We propose an oritatami system that $\Xi = (\mathcal{H}, \alpha, \delta, \sigma, w)$ folds into the n-bit Heighway dragon. The dragon it folds into is actually slanted as illustrated in Figure 1, which is more natural than the conventional (upright) one to be folded over the triangular grid. The oritatami system employs about 2000 bead types, not depending on n. Its delay δ is set to 3 and its arity α is also set to 3.

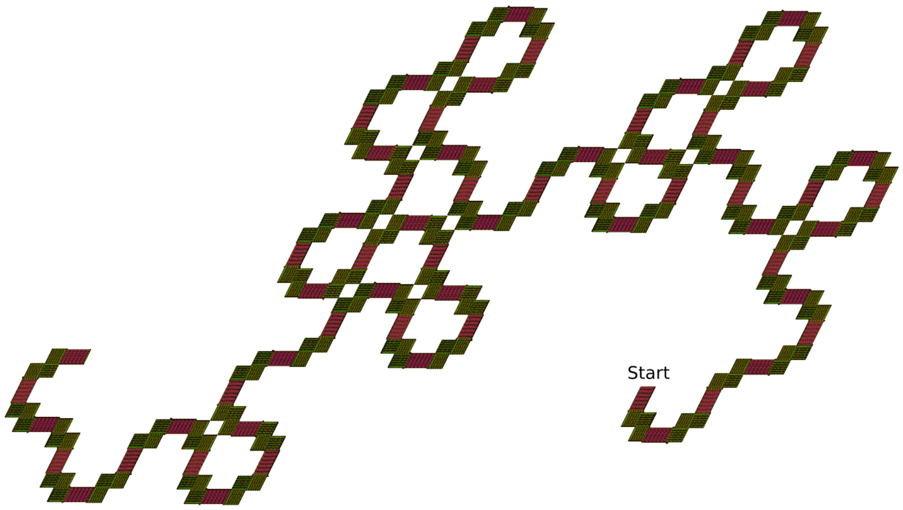

Fig. 1. the Heighway dragon: 6 bits

This oritatami system is periodic, that is, its transcript w is a periodic sequence. Recall that we can draw the Heighway dragon by repeating the following processes:

1. $i \leftarrow i + 1$ (increment)
2. copy i(drawing a segment)
3. compute P_i
4. turn according to P_i

One period of the system consists of four subsequences: Counter, Xerox, DFAO, and Turner, that handle the four tasks, respectively. These subsequences fold as abstracted in Figure 2 into one segment plus one turn of the Heighway dragon.

Counter and Xerox are implemented as done in the binary counter of [**?**].[1] They increment/copy i represented in binary. The resulting i is encoded at the bottom of them as a sequence of bead types; different sequences are thus exposed for 0 and 1.

[1] The oritatami system of [**?**] uses a different but more tractable dynamics than the dynamics used in this paper.

In order to explain how DFAO and Turner are implemented, we have to raise two issues of significance specific to the folding of Heighway dragon by oritatami system. The first issue arises from that the dragon is slanted. The slanted dragon involves two types of left turn as well as two types of right turn: acute and obtuse. Does this mean that we have to implement Turner that is capable of all the four types of turn? That is not the case. Observe that after the (slanted) vertical segment, certainly the left turn is acute and the right turn is obtuse, while after the horizontal segment the left turn is obtuse and the right turn is acute.

