



**HAL**  
open science

# Proving the Turing Universality of Oritatami Co-Transcriptional Folding

Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, Shinnosuke Seki

► **To cite this version:**

Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, Shinnosuke Seki. Proving the Turing Universality of Oritatami Co-Transcriptional Folding. 2017. hal-01567227

**HAL Id: hal-01567227**

**<https://hal.science/hal-01567227v1>**

Preprint submitted on 21 Jul 2017

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Proving the Turing Universality of Oritatami Co-Transcriptional Folding

Cody Geary<sup>1</sup>, Pierre-Étienne Meunier<sup>2</sup>, Nicolas Schabanel<sup>3</sup>, and Shinnosuke Seki<sup>4</sup>

1 California Institute of Technology, Pasadena, CA, USA. [codyge@gmail.com](mailto:codyge@gmail.com).

2 Tapdance, Inria Paris, France. [pe@prijul.org](mailto:pe@prijul.org)

3 CNRS, U. Paris Diderot, France and IXXI, U. de Lyon, France. [www.irif.fr/users/nschaban/](http://www.irif.fr/users/nschaban/)

4 Oritatami Lab, University of Electro-Communications, Tokyo, Japan.  
[www.sseki.lab.uec.ac.jp/](http://www.sseki.lab.uec.ac.jp/)

---

## Abstract

---

We prove that the Oritatami model of molecular folding is capable of embedding arbitrary computations in the folding process itself, by a local energy optimisation process, similar to how actual biomolecules such as DNA or RNA fold into complex shapes and functions.

This result is the first principled construction in this research direction, and motivated the development of a generic toolbox, easily reusable in future work. One major challenge addressed by our design is that choosing the interactions to get the folding to react to its environment is NP-complete. Our techniques bypass this issue by allowing some flexibility in the shapes, which allows to encode different “functions” in different parts of the sequence (instead of using the same part of the sequence).

However, the number of possible interactions between these functions remains quite large, and each interaction also involves a small combinatorial optimisation problem. One of the major challenges we faced was to decompose our programming into various levels of abstraction, allowing to prove their correctness thoroughly in a human readable/checkable form. We hope this framework can be generalised to other discrete dynamical systems, where proofs of such large objects are often difficult to get.



## 1 Introduction

Molecular folding is the biological process that turns one-dimensional sequences into three-dimensional shapes. In the particular context of proteins and RNA, this process has attracted a lot of attention, as it could allow us to engineer our own molecules, and therefore to interact with biological functions. The potential applications range from using bacteria as computing devices or nano-factories, to producing targeted drugs to cure specific diseases with little to no side effects. More fundamentally, understanding “molecular programming” by engineering our own molecules will shed a new light on how these mechanisms, and evolution in particular, work in nature.

If we are to have such an engineering discipline crafting “computing molecules” with arbitrary shapes, we need a theory of these systems to inform of their capabilities and give hints for building actual molecules in the wet lab.

Unfortunately, we seem quite far from a full understanding of these mechanisms. From a practical perspective, the latest efforts to solve the protein design problem [19] are still quite far from a complete general methodology. From a theoretical perspective, it has been shown that, in different variants of the *hydrophobic-hydrophilic (HP) model* [6], the problem of predicting the most likely geometry (or *conformation*) of a sequence is NP-complete [20, 17, 2, 3, 5], both in two and three dimensional models. Approximation algorithms have also been developed [1, 16].

However, the effective speed of molecular folding in actual cells seems to contradict these hardness results. Moreover, its reliability and relative robustness to small changes in conditions or sequences seem to rule out approximations as well.

To understand these phenomena, two essential ingredients of molecular dynamics need to be considered: *thermodynamics*, which governs probability distributions over shapes in the long run, and *kinetics*, which is the step-by-step movements of molecules in solution. Some models of tile assembly, such as the abstract Tile Assembly Model [21, 18] chose to ignore thermodynamics and focused on kinetics, and got excellent results in the lab. Models of molecular folding, like the HP model, focus on hardness results, and for that reason ignore kinetics and work entirely on thermodynamics.

