アヤトリにおけるひも図形変形過程の表現とその処理

山田雅之† RAHMAT BUDIARTO†
伊藤英則† 世木博久†

アヤトリはひもを指に絡めて，その変形を繰り返すことによってさまざまな形を作り出すことができる．本論文では，アヤトリのひもの変形過程の表現の方法と，この表現に基づくひも図形処理の方法について述べる．まず，指の動作によってひもが変形してゆく過程を表現する方法として，いくつかの基本動作を定義し，これらに対応する図形をそれぞれ重ね合わせる方法を提案する．次に，アヤトリの出来上がり形をこの重ね合わせ図形から生成するアルゴリズムについて述べる．最後に，この生成アルゴリズムの有効性をプログラムを作成して確認する．

A Representation Method of Ayatori Process and Its String Diagram Processing Method

MASASHI YAMADA,† RAHMAT BUDIARTO,† HIDENORI ITOH† and HIROHISA SEKI†

In *ayatori*, many possible shapes can result from a simple closed string by several transformation. In this paper, we show a method for representation of *ayatori* process and a method for string diagram processing. We propose the *overlapped* diagram for the representation. of *ayatori* process, and describe its properties by making use of some example. Also, we describe a method to generate *ayatori* shapes from *overlapped* diagram. Lastly, we make sure of the usefulness of this method by experiment.

1. はじめに

ひもを指に絡め，繰り返し変形することによりさまざまな美しい形をつくる遊びとして知られるアヤトリは多くの国で古くから親しまれている．このようなアヤトリの変形過程を表現し，機械的にアヤトリの形を生成することは興味深い．しかし，アヤトリの変形過程を計算機上で表現し，プログラムにより処理する手法はこれまであまり試みられていない[9]．その理由として，1)ひもそのものの表現方法，2)ひもの変形過程の表現方法，および，3)これらの表現された対象の処理とその結果を表示する方法に効果的なものがなかったことが挙げられる．

1)については，3次元空間内のひもの位相的性質を考慮しなければならない．これについては結び目理論[2),4),5),8)]の分野で現在盛んに研究が行われているが，結び目理論の結果であるひもの位相的性質についての不変量による表現のみでは，細部のひも状態を処理する目的では不十分である．また，ひもの状態全体を連続したセルによって表現する方法[3),6)]が報告されているが，これを処理する際は効率上難がある．

2)については，これまでは，ひもの変形過程は変形前と変形後の双方の図形を示し，変形した箇所を明示することが一般的であった．しかし，この方法は人間の視覚による図形処理にとっては便利であるが，計算機上でプログラム処理する際の表現方法としては適当とはいえない．

3)については，ひも状態の表現の方法から変形処理後の図形を表示する方法に効果的なものがなかった．

そこで本論文では，上述の困難さの一部を解決するひもの表現方法と，アヤトリにおけるひもの変形過程表現方法と，およびアヤトリの完成図を表示する方法について述べる．

以下，2章でアヤトリの変形過程を表現する重ね合わせ図形について述べ，3章で重ね合わせ図形からアヤトリの形を生成する方法について述べ，4章でプログラムによるアルゴリズムの確認を行う．

2. アヤトリ変形過程表現

ここではアヤトリの変形過程はいくつかの基本動作の繰り返しであることに着目する．すなわち，いくつ

† 名古屋工業大学知能情報システム学科
Department of Intelligence and Computer Science, Nagoya Institute of Technology

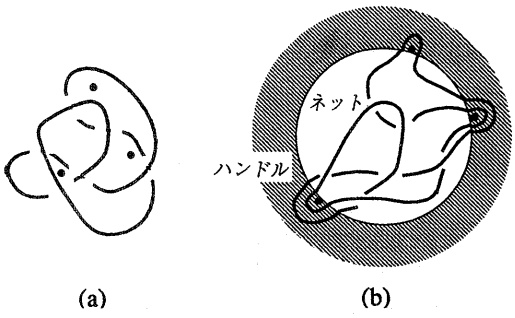


図 1 点を含むひもの図形
Fig. 1 A diagram including some points.