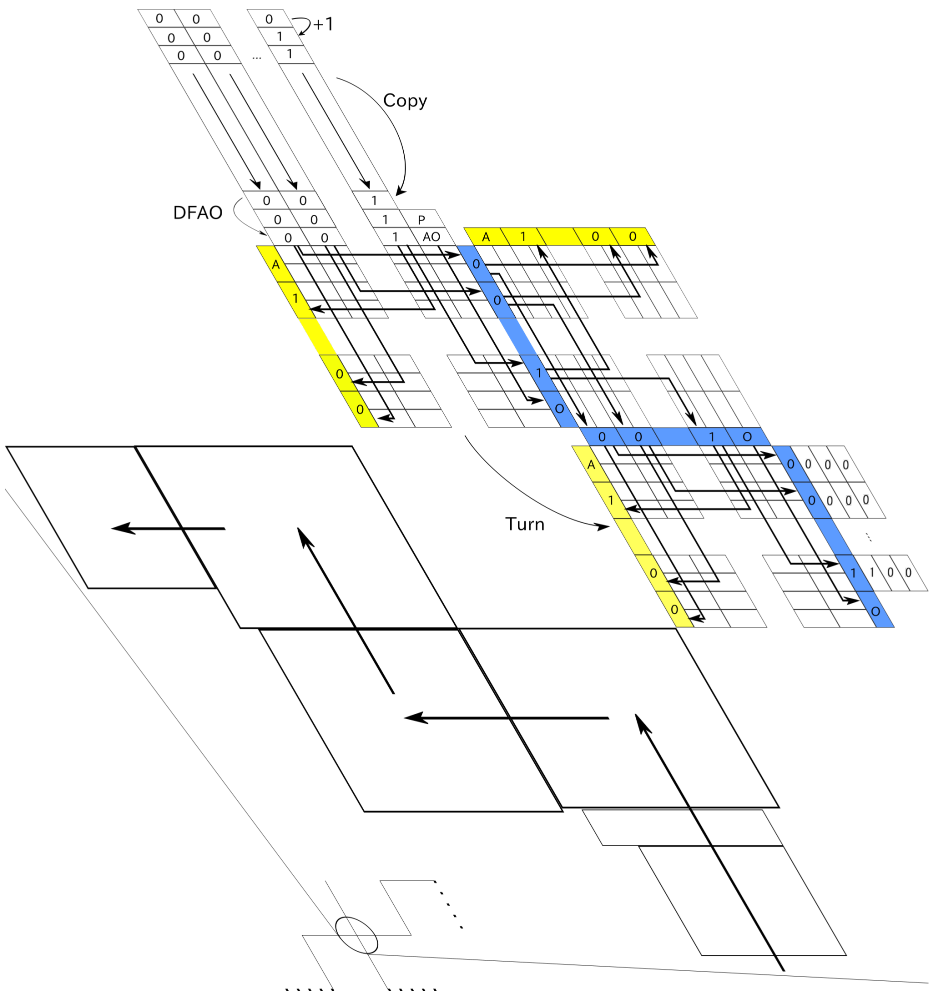

Fig. 2. The abstract of the Heighway dragon

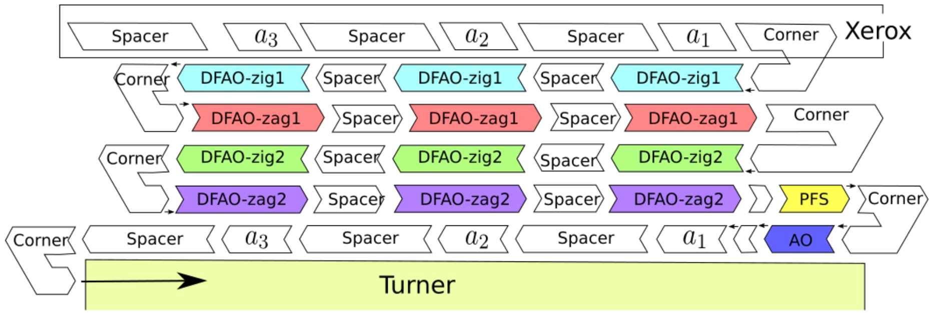

Fig. 3. The abstract of the *DFAO*

0.1 DFAO

DFAO is composed of the six modules: "DFAO-zig1", "DFAO-zag1", "DFAO-zig2", "DFAO-zag2", "PFS", "AO" (or "$\overline{\text{AO}}$"). It folds as abstracted in Figure 3. What the system does in the first zig-zag is to have DFAO-zig1 and DFAO-zag1 read the current count i from its least significant bit and "mark" the first 0, while propagating i. In the second zig-zag, it employs DFAO-zig2 and DFAO-zag2 to check whether the automaton transitions to the state q_2 ($P_i = R$) or else ($P_i = L$). The second zag is to end at the top if $P_i = R$ or at the bottom if $P_i = L$. PFS takes one of the two conformations in Figure 4 and outputs P_i downward. In vertical segments, PFS is followed by AO, while in horizontal segments, it is followed by $\overline{\text{AO}}$. AO interprets the PFS' output $P_i = R$ as acute and $P_i = L$ as obtuse, as shown in Figure 5. $\overline{\text{AO}}$ interprets them other way around. Let us explain briefly how the modules in the zigzags fold to fulfill their roles.

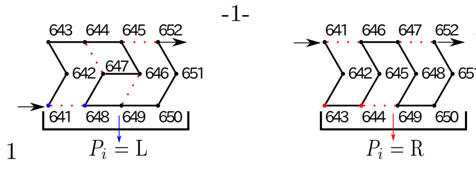

Fig. 4. The possible two conformations of PFS. (left) PFS-L. (right) PFS-R.

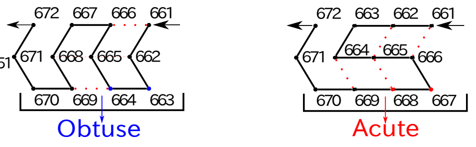

Fig. 5. The possible two conformations of AO. (left) PFS-L. (right) PFS-R.

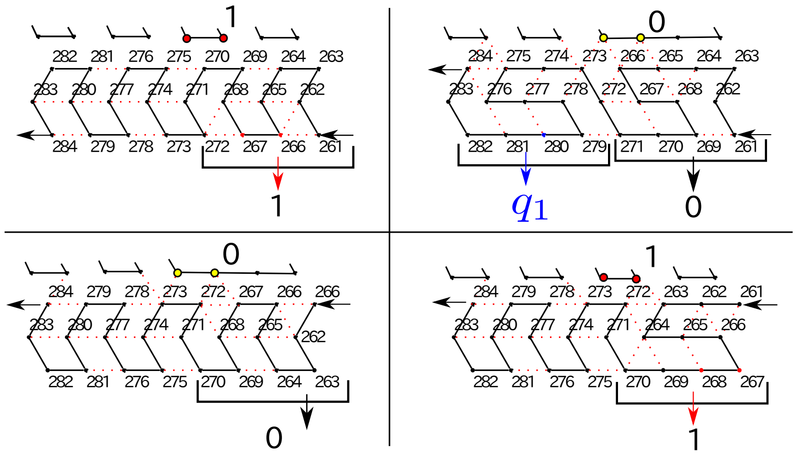

Fig. 6. The possible four conformations of DFAO-zig1. (above left) Dzig1-1. (above right) Dzig1-f0. (below left) Dzig1-20. (below right) Dzig1-21.