Our goal with Oritatami [11] is to try to understand the kinetics of folding, and in the future get a more complete picture including both aspects. The rationale of this choice is that the wetlab version of Oritatami already exists, and has been successfully used to engineer shapes with RNA in the wetlab [10]. The main feature of RNA that motivates our approach is the fact that RNA folds while being produced, which is known as *co-transcriptional folding*. This process has been shown to play an important role in the final shape of biomolecules [12], especially in the case of RNA [7].

### 1.1 Brief overview of the model

In Oritatami, we consider a finite set of *bead types*, and a periodic sequence of *beads*, each of a specific bead type. Beads are attracted to each other according to a fixed symmetric relation, and in any folding (a folding is also called a *conformation*), whenever two beads attracted to each other are found at adjacent positions, a *bond* is formed between them.

At each step, the latest few beads in the sequence are allowed to explore all possible positions, and we keep only those positions that minimise the energy, or otherwise put, those positions that maximise the number of bonds in the folding. “Beads” are a metaphor for domains, i.e. subsequences, in RNA and DNA.

### 1.2 Main results

In this paper, we construct a “universal” set of 520 bead types, along with a single universal attraction rule for these bead types, with which we can simulate any tag system, and therefore any Turing machine  $\mathcal{M}$ , within a polynomial factor of the running time  $\mathcal{M}$ .

This construction motivated the development of a toolbox composed of two things: common structures that can react to their environment, and solutions to combine these structures into larger constructions.

► **Theorem 1.** *There is a finite universal set of beads  $B$  and an attraction rule  $\heartsuit \subseteq B^2$  such that for any Turing machine  $\mathcal{M}$  running in time  $t$  on input  $x$  (where  $t$  is possibly infinite), there is a seed structure  $\sigma \in B^{\mathbb{Z}^2}$  of size  $O(|x|)$ , and a periodic sequence  $w$  of beads from  $B$  (with period of length  $O(1)$ ) such that  $w$  folds into a structure of size  $O(t^3 \log t)$ .<sup>1</sup>*

Our construction is composed of different *modules*, or subsequences, each building different “sub-shapes” of the global conformation. This result had to overcome a number of important challenges, presented in Sections 1.3 and 1.4.

### 1.3 Proving our designs

The main challenge we faced in this paper was the size of our constructions: indeed, while we developed higher-level geometric constructs to program useful shapes, there is a large number of possible interactions between all different parts of the sequence.

Getting solid proofs on large objects is a common problem in discrete dynamical systems, for instance on cellular automata [8, 4] or tile assembly systems [13]. In this paper, we introduce a general framework to deal with that complexity, and prove our constructions rigorously. This method proceeds by decomposing the sequence into different *modules*, and the space into different areas where exactly one module grows. We can then reason on the modules separately, and only deal with interactions at the border between all possible modules that can have a common border.

### 1.4 Design challenges

As shown in our previous results [9], the problem of choosing an attraction rule so that a single sequence folds into different shapes depending on its environment, is NP-complete. That problem is called the *sequence design problem* in [9].

In the present paper, since our sequence is periodic and has a small number of bead types, a single module can interact with a large number of other modules (including previous copies of itself).

We introduce a tool to cope with such situations, called *socks*. The goal is to “shift” the sequence, so that different parts interact with the various environments. Socks work as follows: whenever different copies of a single subsequence  $s_i, s_{i+1}, \dots, s_{i+k}$  (for some integer  $i$ , and with  $k$  equal to a few dozens) have to interact with a large number of different unrelated environments (where an environment is a local configuration of beads), we reduced the number of environments by folding a small part of the molecule, before  $i$ , into a compact useless shape (with the shape of a “sock”), so that only a later part  $s_j, s_{j+1}, \dots$  with  $j > i + k$  of the sequence interact with a subset of all environments.

This allows us to dispatch the different interactions to different parts of the sequence.

