第4回AI若手の集い(人工知能学会主催)
MYCOM2003資料
2003年5月26日，27日

文字列書き換え系における書き換え規則の進化

*杉浦 孔明 †, 鈴木 秀明 ‡, 塩瀬 隆之 †, 川上 浩司 †, 片井 修 †

† 京都大学大学院 情報学研究科
〒 606-8501 京都市左京区吉田本町
sugiura@sys.i.kyoto-u.ac.jp

‡ ATR 人間情報科学研究所
〒 619-0288 京都府相楽郡精華町光台 2-2-2
hsuzuki@atr.co.jp

{shiose, kawakami, katai}@i.kyoto-u.ac.jp

Abstract. 人工生命システム Tierra の基本デザインを改良するために，文字列書き換え系 String-Based Tierra を構築した．本システムは三つの特徴を持つ．即ち，1)Tierra の命令セットを正規表現を用いた書き換え規則に翻訳したこと，2) 各エージェントにゲノムとして文字列を，命令セットとして書き換え規則集合をそれぞれ与え，自身の持つ書き換え規則を用いて自己複製を行わせるようにしたこと，3) よりよい規則集合を得るために規則に対する遺伝的操作および淘汰を導入したことである．本システムを用いてエージェントに自身を自己複製させる実験を行ったところ，ゲノムだけでなく，書き換え規則集合をも進化させることができた．

1 はじめに

進化システムの能力は，設計者が用意した基本関数 (関数記号，機械語命令など) に大きく依存する．よって，設計者はシステムが高い進化性を持つように基本デザインを行う必要がある．しかしながら，人間の用意した基本関数は必ずしも高い進化性を持つとはいえない．

Matsuzaki らは，人工生命システム Tierra [2] で用いられている機械語の命令セットを書き換え規則に翻訳する方法を提案している [1]．また，書き換え系を用いた書き換え規則最適化の研究例として Suzuki の研究が挙げられる [3]．しかし，進化システムを書き換え系によって実現し，書き換え規則を実際に進化させた研究は今までにない．そこで本研究では，Tierra の命令セットを改良するために，文字列書き換え系 Sting-based Tierra を構築する．まず，Tierra で用いられる機械語の命令セットを，等価な書き換え規則に翻訳して，書き換え系を構築する．次に，書き換え規則をシステム自身が改良するしくみを用意する．

2 String-Based Tierra

2.1 Tierra とは

Ray による人工生命システム Tierra は，計算機の中でプログラムが進化することを示し，人工生命という研究パラダイムにひとつの契機を与えた [2]．Tierra におけるディジタル生物は，機械語で書かれた自己複製プログラムを遺伝形質とし，CPU 時間とメモリ空間を求めて競争することにより，自身の遺伝形質をより効率よく複製できるように進化する．各個体は，機械語のシークエンスで表されたゲノムを持ち，“生きているあいだ” ひとつの仮想プロセッサ (Tierra CPU) を与えられる．さらに，それぞれの Tierra CPU は 4 個のレジスタ (ax, bx, cx, dx) と容量 10 のスタックを持ち，ゲノムに書かれた機械語の命令を順に実行することで自身のゲノムを複製する．

2.2 要素集合

本システムで用いる要素集合を表 1 に示す．

命令文字 Tierra で用いられる機械語に対応する．

例: NOP0(no operation) = ‘`f`’, PUSHA(push ax on stack) = ‘`A`’, . . .

レジスタ文字 Tierra CPU が持つポインタ，レジスタ，スタックを表し，命令文字と同様にゲノムに埋め込まれている．本システムでは，レジスタが直接数を保存することはないが，レジスタ文字は，ゲノムにおいて開始文字からその文字までに含まれる，命令文字の数を保持するものとみなされる．レジスタ文字をゲノムに埋め込むことの利点は次の 2 つである．

- ポインタなどの移動を文字列書き換えにより表現できる
- ゲノムの中にポインタを複数持つことにより，命令を並列に実行することを可能にする

開始・終了文字 ゲノムの始まりと終わりを表す．レジスタ文字はこれらの外に出ることはない．

2.3 正規表現を用いた書き換え規則

本研究では，命令の実行を文字列の書き換えによって表現するために，命令の実行と等価な書き換え規則を作成した．これらの規則は以下の特徴を持つ．

1. Perl の正規表現を利用して書き換え規則を作成したため，複雑な書き換えを行うことができる．
2. ひとつの書き換え規則を，正規表現のマッチングまたは置換の論理積として表現するため，個々の正規表現を簡潔に書き表すことができる．よって，本研究で作成した書き換え規則は，いたずらに長く入り組んだ規則よりも，変化に対する許容性が高い．

例: INCA(ax レジスタの値をインクリメント) に対応する書き換え規則は以下のように表される．

```
s/pN/Np/g && s/a($J)/$1a/g
```

ここに，`N` = INCA を表す命令文字，`$N = [`レジスタ文字`]`，`$J = $N*[`命令文字`]$N*`とする．すなわち，`$N` はレジスタ文字に対応する文字クラスを表し，これに`*`をつけることで 0 個以上任意個のレジスタ文字を表す．したがって，`$J` は前後に 0 個以上のレジスタ文字を持った，任意の命令文字 1 つにマッチする．

INCA に対応する書き換えを行う様子を図 1 に示す．まず，前半部の置換によって，INCA を指していたポインタを 1 つ右へ移動させる．次に，後半部によって `a` を 1 つ右に移動させることにより，開始文字 '[' から `a` までの間にある命令文字の数は 1 だけ増加する．これはレジスタの値を 1 だけ増加させたことを意味する．

