Vol. 37 No. 2 Transactions of Information Processing Society of Japan Feb. 1996

Regular Paper

An Interactive System Implementation for Constructing Cat's Cradle Diagrams and its Evaluation

Rahmat Budiarto,[†] Masashi Yamada,[†] Hirohisa Seki[†] and Hidenori Itoh[†]

This paper discusses a string representation and processing method in two-dimensions (2-D) which preserves the topological characteristics of the cat's cradle string diagrams and uses a small data. The method applies three basic operations (namely *pull*, *release* and *exchange*), geometric transformations and a genetic algorithm (GA) for constructing cat's cradle patterns. Based on the method, a CAD system for constructing cat's cradle patterns is implemented and evaluated. The CAD system allows the user to interactively construct cat's cradle patterns. Furthermore, this paper proposes a cat's cradle system *Ayasys* which comprises the CAD system and a CAI system for instructing cat's cradle pattern constructions.

1. Introduction

Recently, there have been some reports about string state representation and its processing based on computer programs. One is a cell-pattern representation which represents the string state by cells in three-dimensions (3-D). It perfectly preserves the detail shape and the topological characteristics of the string, but requires a large data and a complex calculation in its processing[1),2)]. The other one is a crossing set representation which represents the string state by a set of crossings of 2-D. It does not preserve the detail shape of the string. However, it preserves the topological characteristics of the string and requires only a small data and a simple calculation in its processing[3),4)]. Besides, the string state can be represented as an invariant polynomial discussed in knot theory 5) and 6). This representation is useful for calculating an invariant of the topological characteristics of the string state, however, this is a poor method for representing string shapes.

Figure 1 a shows three closed strings constrained by some poles in 3-D. Each string state can be illustrated by a planar diagram, called as *string diagram* shown in Fig. 1 b. A and B can be transformed into a simple closed curve, A' and B' in Fig. 1 c[☆]. However, C cannot be transformed into a simple closed curve, owing to the constraint of the dot which represents a pole. These transformations which consider the dot constraints are useful for simulating cat's cradle transforming processes. If a cell-pattern representation is employed, a complex calculation is needed for simulating these transformations. On the other hand, employing the invariant polynomial cannot represent these transformations since the invariant polynomial represents the topological invariant of the string state, therefore the invariant polynomials of string A and A' are of the same and so are B and B'. This paper distinguishes the string diagrams A from A' and B from B' respectively, since this paper attempts to employ the method which represents a string state of the cat's cradle and simulates its transforming

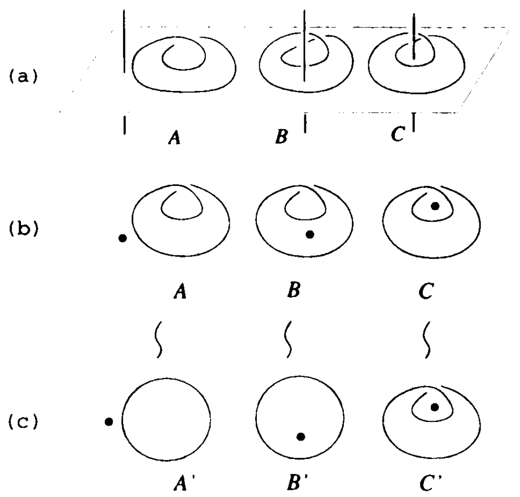


Fig. 1 (a) Closed string constrained by poles, (b) constrained string diagrams and (c) their equivalent diagrams.

† Department of AI and Computer Science, Nagoya Institute of Technology

☆ $D_1 \sim D_2$ denotes that the diagram D_2 is equivalent to the diagram D_1 (there is a continuous deformation through embedding from D_1 to D_2 or D_2 to D_1, see 6)).

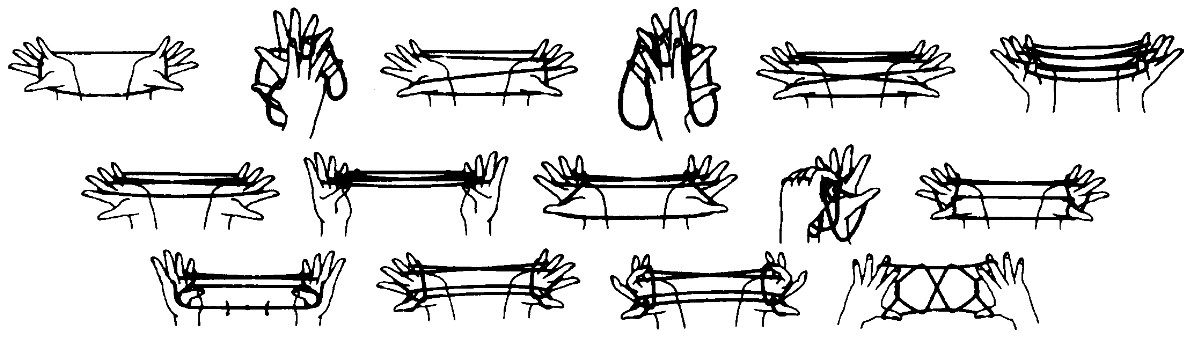


Fig. 2 *Nidan-bashigo* pattern construction by finger operations.

processes. In order to display the string states of the cat's cradle the string diagram is used[3)] and a finger is simply denoted by a dot. The intersections between strings are indicated as *crossings*. Patterns (shapes) in the cat's cradle are obtained by minimizing both the number of crossings and the length of the string in the string diagram. To minimize the number of crossings, a local transformation is employed. To minimize the length of the string, some geometric movements and a genetic algorithm (GA) are employed. The string length minimization is done by placing the crossings at their 'best' positions. If there is one crossing in the diagram, the search space for its position is R^2. Thus, if there are n crossings in the diagram the search space of their positions is R^{2n}. Therefore, the possible combination of the crossing positions increases exponentially when the number of crossings in the string diagram increases. Using the sequential process based on a geometric transformation[7)] often faces a local minimum in case of search in a complex diagram which has a large number of crossings. This paper uses GA, since GA is suitable for solving the 'hard' combinatorial optimization problem whose search space is large and can get out of the local minimum because of its probabilistic searching[8)]. By executing some patterns of cat's cradle, the suitability of the transformation method is verified.

