの囚人のジレンマゲームを考える．ここでゲームのルールに要求されていることは，各プレイヤーの戦略の組に対して，それらを区別してそれぞれの場合に応じて利得表に従った数を返すことである．すなわちゲームのルールは図1の右図のようにif文を2回使うことによって各プレイヤーの戦略を区別し，場合分けをして，それぞれの利得を決定している．

そこで，戦略についてはC(協調)を $\mathbf{T}(= \lambda xy.x)$ で表し，D(裏切り)を $\mathbf{F}(= \lambda xy.y)$ で表すことにして，この真偽値 $\mathbf{T}, \mathbf{F}$ をゲームの戦略C，Dと解釈できるようなλ式 $\mathbf{G}$ を構成する．このλ式 $\mathbf{G}$ に要求されていることは，ふたつの戦略をλ式で与えられると，そのふたつの戦略の対戦をシミュレートして，それの戦略の組に対応した利得を返すことである．そこで利得はBarendregt数で表現することにすると，求めるλ式 $\mathbf{G}$ は式(8)のように構成できる．以下では，ゲームのルールを表現していて各戦略に対して利得を与える関数という意味でこのλ式 $\mathbf{G}$ をgame masterと呼ぶ，

$$\mathbf{G} \equiv \lambda x.x(\lambda y.y\,\mathbf{3}\,\mathbf{5})(\lambda y.y\,\mathbf{0}\,\mathbf{1}) \tag{8}$$

$$\begin{aligned} \equiv\ & \lambda x.x(\lambda y.y\underline{(\lambda z.z\mathbf{F}(\lambda z.z\mathbf{F}(\lambda z.z\mathbf{FI})))}_{\mathbf{3}} \\ & \qquad \underline{(\lambda z.z\mathbf{F}(\lambda z.z\mathbf{F}(\lambda z.z\mathbf{F}(\lambda z.z\mathbf{F}(\lambda z.z\mathbf{FI})))))}_{\mathbf{5}} \\ & (\lambda y.y\ \underline{\mathbf{I}}_{\mathbf{0}} \\ & \qquad \underline{(\lambda z.z\mathbf{FI})}_{\mathbf{1}}) \end{aligned} \tag{9}$$

2つの戦略 S_{self}，S_{opp} が与えられたとき，それぞれの戦略のとる利得は，$\mathbf{G}$ にこれら S_{self}，S_{opp} を代入することで得られる．

$$\mathbf{G}S_{self}S_{opp} \to \mathbf{G}'_{self}S_{opp} \to Reward_{opp} \tag{10}$$

$$\mathbf{G}S_{opp}S_{self} \to \mathbf{G}'_{opp}S_{self} \to Reward_{self} \tag{11}$$

例えばDとCが対戦したときのDの利得は，$\mathbf{GTF} \to (\lambda y.y\mathbf{35})\mathbf{F} \to \mathbf{5}$ と計算される．他の戦略の組に対しても，$\mathbf{GTT} \to \mathbf{3}$，$\mathbf{GFT} \to \mathbf{0}$，$\mathbf{GFF} \to \mathbf{1}$ となり，このgame master($\mathbf{G}$)が図1の利得表を表わしていることがわかる．

3 ランダムに生成したλ式での総当り戦

3.1 モデル

ここでモデルの記述に使用した(型のない)λ計算はtype-freeなものなので，前節で定義したgame master $\mathbf{G}$ は実際には $\mathbf{T}, \mathbf{F}$ 以外の任意の戦略λ式を代入することができる．そこで，

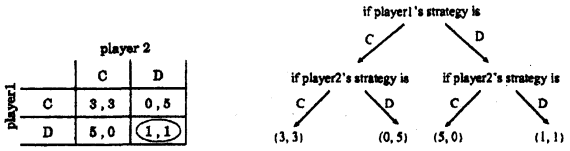

図 1: Two person one-shot prisoners' dilemma game

本研究ではまず，ランダムにλ式を生成してそれらの間で総当り戦をして，game master **G** からどのような利得を得るかを調べた．ただし，ここでは，計算機を使ったシミュレーションなので，以下の制約を設ける．まず，扱うλ式はすべて閉じた式 (コンビネータ) のみであること，生成されたλ式が自由変数を持つ場合はそのλ式は排除される．次に β 変換のステップ数に制限があること，ここでは 1000 ステップを上限とする．またここではすべてのλ式は正規形に変換してから用いることとする．さらにゲームとして成立するための条件として，元々のゲームの戦略である **T**, **F** と対戦した時と，自分自身と対戦した時には Barendregt の意味で自然数を返さなければならないという制約を加える．これはすなわち新しく生成されたλ式 Str に対して (**G***Str***T**), (**GT***Str*), (**G***Str***F**), (**GF***Str*), (**G***Str Str*) の変換の結果がすべて Barendregt 数にならなければいけないということになる．

このような制約の元で以下のようにしてランダムにλ式を生成した．まず，基本要素となる変数 x_i と，コンビネータの集合 ($\mathbf{T}(=\lambda xy.x), \mathbf{F}(=\lambda xy.y), \mathbf{I}(=\lambda x.x), \mathbf{S}(=\lambda xyz.xz(yz)), \mathbf{V}(=\lambda xyz.zxy)$) に対して，関数適用と関数抽象という操作をランダムに繰返して新しいλ式をつくる．次にできたλ式を正規形に変換する．ここでは適用と抽象という操作の繰返し回数 n を $1 \leq n \leq 20$ で動かして合計で 10^7 のλ式を生成した．それらのうち約 8×10^5 がゲームとしての制約を満たし，さらにそれらを正規形に変換して重複を除くと合計で約 4400 のλ式が得られた．

