数理解析研究所講究録
第 856 巻 1994 年 151-163

Combinatorics on Soliton Cellular Automata

Department of Mathematical Sciences, University of Tokyo, Japan
Makoto TORII (鳥居 真)*

1 Introduction

Completely integrable systems have many notable properties. Korteweg-de Vries (K-dV) system, which is described by nonlinear partial differential equation

$$u_t = uu_x + u_{xxx}$$

is one of well-known completely integrable systems. This system has the following properties which is observed among many integrable systems:

1. the existence of N-soliton solutions,
2. the existence of an infinite number of time evolution invariant,
3. exact solvability by analytical methods.

The N-soliton solution describes N wave packets which interact like particles; the orbits of their peaks shift when they collide. These phenomena are called phase shifts. In the N-soliton solution, the wave packets are separated each other after long time evolution, and their individuality are restored.

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

Preservation of individuality comes from The existence of an infinite number of time invariants, and the invariant property comes from the existence of symmetries of a system.

Cellular automata are dynamical systems which consist of time evolution rules of values on lattice. The values are discrete, often only two. This means that cellular automata are discrete system in space, time, and value. Combinatorial properties, which have not been reported in other integrable systems, arises from this discreteness.

A filter automaton is one of the cellular automata; both previous state and a part of present state determines the value on a site at the present state. Soliton phenomena are observed in many filter automata(see [5]). A filter automaton considered in this paper is proposed by Takahashi and Satsuma, which carries only soliton solutions (see [3]). The lattice is one dimensional and only two values "1" or "0" are assigned on every site of the lattice. The number of assigned "1" is finite. Time evolution rules of the automaton are quite simple, which will be mentioned in the next section.

In this paper, we show that the time invariants of the automaton are constructed practically by combinatorial way. In section 2, time evolution rules of the automaton are given. The 2-soliton behavior of the automaton is also given exhibited in this section. In section 3, some combinatorial techniques concerned with the automaton are introduced. These techniques include the Robinson-Schensted algorithm, which plays an interesting role in the field of group representation theory. In section 4, the time invariants of the automaton are constructed using combinatorics explained in the section 3.

2 Time evolution rules

The soliton cellular automaton evolves obeying the following rules:

1. select the unfixed and most left "1";
2. search the right side of the selected "1" and move it to the nearest "0", after this operation, fix the moved "1";
3. if there remains the unfixed "1", repeat the above procedure; if all "1"s are fixed, release all "1"s and repeat from the first.

The time evolution process of a state of the automaton

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

is demonstrated below. First, the most left "1" is selected, which is indicated in the next line by a frame.

$$\cdots 0\,\boxed{1}\,1\,\boxed{0}\,1\,0\,0\,0 \cdots.$$

The selected "1" is moved to the nearest "0" in the right side of the "1" which is indicated above by another frame. We get an intermediate state

$$\cdots 0\,0\,1\,\underline{1}\,1\,0\,0\,0 \cdots,$$

where the underline signifies that the number on it is fixed. Next, the most right and unfixed "1" is selected, which indicated below by a frame.

$$\cdots 0\,0\,\boxed{1}\,\underline{1}\,1\,\boxed{0}\,0\,0 \cdots.$$

The selected "1" is moved to the nearest "0" in the right side of the "1" which is indicated above by another frame. Then we get another intermediate state

$$\cdots 0\,0\,0\,\underline{1}\,1\,\underline{1}\,0\,0 \cdots.$$

Finally, the remained "1" is moved to the nearest right "0" in the same way. Now we get

$$\cdots\ 0\,0\,0\,\underline{1}\,0\,\underline{1}\,\underline{1}\,0\ \cdots,$$

where all the "1"'s are fixed. This is the next state obtained as the result of time evolution. Thereafter we release all the "1"'s and repeat the same procedure above to get the further state.