Furthermore, in applications on the cat's cradle game, there have been no report about user interactivity in constructing and designing cat's cradle patterns. For example, in 4), only the final pattern of the cat's cradle is generated from an overlapping diagram. In this paper the construction process is divided into several steps where in each step basic operations can be performed, and the result of each step can then be observed. Therefore, a CAD system for constructing cat's cradle patterns interactively can be implemented. By making use of this CAD system, a cat's cradle system which has two functions namely the CAD system and a CAI system is generated.

This paper is organized as follows. The next two subsequence sections review a cat's cradle string state representation, basic operations and a string diagram processing. Section 4 describes a CAD system and its evaluation. Section 5 discusses about the proposed cat's cradle system. Finally, Section 6 provides some conclusions.

2. A Cat's Cradle String State Representation and Basic Operations

In a cat's cradle game, some beautiful patterns can be constructed from a simple closed string by several consecutive finger operations (see **Fig. 2**). Representation methods of cat's cradle string state and basic operations have been already described in 7). This section explains again these things briefly.

2.1 A Cat's Cradle String State Representation in 2-D Plane

According to the method discussed in 7), any string diagram which displays a string state of the cat's cradle is divided into two parts: the *handle* part which has some dots (points) and the *net* part which has some crossings. An *edge* is an intersection point of a string and the boundary curve of the net and the handle (see **Fig. 3** a).

A cat's cradle string state is represented by a set of crossings of its string diagram. A crossing c_i is expressed as a square with four corners $C_i{}^1$, $C_i{}^2$, $C_i{}^3$, $C_i{}^4$ (see Fig. 3 b). The diagonal lines in the square describe how the string segments in the diagram intersect each other. Each corner of a crossing is connected to another corner or an edge by a string segment (line).

2.2 Basic Operations of Cat's Cradle

In the real world, cat's cradle patterns can be

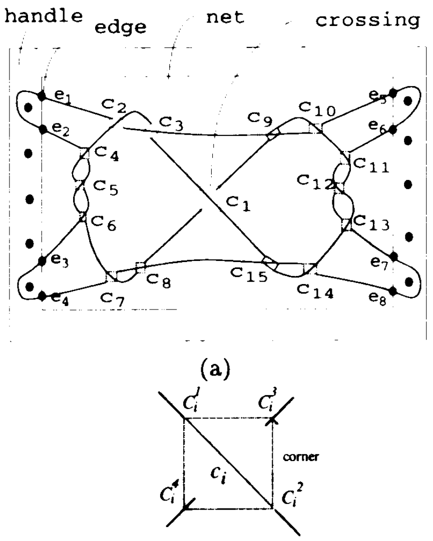


Fig. 3 A cat's cradle string diagram.

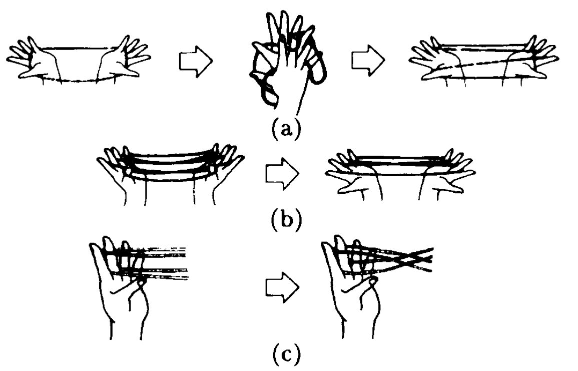


Fig. 4 Examples of cat's cradle operations.

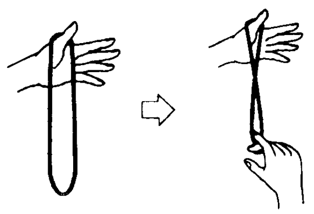


Fig. 5 A twisting operation.

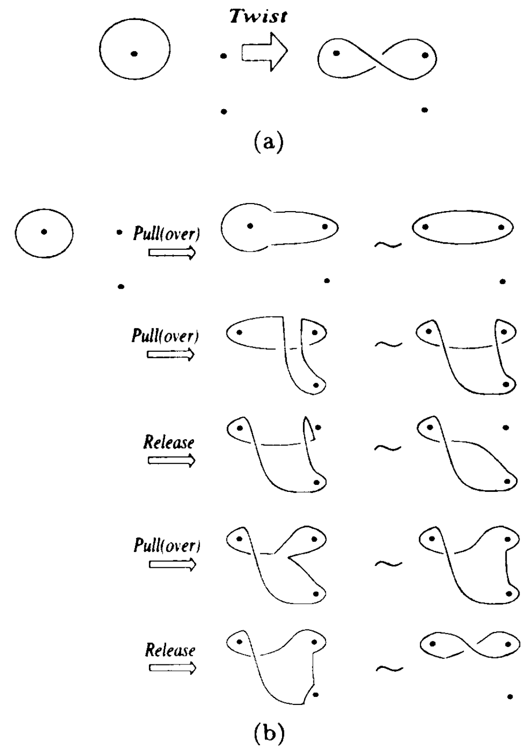


Fig. 6 A twisting operation and an equivalent combination of basic operations.

constructed by several operations such as pull a part (segment) of string and hang it on one finger, or release a part of string from one finger and so on. **Figure 4**a demonstrates the marked string part in the left hand being pulled and then hang on the middle finger of the right hand. Figure 4b shows the hung string on the thumbs is released, and Fig. 4c shows two string parts are exchanged. In 7), three basic operations were defined: *pull*, *release*, and *exchange*. The detail definitions are shown in appendix.