これら 4400 のλ式で総当り戦を行った結果を図 2 に示す．元々の戦略である **T** のこの総当り戦での平均利得は約 0.7 点であり，**F** の平均利得は約 1.8 点であった．また最も高い利得をとった戦略は $\lambda x_1 . x_1(\lambda x_2 . \mathbf{S})\mathbf{VFVF}$ という形をしており，約 2.6 点を得ている．図 2 の左図からは 0 点から 6 点までのほとんどの得点のペアがゲームの対戦によって実現していることを表している．一方で右図では集団の平均利得分布には偏りがあり，いくつかのピークがあることが分かる．

3.2 G′ (=相手に対する態度) の導入

各戦略λ式 S_{self} が相手の得点を計算するときの式 (10)

$$\mathbf{G}S_{self}S_{opp} \rightarrow \mathbf{G}'_{self}S_{opp} \rightarrow Reward_{opp}$$

において，S_{self} がまず **G** と反応してつくる途中のλ式 $\mathbf{G}'_{self}$ に注目する．この $\mathbf{G}'_{self}$ は自分が相手の得点を計算するときに相手に対してどういう態度をとっているのかを表わしてい

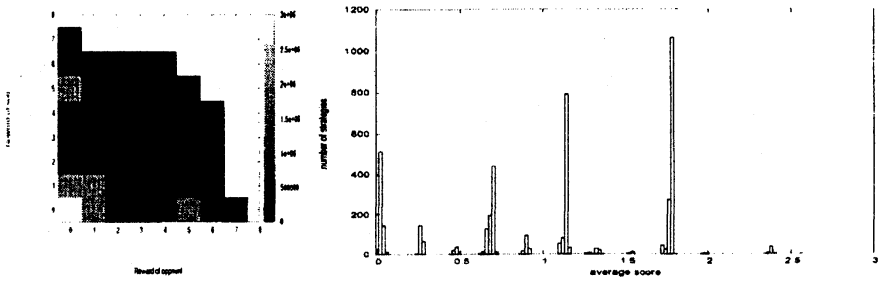

図 2: 総当り戦での各戦略 λ 式の利得の様子．左図は総当り戦で実現したすべての「利得の組」のヒストグラム．横軸と縦軸はそれぞれ対戦した戦略の利得．「利得の組」の出現頻度をグレイスケールで表している．右図は 平均利得のヒストグラム． 横軸は総当り戦での平均利得，縦軸は戦略の個体数．

るといえる．元々の囚人のジレンマゲームで用意された戦略 **T** と **F** のつくる **G′** は **G** の部分 λ 式である if 文 $\lambda x.x\mathbf{35}$，$\lambda x.x\mathbf{01}$ であったが，ここでつくられた各戦略は様々な **G′** を生成する．各戦略がつくる **G′** は，その λ 式の構造と意味から，まず相手に対して reactive なものと相手に対して non-reactive なものにわけることができる．そして reactive な **G′** のなかでも if 文の構造をしているものと，そうでないものとに分類できる．まとめると **G′** を以下の 3 つのタイプに分けることができる．

- Type I (if 文)　　$\mathbf{G}' = \lambda x.x\,\mathbf{m}\,\mathbf{n}$　　$\mathbf{m}, \mathbf{n}$: Barendregt 数
  「相手が **T** を出せば m 点を与え，相手が **F** ならば n 点を与える.」
  例：$\lambda x.x\mathbf{01}, \lambda x.x\mathbf{35}, \lambda x.x\mathbf{51}, \lambda x.x\mathbf{11}, \lambda x.x\mathbf{00}, \lambda x.x\mathbf{50}, \lambda x.x\mathbf{15}$

- Type II (定数関数)　　$\mathbf{G}' = \lambda x.\mathbf{n}$　　$\mathbf{n}$: Barendregt 数
  「相手の戦略にかかわらず，常に n 点を与える.」
  例：$\lambda x.\mathbf{0}$, $\lambda x.\mathbf{1}$, $\lambda x.\mathbf{5}$, $\lambda x.\mathbf{3}, \lambda x.\mathbf{4}, \lambda x.\mathbf{2}$

- Type III (others)
  例：$\lambda x.x\mathbf{F2}(x\mathbf{F4}), \lambda x.x\mathbf{F035}, \lambda x.x\mathbf{F0}(x\mathbf{35}), \lambda x.x\mathbf{F001}, \lambda x.x\mathbf{35}(x\mathbf{01})$

前節で述べたようにこのシミュレーションでは約 4400 の異なる λ 式を生成したが，各 λ 式がつくる **G′** は正規形で比較すると形が異なるものは 57 種類しかなかった．さらに **G′** として Type I の $\lambda x.x\mathbf{01}$ と $\lambda x.x\mathbf{35}$，Type II の $\lambda x.\mathbf{0}$ をつくる λ 式が大半であり，この 3 種で全体の 95%を占めていた．ある 2 つの異なる λ 式がある場合に，それぞれの形は違っていてもそれらがつくる **G′** が同じものであれば，そこから得られる得点は同じなので，他人からみるとその λ 式は同一視されることになる．この **G′** の偏りが図 2 の右図のピークを説明する．つまり総当り戦では集団の大半を占めるこの 3 つの **G′** からどのような得点を得るかで，各 λ 式の平均利得がほぼ決まっていることが分かる．得点をとる過程については進化バージョンと併せて次節で説明する．

