数理解析研究所講究録
第868巻 1994年 116-128

ソリトンセルオートマトンと組合せ論

東京大学数理科学研究科
鳥居 真 (Makoto TORII) *

1 はじめに

十年ほど前から，非常に離散性の強い系であるセルオートマトン系に対してもソリトン現象が発見され，研究されている．([8]) ソリトンを持つオートマトン系の幾つかは完全可積分と言い得る特長を有していて，これから取り扱うオートマトンの場合にはその可算無限個の保存量と呼べるものが構成できる．この保存量を構成する際にあまり今までの可積分系にはみられなかった組合せ論的な側面が見られ，その一つに Robinson-Schensted 対応がある．今回はこの Robinson-Schensted 対応を用いたあるソリトンセルオートマトン系の保存量構成方法とその保存性を証明する．

2 ソリトンセルオートマトン

ここで取り扱うのは，以下に示す時間発展規則に従うセルオートマトンである．このセルオートマトンはソリトンのみを発生するオートマトンの中で，知られている限り最も簡単な時間発展規則に従うものであり，より複雑な時間発展規則を持つソリトンセルオートマトンの組合せ論的な特性を理解する上での試金石としての役割を果たしてくれるのではないかと期待される．([5])

以後扱う空間及び時間は全て離散的で，空間は左右に格子点状に配置されているものとし，時間および空間は共に1次元とする．各格子点における関数値は，1もしくは0の2値のみをとるものとする．空間は左右に無限に続いているものとするが，各時点の状態で1という値をとっている格子点は有限個しかないものとする．すなわち十分左方と右方では格子点上には0という値しか存在しないとする．

2.1 時間発展規則

この設定の下で次の手続きに従って1時刻後の状態を決める．

1. 1という値を持つ格子点の中で，最も左にあるものに注目する．
2. 注目した1のある格子点から右方向を順に見て，最初に現れた0の位置に1を移動する．
3. 次の時刻に進むまでこの移動した1は固定する．
4. 未だ固定されていない1を持っている格子点の中で，最も左にあるものに注目し，以後同様の過程を繰り返す．
5. 全ての1が固定された状態になったら，その状態を1時刻後の状態とし，全ての格子点の1の固定を解除し，最も左にある1から移動を再開する．

*e-mail:torii@sat.t.u-tokyo.ac.jp

具体例として時刻tにおいて $\cdots 01101000\cdots$ という状態であったオートマトンの時刻$t+1$の状態を導く．ここで1の動きを明瞭にするため，1に添字A,B,Cをつけて，オートマトンの状態を

$$\cdots 0\ \ 1_A\,1_B\,0\ \ 1_C\,0\ \ 0\ \ 0\ \ \cdots$$

と表示している．

まず，最初に移動するのは1の中で最も左にある1_Aである．1_Aの右にあり，かつ最も近接した0は1_Bの右隣の0であり，この0の位置へ1_Aは移動する．この結果，時刻$t+1$へ至る中間状態としてオートマトンは次の状態になる．

$$\cdots 0\ \ 0\ \ 1_B\,\overline{1}_A\,1_C\,0\ \ 0\ \ 0\ \ \cdots$$

(1_Aが固定されたことを上線をつけて表している．)

次に未だ固定されていない1_Bと1_Cのうち，左方にあるのは1_Bである．この1_Bの右にあり，かつ最も近接した0は1_Cの右隣にある0であるので，この0の位置へ1_Bは移動する．すると時刻$t+1$へ至る第二の中間状態として，オートマトンは次の状態になる．

$$\cdots 0\ \ 0\ \ 0\ \ \overline{1}_A\,1_C\,\overline{1}_B\,0\ \ 0\ \ \cdots$$

最後に未だ移動していない1_Cを移動する．この1_Cの右にあり，かつ最も近接した0は$\overline{1}_B$の右隣の0である．この0の位置へ1_Cは移動する．この結果，オートマトンは次の状態になる．

$$\cdots 0\ \ 0\ \ 0\ \ \overline{1}_A\,0\ \ \overline{1}_B\,1_C\,0\ \ \cdots$$

これで全ての1の移動は終了したので，この状態をもって時刻$t+1$の状態とし，全ての1の固定を解除する．以後，全く同様の過程を時刻$t+1$の状態に対して実行し，次時刻の状態を導出して行けばよい．また上記の過程を，左を右におきかえ，「最も右にある1を自分の左にあってかつ最も近接した0に移動する」と書き直すと，時間を遡行して発展する規則になる．時刻$t+1$の状態に対してこの遡行規則を実行すれば，上記の過程を全く逆に辿ることになり，時刻tの状態が復元され，この系は可逆系であることがわかる．

2.2 ソリトンの挙動の観察

まず最も簡単な2体の衝突を観察する．

$$\cdots 1100010000000000\cdots$$
$$\cdots 0011001000000000\cdots$$
$$\cdots 0000110100000000\cdots$$
$$\cdots 0000001011000000\cdots$$
$$\cdots 0000000100110000\cdots$$
$$\cdots 0000000010011000\cdots$$