### 1.5 Relationship to other work

This construction generalises our previous results, where we built an arbitrary-width counter with a fixed periodic sequence [9]. In that result, all parts of the structure are densely packed into parallelograms, and these structures react to their environment by folding into a different hamiltonian path in each parallelogram.

That result required tedious manual tweaking of the rule, so that different parts of the sequence interacted nicely with each other. Moreover, finding useful hamiltonian paths is hard, which means that our techniques could not scale well.

In this paper, we solve these issues to a large extent, using the toolbox we introduce in Section 6. Note that the dynamics used is slightly changed compared to [9]. We believe the dynamics used here to be more intuitive, and our previous negative results (NP-completeness of rule design) still hold.

---

<sup>1</sup> The constants in the  $O(\cdot)$ s only depend on the size of the simulated Turing machine.



**2** Definitions and Preliminary Results

The empty word is denoted by  $\varepsilon$ . For  $1 \leq i \leq j \leq n$ , by  $w[i..j]$ , we refer to the factor  $w_i w_{i+1} \cdots w_j$  of  $w$ .

**2.1 Oritatami Systems**

Let  $B$  be a finite set of bead types. A *conformation*  $c$  of a bead sequence  $w \in B^* \cup B^{\mathbb{N}}$  is a directed self-avoiding path in the triangular lattice  $\mathbb{T}$ ,<sup>2</sup> where for all integer  $i$ , vertex  $c_i$  of  $c$  is labelled by  $w_i$ .  $c_i$  is the *position* in  $\mathbb{T}$  of the  $(i + 1)$ th bead, of type  $w_i$ , in conformation  $c$ . A *partial conformation* of a sequence  $w$  is a conformation of a prefix of  $w$ .

For any partial conformation  $c$  of some sequence  $w$ , an *elongation* of  $c$  by  $k$  beads (or *k-elongation*) is a partial conformation of  $w$  of length  $|c| + k$ . We denote by  $\mathcal{C}_w$  the set of all partial conformations of  $w$  (the index  $w$  will be omitted when the context is clear). We denote by  $\mathcal{E}(c, k)$  the set of all  $k$ -elongations of a partial conformation  $c$  of a sequence  $w$ .

**Oritatami systems.** An *Oritatami system*  $\mathcal{O} = (w, \heartsuit, \delta, \sigma)$  is composed of (1) a (possibly infinite) bead sequence  $w$ , called the *primary structure*, (2) an *attraction rule*, which is a symmetric relation  $\heartsuit \subseteq B^2$ , (3) a parameter  $\delta$  called the *delay time* and (4)  $\sigma$ , an initial conformation of  $w$ , called the *seed*.  $\mathcal{O}$  is said *periodic* if  $w$  is infinite and its suffix  $w_{|\sigma|w_{|\sigma|+1}} \cdots$  is a periodic bead sequence. Periodicity ensures that the “program” embedded in the oritatami system is finite (does not hardcode any specific behavior) and at the same time allows arbitrary long computation.

We say that two bead types  $a$  and  $b$  attract each other when  $a \heartsuit b$ . Furthermore, given a conformation  $c$  of  $w$ , we say that there is a *bond* between two adjacent positions  $c_i$  and  $c_j$  of  $c$  in  $\mathbb{T}$  if  $w_i \heartsuit w_j$  and  $|i - j| > 1$ . The *number of bonds* of conformation  $c$  of  $w$  is denoted by  $H(c) = |\{(i, j) : c_i \sim c_j, j > i + 1, \text{ and } w_i \heartsuit w_j\}|$ .

**Oritatami dynamics.** The folding of an oritatami system is controlled by the delay time  $\delta$ . Informally, the conformation grows from the seed conformation, one bead at a time. This new bead adopts the position(s) that maximise the potential number of bonds the conformation can make when elongated by  $\delta$  beads in total. This dynamics is oblivious as it keeps no memory of the previously preferred positions; it differs thus slightly from the hasty dynamics studied in [11]; it might also be considered as closer to experimental conditions.

