'Divergence via Abstraction' Practices for Inventing New Puzzles

Kozo Sugiyama, Atsuhiko Maeda and Akinori Mizumoto

September 25, 2003

KS-RR-2003-003

'DIVERGENCE VIA ABSTRACTION' PRACTICES FOR INVENTING NEW PUZZLES

Kozo Sugiyama
School of Knowledge Science, Japan Advanced Institute of Science and Technology, Ishikawa, 923-1292, Japan.
Sugi@jaist.ac.jp

Atsuhiko Maeda
NTT Network Innovation Labs, NTT Coorporation, Yokosuka, 239-0847, Japan.
Maeda.atsuhiko@lab.ntt.co.jp

Akinori Mizumoto
School of Knowledge Science, Japan Advanced Institute of Science and Technology, Ishikawa, 923-1292, Japan.
A-mizumo@jaist.ac.jp

Abstract

To invent new puzzles, we take a systematic approach called *divergence via abstraction*. Existing popular puzzles called *permutation puzzles* and *cyclic puzzles* are abstracted and converted into other media such as graphs, blocks, sounds, and robots, while preserving their logic. We implement puzzle generators on different media and variations of generated puzzles are shown. Merits and demerits of new puzzles are evaluated comparing with the original puzzles. This systematic approach can flexibly parameterize and creatively extend the puzzles.

Keywords: systematic approach, divergence via abstraction, puzzle invention, media conversion

1. Introduction

Puzzles have been continually developed and extremely polished in a long history. Therefore, popular puzzles have sophisticated logical structures, beautiful shapes, and attractive user interfaces. However, usually it is too difficult to solve the Rubik's cube, for example, and so it is desirable to parametrically change the levels of difficulty of the puzzles. Moreover, it is interesting to realize the puzzles on different media. Thus, to make the world of puzzles further richer and to design new interfaces of puzzles, we have carried out a systematic approach called *divergence via abstraction*. The basic idea of the approach is illustrated in Figure 1 concretely and conceptually. Here existing popular puzzles are abstracted and converted into other media such as graphs, blocks, sounds, and robots, while preserving their logic. Using this approach, we can invent new puzzles [1, 2,3,4].

In this paper, we are concerned with two classes of puzzles, *permutation puzzles* and *cyclic puzzles*. We analyze the operations of the puzzles and derive two abstract (mathematical) models. Based on these models, we implement various puzzle generators on different

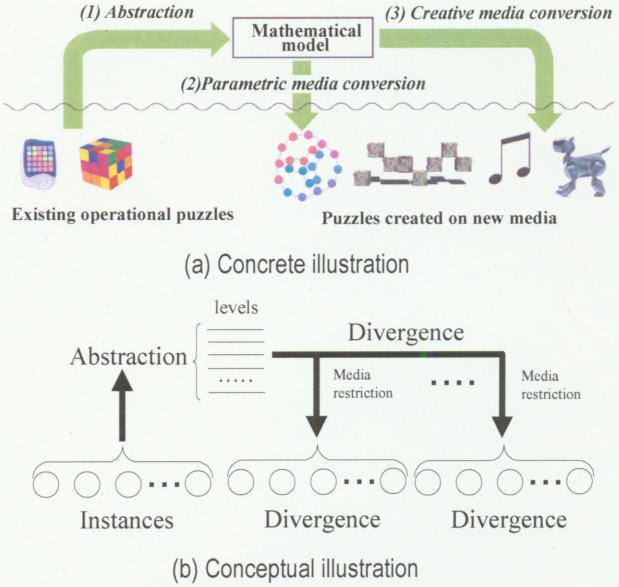

Figure1 Systematic approach for new puzzle invention

media. Using these puzzle generators, we can parametrically change the levels of difficulty and invent new puzzles that have new aspects and/or unexpected attractiveness. Merits and demerits of new puzzles on new media are outlined comparing with the original puzzles. It is concluded that our systematic approach is effective for flexibly parameterizing and creatively extending the original puzzles.