4 進化ダイナミクス

前節ではλゲームでどのような戦略が可能かを調べるために，まずランダムにλ式を生成してその振舞いを調べた．本章ではλ式の集団をゲームの利得を適応度とする環境で進化させたときにどのような戦略の集団が生まれるかを調べる．

具体的にはλ式の集団を用意し，game master **G** からの獲得利得を適応度として遺伝的プログラミング (Genetic Programming, GP) で進化させた場合について計算機シミュレーションを行い，そこで現れたλ式がこのゲームにおいて戦略としてどのような振舞いをするのか，どのようなλ式が集団にひろがるのか，またその進化のパスはどのようになっているのかを調べる．

4.1 モデル

λ式で表現されるひとつの戦略を持つプレイヤーからなる集団を考える．ここではプレイヤーと戦略 (λ式) が同一視されているので，以下では各プレイヤー (個体) のことを戦略と呼ぶ．最初にλ式の集団を用意し，以下の手順で集団を進化させた．

1. 各世代においてそれぞれの戦略λ式は 集団のメンバー全員と一回限りの囚人のジレンマゲームの総当り戦をおこなう．
2. 総当り戦での得点を適応度として自然淘汰をおこない，高得点のλ式を残して得点が低いλ式を置き換える．
3. 2で出来た集団に対して突然変異と交叉によって新しいλ式を導入して次世代の集団を構成する．

ここでの各λ式同士の対戦の得点の計算は前節と同様に **G** にふたつのλ式を代入した結果を Barendregt 数として解釈する．ゲームを構成する自然数や if 文のλ式も前節で説明したものとまったく同じである．また，新しいλ式を構成するための進化オペレータとしては突然変異と交叉がある．λ式に対する突然変異はλ式の再帰的な定義を考えるとその部分λ式に対する「抽象」または「適用」の挿入または除去という操作で定義するのが自然である [3]．本研究では前節でのλ式に対するゲームとしての制約も考慮して，λ式に対する突然変異をその部分λ式に対する次のような操作で定義した．

- 関数抽象の挿入または削除

$$\begin{aligned} M &\rightarrow \lambda x_{new}.M \\ \lambda x.M &\rightarrow M \end{aligned}$$

- 関数適用の挿入または削除

$$\begin{aligned} M &\rightarrow (L_{new}M) \text{ or } (ML_{new}) \\ (MN) &\rightarrow M \text{ or } N \end{aligned}$$

また交叉とは，通常のGPでは2つの木構造の間で互いの部分木を交換することであり，ここでは2つのλ式の間での部分λ式同士の交換操作を交叉と定義する．ただし任意の部分λ式の交換を許すと交叉の結果変数の束縛が壊されてしまい自由変数が生じてしまう可能性があるので，ここでの交叉では交換する部分λ式を閉じたλ式(コンビネータ)だけに制限する．さらに突然変異においても，ある突然変異が新しく自由変数を生み出してしまったり，より上位の部分λ式の束縛の状況を壊したりしてしまわないように関数抽象や関数適用に使われる変数 x_{new} や L_{new} に対して適切な制限を加えた．

4.2 シミュレーション結果

4.2.1 進化の流れ

前節で設定したモデルを用いて行ったシミュレーションからもっとも典型的な一例の進化の様子を図3に示す．左の図は横軸に進化の世代をとり，縦軸に各世代での集団での総当り戦で各戦略が得る一試合当たりの期待利得を集団で平均したものをとったものである．また右の図は同じく横軸が世代で，縦軸はλ式を二分木で表現したときの枝の最大の深さの集団平均を示している．このシミュレーションでは以下のような進化の道筋がみられた．

1. まず，**T** と **F** だけの初期状態から **F** が集団全体に拡がり，集団全体で裏切り合う状態になる．よってこの時点では各戦略の1ゲーム当たりの平均利得は1点であり，通常の囚人のジレンマゲームの進化とほとんど同じである．

2. 次に次第に **F** と実効的に同じ得点をとる変異種が集団に拡がる．
   F と実効的には同じ種とは，自分の $\mathbf{G}'$ としては $\lambda x.x\mathbf{01}$ のif文をつくり，そこから1点をとるような **F** 以外の(そして **F** より長い)λ式のことである．この状態でも集団の1ゲーム当たりの平均利得は1点のままである．

3. さらにこの集団から2点をとるλ式が突然変異と交叉によって生まれる．
   この時点では集団は $\mathbf{G}'$ としてType Iの $\lambda x.x\mathbf{01}$ という0点と1点のif文を持つ戦略で占められているので，この戦略は $\lambda x.x\mathbf{01}$ という $\mathbf{G}'$ から2点を得ていることになる．さらにこの新しく生まれた戦略も自分自身の $\mathbf{G}'$ として $\lambda x.x\mathbf{01}$ を作るので，この戦略は自分自身からも2点を得ることができる．このようにしてこの戦略は集団に拡がる．そして世代が進むにつれてこの2点をとる戦略と実効的には同じ戦略が集団を占める．

4. さらに世代が進むと平均利得として3点をとるλ式が現れる．
   この時点でも各戦略は $\mathbf{G}'$ として $\lambda x.x\mathbf{01}$ を持つものが集団を占めているので，この侵入種は $\lambda x.x\mathbf{01}$ から3点をとっていることになる．またこの侵入種は自分自身の $\mathbf{G}'$ として $\lambda x.x\mathbf{01}$ を持つので自分自身との対戦でも3点をとることができ，集団中に拡がっていく．

5. その後も同様にして自分自身の $\mathbf{G}'$ としては $\lambda x.\mathbf{01}$ を持ち，そこからさらに高得点をとるλ式が生まれて集団に拡がるということが繰り返される．