Formally, given an Oritatami system  $\mathcal{O} = (p, \heartsuit, \delta, \sigma)$ , we consider the *dynamics*  $\mathcal{D} : 2^{\mathcal{C}} \rightarrow 2^{\mathcal{C}}$  that maps every subset  $S$  of partial conformations of length  $\ell$  of  $w$  to the subset  $\mathcal{D}(S)$  of partial conformations of length  $\ell + 1$  of  $w$  as follows:

$$\mathcal{D}(S) = \bigcup_{c \in S} \arg \max_{\gamma \in \mathcal{E}(c, 1)} \left( \max_{\eta \in \mathcal{E}(\gamma, \delta - 1)} H(\eta) \right)$$

The possible conformations at time  $t$  of the Oritatami system  $\mathcal{O}$  are the elongations of the seed conformation  $\sigma$  by  $t$  beads in the set  $\mathcal{D}^t(\{\sigma\})$ .

We say that the Oritatami system is *deterministic* if at all time  $t$ ,  $\mathcal{D}_w^t(\{\sigma\})$  is either a singleton or the empty set. In this case, we denote by  $c^t$  the conformation at time  $t$ , such that:  $c^0 = \sigma$  and  $\mathcal{D}^t(\{\sigma\}) = \{c^t\}$  for all  $t > 0$ ; we say that the partial conformation  $c^t$  *folds (co-transcriptionally) into* the partial conformation  $c^{t+1}$  deterministically. In this case, at time  $t$ , the  $(|\sigma| + t + 1)$ -th bead of  $w$  is placed in  $c^{t+1}$  at the position that maximises the number of bonds that can be made in a  $\delta$ -elongation of  $c^t$ .

We say that the Oritatami system *halts* at time  $t$  if  $t$  is the first time for which  $\mathcal{D}^t(\{\sigma\}) = \emptyset$ . The folding process may only stop because of a geometric obstruction (no more elongation is possible because the conformation is trapped in a closed area).

In this article, we will only consider deterministic periodic Oritatami systems with delay time  $\delta = 3$ .

<sup>2</sup> The triangular lattice is defined as  $\mathbb{T} = (\mathbb{Z}^2, \sim)$ , where  $(x, y) \sim (u, v)$  if and only if  $(u, v) \in \{(x - 1, y), (x + 1, y), (x, y + 1), (x, y - 1), (x + 1, y + 1), (x - 1, y - 1)\}$ . Every position  $(x, y)$  in  $\mathbb{T}$  is mapped in the euclidean plane to  $x \cdot X + y \cdot Y$  using the vector basis  $X = (1, 0)$  and  $Y = \text{rotation}_{-120^\circ}(X)$ .

## 2.2 Skipping Cyclic Tag systems

In the next sections, we demonstrate the existence of a single periodic primary structure that can simulate any Turing computation. Precisely, our construction simulates the following particular type of tag systems which are known to simulate in  $O(T^2 \ln T)$  steps any Turing machine running in  $T$  steps [22].

**Skipping Cyclic Tag systems** A *skipping cyclic tag system* consists of a set of  $n$  productions  $p_0, \dots, p_{n-1} \in \{\emptyset, 1\}^*$  and an initial word  $u^0 \in \{\emptyset, 1\}^*$ . At each time step, the tag system cycles through the productions and decides to append the current production or not according to the letter read. We denote by  $u^t$  the word at time  $t$ . Formally, at time  $t = 0$ , its pointer  $q^0$  is set to 0. At all time  $t$ ,

**Halting step:** If  $u^t$  is the empty word  $\epsilon$ , then the tag system halts and outputs  $q^t$ ; otherwise

**Deletion step:** If the first letter  $u_0^t$  of  $u^t$  is  $\emptyset$ , then set  $q^{t+1} := (q^t + 1) \bmod n$  and  $u^{t+1} := u_1^t \cdots u_{|u^t|-1}^t$ , the suffix of  $u^t$  without its first letter; finally

**Appending step:** if  $u_0^t = 1$ , then the tag system appends the next production to  $u^t$  and skips to the following production, i.e.:  $u^{t+1} := u_1^t \cdots u_{|u^t|-1}^t \cdot p_{(q^t+1 \bmod n)}$  and set  $q^{t+1} := (q^t + 2) \bmod n$ .