2. Abstract Models of Puzzles

2.1 Permutation Puzzles

The Rubik's cube is one of the most popular puzzles and there are several variations such as the Megalinx and the Pyraminx (see Figure 4). They differ in shapes and the number of elements, but have similar structure. These are classified as *permutation puzzles* because the operations on the puzzles can be characterized as permutations between elements.

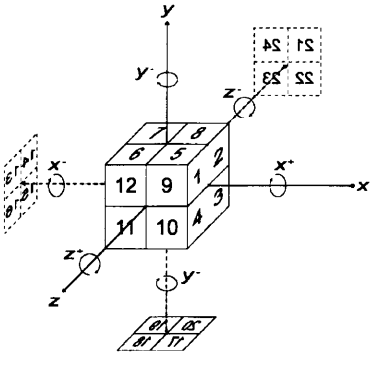

Figure 2 The 2^3 Rubik's cube and a numbering

For simplicity, we consider the 2^3 Rubik's cube, which consists of 2×2×2 blocks. Figure 2 shows the 2^3 Rubik's cube. The elements in the surface of the cube are numbered from 1 to 24. Note that only 6 different colors are enough to color the surface of the cube. The *color mapping* is one of the important factors characterizing the puzzle.

The 2^3 Rubik's cube has 12 operations: 90° clockwise (or anti-clockwise) rotations of four blocks around the positive (or negative) direction of each axis. However, there are redundancies between the operations and it is sufficient to consider only three operations x^+, y^+ and z^+. We define the *operational redundancy* of the 2^3 Rubik's cube as 9/12, Redundancy is another important factor for characterizing puzzles.

Operations x^+, y^+ and z^+ can be denoted as the following *(4·4·4)-type expressions*: sequences of three cyclic permutations, each of length 4.

$$\begin{aligned} x^+ &= (1\ 2\ 3\ 4)(5\ 21\ 20\ 10)(8\ 22\ 17\ 9) \\ y^+ &= (5\ 6\ 7\ 8)(1\ 12\ 14\ 21)(2\ 9\ 13\ 24) \\ z^+ &= (9\ 10\ 11\ 12)(1\ 17\ 16\ 6)(4\ 18\ 13\ 5) \end{aligned} \quad (1)$$

If we insert the elements of the last permutation into the elements of the second permutation in (1), we have the following $(4\cdot8^2)$-type expressions.

$$\begin{aligned} x^+ &= (1\ 2\ 3\ 4)(5\ 8\ 21\ 22\ 20\ 17\ 10\ 9)^2 \\ y^+ &= (5\ 6\ 7\ 8)(2\ 1\ 9\ 12\ 13\ 14\ 24\ 21)^2 \\ z^+ &= (9\ 10\ 11\ 12)(1\ 4\ 17\ 18\ 16\ 13\ 6\ 5)^2 \end{aligned} \quad (2)$$

Similarly, (12^3)-type expressions can be defined as follows (this is *the maximum length expression*).

$$\begin{aligned} x^+ &= (1\ 5\ 6\ 2\ 21\ 22\ 3\ 20\ 17\ 4\ 10\ 9)^3 \\ y^+ &= (2\ 1\ 5\ 9\ 12\ 6\ 13\ 14\ 7\ 24\ 21\ 8)^3 \\ z^+ &= (9\ 1\ 4\ 10\ 17\ 18\ 11\ 16\ 13\ 12\ 6\ 5)^3 \end{aligned} \quad (3)$$

Based on the above analysis (similar analysis can be done for the Megalinx and the Pyraminx), we can derive an abstract operation model M_p of permutation puzzles as follows:

$M_p = (X, C, \varphi, R, \boldsymbol{s}, \boldsymbol{t})$ where

- $X = \{1, 2, \ldots, n\}$: a set of n elements;
- $\varphi : X \rightarrow C = \{c_1, c_2, \ldots, c_p\}$: a color mapping. C is a set of p colors;
- $R = \{r_1, r_2, \ldots, r_m\}$: a set of m operations, and

$$r_i = p_{i1}^{q_{i1}} p_{i2}^{q_{i2}} \ldots p_{ik_i}^{q_{ik_i}} \quad (i = 1,2,\ldots,m) \quad (4)$$

- $\boldsymbol{s}$: an initial state obtained by randomly repeating operations from the goal state;
- $\boldsymbol{t}$: the goal state that is a sequence of the numbers labeled to the elements.