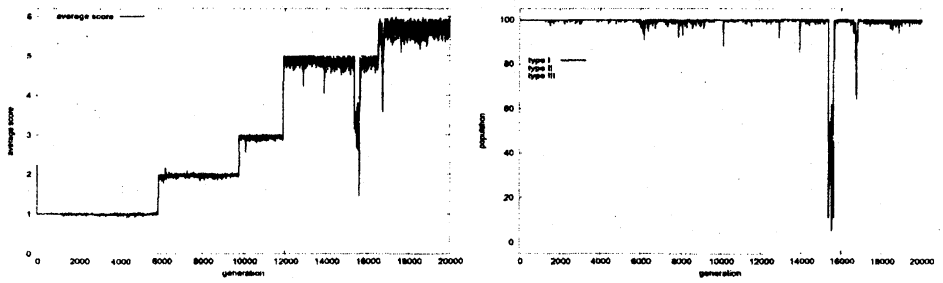

図 3: 各世代での平均利得 (左) と各タイプの $\mathbf{G}'$ を持つ個体の数 (右). 横軸は進化の世代, 縦軸はそれぞれ平均利得 (左), 個体数 (右) を表す.

6. 別のタイプの $\mathbf{G}'$ を持つ λ 式が短期間侵入することがあるが, 最終的には $\lambda x.x\mathbf{01}$ の $\mathbf{G}'$ を持つ λ 式の集団に戻る. (図 3 では 15300 世代から 15700 世代にかけての平均利得が減っているところ)

4.2.2 if 文と干渉する戦略

二つの戦略 S_{self}, S_{opp} が対戦したときの S_{self} の利得の計算過程は

$$\mathbf{G}S_{opp}S_{self} \rightarrow \mathbf{G}'_{opp}S_{self} \rightarrow Reward_{self}$$

となるが, 変換の前半でつくられる $\mathbf{G}'$ については 3.2 節で分類したので, 次にそれぞれの $\mathbf{G}'$ から各戦略が利得を計算する変換の後半部分について考える. この式からわかるように, 各プレイヤーの利得は相手のつくる $\mathbf{G}'$ に強く依存している.

Type II の $\mathbf{G}'$ との対戦

$\mathbf{G}'$ が Type II の場合は $\lambda x.\mathbf{n}$ という形をしているので, 任意の λ 式 Str に対して適用されると $(\lambda x.\mathbf{n})\ Str \rightarrow \mathbf{n}$ と変換が進み, かならず得点 n を返す. すなわち相手がこのタイプの $\mathbf{G}'$ をつくれば, 自分の戦略 Str が何であってもそれに関係なく相手が用意した定数が自分の得点になるし, 自分がこのタイプの $\mathbf{G}'$ をつくればどのような相手に対して常に自分が用意した利得を与えることができることになる.

このタイプの $\mathbf{G}'$ は相手の戦略に対して全く反応せず, 完全に相手の得点をコントロールできるので, ある意味では強い戦略ということができる. なぜなら例えば $\mathbf{G}'$ として $\lambda x.\mathbf{0}$ (すなわち相手に常に 0 点を与える) を生成すれば相手の戦略の利得はかならず 0 点となり, それ以外の得点をとることは不可能だからである. しかし一方でこの $\lambda x.\mathbf{0}$ という $\mathbf{G}'$ を持つ λ 式は自分と対戦した場合でも必ず 0 点ずつの得点になってしまうので, 自分自身と対戦したときに高得点をとることができないというゲームの戦略としては有利でない面をもつ. 一方, $\lambda x.\mathbf{5}$ という $\mathbf{G}'$ をもつ λ 式は自分自身と対戦したときにはお互いに 5 点ずつという高得点を

とることができるが，この $\mathbf{G}'$ は常に相手に 5 点を与えるので，他のすべての戦略に対しても 5 点をとらせてしまう．このように Type II (定数関数) の $\mathbf{G}'$ をつくる λ 式はゲームの戦略としてみると，自分自身と対戦したときにも自分が他人に与える以上の得点を得ることができないという硬直した戦略であるといえる．

succ を構成する戦略

次に Type I (if 文) の形をした $\mathbf{G}'$ から利得を得ることを考える．このタイプの $\mathbf{G}'$ は reactive な関数になっているので，Type II とは違い，代入される戦略によって異なった得点を返す．このタイプの $\mathbf{G}'$ は $\lambda x.x\mathbf{mn}$ という形をしており，これは「相手が $\mathbf{T}$ なら m 点，相手が $\mathbf{F}$ なら n 点を与える」という意味であり，相手の戦略 Str が $\mathbf{T}$，$\mathbf{F}$ の場合の得点の計算は以下のようになる．

$$(\lambda x.x\mathbf{mn})\,\mathbf{T} \to \mathbf{m}, \quad (\lambda x.x\mathbf{mn})\,\mathbf{F} \to \mathbf{n}$$