上の例には，長さが2の連続した1の列が右方向へ速度2で進行し，長さが1の速度1で右方向へ進行している1という列に接近し，衝突する様子が示されている．衝突は上の例の3行目から4行目にかけて起こり，この時刻で“位相のずれ”が生じている．

もっと複雑な3体の衝突も生じる.

$$\cdots 0\,1\,1\,1\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\cdots$$
$$\cdots 0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\cdots$$
$$\cdots 0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\cdots$$
$$\cdots 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\cdots$$
$$\cdots 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,1\,0\cdots$$

上の例には，長さ3である連続した1の列が長さ2の列および長さ1の列の塊に衝突(第2行から第3行)し，その後長さ2の1の列が長さ1の列に衝突(第4行から第5行)する様子が示されている．一般のN体の衝突に対しても同様の過程が見られ，衝突の前に各々の1の列がお互いに十分離れた距離に分離されており，その後衝突が起きたとすると，さらに十分長い時間の経過を待てば1の列が再び分離されて現れ，そのとき現れる1の列の長さの集合は，衝突の前に分離されて存在していた1の列の長さの集合と等しい．すなわち衝突によって1の長さという“孤立波の個性”は失われることがない．

この特徴はこのセルオートマトンがソリトン系であるということを意味していると考えられ，以後この意味で長さnの1の列を長さnのソリトンと呼ぶことにする．

3 組合せ論的手法

オートマトン系がソリトンを持つということは，この系が何らかの意味で可積分系であると考えられる．その一つの証拠として無限個の保存量を構成するが，その手段として用いられる組合せ論である，Dyck言語，スタック表現可能順列，及びRobinson-Schensted対応について紹介する．

3.1 Dyck言語

二つの文字(と)とで作られたある有限列，例えば))(()() や (()()) 等を考える．今挙げた例では，前者は(と)の対応がとれておらず，後者は(と)が正しく対応している．正確に言えば，列に含まれる(と)の数は等しく，かつ列中にある任意の)を一つとったとき，その)よりも左にある(の数と)の数の差は必ず正となっている．

この様に正しく対応のとれている長さ$2n$の(と)の列のことを，(,)をアルファベットとする長さnのDyck言語と呼ぶ．([1],[2],[4]) 例えば，長さ3のDyck言語の要素は次の5列である．

$$((())) \quad (())() \quad (()()) \quad ()(()) \quad ()()()$$

3.2 スタック表現可能順列

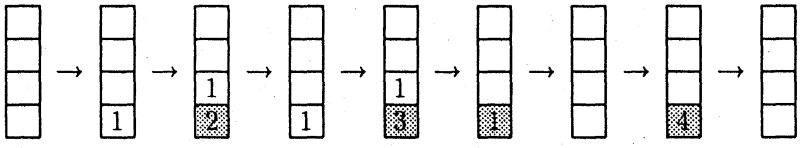
順列$x_1, x_2, \cdots, x_n$を考え，その中から長さ3の部分列x_i, x_j, x_k, $i < j < k$を抽出したとき，どんな部分列を抽出しても$x_i > x_k > x_j$となることが無い(すなわち，要素が大小中なる順序に並ぶ部分列ではない)ならば，順列$x_1, x_2, \cdots, x_n$はスタック表現可能順列であると言う．例えば長さ3の順列の集合では，3,1,2だけがスタック表現可能順列ではない．長さ4の順列の場合，4,3,1,2，3,4,1,2，3,1,4,2，3,1,2,4等は全て部分列として3,1,2を含むので，スタック表現可能順列ではない．縦一列に数を並べたもの(スタック)を考え，その内の数を全て一段上

に上げて最下段に数を挿入する操作をスタックへのプッシュ，最下段から数を削除してその上にあった数を全て一段下げる操作をポップと呼ぶ．スタック表現可能順列は，順列 $1,2,\cdots,n$ を 1 から順にプッシュとポップという二つの操作によってスタックに挿入，削除するときに出力として得られる順列である．

例えば順列 $2,3,1,4$ は，順列 $1,2,3,4$ に対して次のスタックへの一連の操作を行なうことによって得ることができる．

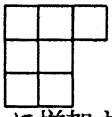
1. “1” をスタックへプッシュ．
2. “2” をスタックへプッシュ．
3. “2” をスタックからポップ．
4. “3” をスタックへプッシュ．
5. “3” をスタックからポップ．
6. “1” をスタックからポップ．
7. “4” をスタックへプッシュ．
8. “4” をスタックからポップ．

この操作の途中でのスタックの状態，及びスタックからの出力 (ハッチングして示した) は，



であり，スタックからの出力結果を順に並べれば，$2,3,1,4$ となる．

3.3 Robinson-Schensted 対応

自然数 n の分割 $\lambda=(\lambda_1,\lambda_2,\cdots,\lambda_k)$ とは，非増加な数列でその総和が n になるものをいう．λ によって定まる Young 図形とは第 k 列に λ_k 個の箱を並べたものである．例えば $\lambda=(3,2,2)$ に対しては  が対応する．n を Young 図形の大きさと呼ぶ．大きさ n の Young 図形に，縦横ともに増加するように 1 から n までの数を記入したものが標準 Young 盤であり，長さ n の順列と等しい台を持つ標準 Young 盤 P,Q の対 (P,Q) は Robinson-Schensted 対応と呼ばれる方法によって全単射対応させることができる．([3],[6]) その方法は多少複雑であるので，いくつかに分けて説明する．