かの基本動作に対応するひもの変化した部分を図形表現する重ね合わせによる方法について述べる．

2.1 ネット部とハンドル部

アヤトリを**図1**(a)のような平面図形を使って表現する．交差する二本のひもの上下関係を陽に示すため，下に位置するひもを区切れた線分で表す．以降，二本のひもが交差する平面上の点を**交差点**と呼ぶ．さらに，ひもを絡める指を図中では点(・)で表す．

図1(a)のような点を含むひもの図形を図1(b)に示すように二つの領域に分ける．一方の領域は互いに交差するひものみが存在し，点は存在しない領域である．もう一方の領域は点とひもが双方存在し，かつ，交差点が存在しない領域である．以降，前者の領域を**ネット(部)**，後者の領域を**ハンドル(部)**と呼び，ネットとハンドルを区別するときはハンドルを斜線で図示する．また，ネットとハンドルの境界をひもが通過する点を**端点**と呼ぶ．

2.2 基本動作

ここでは，三つの基本動作 i)取る，ii)外す，iii)入れ換える，を定義する．

i) 取る：ネット部のひもを指で取る．

ii) 外す：ハンドル部のひもを外す．

iii) 入れ換える：同一の指にかかっているハンドル部の二本のひもの内側と外側を入れ換える．

これらi)～iii)の基本動作に対応するネットとハンドルの変形を**図2**(a)，(b)および(c)に示す．

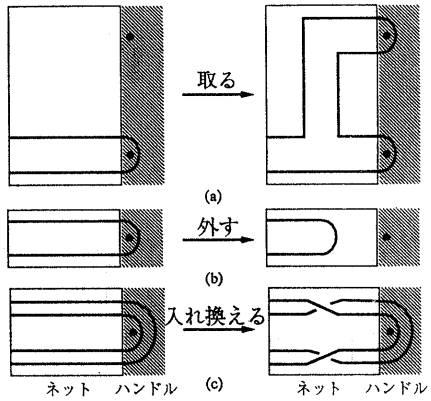


図 2 基本動作
Fig. 2 Basic movements.

2.3 重ね合わせ図形

2.2節で述べた基本動作に対応する図形のネット部を重ね合わせた図形をネット部の重ね合わせ図形と呼ぶ．

ネット部の重ね合わせ図形によるアヤトリの変形過程表現について述べる．ここではアヤトリの代表的な形[1]であるi)基本形，ii)ナバホどり，iii)二段ばしごを例にして基本動作の重ね合わせによりこれらの変形過程が表現できることを示す．

2.3.1 基本形

アヤトリの初期状態を**図3**(a)とする．また，ここ

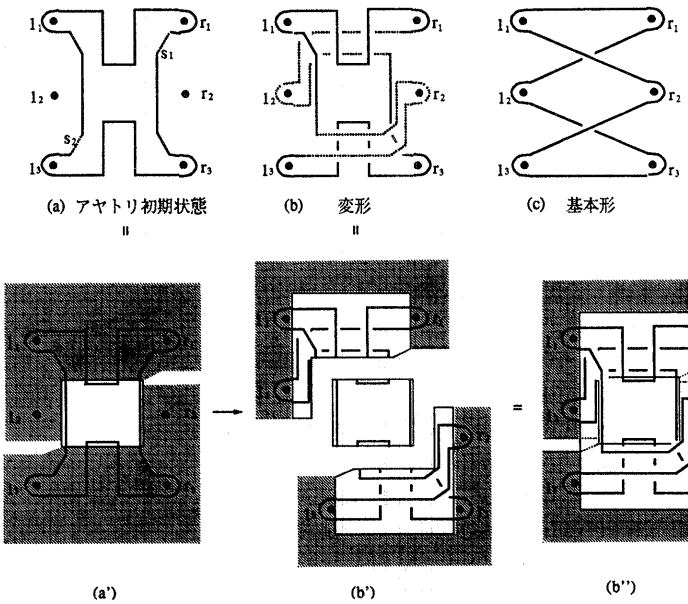


図 3 基本形過程と重ね合わせによる表現
Fig. 3 *Basic* processes and their *overlapped* diagram.

では図3(c)を基本形とする．初期状態から基本形への変形過程は(a)に示す線分 s_1 を指 l_2 で取り，線分 s_2 を指 r_2 で取ることにより表現することができる（図3(b)参照）．この変形過程は図3(a′)，(b′)，(b″)に示すように図3(a′)のハンドルを基本動作の「取る」に対応する図形に置き換えることにより表現できる．

2.3.2 ナバホどり

図4の k_0 のように二本のひもが指に掛かっている状態から変形を繰り返して，**図5**の形を作る取り方である．ナバホどりはi)取る，ii)入れ換える，iii)取る，iv)入れ換える，v)外す，vi)外すの一連の基本動作の繰り返しによって作られる．