しかし，前節のランダムに作った λ 式の総当り戦で最も高い平均利得を得たものは集団の大部分を占める $\lambda x.x\mathbf{01}$ から (用意された 0 点，1 点ではなく) 2 点を得て，次に個体数の多い $\lambda x.x\mathbf{35}$ からは (用意された 3 点，5 点ではなく) 6 点を得ている．また λ 式の集団を進化させたシミュレーションでも前節で述べたように $\lambda x.x\mathbf{01}$ という $\mathbf{G}'$ から 6 点をとる戦略が進化している．これらの戦略はどのようにしてこれら if 文に用意された数字以外の点を得ることが可能になったのだろうか．具体例として図 3 のシミュレーションの世代 15000 での戦略をひとつとりだす．この λ 式は次のようなかたちをしている．

$$\begin{aligned} Str_{dom} \;=\; & \lambda x_1 . x_1(\lambda x_2x_3x_4x_5 . x_5(\lambda x_6 . x_4\mathbf{F})(\lambda x_9 . x_9(x_3(\lambda x_{10}x_{11}x_{12} . \mathbf{F})) \\ & (\lambda x_{15} . x_{15}\mathbf{F}(\lambda x_{18} . x_{18}\mathbf{F}x_4))))(\lambda x_{21}x_{22}x_{23}x_{24} . x_{24}x_1 \\ & (\lambda x_{25} . \mathbf{F})(\lambda x_{28}x_{29} . \mathbf{F}))(\lambda x_{32} . \mathbf{F}) \end{aligned}$$

この戦略 Str_{dom} は平均利得として 5 点を得ており，同じ世代の集団中の λ 式はほぼすべてこの λ 式と実効的に同じものである．この世代では Type I の $\lambda x.x\mathbf{01}$ という $\mathbf{G}'$ (0 点と 1 点の if 文) をつくる戦略が集団を占めているので，この戦略は $\lambda x.\mathbf{01}$ から 5 点をとっていることになる．

この戦略 Str_{dom} は $\lambda x.x\mathbf{01}$ という $\mathbf{G}'$ に代入されたときは $\mathbf{succ}(=\lambda xy.y(\lambda xy.y)x)$ を 4 つ繋げることで以下のような「足す 4」(**plus4**) という関数を構成する．

$$\begin{aligned} \mathbf{plus4} \;=\; & \lambda x\,.\,\mathbf{succ}(\mathbf{succ}(\mathbf{succ}(\mathbf{succ}\,x))) \\ =\; & \lambda x_1x_2 . x_2(\lambda x_3 . x_3\mathbf{F}(\lambda x_4 . x_5\mathbf{F}(\lambda x_6 . x_6\mathbf{F}(\lambda x_7 . x_7\mathbf{F}x_1)))) \end{aligned}$$

つまり戦略 Str_{dom} は $\lambda x.x\mathbf{01}$ という if 文に代入されると用意された **0**，**1** の数字のうち **0** を，自然数としてではなく関数として使うことで **succ**(=「足す 1」) をいくつも構成してしまい，それを $\mathbf{1}(=\lambda x.x\mathbf{FI})$ に作用させて **5** という λ 式をつくり，5 点を得ている．またこの戦略は自分自身の $\mathbf{G}'$ として $\lambda x.x\mathbf{01}$ を構成するので，自分自身との対戦でも 5 点を得ることができ，安定した集団を構成している．これは game master $\mathbf{G}$ が想定していない使い方で if 文の部分 λ 式と干渉を起こし，強引に解釈しなおしていることになる．

2つのif文への代入

ここで重要なのは，どのようなif文 $\lambda x.xPQ$ の前半部分 P に対しても部分λ式とif文と干渉をおこして **succ** を作ればよいという訳ではないことである．なぜなら，相手の得点を計算するときには自分が先に **G** に代入されて $\mathbf{G}'_{self}$ を生成するのであるが，この時は $Q = \lambda x.x\mathbf{01}$ であるから **succ** をつくって **succ** Q としてしまうと，**succ** $Q \to \lambda y\,.\,y\mathbf{F}(\lambda x.x\mathbf{01})$ といった $\mathbf{G}'$ をつくってしまう．この $\mathbf{G}'$ は「$\mathbf{T},\mathbf{F}$ と対戦したときにBarendregt数を与えなければいけない」というλゲームの制約をみたすことができない．(さらにこの場合は自分と対戦したときもBarendregt数を返すこともできない) ところが，実際にはこの戦略 Str_{dom} は $\mathbf{G}'$ としてType Iのif文 $\lambda x.x\mathbf{01}$ を生成し，$\mathbf{T}$ と対戦すれば相手に0点を，$\mathbf{F}$ と対戦すれば相手に1点を与えている．つまり各戦略にとって重要なのは，if文の構成要素すなわち $\lambda x.xPQ$ の P と Q に応じて，そのif文に対する振舞いを変えられるようにすることなのである．たとえばこの戦略 Str_{dom} は **G** というif文に代入されたときはif文の後半部分 Q をそのまま返し，$\lambda x.x\mathbf{01}$ というif文に代入されたときは前半部分 P と干渉して **succ** をつくり，それを後半 Q に適用させる，と振舞いを変えている．

Type I (if文) の $\mathbf{G}'$ をつくる戦略 (これが進化においては集団の大多数を占める) と対戦する場合を考えると，各戦略λ式は次のように2つの場面でif文に代入されることになる．

1. 相手の得点を計算するとき．game master($\mathbf{G}$) に代入されて $\mathbf{G}'_{self}$ をつくる．
2. 自分の得点を計算するとき．相手のつくる $\mathbf{G}'_{opp}$ に代入されて自分の得点を返す．

これらのふたつの各局面で，もっとも単純な戦略である **T** や **F** は，if文の用意した2つの選択肢の内容 P,Q をみることはなく，その中身に関係なく片方を選ぶだけである．$\#$ で任意のλ式を表わすことにすると $\mathbf{T},\mathbf{F}$ のif文に対する振舞いは次のように書ける．

$$(\lambda x\,.\,xP\#)\mathbf{T} \to P, \qquad (\lambda x\,.\,x\#Q)\mathbf{F} \to Q \tag{12}$$