Almost all of operations in the cat's cradle are equivalent to a combination of these three basic operations. **Figure 5** shows a finger operation twisting the string. This operation is illustrated by 2-D diagram such as **Fig. 6**a. Moreover, this operation is equivalent to a combination of basic operations such as Fig. 6b.

In each construction step, a string diagram produced by some basic operations is transformed into its equivalent string diagram to obtain a pattern of the cat's cradle. The transformation is described in the next section. The transformed string diagram is referred to as the *best pattern*.

3. String Diagram Processing

In 7), the best pattern of any string diagram satisfies the following conditions.

(1) The number of crossings is minimal.

(2) The total string length is minimal.

For (1) applying the Reidemeister moves will minimize the number of crossings in the string diagram 7). For (2), this paper uses geometric transformations and genetic operations of GA.

3.1 Overview of GA

GA is a search algorithm based on the mechanics of natural selection processes and genetics. A *population* consisting of *units* is considered. The unit has a *chromosome* which consists of a *genes*. GA repeats a reproduction of the population by using three genetic operations: *selection*, *crossover* and *mutation*. The selection is made in a way such that units having higher *fitness degree* are more likely to be selected. The fitness degree is determined by a *fitness function* which judges the quality of the units. The *crossover* and *mutation* generate new units from selected units by operating their chromosomes.

3.2 GA for Obtaining the Best Pattern

In the implementation of the GA, a string diagram is considered as a unit in population. The unit has a chromosome expressed as a list of crossing positions p_1, p_2, …, p_{N_c} where p_i denotes a position of a crossing c_i in the net area. This paper defines the net area as $\{(x,y) \mid 0 \leq x \leq 480, 0 \leq y \leq 390\}$. Thus $p_i \in \{(x,y) \mid 0 \leq x \leq 480, 0 \leq y \leq 390\}$. A crossing position corresponds to a gene. To define the fitness function, first the total string length is defined as follows.

Let ${d_i}^n$ denotes the distance between a corner ${C_i}^n$ and a corner connected with ${C_i}^n$ or between ${C_i}^n$ and an edge connected with ${C_i}^n$. The sum of the length of the string segments connected to c_i, denoted by d_{c_i}, is defined in Eq. (1) (see 7)).

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

Let d_{e_k} is a distance between an edge e_k and a corner connected with e_k or a distance between e_k and another edge connected with e_k. The total string length L of a string diagram D is defined in Eq. (2) (see 7)),

$$L = \frac{1}{2}\left\{\sum_{i=1}^{N_c} d_{c_i} + \sum_{k=1}^{N_e} d_{e_k}\right\} \tag{2}$$

where N_c, N_e denote the number of crossings and the number of edges in the string diagram D, respectively. Thus L is not a constant but a variable. The fitness function then is defined in Eq. (3),

$$f(L) = a/(L-b)^2 \tag{3}$$

where a, b are constant numbers.

GA for obtaining the best pattern, first, generates an initial population which contains *Psize* units (*Psize* is population size). The initial population is generated by duplicating *Psize* times the string diagram obtained from the basic operations. Therefore, each unit in the initial population has the same crossing positions (genes). The GA then repeats the following steps 1 ~ 6 *Gmax* times (*Gmax* is the maximum generation number). Lastly, the GA outputs the fittest unit of the *Gmax*-th generation as the best pattern.

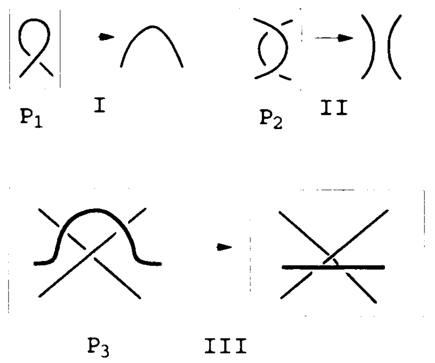


Fig. 7 Reidemeister moves.

Step 1: *Reidemeister moves*: Perform Reidemeister moves (see **Fig. 7**) for each unit in the population as follows.

```
while pattern $P_1$ or $P_2$ exists in the unit
    perform Reidemeister move I or II
    if pattern $P_3$ exists in the unit
        perform Reidemeister move III
endwhile
```

As can be seen from Fig. 7 the Reidemeister moves may decrease the number of crossings in each unit. Then, the units which have the minimum number of crossings are obtained.

Step 2: *geometric moves*: Move crossings of each unit N_c times in turn. Their order is decided randomly. The varying order among the units produces the diversity of the population.
The crossing is moved as shown in **Fig. 8**. Let a crossing c_i is connected to four points b_1, b_2, b_3, b_4. If b_1 b_2 b_3 b_4 is a convex quadrangle then move c_i to the intersection point of its diagonals. And if b_1 b_2 b_3 b_4 is a concave quadrangle then move c_i to the concave vertex of the quadrangle.

Step 3: *selection*: Choose two parent units, *parent*^{1} and *parent*^{2} from the population. The unit which has the higher fitness degree, has the higher probability for being chosen.

Step 4: *crossover*: Produce two new chromosomes from two parent's chromosomes as follows. Choose h crossings c_s, c_{s+1}, …, c_{s+h-1}, from the crossings c_1, c_2, …, c_{N_c} $(1 \leq s \leq N_{c-h+1}, h \leq N_c)$

of the two parent units. Exchange the crossing positions of the two parent units $p^1{}_s$, $p^1{}_{s+1}$, …, $p^1{}_{s+h-1}$ and $p^2{}_s$, $p^2{}_{s+1}$, …, $p^2{}_{s+h-1}$, where $p^k{}_i$ denotes a position of crossing c_i of parent unit *parent*^{k} (k=1 or 2). Then the crossover produces two child units *child*^{1} and *child*^{2} which have the produced chromosomes.