For instance, the skipping tag system  $p = \langle 11\emptyset, \epsilon, 11, \emptyset \rangle$  has the following execution  $(\overset{[q^t]}{u^t})_t$  on input word  $u^0 = \emptyset 1\emptyset$ :

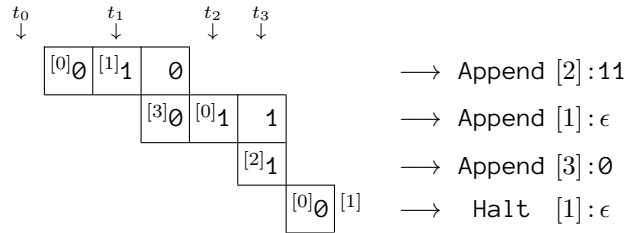
$${}^{[0]}\emptyset 1\emptyset \rightarrow {}^{[1]}1\emptyset \xrightarrow[\text{[2]:11}]{\text{Append}} {}^{[3]}\emptyset 11 \rightarrow {}^{[0]}11 \xrightarrow[\text{[1]:}\epsilon]{\text{Append}} {}^{[2]}1 \xrightarrow[\text{[3]:}\emptyset]{\text{Append}} {}^{[0]}\emptyset \rightarrow {}^{[1]}\text{Halt}$$

and outputs thus production  $p_1 = \epsilon$ .

**Annotated trimmed space-time diagram.** Given a SCTS  $(p_0, \dots, p_{n-1}; u^0)$ , we denote by  $0 \leq t_1 < t_2 < \dots$  all the times  $t$  such that the word  $u^t$  starts with letter 1 and set  $t_0 = -1$  by convention. Let us now compress the deletion steps occurring during steps  $t_i + 1$  and  $t_{i+1} - 1$  by simply indicating in exponent the production index for each deleted letter:

$${}^{[0]}\emptyset \overset{[1]}{1}\emptyset \xrightarrow[\text{[2]:11}]{\text{Append}} {}^{[3]}\emptyset \overset{[0]}{1}1 \xrightarrow[\text{[1]:}\epsilon]{\text{Append}} {}^{[2]}1 \xrightarrow[\text{[3]:}\emptyset]{\text{Append}} {}^{[0]}\emptyset \overset{[1]}{1} \text{Halt}$$

and align the resulting words in a 2D diagram according to their common parts:



we obtain the *annotated trimmed space-time diagram* for the SCTS  $(p, \emptyset 1\emptyset)$ . The following lemma gives a formal definition:

► **Lemma 2.** *The annotated word on row  $i$  (indexed from  $i = 0$ ) of the annotated trimmed space-time diagram is: (the production indices in exponent are computed modulo  $n$ )*

- **if  $u^{1+t_i} = \emptyset^r 1 \cdot s$  for some  $r \geq 0$  and  $s \in \{\emptyset, 1\}^*$ :** then,  $r = t_{i+1} - t_i - 1$  and the annotated word on row  $i$  is  ${}^{[i+1+t_i]}\emptyset \dots {}^{[i-1+t_{i+1}]}\emptyset {}^{[i+t_{i+1}]}\emptyset 1 \cdot s$  whose first letter is placed in column  $t_i + 1$  (where the leftmost column is indexed by 0);
- **if  $u^{1+t_i} = \emptyset^r$  for some  $r > 0$ :** then, row  $i$  is the last row of the diagram and its annotated word is  ${}^{[i+1+t_i]}\emptyset \dots {}^{[i+t_i+r]}\emptyset {}^{[i+t_i+r+1]}\emptyset$  and starts at column  $t_i + 1$ .

