# Bit bifurcation by cotranscriptional folding*

Yusei Masuda, Shinnosuke Seki**, and Yuki Ubukata

Department of Computer and Network Engineering, The University of Electro-Communications, 1-5-1, Chofugaoka, Chofu, Tokyo, 1828585, Japan
s.seki@uec.ac.jp

**Abstract.** We demonstrate cotranscriptional folding of a finite part of the Heighway dragon, a fractal also-known as the paperfolding sequence $P = LLRLLRRL\cdots$ by an oritatami system. The $i$-th element of $P$ can be obtained by feeding $i$ in binary to a 4-state deterministic finite automaton with output (DFAO). We implement this DFAO and a bit-string bifurcator as modules of oritatami system. Combining them with a known binary counter module yields the target oritatami system.

**Keywords:** Heighway dragon, Fractal, Cotranscriptional folding, Oritatami system, Automatic sequence, Bitstring bifurcation

### 0.1 Verificaion

We explain the correctness of the folding. The folding of each module depends on its environment, i.e. on the components already folded nearby and on the current minimum energy conformations output by previous step. Algorithm 1 is an algorithm of designing the $n$-th iteration of the Heighway dragon. It shows all possible environments of each module and component. Using the simulator developed for [1], we have verified that all of the components fold correctly in all possible environments.

### 0.2 Simulation

These modules can be checked by simulators on our website.

## Appendix

### 0.3 Turner

Turner consists of two parts: bit string bifurcator and steering arm (colored in blue in Fig. 1).

* This work is in part supported by JST Program to Disseminate Tenure Tracking System, MEXT, Japan, No. 6F36 and JSPS Grant-in-Aid for Young Scientists (A) No. 16H05854 to S. S.

** Corresponding author

**Algorithm 1** The $n$-th iteration of the Heighway dragon

```
Require: n ≥, length ≤ 0
  current ⇐ 0
  while current < 2^n − 1 do
    carry = 1
    for i = 1 to n do
      if carry = 1 then
        if above component is body-rpx-0 then
          Cozig-11 is folded.
          carry ⇐ 0
        else if above component is body-rpx-1 then
          Cozig-10 is folded.
        end if
      else if carry = 0 then
        if above component is body-rpx-0 then
          Cozig-00 is folded.
        else if above component is body-rpx-1 then
          Cozig-01 is folded.
        end if
      end if
      Spacer is folded.
    end for
    if carry = 1 then
      the folding stops (counter capacity exceeded)
    end if
    current ⇐ current + 1
  end while
```

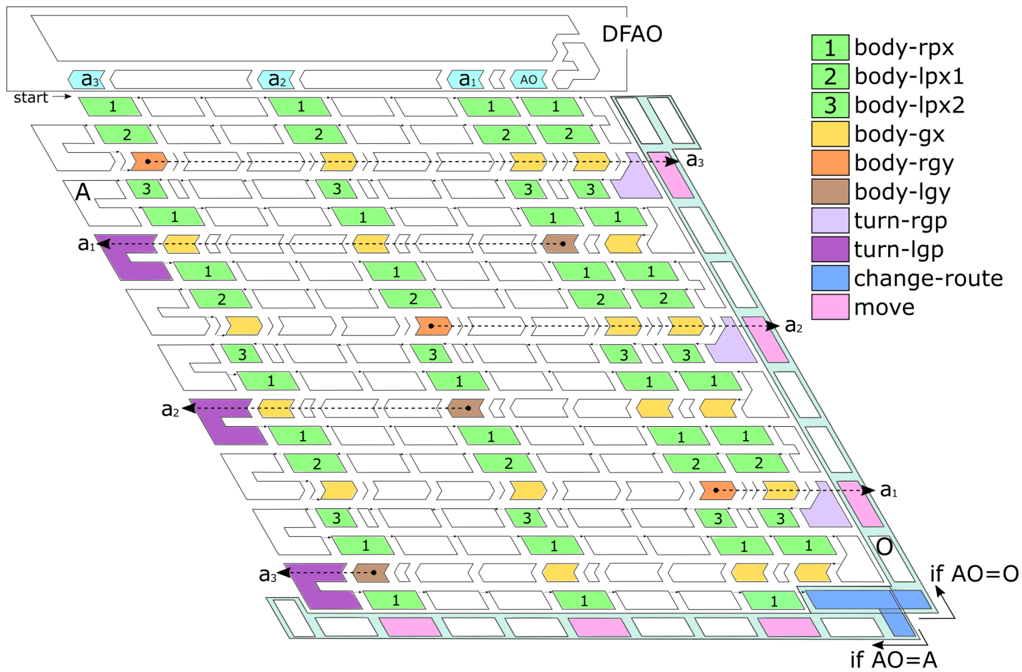


**Fig. 1.** The module-level abstraction of folding of Turner. All the white modules in the middle are spacers, some of which are implemented in the shape of parallelogram instead of glider.

The bifurcator sends binary information as in Fig. **??** while folding into zigzags. For that, it employs four types of module that propagates 1bit vertically (body-rpx, body-lpx1, and body-lpx2 in Fig. 1), that lets 1bit cross another 1bit (body-gx), that forks 1bit vertically and horizontally (body-rgy, body-lgy), and that undergoes transition from a zig to a zag or from a zag to a zig and exposes 1bit outside (turn-rgp, turn-lgp). The first two of them have already been implemented (see, e.g., [1]) so that we shall explain the others.

The module body-rgy takes one of the two conformations in Fig. 2 (left) depending on the 1bit encoded in the two beads above. Output below, the 1bit is encoded as a type of the second bead from left, while output right, it is encoded as whether this module ends top or bottom. Its zag-variant, body-lgy, is implemented analogously; for its conformations, see Fig. 2 (right).