ナバホどりはこれら一連の基本動作に対応する図形を重ね合わせることにより表現できる．ナバホどりの変形過程の表現は以下となる（図4参照）．

i) k_0 のハンドルを基本動作「取る」に対応する図形に置き換えることにより k_1 が得られる．このとき k_0 のネット n_0 は k_1 にそのまま保存される．

ii) k_1 のハンドルを基本動作「入れ換える」に対応する図形に置き換えることにより k_2 が得られる．このとき k_1 のネット n_0, n_1 は k_2 にそのまま保存される．

iii) k_2 のハンドルを基本動作「取る」に対応する図形に置き換えることにより k_3 が得られる．このとき k_2 のネット n_0, n_1, n_2 は k_3 にそのまま保存される．

iv)～vi) 同様に，基本動作「入れ換える」，「外す」，「外す」に対応する図形を順次ハンドルと置き換えることにより，ナバホどりの変形過程が表現される．

2.3.3 二段ばしご

図6(b)の形を二段ばしごと呼ぶ．二段ばしごはアヤトリの初期状態から基本形を作り，さらに，基本動作，ナバホどりを繰り返すことにより作られる（図6(a)参照）．

以上より以下の性質が言える．

[性質1] 変形過程を基本動作に対応する図形を重ね合わせて表現できるアヤトリが存在する． □

図3(c)，図5，図6(b)は基本動作が適用された過程に従い指を左右に引っ張りひもを変形したときできる形である．以降，このような図形を**完成図形**と呼ぶ．

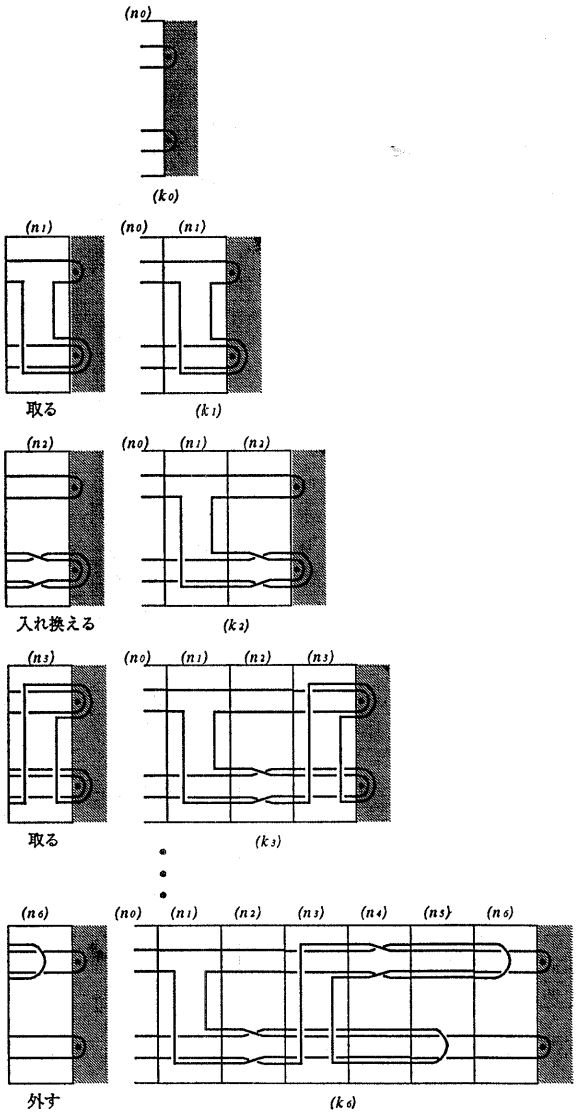


図4 ナバホどり過程の重ね合わせ図形
Fig. 4 An *overlapped* diagram of *Nabaho-dori*.

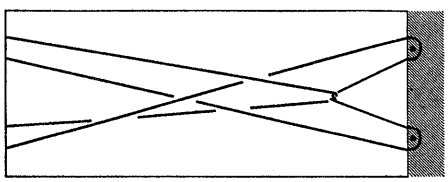
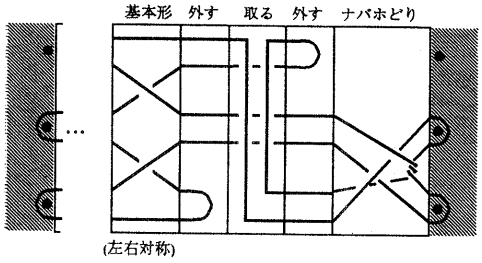
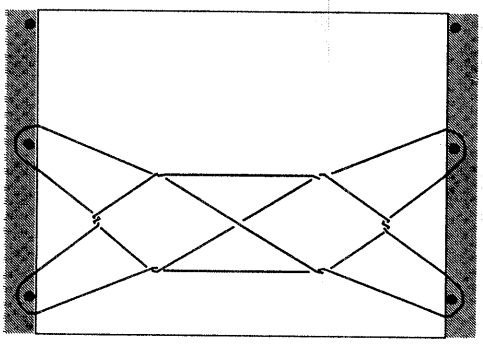