In (4), each permutation p_{ij} corresponds to a cycle. Thus we can define a *puzzle graph*, which consists of a set of cycles.

2.2 Cyclic Puzzles

We can classify the puzzles such as the Lightsout, the Lightsout cube, and the Rubik's clock in Figure 7 as

cyclic puzzles. This is because when we repeat the same operation on the puzzle, the state of the puzzle changes cyclically and returns to the original state after a fixed number of operations.

For example, the Lightsout has 5×5 buttons with lights and the Lightsout cube has 3×3×6 buttons with lights. The goal of the puzzle is to turn off all the lights or change all the colors of the lights to the same one. Pressing one button toggles the lights of buttons of the one selected and its neighbors.

The method for solving the Lightsout puzzle has been studied in [5]; by pressing each button twice, the ON/OFF state returns to the original state. A solution can be determined whether each button is pressed once or not. Therefore, the Lightsout puzzle can be formulated as follows: for a given graph $G=(V, E)$, we define the matrix $A=[a_{ij}]$ as

$$a_{ij} = \begin{cases} 1 & (i = j \text{ or } e = (v_i, v_j) \in E) \\ 0 & (e = (v_i, v_j) \notin E) \end{cases} \quad (5)$$

The matrix A represents the toggling rules in regards to the buttons. We denote the solution vector as $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ where $x_i = 1$ (pushed) or $x_i = 0$ (not pushed). Further we denote an input vector as $\boldsymbol{b} = \{b_1, b_2, \ldots, b_n\}$ which represents an initial state of the buttons. To obtain a solution, we define an equation $A\boldsymbol{x}=\boldsymbol{b}$ where

$$b_i = a_{1i}x_1 \oplus a_{2i}x_2 \oplus \cdots \oplus a_{25i}x_{25}, \quad i = 1,\ldots,25 \quad (6)$$

The equation $A\boldsymbol{x}=\boldsymbol{b}$ has a solution if and only if $r(A)=r([A\ \boldsymbol{b}])$[5].

We denote the current state of the buttons as $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ and when $push(v_i)$ is performed, the state $\boldsymbol{y}$ is replaced by

$$y_j \leftarrow y_j \oplus a_{ij} \pmod 2, \quad j = 1, \ldots, 25. \quad (7)$$

Figure 7(c) shows the Rubik's clock, which has more complicated structure. Each clock has 12 states, and ON/OFF states of four switches changing toggling rules between clocks. There are 16 different sets of rules and three types of effects: positive, negative, and no effect. However, the basic structures of these three puzzles are similar. It has been shown that the operational redundancy of the Rubik's clock is 112/128 while that of the Lightsout is 2/25[2].

Based on the analysis, we can define an abstract model M_c of cyclic puzzles as

$M_c = (\boldsymbol{y}, q, A, R, \boldsymbol{b}, \boldsymbol{t})$ where

- $\boldsymbol{y} = [y_1, y_2, \ldots, y_m]$, $y_i \in \{1, 2, \ldots, q\}$: a vector of element states;
- $A=\{A_1, A_2, \ldots, A_p\}$: a set of n x m adjacency matrices;
- $R=\{r_1, r_2, \ldots, r_n\}$: a set of operations where

  $r_i : y_j \leftarrow y_j \oplus a_{ij} \pmod q$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. (8)
- $\boldsymbol{b}$: an initial state (or input pattern)
- $\boldsymbol{t}$: the goal state.