Figure 9 illustrates an example of the crossover operation. Parent units *parent*^{1} and *parent*^{2} have three crossings c_1, c_2 and c_3. In this example, two child units are produced by exchanging crossing position $p^1{}_2$ and $p^2{}_2$ of two parents.

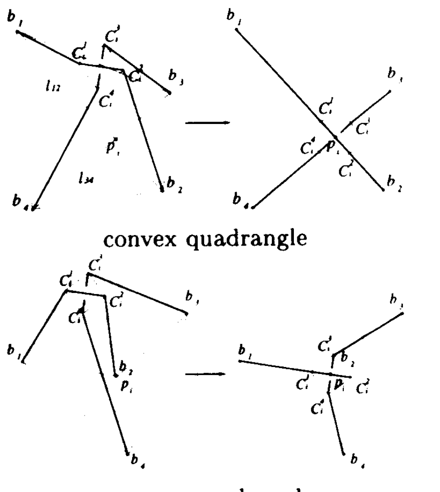


Fig. 8 Crossing movements.

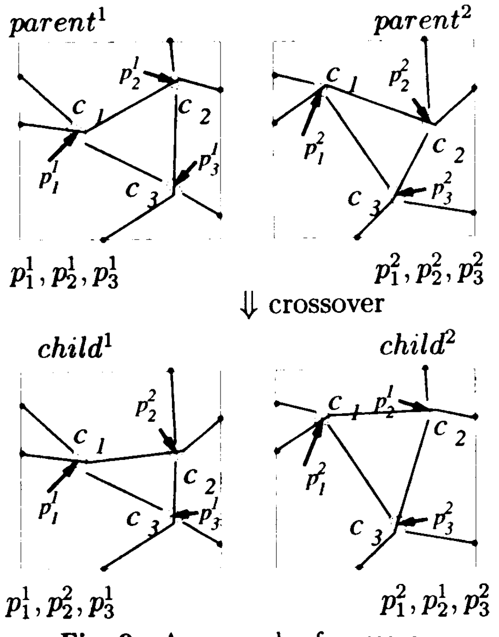


Fig. 9 An example of crossover.

Step 5: *mutation*: Changes a crossing position of the unit into a random position within an area with radius r in the net area, where $r = 10$ unit length. Mutation occurs to each produced child unit with a certain probability.

Step 6: Repeat step 3 ~ 5 until the sum of produced child units comes to the population size *Psize*. The child units produced by the above steps compose new population of the next generation.

Crossover and *mutation* do not produce any fatal (lethal) chromosomes, since a produced chromosome is a combination of crossing positions in the net area. Then, a unit consisting of the produced chromosome corresponds to a string diagram.

4. A CAD System for Constructing the Cat's Cradle Patterns

A CAD system is implemented by using the string diagram processing method described in Section 3. Interactively, cat's cradle operations (*pull, release and exchange*) are operated through the user interface module.

4.1 A Process Scheme of the CAD System

A process scheme of the CAD system is shown in **Fig. 10**. The process has two subprocesses. The first subprocess performs basic operations and the second subprocess performs the GA. Starting from an *initial pattern* a cat's cradle pattern can be constructed step by step. The initial pattern represents a simple closed string with ten fingers (see **Fig. 11** a). In the first step, the initial pattern is processed by the first subprocess. The output from this subprocess,

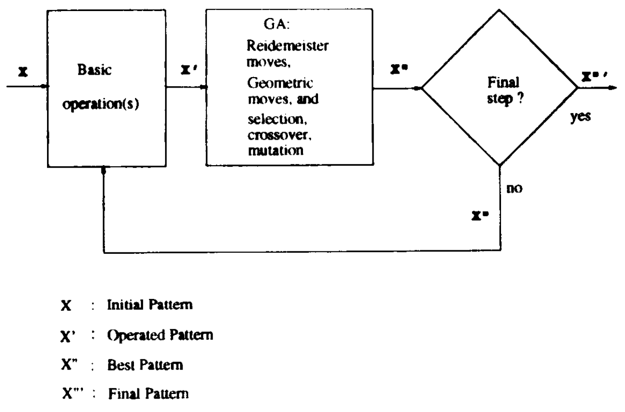


Fig. 10 A process scheme for a cat's cradle construction.

Table 1 Construction steps of *Nidan-bashigo* pattern.

Step	Basic Operations
1	*pull(over)* the marked string segment to R_3
2	*pull(over)* the marked string segment to L_3
3	*release* the string segments from L_5 and R_5
4	*pull(over)* the marked string segments to L_5 & R_5
5	*pull(over)* the marked string segments to L_5 & R_5
6	*exchange* the string segments in L_5 and R_5
7	*release* the string segments from L_1 and R_1
8	*pull(under)* the marked string segments to L_1 & R_1
9	*release* the string segments from L_3 and R_3

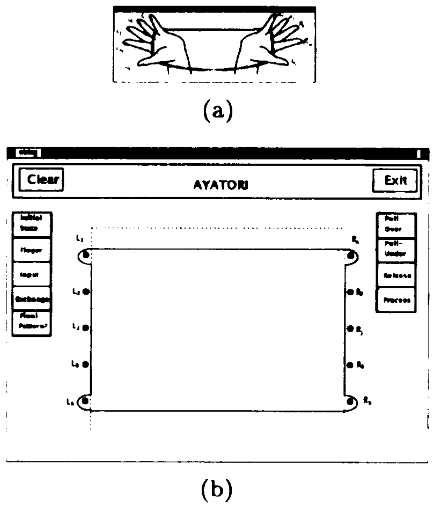


Fig. 11 An *initial pattern* and the screen layout of the CAD system.