図5 ナバホどりが作る形
Fig. 5 A final pattern of *Nabaho-dori*.



(a)



(b)

図 6 二段ばしごの重ね合わせ図形と二段ばしご
Fig. 6 An *overlapped* diagram of *Nidanbashigo* and *Nidanbashigo*.

あるひとつの完成図形に至る基本動作の適応順序は一般に一意ではないことより,

[性質 2] ある完成図形に対し基本動作の図形の重ね合わせによる変形過程の表現は一意ではない. □

ここで述べた, 基本動作に対応する図形を重ね合わせる方法は1)記述が容易, 2)動作履歴が明確である利点を持つ.

3. 重ね合わせ図形の処理

この章では重ね合わせ図形から完成図形を生成する方法について述べる.

完成図形を以下に示すフェーズ 1, フェーズ 2 によって生成する. まずフェーズ1では, ライデマイスター移動[2] によるいくつかの位相的前処理を重ね合わせ図形に施し, フェーズ2では, ネット部の交差点位置の移動変形処理を行う.

3.1 ひも状態の表現

ひも図形は, 交差点 c_i と角 C_i^n, さらに, 角と角, および角と端点の連結状態を要素とする集合とする. ここでは交差点 c_i (i は交差点の番号) はその近傍にある程度の面積を持ち, 四角形で表す (**図7**参照)*.

* 四角形を十分縮小すれば近似的に点として処理できる.

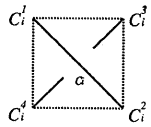


図 7 交差点 c_i
Fig. 7 A crossing c_i.

四角形の四つの角 (カド) C_i^n ($n=1,2,3,4$) のうち, 互いに対角にある角 C_i^1 と角 C_i^2, 角 C_i^3 と角 C_i^4 を結ぶ対角線はそれぞれ交差点の上方に位置するひもと下方に位置するひもに対応する. またおのおのの角 C_i^n ($n=1,2,3,4$) には一つのひもが必ず存在し, それぞれ, i)他の交差点の角 C_j^n ($i\neq j$, $n=1,2,3,4$) かまたは, ii)自分自身の交差点の他の角 C_i^m ($n\neq m$), iii)端点 e_k (k は端点の番号), のいずれか一つと連結している (線分によってつながっている).

3.2 完成図形の生成

3.2.1 フェーズ 1: ライデマイスター移動による前処理

フェーズ1では重ね合わせ図形に三種類のライデマイスター移動[2] (**図 8** 参照) の一部を施して位相的変形を行い, 交差点数が極小となる図形 (以降, **交差点極小図形**と呼ぶ) を生成する.

これらのライデマイスター移動のうち交差点数が減少するか, または, 増加しない移動を用いる. つまり, 図8で示すI, IIは交差点を減少させる方向である左から右への変形処理 r_1, r_2 のみを用い, さらに減少も増加もしないことより III は両方向の移動を用いる. なお, このフェーズでは交差点の位置は固定したままで変形処理を行うことにより, ライデマイスター移動IIIについては変形は**図9**に示す r_3 となる.

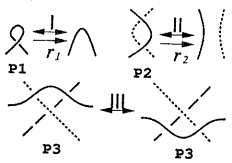


図 8 ライデマイスター移動と変形 r_1, r_2
Fig. 8 *Reidemeister moves, I, II, III* and transformations r_1, r_2.

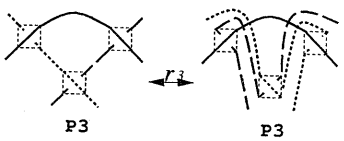


図 9 変形 r_3
Fig. 9 A transformation r_3.