3. Generation and Evaluation

3.1 Permutation Puzzles

Puzzle Generator: Based on the model, we implement a puzzle generator that allows a user to define their own puzzle. Figure 3 shows a user interface of the permutation puzzle generator. To define a new puzzle, the user needs to input the number of elements, expressions for the operations, and a color mapping. An example of defining a puzzle is shown in the left window. We use a spring algorithm [6] for the layout. However, we observe that it sometimes fails to achieve a good layout. Thus we allow the user to interact with the drawing to improve the layout. An example of the layout is shown in the right window. To operate the layout as a puzzle, the user can simply use drag and drop to move an element or to rotate a cycle. Then the cycle that contains the element and the other cycles that are included in the operation rotate together.

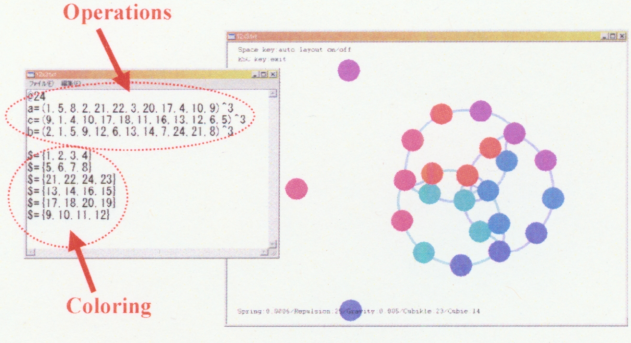

Figure 3 User interface of a permutation puzzle generator

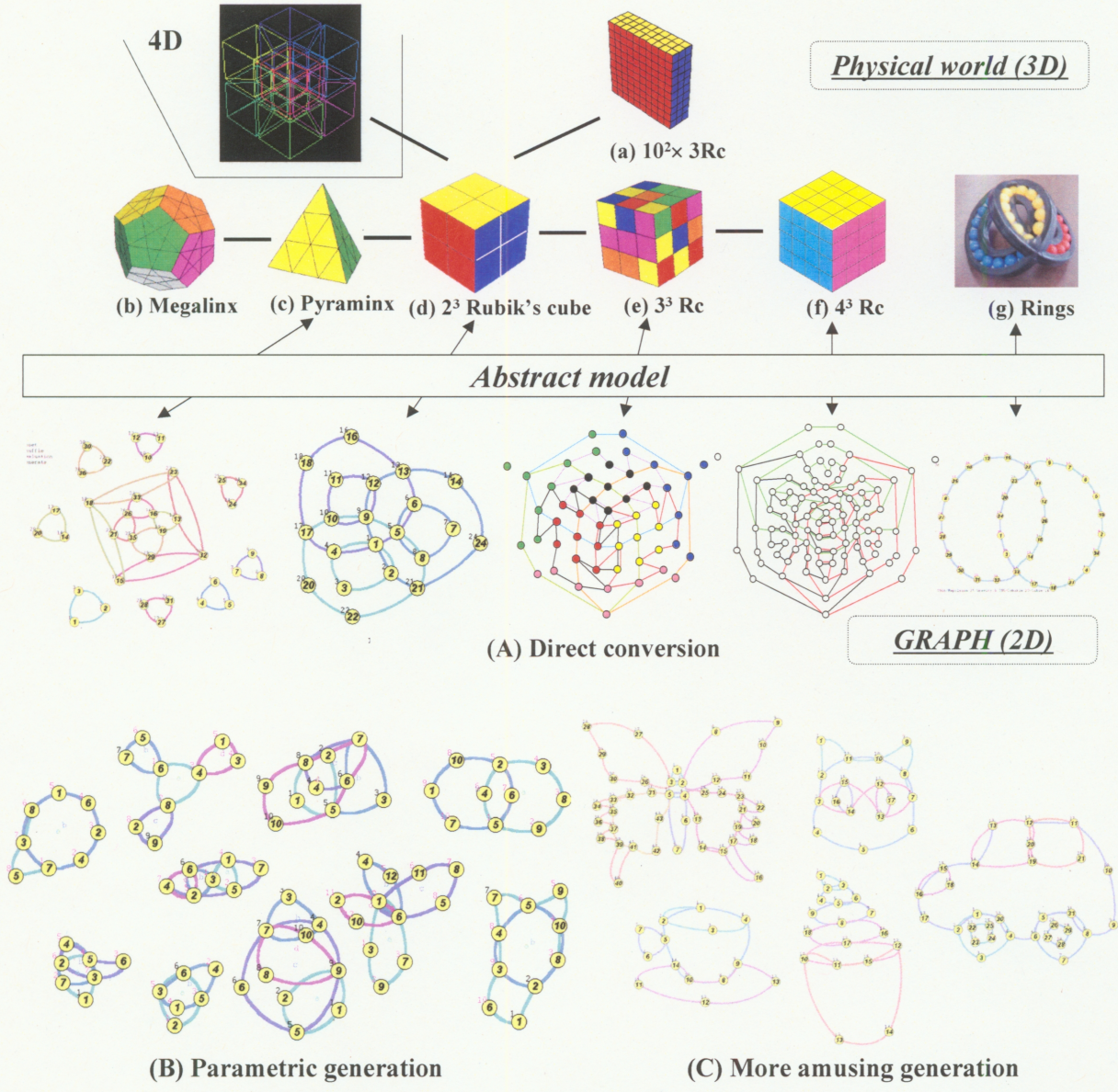

Figure 4 Existing permutation puzzles and generated puzzles on graph media

Puzzles on Graph Media: In Figure 4, the existing permutation puzzles are shown in the upper part and generated puzzles on graph media are shown in the lower part. In (A), direct conversions of the existing puzzles are presented where all graphs in (A) are planar. In fact, the graphs corresponding to the $(4{\cdot}8^2)$-type and the (12^3)-type expressions of the 2^3 Rubik's cube are planar and the longest expression of the n^3 Rubik's cube can be modeled as a planar graph for any $n \geq 2$ [3]. Figure 4(B) shows examples of the new puzzles with small size. Note that sometimes hand drawn drawings can be more amusing and attractive for the users as shown in Figure 4(C).