まず，一つの行に対する挿入アルゴリズムを説明する．単調増加するように数が記入されている一行の箱を考える．この行に数 x を以下の手続きに従って挿入する．

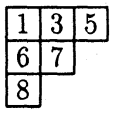
1. x を挿入すべき行に何も要素が存在しない場合は，新たに箱を一個作成し，その中に x を記入する．
2. 行の要素が空でない場合，左から順次行の各箱中の数を x と比較する．
   (a) 箱の中の数が x 以下である場合には何の操作せず，右隣の数と x との比較に移行する．

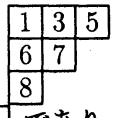
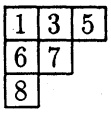
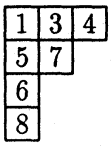
(b) x よりも箱の中の数が大きい場合，箱の中の数を行から排出し，排出後の空箱へは代わりに x を記入する．

3. 行の右端まで比較を行なっても x よりも大きな数が存在しない場合，行の右端に新たに一個の箱を作り，その中に x を記入する．

この行に関する挿入操作を用いて，標準 Young 盤へ数 x を挿入する．

1. まず，与えられた標準 Young 盤 P を行の集合に分割する．
2. 第一行に対して x を挿入する．
3. もし第一行から排出された数があれば，その数を x' とし，第二行に対して x' を挿入する．排出された数がある場合は順次その排出された数を次行に挿入し続ける．
4. 排出される数が存在せず，挿入された数が新たに作成された箱に記入された時点で標準 Young 盤への挿入操作を終了する．

標準 Young 盤  に 4 を挿入する場合を例にとり，標準 Young 盤への挿入操作を行なってみる．

1. 与えられた標準 Young 盤  を行の集合に分割すると，第 1 行は $\boxed{1}\boxed{3}\boxed{5}$，第 2 行は $\boxed{6}\boxed{7}$，第 3 行は $\boxed{8}$ であり，第 4 行以降は全て要素が空の行である．
2. 第 1 行 $\boxed{1}\boxed{3}\boxed{5}$ に 4 を挿入すると，第 1 行は $\boxed{1}\boxed{3}\boxed{4}$ となり，5 が排出される．
3. 第 1 行から排出された 5 を第 2 行 $\boxed{6}\boxed{7}$ に挿入すると，その結果第 2 行は $\boxed{5}\boxed{7}$ となり，6 が排出される．
4. 第 2 行から排出された 6 を第 3 行 $\boxed{8}$ に挿入すると，その結果第 3 行は $\boxed{6}$ となり，8 が排出される．
5. 第 4 行は要素が空なる行であるので，8 を挿入すれば一個の箱だけからなる行 $\boxed{8}$ が得られる．このとき，第 4 行から排出するべき数は存在しないから，ここで標準 Young 盤への挿入を終了する．
6. 4 を  に挿入した結果最終的に得られた Young 盤は，  となる．

このとき得られた新しい Young 盤は，再び標準 Young 盤となっている．

長さ n の順列 $x_1, x_2, \cdots, x_n$ を順に挿入することによって Robinson-Schensted 対応が得られる．

1. まず初期状態として，空の Young 盤の対 (P, Q) をとる．
2. Young 盤に順列 $x_1, x_2, \cdots, x_n$ の要素を，まず x_1，次に x_2 と左の要素から順次挿入した結果得られる標準 Young 盤を P とする

3. x_k を P に挿入する過程を終了するとき，P に新たな箱が作成される．Q の同じ位置にも新たに箱を作成するが，その箱の中には数 k を記入する．

順列 $6,3,7,1,4,2,5$ を例に用い，Robinson-Schensted 対応を実際に実行する．

まず，初期には標準 Young 盤の対 (P,Q) は共に空である．P に順列の最左端にある数 6 を挿入すると $\boxed{6}$ となる．Q にも同じく箱を一個新たに作成し，1 を記入する．(P,Q) は次の様になる．順列の数の上に付したハーケン記号は，既に挿入を終えたことを表す．

$$(\check{6},3,7,1,4,2,5) \quad \left(P=\boxed{6},Q=\boxed{1}\right)$$

上の ($\boxed{6}$, $\boxed{1}$) に対して，順列の左から 2 番目にある数 3 を挿入する．すると，次の (P,Q) が得られる．

$$(\check{6},\check{3},7,1,4,2,5) \quad \left(P=\begin{array}{|c|}\hline 3\\\hline 6\\\hline\end{array},Q=\begin{array}{|c|}\hline 1\\\hline 2\\\hline\end{array}\right)$$