重ね合わせ図形から交差点極小図形を生成するアルゴリズムを以下に示す．

3.2.2 フェーズ1処理のアルゴリズム

図形にパターン **P1**, **P2** がいずれも存在しなくなるまで，変形 r_1, r_2 を繰り返し施す．次に変形 r_3 を有限回施してできる図形を求め，このとき，再度パターン **P1**, **P2** が出現すれば変形 r_1, r_2 を施す．以上の処理を繰り返し，交差点数極小の図形を生成する．このアルゴリズムでは同数の交差点をもつ交差点極小図形が複数個得られる．これらのおのおのの交差点極小図形に対しフェーズ2の処理を施す．

[例1] フェーズ1の処理過程例を**図 10** に示す．図形 D_0 が与えられたとする．D_0 にはパターン **P1**, **P2** が存在する．よって変形 r_1, r_2 を施し，その結果 D_1 となる．D_1 にはパターン **P3** が存在する．変形 r_3 を施すことにより，D_2* が得られる．D_1 に対し変形 r_3 を繰り返し施してできる図形は D_2 と自分自身 D_1 のみである．さらに，D_1 と D_2 はどちらもパターン **P1**, **P2** が存在しないことから D_1 と D_2 は図形 D_0 に対する交差点極小図形である．

交差点極小図形生成アルゴリズム：

input K: 重ね合わせ図形；
output S: 交差点極小図形の集合；
begin
　while (K に **P1** または **P2** が存在する) **do**
　　r_1, r_2 を用いて K を変形し，得られた図形を K とする；
　end
　if (K に **P3** が存在する) **then**
　　手続き $M(K, S)$；
　else
　　S:=$\{K\}$；
end
procedure $M(K, S)$
　input K: **P1** と **P2** どちらもが存在しない図形；
　output S: 交差点極小図形の集合；
begin

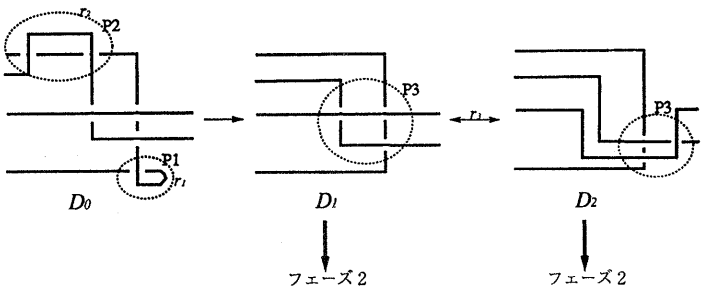


図 10 フェーズ1の処理過程
Fig. 10 A phase 1 processing.

　Q:=$\{K_i | K_i$ は K から r_3 の有限回の繰り返しで得られる図形$\}$；
　T:=$\{K_i | K_i \in Q$, K_i に **P1** または **P2** が存在する$\}$；
　if (T=空集合 ϕ) **then**
　　S:=Q；
　else
　　手続き $G(T, S)$；
end
procedure $G(T, S)$
　input T: **P1** または **P2** が存在する図形の集合；
　output S: 図形の集合；
begin
　if (T=空集合 ϕ) **then**
　　S:=空集合 ϕ；
　else
　begin
　　T のある要素 K_i に対して；
　　K_i^{copy}:=K_i；
　　while (K_i^{copy} に **P1** または **P2** が存在する) **do**
　　r_1, r_2 を用いて K_i^{copy} を変形し，得られた図形を K_i^{copy} とする；
　　end
　　手続き $M(K_i^{copy}, S_i)$；
　　手続き $G(T-\{K_i\}, S_s)$；
　　S:=$\{K' | K'$ は $S_i \cup S_s$ において交差点数が最小の図形$\}$；
　end
end

3.2.3 フェーズ2：位置の移動処理

フェーズ2ではフェーズ1により得られた交差点極

---

* 図形を明瞭に示すため変形 Ⅲ（図8参照）を用いた．実際の処理は交差点の位置を固定したままの変形 r_3（図9参照）を用いる．

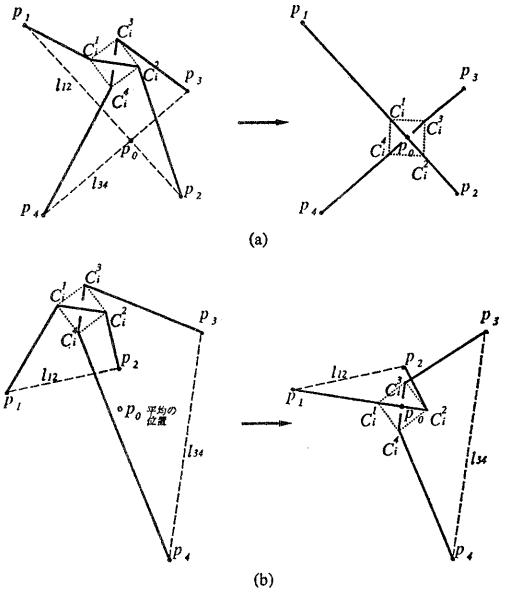


図 11 交差点 c_i の移動
Fig. 11 Moving a crossing c_i.