The 2-soliton behavior of the automaton is exhibited in the following example. The high speed soliton "11" collides with the low speed soliton "1". We observe a phase shift between the third row and fourth row.

$$\begin{gathered}
\cdots 1\,1\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \cdots \\
\cdots 0\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \cdots \\
\cdots 0\,0\,0\,0\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0 \cdots \\
\cdots 0\,0\,0\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,0 \cdots \\
\cdots 0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0 \cdots \\
\cdots 0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1 \cdots
\end{gathered}$$

3 Combinatorial techniques

In this section some combinatorial materials which are useful to treat the automaton are introduced. They are available to construct the time invariants of the automaton.

3.1 Dyck language

The first material is some sequence of two letters called as the Dyck language (see [1]).

The letters of sequences are written as “(” and “)”. The following conditions define the Dyck language;

- each sequence has the same number of “(” and “)”,
- the letter “(“ appears more or the same times than the letter “)” in the left side of any letters in each sequence.

For example, the next 5 sequences are whole elements of Dyck language of 6 letters.

((( ))), (( )) ( ), (( ) ( )), ( ) (( )), ( ) ( ) ( )

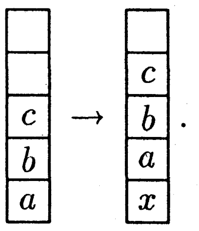
3.2 Stack representative permutations

Second material is a property which a part of permutations have, called stack representativity.

The sequence $(x_1, x_2, \ldots, x_n)$ denotes a permutation of $(1, 2, \ldots, n)$. A subsequence of length k in this permutation is given by $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ with $i_1 < i_2 < \cdots < i_k$. The stack representativity is defined by subsequences in a permutation. If the case $x_i > x_k > x_j$ never occurs for any subsequence of length 3, (x_i, x_j, x_k), then the permutation is called stack representative.

3.3 Non-crossing arcs on the half plane

The naming of stack representativity comes from the fact that any stack representative permutation can be generated by two operations to a stack:pushing and popping. A stack can be displayed as piled up boxes. The operation pushing x to this stack is illustrated as follows:



Popping x from the stack is the inverse operation, that is,

$$\begin{array}{|c|}\hline \\ \hline c \\ \hline b \\ \hline a \\ \hline x \\ \hline \end{array} \longrightarrow \begin{array}{|c|}\hline \\ \hline \\ \hline c \\ \hline b \\ \hline a \\ \hline \end{array}.$$

In a pushing operation, x is called an input to the stack and in a popping ,an output from the stack. The hight of a number in a stack is defined as the number of boxes below the number. It is regarded that the number in the lowest box has the hight 0.

A stack representative permutation can be obtained as numbers which are assigned to non-crossing arcs on the half-plane $\{(t,h)|t,h \in \mathbb{R}, h \geq 0\}$. Considering t as the time value of operations to a stack and h as the hight of the numbers in the stack, the orbits of numbers draw non-crossing arcs on the half-plane. The both ends of each arc lie on the edge of the half-plane. We assign natural numbers $1, 2, \cdots, n$ to the left ends of these arcs from left to right, where n is the number of the arcs. Thereafter the same number given to the left end of an arc is assigned to the right end. A stack representative permutation is obtained by picking up the numbers of right ends from left to right.

For example, a stack representative permutation $(2,1,3)$ is obtained by the following stack operations shown in figure 1. As seen from this figure,

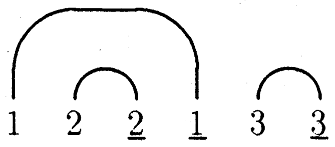


Figure 1:

the left end of each arc means that the number assigned on the arc is pushed to the stack and the right one, the underlined end, means that the number is popped. That is, an arc is an orbit of a number which assigned on the arc. By the method mentioned above, one can obtain a stack representative permutation from operations to a stack.

3.4 Standard Young tableaux

Third material is the Robinson-Schensted algorithm, which extracts a property of permutations. Let $(\lambda_1, \lambda_2, \cdots, \lambda_k)$ be a partition of a natural number n that is a sequence of non-increasing natural number and their total sum is equal to n.