以後同様に $7,1,4,\cdots$ を挿入して行けばよい．挿入の結果だけを示す．

$$(\check{6},\check{3},\check{7},1,4,2,5) \quad \left(P=\begin{array}{|c|c|}\hline 3&7\\\hline 6\\\cline{1-1}\end{array},Q=\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\right)$$

$$(\check{6},\check{3},\check{7},\check{1},4,2,5) \quad \left(P=\begin{array}{|c|c|}\hline 1&7\\\hline 3\\\cline{1-1}6\\\cline{1-1}\end{array},Q=\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}4\\\cline{1-1}\end{array}\right)$$

$$(\check{6},\check{3},\check{7},\check{1},\check{4},2,5) \quad \left(P=\begin{array}{|c|c|}\hline 1&4\\\hline 3&7\\\cline{1-2}6\\\cline{1-1}\end{array},Q=\begin{array}{|c|c|}\hline 1&3\\\hline 2&5\\\cline{1-2}4\\\cline{1-1}\end{array}\right)$$

$$(\check{6},\check{3},\check{7},\check{1},\check{4},\check{2},5) \quad \left(P=\begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\cline{1-2}6&7\\\cline{1-2}\end{array},Q=\begin{array}{|c|c|}\hline 1&3\\\hline 2&5\\\cline{1-2}4&6\\\cline{1-2}\end{array}\right)$$

$$(\check{6},\check{3},\check{7},\check{1},\check{4},\check{2},\check{5}) \quad \left(P=\begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4\\\cline{1-2}6&7\\\cline{1-2}\end{array},Q=\begin{array}{|c|c|c|}\hline 1&3&7\\\hline 2&5\\\cline{1-2}4&6\\\cline{1-2}\end{array}\right) \tag{1}$$

4 保存量の構成

今まで説明した組合せ論の方法を使ってソリトンセルオートマトンの保存量を構成する．

4.1 セルオートマトンと Dyck 言語との対応

第一段階として，ソリトンセルオートマトンの状態と Dyck 言語の要素を対応させる．

1. 1 を全て (に置換する．

2. 0 を次の規則に従って左から順次) に置換する．

   - 対象となっている 0 の左方にある (と) の数の差が正であるならば，その 0 は) に置換する．
   - それ以外の場合，すなわち 0 の左方にある (と) の数が等しいか，もしくはそれらの差が負である場合，その 0 は消去する．

例えば $\cdots 0011010001000\cdots$. というオートマトンの状態は次のように Dyck 言語に変換される.

$$\begin{array}{ccccccccccccccc} \cdots & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots \\ \cdots & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \cdots \\ \cdots & & & (& (&) & (&) &) & & (&) & & \cdots \end{array}$$

4.2 Dyck 言語とスタック表現可能順列との対応

Dyck 言語の要素を更に以下のようにしてスタック表現可能順列へと対応させる.

- (をスタックへの挿入 (プッシュ)
-) をスタックからの削除及び出力 (ポップ)

へと対応させ, Dyck 言語の要素である (,) の列を左から右へと順に読みながら対応するスタック操作を行ない, このとき出力されたスタック表現可能順列をとる.

例えば Dyck 言語の要素 (( ) (( ))) ( ) は, スタック表現可能順列 $2,4,3,1,5$ に対応させられる.

4.3 スタック表現可能順列に対する Robinson-Schensted 対応

得られたスタック表現可能順列に対して, 更に Robinson-Schensted 対応を行なう. この結果, ソリトンセルオートマトンの保存量が Young 図形の形で取り出される.

長さ 2 のソリトンと長さ 1 のソリトン 2 個の衝突を例をもちいて, 上記の 3 つの対応がどの様な結果を与えるか見ることにする. オートマトンの時間発展は,

$$\begin{array}{l} \cdots 0110010001000000000\cdots \\ \cdots 0001101000100000000\cdots \\ \cdots 0000010110010000000\cdots \\ \cdots 0000001001101000000\cdots \\ \cdots 0000000100010110000\cdots \end{array}$$

Dyck 言語の要素, スタック表現可能順列, 標準 Young 盤に対応させると,

$$\begin{array}{lll} (()())()() \to & 2,1,3,4 & \to \left(\begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2\\\cline{1-1}\end{array}\ , \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2\\\cline{1-1}\end{array} \right) \\ (()()())() \to & 2,3,1,4 & \to \left(\begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2\\\cline{1-1}\end{array}\ , \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3\\\cline{1-1}\end{array} \right) \\ ()(()())() \to & 1,3,2,4 & \to \left(\begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3\\\cline{1-1}\end{array}\ , \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3\\\cline{1-1}\end{array} \right) \\ ()(()()) \to & 1,3,4,2 & \to \left(\begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3\\\cline{1-1}\end{array}\ , \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4\\\cline{1-1}\end{array} \right) \\ ()()(()) \to & 1,2,4,3 & \to \left(\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4\\\cline{1-1}\end{array}\ , \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4\\\cline{1-1}\end{array} \right) \end{array} \tag{2}$$

となる.