小図形*の交差点位置を移動させながら交差点間，端点交差点間のひもの長さの総和を小さくさせる処理を行う．このための交差点位置の移動規則を以下に示す．

a） **交差点位置の移動規則**（図 11 参照）

角 C_i^n とこれに連結する他の交差点の角または端点との距離を d_i^n とする．このとき交差点 c_i と連結するひもの長さ d_{ci} を以下とする．

$$d_{ci}=\sum_{n=1}^{4} d_i^n. \tag{1}$$

また，端点 e_k とこれに連結する角または他の端点との距離を de_k とし，さらに図形中に存在する交差点と端点の個数をそれぞれ N_c, N_e とする．このとき，ひもの長さの総和 L を以下とする．

$$L=\frac{1}{2}\left(\sum_{i=1}^{N_c} d_{ci}+\sum_{k=1}^{N_e} de_k\right). \tag{2}$$

交差点 c_i の各角 C_i^n $(n=1\sim4)$ が他の交差点の角または端点 p_m $(m=1\sim4)$ に連結しているとき，以下の二つの規則によって c_i を移動する（n と m の値は順不同とする）．

移動規則 1：p_1 と p_2 を結ぶ線分 l_{12} と p_3 と p_4 を結ぶ線分 l_{34} が一点で交わるとき，その点を交差点 c_i の新しい位置 p_0 とする*（図 11(a) 参照）．

移動規則 2：上記の l_{12} と l_{34} が一点で交わらないとき，p_m $(m=1\sim4)$ の相加平均の位置を交差点 c_i の新しい位置 p_0 とする**（図 11(b) 参照）．

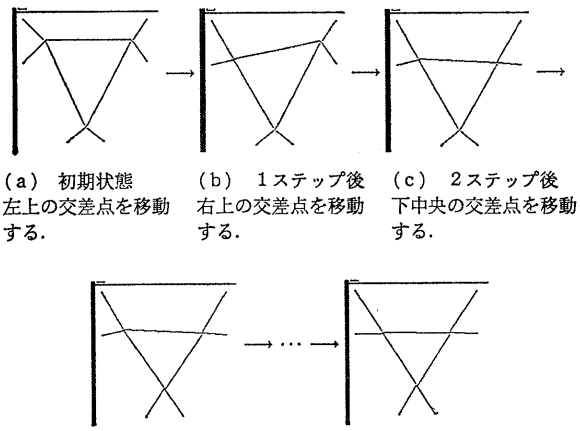
移動規則 1 における移動先は交差点 c_i に連結するひもの長さ d_{ci} を明らかに最小とする．また，移動規則 2 における p_0 の位置は d_{ci} を極小とする[7]．さらに d_{ci} が交差点移動により減少するとき，ひもの長さの総和 L もまた減少する．

交差点 c_i の四つの角 C_i^n $(n=1,2,3,4)$ の移動後の位置は四角形が十分小さいときは便宜的に次のように決めることができる．

交差点移動後の角の位置：交差点 c_i の移動先の位置 p_0 と p_1 を結ぶ線分上に C_i^1 を決め，C_i^3, C_i^2, C_i^4 をそれぞれ $\pi/2$, π, $3\pi/2$ と右回りにして移動後の位置とする（図 11 参照）．

b） **長さ縮小処理**

フェーズ 1 で得られた交差点極小図形に対し以下に示す長さ縮小処理を施し完成図形を生成する．逐次的に交差点を移動させて長さ縮小処理を実行するため，



(a) 初期状態 左上の交差点を移動する．
(b) 1 ステップ後 右上の交差点を移動する．
(c) 2 ステップ後 下中央の交差点を移動する．
(d) 3 ステップ後 左上の交差点を移動する．
(e) 変化しなくなる．

図 12 フェーズ 2 の処理過程
Fig. 12 A phase 2 processing.