Table 1 Comparison between media for permutation puzzles

	Physical Media	Graph Media
Symmetry	3D symmetry	2D symmetry
See the whole	Can't see the whole at a glance	+Can see the whole at a glance
Variations	Rigid to change	+Easy to change
Color mapping	Each plane with a same color	+Flexible to paint
Size	Fixed	+Easy to change
Decomposition of problem	Physically fixed	+Sometimes reveal
Layout	Fixed	+Many variations

Table 1 shows comparisons between media of existing puzzles (physical) and generated puzzles (graph). From Table 1 we can see that graph media have merits in most items comparing with existing puzzles (see + in the table).

Implementation on Robot: We have preliminarily tried to map a permutation puzzle onto elements (a head lamp and four foots) of a robot called AIBO that is a commercial product by SONY. Figure 5 shows a scene, where the upper-left graph puzzle is mapped onto the AIBO. As the AIBO has a voice sensor, the AIBO elements are operated with voice commands. *In the robot puzzle, information about connectivity is lost. This makes the puzzle more difficult but interesting.*

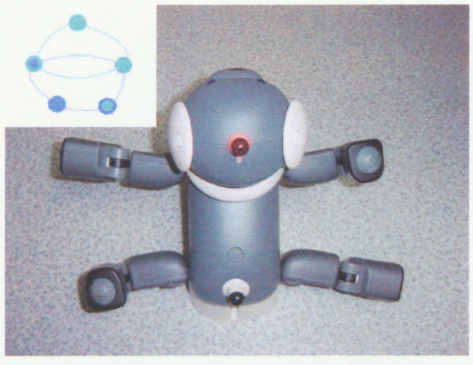

Figure 5 A graph puzzle and its corresponding robot puzzle

3.2 Cyclic Puzzles

Puzzles on Graph/Block Media: Figure 6 shows a user interface for a cyclic puzzle generator. A puzzle (i.e. toggling rules) is defined in the lower-left window and the rules are illustrated using *L-mapping* [6] in the lower-right window. An 'up and down box puzzle' is shown in the upper window.