この例には，一つの初期状態から時間発展させた結果生じるオートマトンの各状態を対応させた結果最終的に得られる標準 Young 盤の台は，どれも同一であることが示されている．すなわちこの Young 図形そのものが，セルオートマトンの保存量となっている．任意の Young 図形の形を指定するには，全ての列の高さを指定しなければならず，空の列はその高さが 0 であると考えれば，一つの Young 図形は各列の高さという無限個の保存量を定めていると考えることができる．

5 保存性の証明

以後上で構成した Young 図形が時間発展に対して不変であることを証明する．そのためにまず，時間発展の規則を Dyck 言語の補完規則に書き換える．その後，Dyck 言語の部分列を削除することで対応する標準 Young 盤がどう変化するかを調べ，時間発展，遡行の両規則によって標準 Young 盤の台が形を変えないことを示す．

5.1 Dyck 言語による時間発展の記述

時間発展の規則が Dyck 言語上で (を与えたときに) を補完する操作に等しいこと，すなわちオートマトンの各状態を Dyck 言語に対応させるとき，) に置換される 0 は次の時刻に 1 が移動する位置であることを示す．証明の便宜上，Dyck 言語に変換する時に消去すべき 0 を − に置換して表し，残したまま考えることにする．証明は帰納法による．

1. 時刻 t の状態で，もっとも左にある 1 のさらに左にある無限個の 0 は，(の数も) の数も 0 であるから全て − に置換される．

2. オートマトンが時刻 t から $t+1$ へ移る中間段階で，k 個の 1 を移動し終って固定した状態を考える．またオートマトンの状態を Dyck 言語に変換する中間段階で k 個の 0 を) に置換した状態を考える．これらの両中間段階までは移動を終了して固定されている 1 のある位置は，) に置換された 0 の位置と一致していたと仮定する．この仮定は，現状態が Dyck 言語への変換の中間段階であることから，どの) の左にも未変換の 0 は存在せず，全て) もしくは − となっていることを意味している．このとき，以下のことが成り立つ．

   - 次の中間段階に至る際に移動されるべき 1 が，その移動すべき先に現段階で存在している 0 について考えると，この 0 の左にある (の数と) の数の差は必ず非負である．なぜならば，もしも (の数が) の数よりも小さかったならば，) に対応するすでに固定された 1 が移動される前に占めていた位置は，必ず 1 が右に移動することから 0 の左にあるこれら (のいずれかでなければならず，矛盾をきたすからである．
   - また，この 0 がすでに (に対応していることはありえない．なぜならばもし (に対応しているならば，現段階でこれが 0 であるということは，以前に存在した 1 が移動した後に現れた 0 だということになり，次の中間段階に至る際に移動されるべき 1 より右にある 1 が以前に移動されていることになる．これはもっとも左にある 1 を選んで移動するという時間発展の規則に矛盾する．
   - 逆に現段階で未だ) もしくは − に置換されておらず，かつ最も左に位置する 0 について考える．もし，この 0 の左にある (の数と) の数の差が正であるならば，この 0 の左には未移動の 1 があることになる．この未移動の 1 は次の段階で必ずこの 0 の位置へ移動しなくてはならない．なぜならば，もし 1 がこの 0 の左で既に − に置換された 0 の位置へ移動するとすると，この − へ置換された 0 の左には同数の (と) があること，すなわち既に移動を終えた 1 しかなかったことに矛盾するからである．

- これより，現段階で未だ) もしくは − に置換されていない 0 はその左にある (と) の数が等しいときには − に変換されていくので，次に) に置換される 0 が存在するときには，その 0 はオートマトンの次の中間段階で 1 が移動してくる位置になっている．

3. 以上より，次の中間段階に移るときに移動する $k+1$ 個目の 1 が移動する先は， Dyck 言語への変換の中間段階で $k+1$ 番目の) へと置換される 0 の位置に等しい．

4. 結局時刻 $t+1$ の状態になったときに 1 が占めている位置は，時刻 t の状態を Dyck 言語へ変換したときに) が占めている位置に等しい．

時刻 t に 1 が存在する位置を (に置換し，) を補って Dyck 言語を作る操作を右補完と呼ぶことにする．逆に右補完操作の左右を置き換えて考え，時刻 t に 1 が存在する位置を) に置換し，左に (を補って Dyck 言語を作る操作を左補完と呼ぶことにする．時間発展規則の左右を置き換えて時間遡行の規則にし，上の過程を同様にたどれば，左補完によって (に対応させられる 0 の位置が時刻 $t-1$ において 1 が占めていた位置であることがわかる．

5.2 Dyck 言語と標準 Young 盤

Dyck 言語の要素をスタック操作と考えるとき，その操作からスタック表現可能順列を作る操作を逆にたどれば，スタック表現可能順列から Dyck 言語を作ることができる．つまり， Dyck 言語の要素とスタック表現可能順列は全単射対応している．