* 図 10 の例においては，D_1 と D_2 それぞれに対しフェーズ 2 の処理が行われる．

* p_m $(m=1\sim4)$ を頂点とする凸四角形が存在する．移動先はこの凸四角形における対角線の交点である．

** p_m $(m=1\sim4)$ を頂点とする凹四角形が存在する．p_m $(m=1\sim4)$ の位置をそれぞれ (x_m, y_m) $(m=1\sim4)$ とするとき，これらの相加平均の位置 (x_0, y_0) に移動する．ただし，

$$x_0=\frac{x_1+x_2+x_3+x_4}{4},\quad y_0=\frac{y_1+y_2+y_3+y_4}{4}.$$

この処理ではプログラム自身で停止時点を決定することはできないことより，交差点を順次移動させる過程を表示し，人間が表示画面を確認した時点で会話的に停止させる．

長さ縮小処理：

step 1 交差点極小図形の交差点を任意に順序付ける．

step 2 交差点移動規則1および2に従い交差点を順次移動させる．

step 3 外部から停止指示があれば処理を停止する．停止指示がなければ step 2 へ行く．

[例2] フェーズ2の長さ縮小処理の過程例を**図 12** に示す．この例では1ステップ目に左上の交差点を移動し，2ステップ目で右上の交差点を移動し，3ステップ目で中央下の交差点を移動している．3個の交差点をこの順序で繰り返し移動し，(e)へ推移する．

4. 実 行 例

ナバホどりの重ね合わせ図形にフェーズ1，フェーズ2の処理を施した結果を示す．目標となる図形は図5である．ナバホどりの重ね合わせ図形（図 4 k_6 参照）にフェーズ1の処理を施したところ，4個の交差点極小図形が得られた．4個の交差点極小図形を**図 13**(a) の D_1, D_2, D_3, D_4 に示す．D_1 と D_2, D_3 と D_4 はそれぞれ互いに各角の連結のしかた，すなわち平面的位相が等しく，かつ，交差点の位置が異なる．D_1, D_2, D_3, D_4 にフェーズ2の処理を施し生成された完成図形をそれぞれ図 13(b) の D_1', D_2', D_3', D_4' に示す．

フェーズ2により得られる完成図形は，どの交差点を始めに移動させるかによって多少異なるが，交差点極小図形の平面的位相と端点の位置に強く依存することが実験により確かめられた．

5. おわりに

ひもの変形に関する計算機シミュレーションの研究報告は少ないが，自動的に変形を行ったシミュレーションの報告は文献 3) でなされている．文献 3) では3次元デジタル空間を考え，ひもを空間内で接するセルの集合により表現し，ひもの長さを短くする処理を，セルを減少させるアルゴリズムにより実現している．

これに対し，ここでは，ひもを表現する平面図形の位相パターン，および幾何学的構成に着目し，2次元

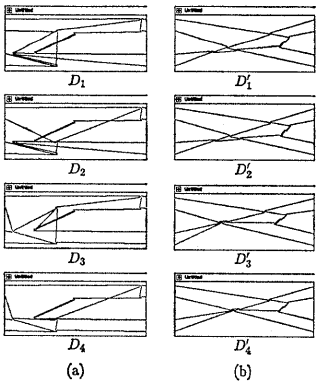


図 13 完成図形生成例
(a) ナバホどりの重ね合わせ図形から生成された交差点極小図形 $D_i (i=1,2,3,4)$, (b) D_i から生成された完成図形 D_i'

Fig. 13 An example of generating final pattern diagrams.
(a) Generated minimum crossing number diagrams $D_i (i=1,2,3,4)$ from an *overlapped* diagram of *Nabaho-dori*, (b) Generated final pattern diagrams D_i' from D_i.

平面上での変形を行った．その結果，3次元空間内で複雑な位相を持つひもを表現する図形の効率的な変形処理が可能となった．また妥当な完成図形を生成することができた．

以上，本論文では基本動作に対応する図形の重ね合わせにより，アヤトリ変形過程を表現する方法を提案した．また，基本動作の履歴を保存した重ね合わせ図形から完成図形を生成する方法について述べた．さらに，実験プログラムによりその妥当性を確認した．