一方，上記の15000世代付近で集団を占めている Str_{dom} は自分が代入されるif文の内容を見て以下のように振舞いを変えることが出来る．

$$(\lambda x\,.\,x\,\underline{(\lambda x\,.\,x\#\#)}\,Q)\,Str \;\to\; Q \tag{13}$$

$$(\lambda x\,.\,x\;\underline{\quad(\lambda x\,.\,x)\quad}\,Q)\,Str \;\to\; (\mathbf{plus4})\,Q \tag{14}$$

つまりこの戦略λ式は "if x then P else Q" の部分λ式 P がif文であるかまたは1以上の自然数である場合 ($P = \lambda x.x\#\#$) は Q をそのまま返し，もし P が0の場合 ($P = \lambda x.x = \mathbf{I}$) はif文から **plus4** をつくって Q に作用させることで結果的に $Q+4$ をつくることを意味している．

この場合 game master **G** は2重のif文なので式13の場合に相当し，対戦相手のつくる $\mathbf{G}'$ が $\lambda x.x\mathbf{01}$ の時は式14の場合に相当する．よってこの戦略は，相手の得点を計算する時は自分の $\mathbf{G}'_{self}$ として $\lambda x.x\mathbf{01}$ (= "if x then 0 else 1") をつくって **F** のように振舞い，一方で自分の得点を計算する時は相手の $\mathbf{G}'_{opp}$ のif文 ("if x then 0 else 1") の中にある数を操作して **succ** を合成してそれをもとのif文の数に作用させることでより多くの得点を得ることを可能にしている．

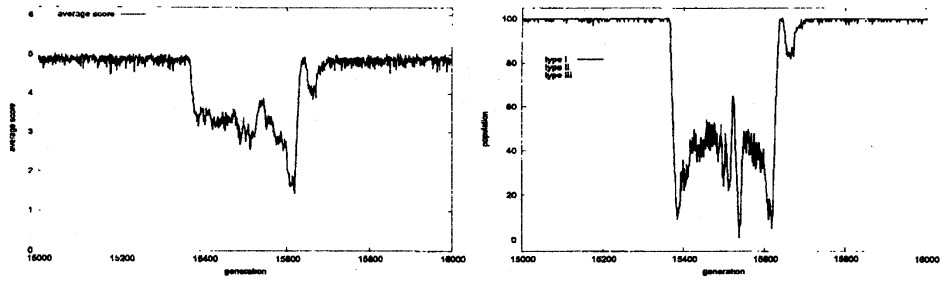

図 4: Type I 以外の戦略の侵入．平均利得 (左) と $\mathbf{G}'$ のタイプ別の個体数 (右)．横軸は世代，縦軸はそれぞれ平均利得 (左)，タイプ別の個体数 (右) を表している．

4.2.3 別のタイプの $\mathbf{G}'$ を持つ戦略の侵入と Type I のロバストさ

前述したように，ほとんどすべての世代では自分自身の $\mathbf{G}'$ として $\lambda x\,.\,x\mathbf{01}$ を作り，そこから succ をいくつか繋げたものを構成して得点を得るものが集団を占めているが，時々別の $\mathbf{G}'$ を持つ λ 式が侵入することがある．例として図 3 の 15300 世代から 15700 世代までを拡大したものを図 4 に示す．

侵入前の集団を占めている λ 式を Str_{dom} とおくと，この戦略は上で説明したものと機能的には同じで，$\mathbf{G}$ に代入されると自分自身の $\mathbf{G}'$ として $\lambda x\,.\,x\mathbf{01}(=\mathbf{G}'_{dom})$ を作り，$\lambda x\,.\,x\mathbf{01}$ に代入されると **plus4** を構成して 5 点を得る λ 式である．ここで新たに侵入してきた戦略を Str_{invade}，その $\mathbf{G}'$ を $\mathbf{G}'_{invade}$ とおく．具体的には次のような λ 式である．

$$
\begin{aligned}
Str_{invade} &= \lambda x_1\,.\,x_1(\lambda x_2x_3x_4\,.\,x_4\mathbf{F}(\lambda x_7\,.\,x_7(\lambda x_8\,.\,x_2(\lambda x_9x_{10}x_{11}\,.\,x_{10}\mathbf{I}\\
&\quad (\lambda x_{13}x_{14}x_{15}x_{16}\,.\,x_{15}(\lambda x_{17}\,.\,\mathbf{F}))\mathbf{F}(\lambda x_{22}\,.\,x_{11}\mathbf{I})x_9(\lambda x_{24}x_{25}\,.\,\mathbf{F})))\\
&\quad (\lambda x_{28}\,.\,x_1x_{28}\mathbf{F}(\lambda x_{31}\,.\,x_{31}\mathbf{F}x_3))))\mathbf{F}\\
\mathbf{G}'_{invader} &= \lambda x_1\,.\,x_1\,\mathbf{3\,5\,F\,6}.
\end{aligned}
$$

この $\mathbf{G}'_{invade}$ は Type III で，if 文でも定数関数でもない．しかしこの戦略はそれまでの集団の支配種と同じく $\lambda x\,.\,x\mathbf{01}$ から **plus4** を作り，5 点をとることができる．そして自分自身のつくる $\mathbf{G}'_{invade}$ からは 4 点をとることができる．一方，それまで集団を占めていた $\mathbf{G}_{dom}$ はこの $\mathbf{G}'_{invade}$ からは 2 点しかとることができない．この様にして新しい Type III の $\mathbf{G}'$ を持つ Str_{invade} が集団に拡がっていく．

$$
\begin{aligned}
\mathbf{G}'_{dom}Str_{dom} &\to \mathbf{5}, \quad & \mathbf{G}'_{dom}Str_{invade} &\to \mathbf{5},\\
\mathbf{G}'_{invade}Str_{dom} &\to \mathbf{2}, \quad & \mathbf{G}'_{invade}Str_{invade} &\to \mathbf{4}.
\end{aligned}
$$