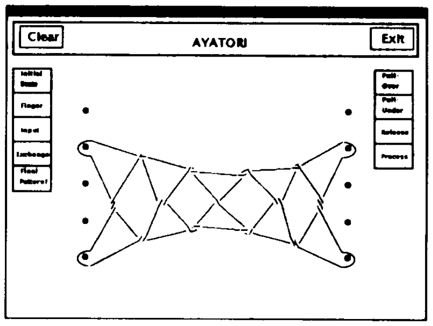


Fig. 12 Final pattern of *Yondan-bashigo*.

named *operated pattern* is fed into the second subprocess. Then, its best pattern is generated by using the GA. The best pattern in the step is displayed. If the step is not the final construction step then this best pattern is fed into the first subprocess of the next construction step. If the step is the final construction step then the best pattern of the final construction step is the final pattern of the cat's cradle being constructed.

4.2 A User Interface for the CAD System

The screen layout of the CAD system is designed as shown in Fig. 11 b. Some buttons are attached on the screen, and the mouse is used for operating them. The dots on the left and the right side represent fingers. L_i represents a left finger and R_i represents a right finger. The area inside the dash-lined rectangle represents the net and the rectangle itself represents the edge border. The edge position is the intersection coordinate between any string and the edge border. The first step for constructing a cat's cradle pattern is to invoke the initial pattern. To do a *pull* operation, a finger which the string will be hung is selected. Then a part of string which will be pulled is clicked, and the 'pull' button is clicked. To do a *release* or *exchange* operation, a finger from which a part of string will be released or exchanged is selected. Then the 'release' or 'exchange' button is clicked. The operated pattern is displayed. An appropriate best pattern in a step can be constructed and displayed by clicking the 'process' button. Finally, after all construction steps are completely done, a final pattern of cat's cradle is displayed.

4.3 Executed Examples

The CAD system is implemented on the computer. The X-Window programming is used for user interface. Some cat's cradle patterns are constructed. **Figure 12** shows the final pattern of a cat's cradle called as *Yondan-bashigo*. **Figure 13** exhibits a construction of *Nidan-bashigo* (bridge) pattern. The figures on the left are the previous patterns. The figures in the middle are operated patterns resulted by operating basic operations and the figures on the right are the best patterns obtained from the middle by using GA. Each row shows one step in the construction progress. The construction steps of the *Nidan-bashigo* pattern are shown in **Table 1**.

4.4 Evaluation of the CAD System

To evaluate the CAD system's performance the response time for some cat's cradle con-

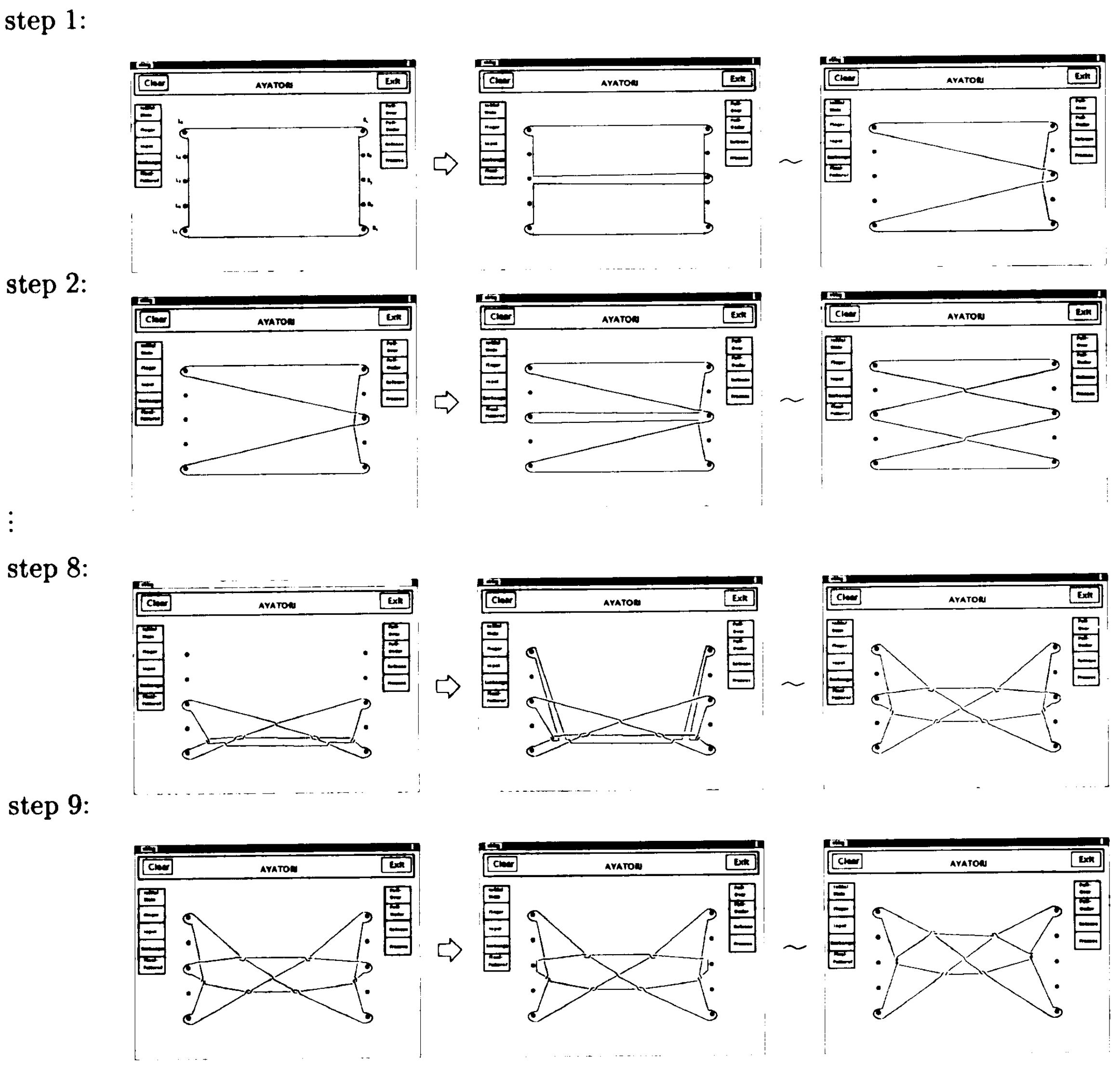


Fig. 13 *Nidan-bashigo* pattern construction.