なお本論文では，ここで示した完成図形生成方法が適用可能なアヤトリが存在することを比較的複雑で代表的なものとして知られている例を用いて示した．しかし，どの程度の範囲のアヤトリに適用可能であるかについては未解決の問題として残された．本手法が適用できないアヤトリとしては，例えば a) 立体的な形の表現を重視するアヤトリ，b) 出来上がり時において，ゆるんだひもにより形を表現するアヤトリなどがある．a) に含まれるアヤトリに対しては，本手法におけるひもの2次元的表現では不十分である．また本手法はひもを両側から引っ張り出来上がり形を作るアヤトリを対象としており，b) に含まれるアヤトリは扱っていない．しかし a)，b) に含まれない他のいくつかのアヤトリ（二段ばしごなど）に対しては良好な結果が得られており，本手法がかなりの範囲に対し適用可能であると考えられるが，これを実証することは

今後の課題である．

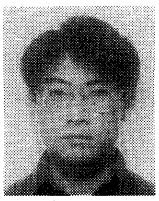
最後に，本論文における主張点を以下にまとめる．

1) 履歴が明確である重ね合わせ図形により変形過程が表現できるアヤトリが存在する．
2) ここで示したひも状態の表現方法により重ね合わせ図形を表現し，処理することができる．
3) 本完成図形生成方法により妥当な完成図形を生成できるアヤトリが存在する．

なおここで示した手法はアヤトリに限らず，一般のひも図形や複雑な位相を持つ図形の変形処理に応用でき得る．また，フェーズ2の長さ縮小処理は交差点の配置問題と考えることができ，現在，この観点からより良いアヤトリの形を生成する手法の拡張を試みている．

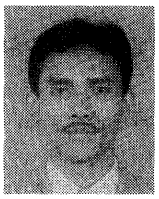
参考文献

1) 野口 広：母と子のやさしいあやとり遊び，梧桐書院 (1987).
2) Kauffman, L. H.: *On Knots, Annals of Mathematics Studies*, Vol. 155, Princeton Univ. Press, Princeton (1987).
3) 斉藤豊文，横井茂樹，鳥脇純一郎：3次元ディジタル線図形のトポロジー—結び目の解析，電子情報通信学会技術研究報告，PRU 90-83 (1990).
4) Hoste, J. and Przytycki, J. H.: An Invariant of Dichromatic Links, *Pro. Am. Math. Soc.*, Vol. 105, No. 4, pp. 1003-1007 (1989).
5) Miyazawa, Y.: Symmetry of Dichromatic Links, *Pro. Am. Math. Soc.*, Vol. 114, No. 4, pp. 1087-1096 (1992).
6) Hasslacher, B. and Meyer, D. A.: *Knot Invariants and Cellular Automate, Cellular Automate Theory and Experiment*, The MIT Press, pp. 328-344 (1991).
7) 岩田至康：幾何学大辞典 第1巻 基本定理と問題—平面—，pp. 364-365, 槙書店 (1978).
8) 山田雅之，R. Budiarto, 伊藤英則，世木博久：点で制約されている結び目の特性化について，情報処理学会第46回全国大会，3-13 (1993).
9) 山田雅之，R. Budiarto, 伊藤英則，世木博久：アヤトリの表現方法とその処理，形の科学会報，Vol. 8, No. 1, pp. 7-8 (1993).



山田 雅之（学生会員）

1992年名古屋工業大学工学部電気情報工学科卒業．現在同大大学院博士前期課程在学中．図形処理，遺伝的アルゴリズム等に興味を持つ．人工知能学会会員．



Rahmat Budiarto

1986年 Bandung 工業大学数学科卒業．Bandung 工業専門学校講師，Nusantara Aircraft Industries Ltd. 勤務．現在名古屋工業大学工学部大学院博士前期課程在学中．システム設計，結び目理論に関する研究に従事．

伊藤 英則（正会員）

1974年名古屋大学大学院工学研究科博士課程電気・電子専攻満了．工学博士号取得．1974年日本電信電話公社横須賀研究所勤務．1985年(財)新世代コンピュータ技術開発機構出向．1989年名古屋工業大学教授．現在知能情報システム学科所属．この間，数理言語理論，計算機ネットワーク通信 OS，知識ベースシステムなどの研究開発に従事．電子情報通信学会，人工知能学会，ファジー学会，形の科学学会各会員．

世木 博久（正会員）

1979年東京大学工学部計数工学科卒業．1981年同大学院工学系研究科修士課程修了．同年4月より三菱電機(株)中央研究所に勤務．1985年～1989年(財)新世代コンピュータ技術開発機構に出向．1992年4月より名古屋工業大学工学部知能情報システム学科助教授．工学博士．論理プログラミング，演繹データベース等に興味を持つ．電子情報通信学会，人工知能学会，ACM, IEEE Computer Society 各会員．