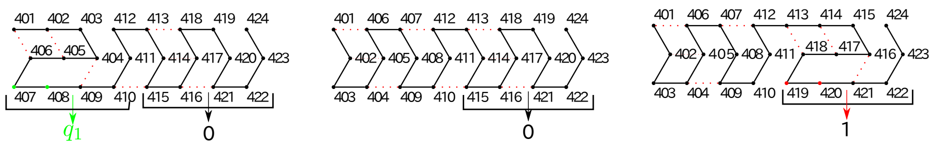

Fig. 7. The possible three conformations of DFAO-zag1. (above left) Dzig1-a. (above right) Dzig1-b. (below left) Dzig1-c. (below right) Dzig1-d.

DFAO-zig1s detect the first 0 in two phases. Phase1 is to copy all the 1's before the first 0 and Phase2 is to copy 0s and 1's after the first 0. These phases are distinguished by the first bead of a molecule (in Phase1 it is at the bottom, while in Phase2 it is at the top, as suggested in Figure 6). That is, in Phase1, DFAO-zig1s take the conformation Dzig1-1 (the top left conformation in FIgure 6). At the first 0, a DFAO-zig1 takes Dzig1-f0, starting at the bottom but ending at the top (phase transition). Each of the succeeding DFAO-zig1s takes one of the remaining two conformations Dzig1-20 and Dzig1-21 for copy in Phase2. In the first zag, DFAO-zag1s propagate 0's, 1's, and the first 0 by taking the proper one of the three conformations in Figure 7

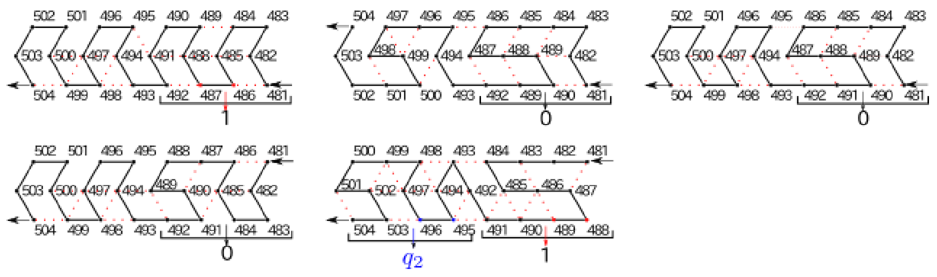

Fig. 8. The possible five conformations of DFAO-zig2. (above left) Dzig2-1. (above right) Dzig2-f0. (below left) Dzig2-0. (below right) Dzig2-f00. (center) Dzig2-f01.

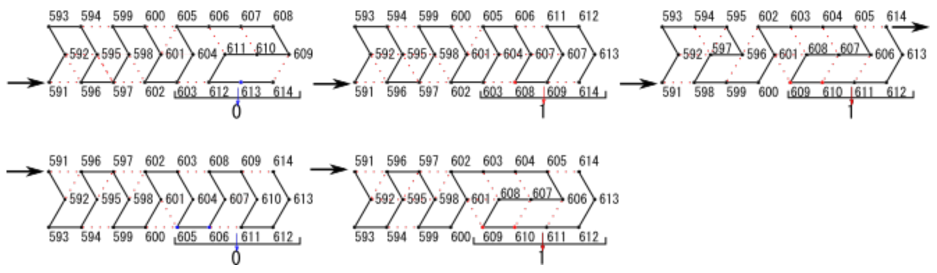

Fig. 9. The possible five conformations of DFAO-zag2. (above left) Dzag2-L0. (above right) Dzag2-L1. (below left) Dzag2-T1. (below right) Dzag2-R0. (center) Dzag2-R1.

In the second zig, DFAO-zig2s check whether the first 0 is followed by 0 or 1, being read from the least significant bit in two phases as well. When DFAO-zig2 reads 0 after the first 0, it takes the conformation Dzig2-f00 (the bottom left in Figure 8). On the other hand, when DFAO-zig2 reads 1 after the first 0, it takes Dzig2-f01 (the automaton transitions to q_2). In the second zag, DFAO-zag2s output P_i to PFS as stated before, while propagating the current count i. This zag starts to fold at the bottom. In the case of $P_i = R$, DFAO-zag2 folds into a shape Dzag2-T1 (the top right in Figure 9), ending at the top when DFAO-zag2 read Dzig2-f01. By contrast, the rightmost conformation of DFAO-zag2 finishes to fold at the bottom when $P_i = L$.