The 1bit forked rightward by a body-rgy transfers till the end of the zig without being jammed because all remaining modules in the zig are designed in such a way that they start and end at the same height like the glider. The module turn-rgp receives the 1bit (top or bottom), and exposes it by taking one of the two comformations in Fig. 3 (bottom). The module turn-lgp functions analogously in zags as being holded in Fig. 3 (top).

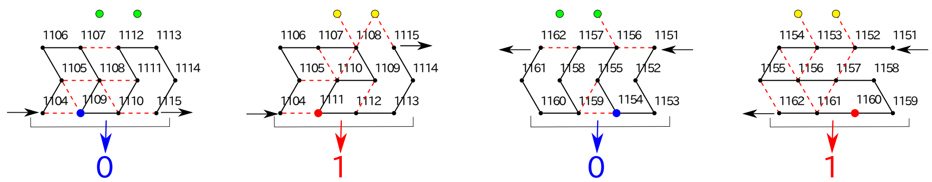


**Fig. 2.** (left) The possible two conformations of body-rgy and (right) of body-lgy.

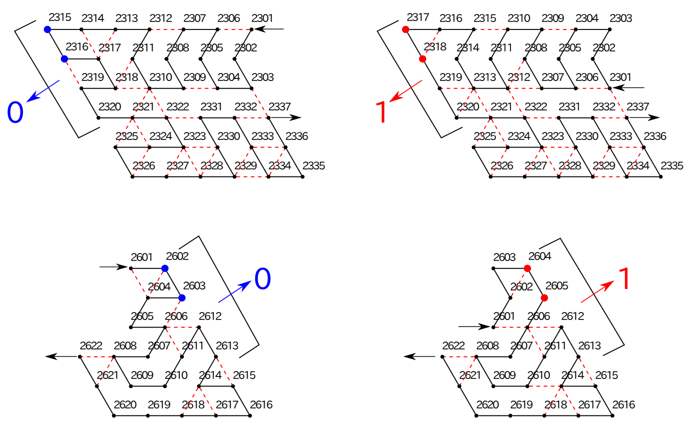


**Fig. 3.** (top) The possible two conformations of turn-lgp and (bottom) of turn-rgp.

The bifurcator also propagates a signal, A or O, fed by the DFAO, to tell the steering arm which way it should take. Specifically, the signal has the first module, change-route, of the steering arm take one of the two conformations in Fig. 4, guiding the rest of the steering arm in the ordered direction. The steering arm is provided with move modules, which let the bifurcated bit string pass through. Note that the Turner does not have to bifurcate AO. Indeed, the second Turner is supposed to turn in the same manner as the first one. It hence suffices to append A and O to the bifurcated bit strings on the acute side and obtuse side, respectively, as shown in Fig. 1.

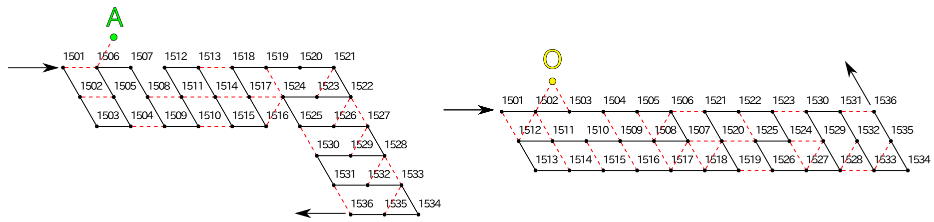


**Fig. 4.** The possible two conformations of change-route.

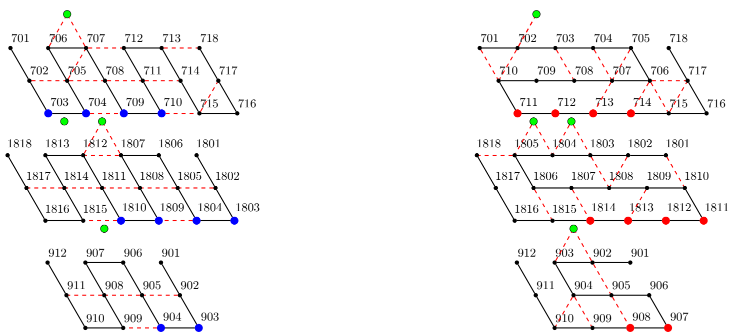


**Fig. 5.** (top) The possible two conformations of body-rpx, (middle) The possible two conformations of body-lpx1, (bottom) The possible two conformations of body-lpx2.

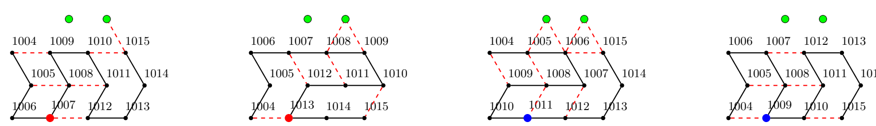


**Fig. 6.** The possible four conformations of body-gx.


**Fig. 7.** The possible two conformations of move.

*Remark 1.* In fact, as suggested in Fig. 1, the bifurcator outputs the bit string also downward. That is, it bifurcates a given bit string into three directions. This provides a more space-efficient way to replicate a bit string many-folds.

## References

1. Han, Y.S., Kim, H., Ota, M., Seki, S.: Nondeterministic seedless oritatami systems and hardness of testing their equivalence. In: Proc. 22nd International Conference on DNA Computing and Molecular Programming (DNA22). LNCS, vol. 9818, pp. 19–34. Springer (2016)