**Proof.** Observe that  $q^{t_i} = i + t_i \bmod n$  as exactly  $t_i$  letters have been read and  $i$  appending steps occurred before reading the  $i$ -th 1. ◀

## XX:6 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

Finally, we use a result by Cook [4], and Neary and Woods [22, 14] to show that simulating a skipping cyclic tag system is sufficient to simulate universal Turing machines efficiently:

► **Lemma 3.** *Let  $M$  be a Turing machine running in time  $t$ . There is a skipping cyclic tag system  $\mathcal{S}$  simulating  $M$  in  $O(t^2 \log t)$  steps. Moreover, the number of productions of  $\mathcal{S}$  is a multiple of 4.*

**Proof.** The original cyclic tag system by Cook [4] differs from the skipping cyclic tag system only in that in the original, the list rotates by 1 no matter which letter the current word begins with. By a classical result about tag systems, 2-tag system with  $m$  productions (i.e. over an  $m$  letter alphabet) can be simulated by a cyclic tag system with  $2m$  productions: that simulation works by encoding the  $m$  letters as  $10^{m-1}, 010^{m-2}, 0^210^{m-3}, \dots, 0^{m-1}1$ , respectively. We can in turn simulate a cyclic tag system with  $n$  productions  $p_0, p_1, \dots, p_{n-1}$ , starting from an input  $u$ , by a skipping cyclic tag system with  $2n$  productions  $\varepsilon, f(p_0), \varepsilon, f(p_1), \varepsilon, \dots, f(p_{n-1})$ , starting from the word  $f(u)$ , where  $f$  is the automorphism over  $\{0,1\}^*$  defined as  $f(0) = 00$  and  $f(1) = 1$ .

Finally, the result by Neary and Woods [15, 22] on cyclic tag systems implies that 2-tag systems can simulate  $t$  steps of a Turing machines in  $O(t^2 \log t)$ . ◀

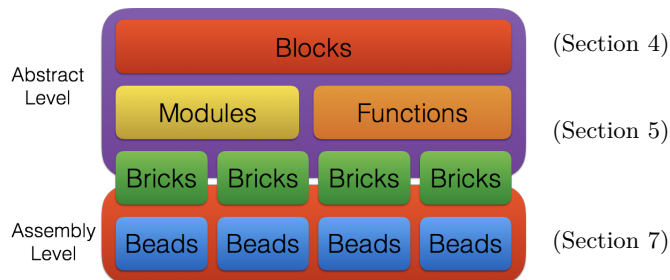
### 3 Proof structure

Figure 1 shows the global structure of our construction: at the abstract level (Section 4), we will show how the geometrical arrangement of big *blocks* (regions of the plane) simulates a tag system. The construction then becomes local: we only have to construct a molecule that correctly folds into the blocks, and interacts with neighbouring blocks as planned.

In Section 5, we will introduce *modules* (parts of the sequence), functions (possible conformations of a module in response to the environment in which it is folded), and *bricks* (partial conformations contained inside a block). Finally, in Section 7, we will show the actual sequences and attraction rule that implement all bricks, and show that our choice of geometric parameters guarantees that important parts of the sequence always fold in the same environments, in all possible conformations and inputs.

Section 8 gives a proof that the bricks actually implement the blocks, and shows how we verified the assembly level using a program on a specific tag system that exhibited all possible interactions between modules. This last step of our proof will follow the following steps:

1. Enumerate all the surroundings for each brick of each module
2. Enumerate all possible modules following the module
3. Generate automatically human-readable certificate of the correctness of the folding for each possibility, in the form of *proof trees*.
4. In the few cases where the surrounding may vary, prove that it has no incidence in the folding of the brick.



■ **Figure 1** Programming framework.

## 4 Simulating Tag Systems with Blocks

This section presents the simulation logic and the global geometry, *without any folding*. The simulation is decomposed into blocks, which are regions of the plane.