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0, \quad \sum_{i=1}^{k} \lambda_i = n.$$

A Young diagram is square arrayed boxes which has λ_i boxes in the i-th row. The number of boxes in the Young diagram is denoted by n.

A Young diagram which has numbers in its boxes is called Young tableau. Let us put numbers $1, 2, \ldots, n$ in each box in a Young diagram by obeying the following regulations;

- each number appears just once,
- the numbers in each row are strictly increasing from left to right,
- the numbers in each column are strictly increasing from top to bottom.

A Young tableau satisfying the regulations is called the standard Young tableau.

3.5 The Robinson-Schensted algorithm

The Robinson-Schensted algorithm gives an correspondence between a permutation and a pair of standard Young tableaux (P, Q)(see [2],[4]). The left one of this pair is named the P-symbol and the right one the Q-symbol.

This algorithm is complicated a little. However, the algorithm is made by repeating simple procedure, row insertion. The row is a sequence of boxes, and each box has numbers arranged in a strictly increasing order.

Suppose that we insert a number x to a row. The row insertion rules are given as follows;

1. if the row is empty, make a new box and put the number x in it,
2. otherwise pick up the number in the boxes from left to right and compare x with each of them successively, then
   - if x is greater than the number in the box, go on comparison x with the number in the next right box,
   - if x is less than the number in the box, exchange the number in the box for x and eliminate the number in the box from the row,
3. if there is no greater number than x in the row, add a new box to the right end of the row and put x in it.

For example, the insertion of 4 to $\boxed{2}\boxed{3}\boxed{5}\boxed{8}$ yields the row $\boxed{2}\boxed{3}\boxed{4}\boxed{8}$ and eliminates the number 5. In a case of the insertion of 9, $\boxed{2}\boxed{3}\boxed{5}\boxed{8}\boxed{9}$ is obtained and no number is eliminated.

We can apply these row insertion rules, to a standard Young tableau as follows;

1. split the standard Young tableau into the rows,
2. insert x to the first row,
3. if a number is eliminated from the first row, insert the eliminated number to the second row, until no number is eliminated and a new box is created, insert the eliminated number to the next row.

For example, by insertion of 4 to the standard Young tableau $\begin{array}{|c|c|c|}\hline 1&3&5\\\hline 6&7\\\cline{1-2} 8\\\cline{1-1}\end{array}$,

one obtains the new standard Young tableau $\begin{array}{|c|c|c|}\hline 1&3&4\\\hline 5&7\\\cline{1-2} 6\\\cline{1-1} 8\\\cline{1-1}\end{array}$.

The Robinson-Schensted algorithm is defined by applying this insertion procedure to a standard Young tableau successively. A permutation $(x_1, x_2, \ldots, x_n)$ is given the correspondence a pair of standard Young tableaux by the following procedure;

1. in an initial state, let P and Q be empty tableaux,

2. insert the elements of the permutation to the P-symbol from x_1 to x_n,

3. when the insertion of x_i is done, a new box is created in P, at the same position of the new box in P, create a new box in Q and put i in it.

A demonstration would explain the algorithm well. Let us consider the permutation $(5,6,3,1,2,4)$ for example. In initial state, $(P,Q) = (\emptyset, \emptyset)$. After the first insertion, insertion of 5, we obtain

$$(P,Q) = \left(\boxed{5}\,,\boxed{1}\right).$$

Next, we insert the 6 to this tableaux. Since 6 is greater than 5, the greatest number in the first row, 6 makes a new box at the right end of the first row. Thus we obtain

$$\left(\begin{array}{|c|c|}\hline 5&6\\\hline\end{array}\,,\begin{array}{|c|c|}\hline 1&2\\\hline\end{array}\right).$$

We then insert 3 to this tableaux. 3 is smaller than 5. So 3 eliminates 5 from the first row and the eliminated 5 makes a new box in the second row. The tableaux

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

is obtained as the result. Next, the turn of 1. Since 1 is smaller than 3, which is the first element of the first row, it eliminates 3 and the eliminated 3 goes to the second row. But 3 is smaller than 5, which is the elements of the second row. Hence 3 eliminates 5 from the second row. The eliminated 5 makes a new box in the third row. Thus we obtain

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

By following the similar procedure, we can complete the Robinson-Schensted algorithm to obtain

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

for the permutation $(5, 6, 3, 1, 2, 4)$.