Figures 7(C_1)-(C_3) show variations of the layouts of cyclic puzzles that correspond to *n*-color cyclic (or multi-state) puzzles. We applied various graph drawing methods to achieve a variety of layouts for the puzzle. Figure 7(B_1) shows an *orthogonal* layout [6] and Figure 7(B_2) shows a *visibility representation* [6]. Figure 7(A) shows a symmetric layout of the Lightsout cube, which uses the concept of concentric circles.

Table 2 shows comparisons among media for cyclic puzzles: Lightsout, Rubik's clock, and graph/block puzzles. From Table 2 we can see that graph/block media have merits in several items comparing with existing puzzles (see + in the table).

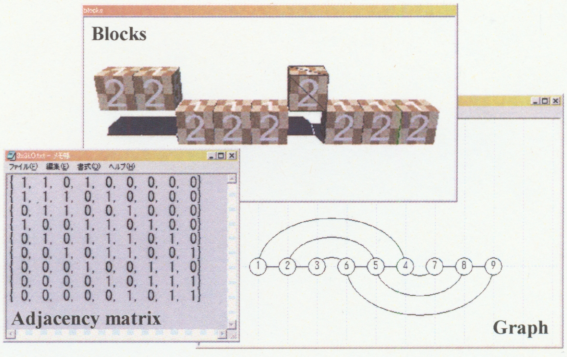

Figure 6 User interface for cyclic puzzles

Table 2 Comparison among media for cyclic puzzles

	Lightsout	Rubik's clock	Graph/block
Adjacency	See explicitly	Implicit	Can choose either
Planarity	Planar	Non-planar	Planar a.m.a.p.
See the whole	possible	impossible	+possible
Variations	Fixed	Fixed	+Easy to change
Size	Fixed	Fixed	+Easy to change
Layout	Fixed	Fixed	+Many variations

Sound puzzles: We generated a sound puzzle where a familiar melody with 7 notes (see Figure 7(c_4)). Note that this puzzle is very similar to 'up and down' puzzle in Figure 7(C_1) except listening instead of viewing. We asked subjects to solve the puzzle. The subjects all are experts of music. Their impressions on this puzzle are as follows: This is very fresh and interesting. But it is difficult to solve even with 7 notes. 4 notes are adequate for beginners. This puzzles seems very useful for music education of both children and adults.

4 Concluding Remarks

We present a new application for inventing new puzzles. We define abstract models for permutation and cyclic puzzles which can be modeled as puzzle graphs. Based on these models, we implement two puzzle generators and produce various puzzles on various media. Using the puzzle generators, we create new puzzles. Moreover by applying various graph drawing algorithms [6], we create new user interfaces for a puzzle with different attractions.

Our ultimate goal is to develop software that implements the puzzle generators and various layouts, so that the users can define their own puzzles and then communicate the puzzles with small communication devices, such as PDA or mobile phones.