Dyck 言語は次の二つの操作を繰り返すことで作り出すことができる．

1. 既に得られている Dyck 言語を中に含むように最外枠に一つの (と) を追加する．

2. 既に得られている二つの Dyck 言語を並列する．

例えば操作 1は， (( )) ( ) → ((( )) ( )) 操作 2は ( ), (( )) → ( ) (( )) と表せる．このスタック操作の変更によって，得られるスタック表現可能順列もまた変化する．すなわち操作 1によって，スタック表現可能順列はその全ての数に 1 が加えられ，かつ最後尾に数 1 を付け加えた順列に変化する．例えば上の例では，スタック表現可能順列 $2,1,3$ が $3,2,4,1$ に変化している．操作 2によって Dyck 言語 A と B が連結されて AB になったとすると (例えば上の例では $A =$ ( ), $B =$ (( )) である) 連結後のスタック操作は操作 A に引き続いて操作 B を行なうというスタック操作に他ならない．この結果生じる順列の変化は， B に対応するスタック表現可能順列の各数に， A に対応するスタック表現可能順列の最大数を加え，その後 A に対応するものを左に， B に対応するものを右に並列配置したものに等しい．上例では，スタック表現可能順列 1 と 2,1 から，新たなるスタック表現可能順列 $1,3,2$ が得られている．

前述の二種類のスタック操作の変更を加えたとき， Robinson-Schensted 対応の結果得られる標準 Young 盤の台がいかに変化するかを考える．

1. 最外枠に括弧を追加する場合には，得られる標準 Young 盤の第一列の高さのみが以前よりも 1 だけ増加する．

2. 二つのスタック操作を並列する場合には，得られる標準 Young 盤の各列の高さは，並列する以前の二つの標準 Young 盤の各列の長さの離散和集合 (すなわち，同じ要素の重複も認めた和集合) となっている．

再び上の例でこれをみてみることにする．両者の Robinson-Schensted 対応の結果は，

$$2,1,3 \leftrightarrow \left(\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\ ,\ \begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\right)$$

$$3,2,4,1 \leftrightarrow \left(\begin{array}{|c|c|}\hline 1&4\\\hline 2\\\cline{1-1} 3\\\cline{1-1}\end{array}\ ,\ \begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1} 4\\\cline{1-1}\end{array}\right)$$

となる．

次にスタック表現可能順列 $2,3,1$ と $2,1,3$ の二つの並列して $2,3,1,5,4,6$ を作った場合について Robinson-Schensted 対応の結果を示す．

$$2,3,1 \leftrightarrow \left(\begin{array}{|c|c|}\hline 2&3\\\hline 1\\\cline{1-1}\end{array}\ ,\ \begin{array}{|c|c|}\hline 1&2\\\hline 3\\\cline{1-1}\end{array}\right)$$

$$2,1,3 \leftrightarrow \left(\begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\ ,\ \begin{array}{|c|c|}\hline 1&3\\\hline 2\\\cline{1-1}\end{array}\right)$$

$$2,3,1,5,4,6 \leftrightarrow \left(\begin{array}{|c|c|c|c|}\hline 1&3&4&6\\\hline 2&5\\\cline{1-2}\end{array}\ ,\ \begin{array}{|c|c|c|c|}\hline 1&2&4&6\\\hline 3&5\\\cline{1-2}\end{array}\right)$$

上記の例では，$2,3,1$ を Robinson-Schensted 対応したとき，その標準 Young 盤の台の第 1 列の高さは 2，第 2 列の高さは 1 であり，列の高さは集合として $\{2,1\}$ と表される．同様に $2,1,3$ についても，第 1 列は高さ 2，第 2 列は高さ 1 であり，列の高さの集合は $\{2,1\}$ である．そしてこれらを並列したスタック操作によって得られるスタック表現可能順列 $2,3,1,5,4,6$ に対しては，第 1 列，第 2 列が共に高さ 2，第 3 列，第 4 列が共に高さ 1 であり，集合としては $\{2,2,1,1\}$ である．これは丁度前二者の要素の重複を許した和集合となっている．

以下に上の二つの場合に生じる標準 Young 盤の変化についてその証明を行なう．まず第一に最外枠に弧を追加する場合について述べる．最外枠に弧が追加されることは，スタック操作に翻訳すれば，最初の操作で “1” を挿入し，以後は追加前と全く同じスタック操作を行ない，最後に残っている “1” をスタックから削除することに他ならない．すなわちスタック表現可能順列としては最後尾に 1 が付け加わり，それより前の順列は全て 1 だけ増えたものになっている．このスタック表現可能順列に対して Robinson-Schensted 対応を行なったとき，最後尾の 1 を対応させる直前にできている標準 Young 盤は全く以前と変わりなく，ただ盤に記入されている数が各々 1 だけ増えているだけである．この盤に 1 を挿入すると 1 は全ての盤上の数よりも小であるから，1 が占めるべき位置は第 1 行の第 1 列であり，第 1 行第 1 列にあった要素は第 2 行に排出される．ところがこの排出された要素は，第 1 行の第 1 列に存在していた要素であるから，当然第 2 行の第 1 列の要素より小であり，第 1 行から排出された要素が第 2 行において占めるべき位置は第 1 列であって，第 2 行第 1 列に存在していた要素は第 3 行へ排出される．以上が反復される結果，結局第 1 列のみが 1 段下へと下がり，第 1 列の高さのみが高さが 1 増加する．