この新しい種 Str_{invade} はそれまで集団を支配していた種 Str_{dom} に対しては Str_{dom} と同じ様に振舞い，自分の $\mathbf{G}'$ としては相手に「解釈されにくい」Type III の $\mathbf{G}'_{invade}$ を作り，自

分たち同士ではそこから高得点を得ることに成功している．しかし図4からもわかるように，最終的には $\lambda x\,.\,x\mathbf{01}$ の $\mathbf{G}'$ を持つλ式が再び集団の中で拡がり，$\mathbf{G}_{invade}$ を駆逐して支配種になる．これは，自身の $\mathbf{G}'$ として $\lambda x\,.\,x\mathbf{01}$ を持つλ式に突然変異が起こり $\mathbf{G}'_{invade}$ から7点をとる種が生まれるからである．つまりこの新しい支配種 Str_{newdom} は次のような振舞いをすることができる．

$$\mathbf{G}'_{dom}Str_{newdom} \rightarrow \mathbf{5}, \qquad \mathbf{G}'_{invade}Str_{newdom} \rightarrow \mathbf{7} \tag{15}$$

興味深いのは，このように $\mathbf{G}'_{invade}$ から6点以上をとる戦略は $\lambda x\,.\,x\mathbf{01}$ を自分自身の $\mathbf{G}'$ にもつ Type I のλ式にのみ現れ，他の Type II，Type III のλ式には現れないことである．$\mathbf{G}'_{invade}$ をつくる Str_{invade} は Type III であるにもかかわらずである．この現象はこのパス以外の進化でも観察され，シミュレーションでは常に支配種は $\lambda x\,.\,x\mathbf{01}$ を $\mathbf{G}'$ に持つ Type I のλ式であった．またそれらのλ式は一時的に他の Type II や Type III の侵入を許しても最終的にはそれらに対応して得点をとるように進化して再び集団に拡がっていく．このような点から $\lambda x\,.\,x\mathbf{01}$ を作る Type I のλ式は侵入種に対してロバストな戦略と言うことができる．

図5は，横軸に世代をとり，縦軸に各λ式の枝のうち $\mathbf{G}'$ から **plus4** を構成するために必要な枝の深さ (左図) と $\mathbf{G}$ から $\mathbf{G}'$ を構成する際に使用するλ式の枝の深さ (右図) の支配種 Str_{dom} と侵入種 Str_{invade} のそれぞれの種ごとの平均を示したものである．この図からは支配種と侵入種の間には各λ式の枝のうちどれだけを **plus4** の構成に使うかについてはほとんど違いがないのに対して，$\mathbf{G}'$ の構成に必要な枝の深さについては大きな差があることがわかる．すなわち Type III の $\mathbf{G}'$ を持つ戦略は Type I の $\mathbf{G}'$ をもつ戦略と比較すると $\mathbf{G}'$ の構成に枝のより深いところを使用している．これは $\lambda x\,.\,x\mathbf{01}$ は game master $\mathbf{G}$ の中に部分λ式としてすでに存在しているので，Type I の戦略は $\mathbf{G}'_{self}$ を構成する際にはあまり深い枝まで使わないかわりに，相手のつくる $\mathbf{G}'_{opp}$ への対応を複雑にすることができるからだと思われる．またこれは最初に **succ** を作って $\lambda x\,.\,x\mathbf{01}$ から2点をとる戦略が例外なく Type I の戦略から生まれたことからもわかる．この事が Type I の戦略のロバストさの原因を示唆していると考えている．

5 まとめ

本研究ではλ計算でゲームを記述した体系を作り，計算機シミュレーションによってその性質を調べた．ゲームの利得，利得関数，プレイヤーの戦略といった異なる階層に属するゲームの構成要素をすべてλ式で記述することによって「ゲームのルールの記述」を明示的に組み込んだ形でのゲームのモデルを構築することが可能になった．これにより，従来のゲームのシミュレーションで提示されてきたような記憶の深さや相手のモデルの不定性ではなく，ルールの記述のレベルそのものを読みかえることによって多様な振舞いをする戦略があらわれた．これはルールの openness が戦略の多様さを可能にするというゲームの一つの解釈を示していると言えよう．また同様の設定でλ式の集団を用意し，それらをこのゲームにおける戦略として遺伝的プログラミング (GP) の手法で進化させることで，さらに様々な振舞いをする戦

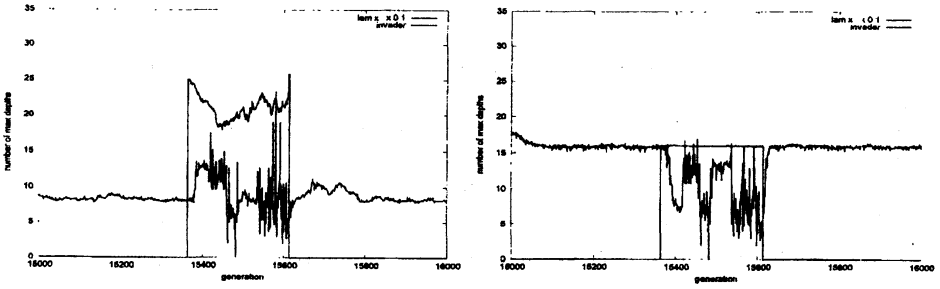

図 5: Type I の支配種と Type III の侵入種の λ 式の枝の最大の深さ．横軸は世代で縦軸は λ 式の枝の最大の深さのそれぞれの種ごとの平均を示している．左図は λ 式の枝のうち **plus4** を構成するのに必要な枝の深さ，右図は同じくそれぞれの戦略が **G′** を構成するのに必要な枝の深さを示している．