表 1: 要素集合

	本システム	Tierra
命令文字	`t, f, A,..., Z, w, x, y, z`	機械語
レジスタ文字	`p`	ポインタ
	`a, b, c, d`	レジスタ
	`0, 1,..., 9`	スタック
開始・終了文字	`[, ]`	

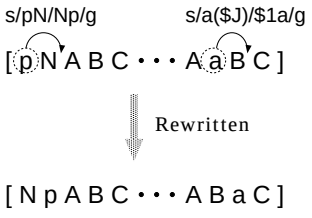

図 1: INCA の実行

2.4 淘汰

本研究では，各個体がそれぞれ独自の書き換え規則集合 **R** を持ち，**R** を用いて自己複製を行う．本研究では，進化を用いてより優れた規則集合を得るために，規則集合に対する選択を導入する．ここに，個々の書き換え規則の適合度を，その規則の適用回数と定義する．書き換え規則集合 **R** における並び替えの手順は以下の通りである (図 2).

1. 規則を適用回数にしたがってソートする
2. 下から l 個の規則を削除する
3. 上位 l 個の規則を選び，交叉や突然変異などの操作を加えて，新たに l 個の規則を生成する
4. 新規則を **R** の中央に挿入する

2.5 遺伝的操作

Tierra では，遺伝的操作はゲノムに対してのみ行われていた．それに対し本システムでは，操作を行う対象はゲノムと書き換え規則の 2 つである．書き換え規則に対する操作として，1) 重複，2) 削除，3) 正規表現の変化，の 3 種類を用意した．

3 実験および考察

実験のパラメータとして，最大個体数 $N_{max} = 32$，ゲノムに対する突然変異率 $m_g = 0.01$，規則に対する突然変異率 $m_r = 0.5$，1 タイムステップで規則の淘汰が行われる確率 $p = 0.02$，新たに差し換えられる規則の数 $l = 14$，の 5 つを用いた．

図 3 は，1 タイムステップにおいて適用された書き換え規則の数を，ステップ数に対してプロットしたものである．図 3 より，適用回数の最大値は初期ステップでは約 5 であったが，18 万タイムステップ後には約 14 に増加している．これは，規則集合から不要な規則が排除され，有用な規則が獲得されたためであると考えられる．

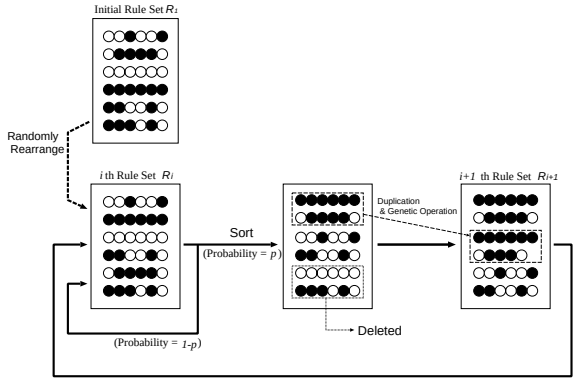

図 2: 規則の淘汰

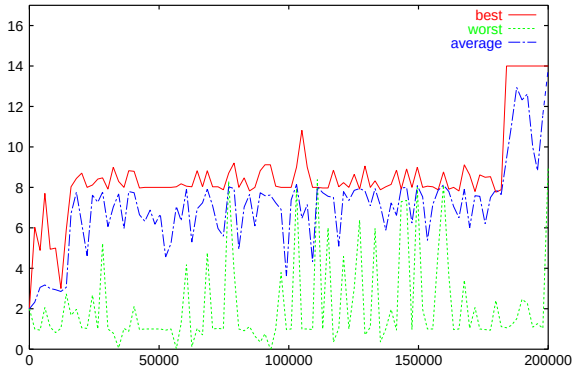

図 3: 規則の適用回数の変化

4 おわりに

本研究では，Tierra の基本デザインを改良するために，Tierra をもとにした書き換え系を構築し，書き換え規則の進化について検討した．システムを構築するために 3 つのステップを踏んだ．第一に，Tierra において用いられる 32 種類の命令語を，互いに独立な 140 の書き換え規則に翻訳した．各規則は，正規表現のマッチングまたは置換を論理積演算子でつないだものとして表現される．第二に，各個体にゲノムとして文字列を，命令セットとして書き換え規則集合をそれぞれ与え，自らの規則集合を用いて自己複製を行わせるようにした．第三に，書き換え規則に対する遺伝的操作および淘汰を導入することによって，書き換え規則をシステムが改良するしくみを用意した．本システムを用いて実験を行ない，書き換え規則の適用回数の増加という結果を得た．このことは，本システムを用いることにより，書き換え規則集合を進化させることができたことを示している．

参考文献

[1] S. Matsuzaki, H. Suzuki, and M. Osano. Tierra instructions implemented using string rewriting rules. In *Proceedings of the Fifth International Conference on Humans and Computers*, pp. 167–172, 2002.

[2] T. S. Ray. An approach to the synthesis of life. In C. G. Langton, et al., editors, *Artificial Life II: Proceedings of the Second Artificial Life Workshop*, pp. 371–408. Addison-Wesley, 1992.

[3] H. Suzuki. String rewriting grammar optimized using an evolvability measure. In J. Kelemen, et al., editors, *Advances in Artificial Life*, pp. 458–468. Springer-Verlag, Berlin, 2001.