struction steps and the response time for some number of crossings are measured. The system is running on SUN SPARCstation 5. Parameter values of GA are given as follows: the population size (*Psize*) is 120 units, maximum generation (*Gmax*) is 10.

Figure 14a shows the response times for constructions of 6 cat's cradle patterns in each step. Figure 14b shows executing times for some number of crossings in string diagrams.

In practice, generally the cat's cradle string diagrams contain about 35 crossings at the most. And from the experimental results the response time for about 35 crossings in the diagram is around 3.5 second approximately. Therefore, the CAD system can be implemented for a real time application. Furthermore, these experimental results show some constructed cat's cradle patterns are suited to their real patterns. **Table 2** shows the comparison between the sequential process based on the geometric transformation method[7] and the transformation based on the GA method. It can be seen from the table that for a large number of crossings, the method based on GA provides the shorter total string length. Even

Table 2 Obtained total string length without GA and using GA.

number of crossings	without GA (unit length)	using GA (unit length)
2	2891	2891
6	2989	2989
10	3154	3151
13	2368	2365
15	1979	1952
20	3322	3176
25	2531	2501
29	2897	2612
33	2810	2796

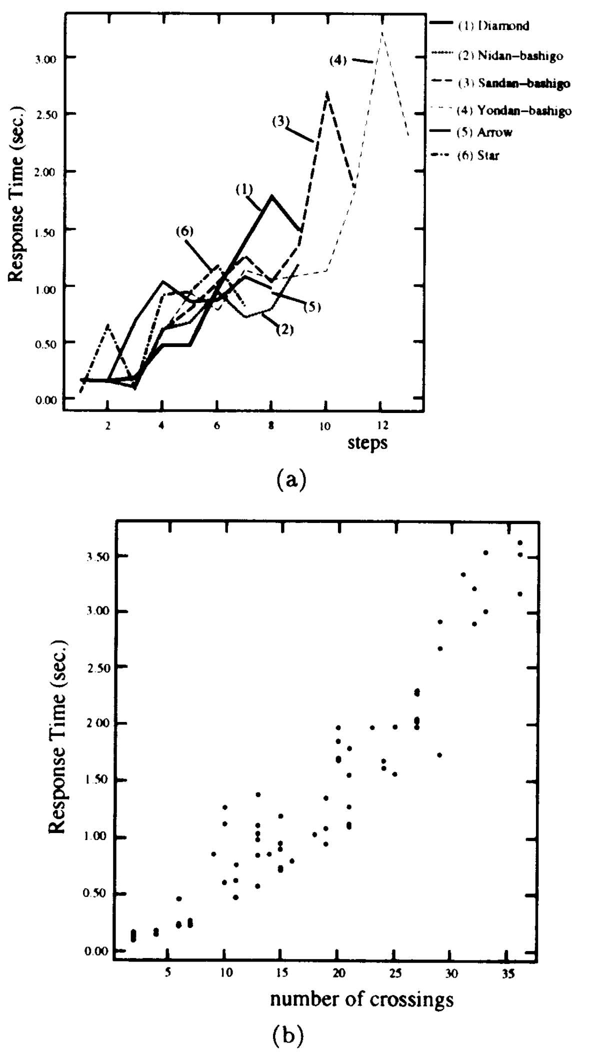


Fig. 14 Response time of the CAD system.

though the difference in the total string length is little bit, the method using GA provides more suitable shapes.

5. A Cat's Cradle System —*Ayasys*—

Ayasys aims to provide convenience to the user so that he can examine topological characteristics of string diagram easily. By making use of the CAD system a data base of cat's cradle is generated. The data base is useful for implementing CAI cat's cradle system. This idea comes from the cat's cradle book such as 9), 10). The data base is an implementation of the figures in the cat's cradle book. Seventy cat's cradle shapes selected from 9) and 10) have been constructed by making use of this system, and they have been stored in the data base. Then, the system can be used for instructing the construction of those cat's cradle shapes interactively. Moreover, the user can examine the topological characteristics of those

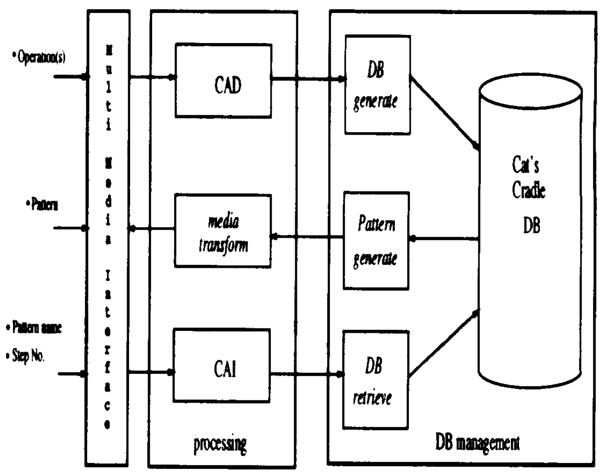


Fig. 15 The scheme of *Ayasys*.

string diagrams.