略が現れた．

記述レベルの読みかえは，具体的にはルールである if 文の階層を超えて，その「中」を見て振舞いを変えることによって実現される．例えば，相手の得点を計算する時には元のゲームの if 文をこわして定数関数を返すことで相手の得点を一定にしてしまったり，自分の得点を計算する時には利得表の要素である自然数を関数として使うことにより，ルールの if 文から **succ** を合成することによって高得点を得る，といった戦略があらわれた．さらに GP を用いて集団で進化させた場合には **succ** を複数個生成して繋げることによってさらに高得点を得る戦略が生まれた．

またゲームにおける戦略集団の進化ダイナミクスでは，進化的安定性が問題になるが，本シミュレーションにおいても支配種に対する変異種の侵入とそれらを克服する新たな支配種の出現といった興味深い進化ダイナミクスを観察することができた．この場合に，侵入種に対する戦略のロバストさというのは，侵入種に対する適応のしやすさで決まることになる．これは「相手の得点の計算」と「自分の得点の計算」というゲームの 2 つの局面に対応する 2 つの if 文への代入に対して，どちらにより戦略のリソース (ここでは λ 式の枝の深さ) を割くかという点から解釈される．このように，ルールの解釈とそこからどのように利得をとるか，という 2 つの過程にひとつの戦略で対応する場合にどちらの過程により多くのリソースを割くかというのは，開かれたゲームの状況では重要であり一般的な問題であると言えよう．

導入で述べたような子供の遊び (プレイ) のようにルールが開かれた相互作用とはすなわち言い換えるとルールの解釈を許すようなゲームだといえる．これは各プレイヤーの戦略として，ルールによって固定されて利得表に代入される単なる変数ではなく関数のように reactive なものが生まれてくる状況である．そのような状況下では，与えられた定義域の範囲にとどまらず，ルールの解釈レベルを読みかえてしまってルールをだましたりルールと干渉するような戦略がでてきてしまうので，その過程を視野に入れて解析することが必要であると言えるだろう．そのための枠組みとして，ここで提案した λ ゲーム，あるいはこれを拡張したアプローチが有効であると考える．

参考文献

[1] Axelrod, R., *The Evolution of Cooperation.* Basic Books, New York, 1984.

[2] Barendregt, H. G., *The Lambda Calculus: Its Syntax and Semantics*, Amsterdam, North Holland, 1981.

[3] Heiss-Gzedik, D., Fontana, W., Evolution of λ-expressions through Genetic Programming, `http://www.santafe.edu/~walter/Pages/publications.html`.

[4] Fontana, W., Buss L. W., The arrival of the fittest: Toward a theory of biological organization, *Bulletin of Mathematical Biology* 56, 1–64, 1994.

[5] Fontana, W., Buss L. W., What would be conserved if 'the tape were player twice'?, *Proc. Natl. Acad. Sci. USA* 91, 757–761, 1994.

[6] Fontana, W., Buss L. W., THe barrier of objects: From dynamical systems to bounded organizations, in *Boundaries and Barriers*, J. Casti and A. Karlqvist (eds.), pp. 56-116, Addison-Wesley, 1996.

[7] Hofstadter, D.R., *Metamagical Themas*, BasicBooks, 1985.

[8] Howard, N., The theory of meta-games, *General Systems* 11, 167–186, 1966.

[9] Howard, N., The mathematics of meta-games, *General Systems* 11, 187–200, 1966.

[10] Howard, N., *Paradoxes of Rationality: Theory of Metagames and Political Behavior.* MIT Press, Cambridge, MA. 1971.

[11] Ikegami, T., From genetic evolution to emergence of game strategies, *Physica D* 75, 310–327, 1994.

[12] Ikegami, T., Taiji, M., Structures of Possible Worlds in a Game of Players with Internal Models. *Acta Poly. Scan.* MA. 91, 283–292, 1998.

[13] Ikegami, T., Taiji, M., Imitation and Cooperation in Coupled Dynamical Recognizers, in *Advances in Artificial Life.* (eds. Floreano,D. et al.), 545–554, Springer-Verlag, 1999.

[14] Lindgren, K., Evolutionary phenomena in simple dynamics, 295–311, in *Artificial Life II*, Langton, C.G. et al.(eds), Addison-Wesley, CA. 1992.

[15] Speroni di Fenizio, P., A Less Abstract Artificial Chemistry, in *the proceedings of Artificial Life VII*, 49–53, 2000.

[16] Taiji, M. and Ikegami, T., Dynamics of internal models in game players, *Physica D* 134, 253-266. 1999.

[17] Vreeswijk, G.A.W., Several experiments in elementary self-modifying protocol games, such as Nomic, Technical report CS 95-06, Department of Computer Science, FdAW, University of Limburg, The Netherlands, 1995

[18] Vreeswijk, G.A.W., Formalizing Nomic: working on a theory of communication with modifiable rules of procedure, Technical report CS 95-02, Department of Computer Science, FdAW, University of Limburg, The Netherlands, 1995.