4 Construction of time invariants

Using combinatorial techniques explained above, we can construct a time invariants of the automaton. The construction process consists of three correspondences: from a state of the automaton to an element of Dyck language, from an element of Dyck language to a stack representative permutation, and from a stack representative permutation to a pair of standard Young tableaux. A time invariant is obtained as the shape of the Young tableau. Regarding empty columns have zero boxes, a shape of Young tableau gives an infinite number of time invariants.Namely, the number of boxes in the i-th column.

The way of construction is explained by demonstrating the following example of the state of the automaton,

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

4.1 Correspondence to an element of Dyck language

We obtain an element of Dyck language by rewriting "0" and "1" in a state of the automaton as follows;

1. rewrite all the "1"'s in the state to "("'s,
2. rewrite "0"'s from left to right by the following manner,
   - if the number of "("'s which is in the left side of the present "0" is greater than that of ")"'s, rewrite the "0" to ")",
   - otherwise, erase "0".

By applying these rules to the example, we obtain the sequence $(\,(\,)\,(\,)\,)\,(\,)$ as shown below;

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

4.2 Correspondence to stack representative permutations

Considering "(" and ")" as pushing to and popping from a stack, a stack representative permutation is obtained. In the case of this example, $(\,(\,)\,(\,)\,)\,(\,)$ is translated to the following operations to a stack(see section 3.3).

1. push "1" to the stack.
2. push "2" to the stack.
3. pop "2" from the stack.
4. push "3" to the stack.

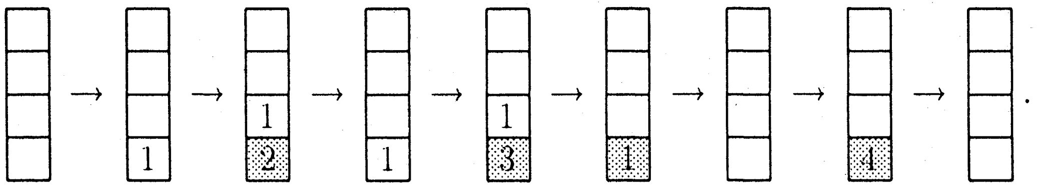
5. pop "3" from the stack.

6. pop "1" from the stack.

7. push "4" to the stack.

8. pop "4" from the stack.

These operations are displayed pictorially as follows:



Where the hatched numbers are the outputs from the stack. As a result of the operations, we obtain a stack representative permutation $(2,3,1,4)$.

4.3 Correspondence to Young tableaux

A stack representative permutation is given a correspondence to a pair of Young tableaux by the Robinson-Schensted algorithm(see section 3.5). The stack representative permutation which we obtained above corresponds to a pair of Young tableaux

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

The shape of these Young tableaux are invariants of time evolution. Indeed the state of the automaton which comes after the state of this example is

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

which gives the Young tableaux

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

We see that this pair of Young tableaux have the same shape as that of the initial state.

5 Conclusions and prospect

We have shown that the invariants of the soliton cellular automaton are given as shapes of the Young tableau through three combinatial correspondences: Dyck language, stack representative permutation (stack operations) and the Robinson-Schensted algorithm. These conbinatorics are related to some algebras, representation theory, and symmetric group and Bruhat ordering on it(for example, see [4]). It would be an interesting problem to investigate the relation between our results and such mathematical concepts.

References

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

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

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

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

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