Based on these ideas the *Ayasys* is designed as follows. The system has three parts, an interface part, a processing part and a data base management part. The interface part provides the user an interactive media. Here, a mouse and a window in the screen are used. The processing part comprises CAD, CAI and *media transform*. The *media transform* transforms the code data of the cat's cradle pattern into pixels and displays them in the window. The data base management part generates and retrieves cat's cradle data. The cat's cradle data consists of terms for cat's cradle name, step number, operation(s) and pattern. The data base management part comprises three components, the *DB generate* for generating cat's cradle data base, the *DB retrieve* for retrieving cat's cradle data from cat's cradle data base and the *pattern generate* which generates a pattern from the cat's cradle data. **Figure 15** shows the system scheme.

The *Ayasys* has two main functions. One is a CAD system for designing and constructing cat's cradle diagrams. The other is a CAI system for instructing the cat's cradle game to the user.

5.1 The CAD System

The CAD system described in the previous section is employed. The CAD system gets input data (operation(s)) from the multimedia interface. A generated pattern is fed into *DB generate* function, then the cat's cradle data is stored in the data base. Meanwhile, the system displays the pattern by using the *pattern generate* function and the *media transform* function (see Fig. 15). Displaying pattern in each step aids the user for designing cat's cradle patterns.

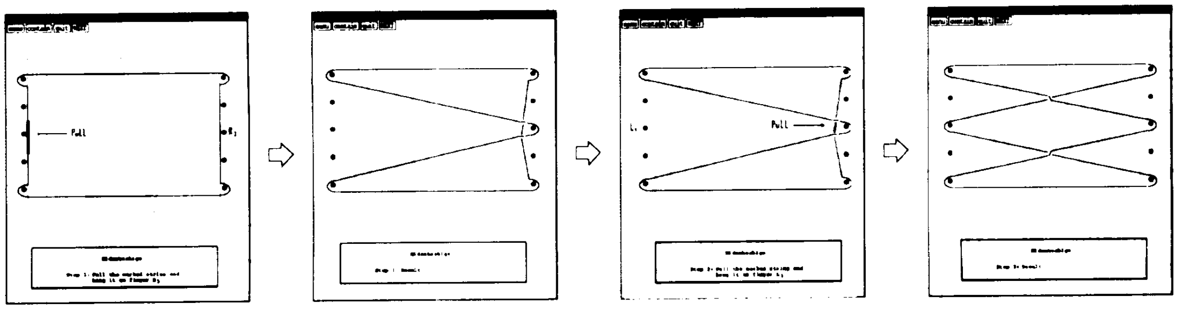


Fig. 16 The first two steps of *Nidan-bashigo* construction in CAI system.

5.2 The CAI System

By making use of the data base built by the CAD system, the CAI system for instructing cat's cradle construction is implemented. The CAI system gets cat's cradle name from the user and retrieves associated data from the data base. Also by making use of the *pattern generate* and the *media transform*, a pattern of each step along with some messages are displayed (see Fig. 15).

The screen layout of the CAI system is designed as two windows to provide informations in a convenient manner (see **Fig. 16**). The upper window shows a pattern and basic operations, while the lower window (smaller one) shows a message and explanation of the current step. Some pattern constructions are available in the 'contain' button. When the 'next' button is clicked, a transformed pattern in that step is displayed. When the 'next' button is clicked again the next step of the cat's cradle construction is invoked, and a pattern and its explanation are displayed. This is repeated until all construction steps are completely done. Figure 16 shows a typical startup screen, showing the first two steps of *Nidan-bashigo* pattern construction.

6. Conclusion

Based on the method described in 7), the interactive method for constructing cat's cradle patterns in 2-D is proposed. The construction process is divided into several steps and in each step basic operations are performed. Some experimental results verified that the constructed cat's cradle patterns are more suitable than the results from method in 7). This paper evaluated the response time for each construction step and for the number of crossings in cat's cradle string diagrams and verified that a real time interaction can be realized.

Moreover, a cat's cradle system *Ayasys* has been implemented. The system has two main functions, the CAD system for designing and constructing cat's cradle patterns and the CAI system for instructing the user how to play the cat's cradle game on the computer. Besides, the topological characteristics of the string diagram in R^2 can be examined.

Actually, in the cat's cradle game the string length is fixed. Having done some operations, a cat's cradle pattern is then constructed by tightening the string or arranging the distance between two hand positions. This causes to use a method which based on the assumption where the total string length is minimal. However, there are some patterns of cat's cradle constructed without tightening the string enough. In this case the string length in the string diagram is not minimal and the method discussed in this paper is not suitable. Then, a future work is to extend the method so that the method can solve the above problem.

Acknowledgments We would like to express our thanks to The Hayao Nakayama Foundation For Scince & Technology And Culture for supporting this research.

References

1) Toriwaki, J., Yokoi, S. and Saito, T.: Understanding Forms by Man and Computer Using Computer Graphics and Image Processing, *Proc. 2nd Int. Symposium for Science on Form* (Ishizaka, S.E.(ed.)), pp.219–231 (1990).
2) Saito, T., Yokoi, S. and Toriwaki, J.: Topology of 3D Digital Line Figure – An Analysis of Knot, Technical Report Comp Pru 90-83, IEICE (1990) (in Japanese).
3) Yamada, M., Itoh, H., Seki, H. and Budiarto, R.: A Cat's Cradle String Diagram Display Method Based on a Genetic Algorithm, *Forma*, Vol.9, No.1, pp.11–28 (1994).
4) Yamada, M., Budiarto, R., Seki, H. and Itoh, H.: A String Diagram Transformation Process Using a Genetic Algorithm – A Cat's Cradle Diagram Generating method, *PRICAI '94 Proceedings*, pp.429–434 (1994).