次に二つの弧を並列する場合について述べる．二つの弧を並列する場合のスタック操作は，前半部分で表されるスタック操作を全て終了した後，空となったスタックに対して後半部分のスタック操作を行なうというスタック操作である．よってこの並列されたスタック操作から得られるスタック表現可能順列は上記の様に前半部分は全く等しく，後半部分が前半部分の最大数，すなわち最後にスタックに挿入された数の分だけ底上げされた順列になる．二つの弧を並列した場合に得られるスタック表現可能順列の前半部分は，並列する以前の二つの弧によって得られるス

タック表現可能順列と全く変わらない．よって並列した弧の Robinson-Schensted 対応を得る途上で前半部分の挿入を終了した状態での標準 Young 盤は全く変化していない．

この標準 Young 盤に新たに並列した弧から得られるスタック表現可能順列の後半部分を挿入していく．ところが，後半部分の数は前半部分の最大数だけ底上げされているから，後半部分の数を新たに挿入しても前半部分の数は何一つ次の行へ排出することができない．すなわち，新たに挿入された後半部分の数は，各行の前半部分の数の後尾に付加される形でしか盤上の位置を確保できないのである．よって前半部分の各列の高さは，後半部分の挿入の結果によらない．

しかし後半部分の数同士に対しては行からの排出，行への挿入が行われる．しかもこの各行からの排出および次の行への挿入の過程は，前半部分の数が行の左方を占有しているということを除いて，並列以前と全く同様である．この結果，後半部分を挿入して得られる標準 Young 盤は，前半部分と後半部分の二つの標準 Young 盤を横に並べ，左に各行を揃えたものになっている．

5.3 左右補完における一致性

以上の結果を用いて，右補完によって得られる時刻 t の状態の Dyck 言語の要素に対応する標準 Young 盤の台と，左補完によって得られる時刻 $t-1$ の状態の Dyck 言語の要素に対応する標準 Young 盤の台とが常に一致することを示す．標準 Young 盤の台が等しい Young 図形であることを示すには，各々の Young 図形の各行の長さが全て等しいことを言えばよい．

今第 1 行の長さが Dyck 言語においてどう捉えられるかを見ることにする．Dyck 言語は二つの操作 1及び 2を繰り返して作ることができる．既に空でない Dyck 言語に新たに操作 1で括弧を付け加える場合，第 1 行の長さは変わらない．また操作 2で二つの Dyck 言語を並列する場合は第 1 行の長さは二つの Dyck 言語の第 1 行の長さの和であるから，各々について長さを考えればよい．結局ある Dyck 言語のから得られる盤の第 1 行の長さは，最外郭の括弧を外す操作と並列された状態になっている Dyck 言語を切り離す操作を繰り返して最終的にできる Dyck 言語である ( ) という最も簡単なもの ($\square$ という盤に対応する) の切り離された和の個数になる．これは初めの Dyck 言語に含まれていた ( ) という部分列の個数に他ならない．

Dyck 言語の部分列 ( ) は，左補完の場合セルオートマトンの状態上の 01 という部分列に，右補完の場合には 10 という部分列に対応している．ところがこれらの数は互いに等しいことが次のようにして示せる．まずセルオートマトンの状態は十分左方及び右方では全て 0 であり，1 の列の個数は有限であった．このセルオートマトンの状態を左から右に見ていくと，0 から 1 に変化する箇所があれば，その後に連続する 1 の列の後で必ず 1 から 0 に変化しなくてはならない．もしそうならないならば，1 の列が無限に続くことになるからである．この結果 0 から 1 への変化回数は 1 から 0 への変化回数に等しい．

例えば $\cdots 0\,\overline{0}\,\overline{1}\,\underline{1}\,\underline{0}\,\overline{1}\,1\,\underline{1}\,\underline{0}\,0\cdots$ という列の場合，上線で示した 01 という列は 2 列，下線で示した 10 という列も 2 列であり，両者の個数は等しい．

これで第 1 行の長さが等しくなることは証明された．次に第 2 行の長さが等しいことを証明する．

Dyck 言語を操作 1と操作 2で作る過程を考える．このとき，対応する Young 盤で第 2 行が初めて作られるのは，操作 1で空でない Dyck 言語の最外郭に初めて括弧を加えたときである．これは ( ) という部分列を削除したときに ( ) という部分列になっている括弧を加えることに他ならない．第 2 行が作られた後の操作 1で第 2 行の長さが変わらないこと，および操作 2で第 2 行の長さが各々の和になることは第 1 行の場合と同じであり，結局第 2 行の長さとはもともとの Dyck 言語から ( ) という部分列 (第 1 行に相当する) を削除した Dyck 言語に含まれる ( ) という部分列の個数に等しい．

これをセルオートマトンの状態において考えれば，（ ）を全て削除することは，左補完の場合においては01を，右補完の場合においては10を削除することに他ならない．ところが01を削除した場合も，10を削除した場合も，結果として得られる1と0の列は必ず等しくことが次のように示せる．