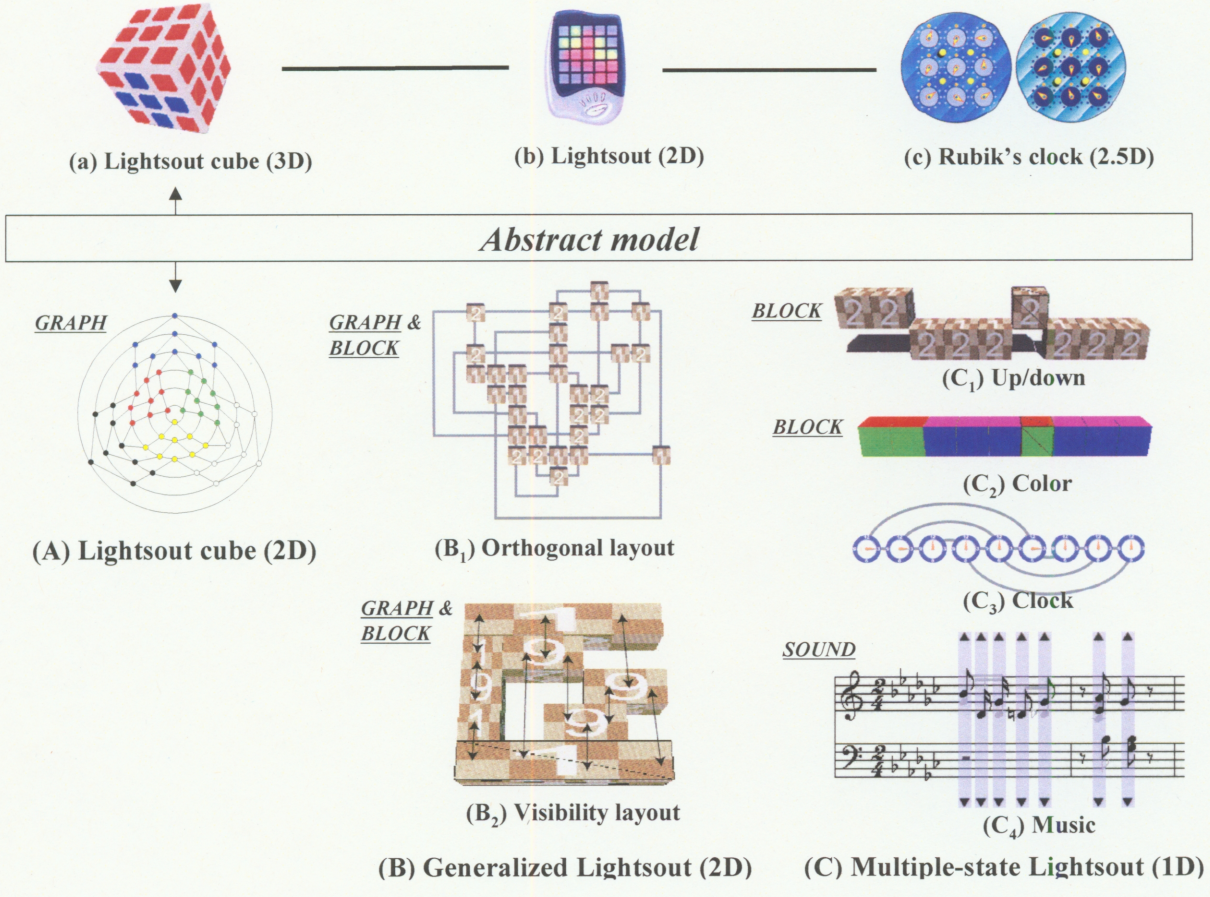

Figure 7 Existing cyclic puzzles and generated puzzles on various media

References

[1] Maeda, A., Sugiyama, K. and Mase, K.: Conversion of permutation puzzles and development of permutation puzzles generators, IPSJ SIG Notes Human Interface, No.101, 33-40, 2002. (in Japanese)

[2] Maeda, A., Sugiyama, K. and Mase, K.: Conversion of cyclic puzzles and development of cyclic puzzles generators, IPSJ SIG Notes Human Interface, No.101, 41-48, 2002. (in Japanese)

[3] Sugiyama, K., Hong, S. and Maeda, A.: The puzzle conversion and layout problem, Research Report KS-RR-2003-002, JAIST, 13p, 2003.

[4] Sugiyama, K., Hong, S. and Maeda, A.: The puzzle layout problem, Proc. of GD'03, Perugia, Sept. 2003. (to appear in Springer LNCS)

[5] Aoki, S., A location problem with variables on Boolean field, B. Eng. Thesis in Information Engineering, Toyohashi University of Technology, 1997. (in Japanese)

[6] Sugiyama, K.: Graph drawing and applications for software and knowledge engineers, World Scientific, 2002,