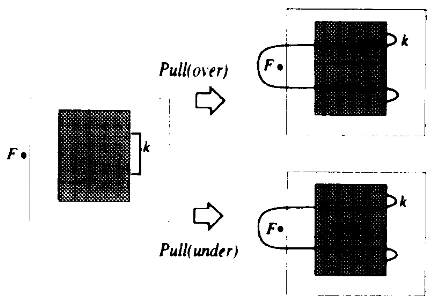
5) Budiarto, R., Yamada, M., Itoh, H. and Seki, H.: Constrained Knot Representation and Its Process, *ICARCV '94 Proceedings*, Singapore, pp.1457-1461 (1994).
6) Kauffman, L.H.: *On Knots, Annals of Mathematics Studies*, Princeton Univ. Press, Princeton (1987).
7) Yamada, M., Budiarto, R., Seki, H. and Itoh, H.: A Representation Method of Ayatori Process and Its String Diagram Processing Method, *Trans. IPSJ*, Vol.35, No.3, pp.497-504 (1994) (in Japanese).
8) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts (1989).
9) Noguchi, H.: *Cat's Cradles Played by Mother and Child*, Goring-shoin Press, Tokyo (1987).
10) Ikeda: *Tanoshiku Asobu Ayatori Zenshu*, Ikeda Publisher, Tokyo (1994).

Appendix

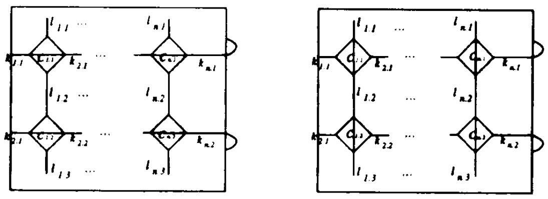
Basic operations of cat's cradle string diagrams are defined as follows.
pull: Pull a part of string and hang it on a finger. It is divided into two types as follows.
pull(over): the string is pulled from the topside and crosses over the lines which it intersects, and
pull(under): the string is pulled from behind and crosses under the lines which it intersects.

Both operations can influence the number of crossings and the number of lines in the diagram. **Figure 17**a shows the string segment (line) k is *pulled(over)* to finger F. Figure 17b illustrates how the *pull(over)* operation works in the meshed area of Fig. 17a. k and $l_1, \ldots, l_n$ are divided into $k_{1.1}$, $k_{2.1}$, $ldots$, $k_{n.1}$, $k_{1.2}$, $k_{2.2}$, $\ldots$, $k_{n.2}$ and $l_{1.1}$, $l_{1.2}$, $l_{1.3}$, $l_{2.1}$, $l_{2.2}$, $l_{2.3}$, $\ldots$, $l_{n.1}$, $l_{n.2}$, $l_{n.3}$ respectively. New crossings $c_{1.1}$, $c_{2.1}$, $\ldots$, $c_{n..1}$, $c_{1.2}$, $c_{2.2}$, $\ldots$, $c_{n.2}$ are produced. The *pull(over)* operation represents the finger operations shown in Fig. 4a. Figure 17c illustrates how the *pull(under)* works in the meshed area of Fig. 17a.
release: Release a part of string from a finger. **Figure 18** shows string k is released from the finger F. This operation represents the finger operations shown in Fig. 4b.
exchange: Exchange the outside and the inside part of the string hung on the same finger. The exchange operation is shown in **Fig. 19**. This operation represents the finger operations shown in Fig. 4c.

(Received May 9, 1995)
(Accepted November 2, 1995)



(a)



(b) *pull(over)* (c) *pull(under)*

Fig. 17 Pull operation.

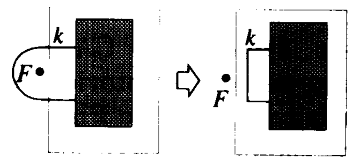


Fig. 18 Release operation.

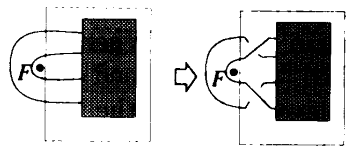


Fig. 19 Exchange operation.

Rahmat Budiarto received the B.Sc. degree from the Bandung Institute of Technology (ITB), Indonesia in 1986 and an M.Eng. degree from the Nagoya Institute of Technology (NIT). in 1995. Currently he is a doctoral candidate at NIT. From 1987 to 1988, he had worked at Poly-technique Education Devolopment Center. From 1989 to 1992 he worked at Nusantara Aircraft Industries Ltd. in Indonesia. His current research interests are computer graphics and computer vision. He is a member of the Information Processing Society of Japan (IPSJ).

Masashi Yamada received his B.Eng. and M.Eng. degree from the Nagoya Institute of Technology in 1992 and 1994, respectively. Since 1994, he has been an Assistant Professor in Department of Artificial Intelligence and Computer Science at Nagoya Institute of Technology. He is a member of IPSJ and JSAI.

Hirohisa Seki received the B.Eng., M.Eng. and Dr. Eng. degrees from the University of Tokyo in 1979, 1981 and 1991, respectively. He joined the Central Research Laboratory of Mitsubishi Electric Corporation in 1981 and engaged in the research on artificial intelligence. From 1985 to 1989, he was with the Institute for New Generation Computer Technology (ICOT). Since 1992, he has been an Associate Professor in Department of Artificial Intelligence and Computer Science at Nagoya Institute of Technology. His current research interests include logic programming, deductive databases and automated deduction. He is a member of ACM, IEEE Computer Society, IPSJ and JSAI.

Hidenori Itoh received a Ph.D. degree in 1974 from Nagoya University. From 1974 to 1985, he worked at Nippon Telephone and Telegraph laboratories, developing operating systems. From 1985 to 1989, he worked at ICOT, the Fifth Generation Computer Project, developing knowledge base systems. Since 1989, he has been a professor in Department of Artificial Intelligence and Computer Science at Nagoya Institute of Technology. Profs. Itoh is a member of IPSJ, JSAI and Japan Society for Fuzzy Logic.