01を削除することを考えとき，次の二つの場合が生じ得る．

1. 連続する1の列同士が削除の結果連結される．

2. 連続する1の列同士は削除を行なっても連結されない．

例えば $\cdots 0\,1\,1\,\overline{0}\,\overline{1}\,1\,1\,0 \cdots$ という列においては，上線によって示した01を削除することによって長さ2の1の列と長さ3の1の列同士の連結が生じ，結果として $\cdots 0\,1\,1\,1\,1\,0 \cdots$ という長さ4の1の列に変化する．この連結という現象は，連続する1の列の中間に0が1個のみしか存在しない場合にのみ生じる．

また $\cdots 0\,1\,1\,0\,\overline{0}\,\overline{1}\,1\,1\,0 \cdots$ という列の場合には，上線によって示した01を削除しても1の列の連結は生じない．すなわち，連続する1の列の中間に0が2個以上存在する場合には連結という現象は生じない．

次に10を削除する場合を考えると，次のことが言える．

1. 01を削除する場合に1の列同士が連結されるならば，10を削除しても1の列同士は連結される．

2. 01を削除しても1の列同士が連結されないならば，10を削除しても連続する1の列同士は連結されない．

何故ならば，01を削除して1の列同士が連結される場合には1の列同士の中間には1個の0しか存在していない．この0は10の削除によっても消去されるので，結果として1の列同士は連結されなければならない．

例えば $\cdots 0\,1\,\underline{1}\,\underline{0}\,1\,1\,1\,0 \cdots$ という列においては，下線によって示した10を削除することによって長さ2の1の列と長さ3の1の列同士の連結が生じ，結果として $\cdots 0\,1\,1\,1\,1\,0 \cdots$ という長さ4の1の列に変化する．

また，01を削除しても1の列同士が連結されない場合には1の列同士の中間には2個以上の0が存在する．この0は10の削除によって全て消去されることはないので，1の列同士が連結されることはない．

例えば $\cdots\ 0\ 1\ \underline{1}\ \underline{0}\ 0\ 1\ 1\ 1\ 0 \cdots$ という列において下線で示した10を消去しても，$\cdots 0\,1\,0\,1\,1\,1\,0 \cdots$ という列が生じるだけである．

以上の例は連結される1の列が2列の場合であるが，連結がさらに多くの1の列同士の間で行なわれる場合でも全く同様である．すなわち，01を削除して連結される1の列は10を削除しても連結される．

また，01の削除によって，1の列の長さは1減少する．これは10の削除の場合でも全く同様である．

長さ k の1の列と，長さ l の1の列に連結が生じる場合には，連結された1の列の長さは $k+l-2$ となる．これは01を削除する場合の連結でも，10を削除する場合の連結でも同じであり，連結がより多くの1の列の間で生じる場合も同様である．

連結が生じない場合，1の列と1の列の中間に存在する0の列の長さ（最低でも2以上ある）は，01を削除する場合でも10を削除する場合でも必ず1減少する．最左方に無限個数存在する

0の列のみは，10の削除で個数が減少することがなく，01の削除でのみ個数が減少することになるが，結局無限個数であることに変わりはない．最右端にある無限個の0の列についても同様である．

以上より結局01を削除する場合でも，10を削除する場合でも得られる1と0の列は同一である．

例えば $\cdots 0\,\overline{0}\,\overline{1}\,\underline{1}\,\underline{0}\,\overline{1}\,1\,\underline{1}\,\underline{0}\,0\cdots$ という列の場合，上線で示した01を削除しても，下線で示した10を削除しても，得られる結果は $\cdots 0\,1\,1\,1\,0\cdots$ という列であり，同一である．

このことから，左補完の場合は01，右補完の場合は10を削除した後の列においても，左補完で( )に対応する01の個数と，右補完で( )に対応する10の個数は等しい．すなわち削除前の列から作られる盤の第2行の長さは，左補完の場合も右補完の場合も等しくなる．

第3行の長さ以降も全て等しいことは，左補完の場合には01を，右補完の場合には10を削除するという過程を同様に続けていくことによって示すことができる．すなわちこれは左補完で作られるDyck言語に対しても，右補完で作られるDyck言語に対してもYoung図形の形が等しいこと，すなわち時刻$t-1$のDyck言語から作られるYoung図形も時刻tから作られるYoung図形もその形が等しいことに他ならない．以上によってYoung図形の形が時間発展についての保存量であることが証明できた．

参考文献

[1] Aho, Alfred V. - Hopcroft John E. - Ullman, Jeffrey D., The design and analysis of computer algorithms, Addison-Wesley,1974

[2] Hopcroft, John E. - Ullman, Jeffrey D., Formal languages and their relation to automata, Addison-Wesley, 1969

[3] Knuth, Donald Ervin, The art of computer programming vol.3 Addison-Wesley, 1973

[4] Lothaire, M, Combinatorics on words, Addison-Wesley, 1983

[5] Takahashi, Daisuke and Satsuma, Junkichi, A Soliton Cellular Automaton, Journal of The Physical Society Japan, **59** 10 (1990), 3514-3519

[6] Sagan, Bruce E., The symmetric group, Wadsworth & Brooks/Cole, 1991

[7] Stanton, Dennis and White, Dennis, Constructive combinatorics, Springer-Verlag, 1986

[8] Wolfram, Stephen (ed.), Theory and applications of cellular automata, World Scientific, 1986