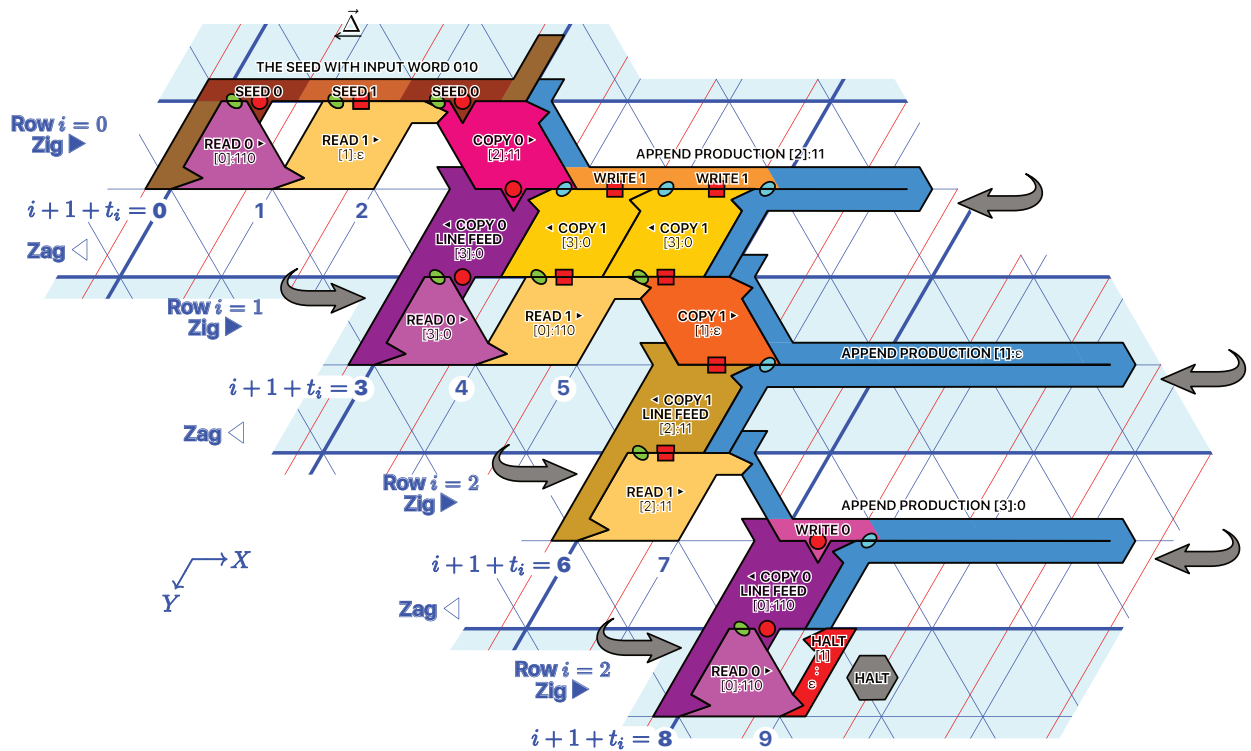
In the simulation, these blocks only interact at their border. In Section 5, we will show how to implement each block, and interactions between blocks, using actual folding. The names of blocks, introduced here, will be the same between the different parts of the paper.

### 4.1 The Blocks

#### 4.1.1 Overview

Our simulation proceeds by mimicking the annotated trimmed space-time diagram of the skipping cyclic tag system to be simulated. The folding walks to the South, i.e. each new step is below the previous step.

At each step, the simulation starts in a general movement from left to right. Leading zeros are trimmed off, and the simulation halts if the remaining word is empty. If the remaining word is not empty, there is at least one 1 in the word. The simulation removes the leading 1, skips over (and copies) the rest of the word, and appends the relevant production at the end of the word. Finally, the sequence is folded again in the opposite direction (i.e. right to left), and copies the computed word for the next step to start. See Figure 2 for an example.



■ **Figure 2** Execution of the block automaton simulating the SCTS ( $p = \langle \epsilon, 100, 1, 0 \rangle; u^0 = 010$ ). Every other row is shaded in blue. Each row encodes one step of the tag system, and each row is divided into a “zig part” (on top) and a zag part (on the bottom). The word at the end of each step can be read at the bottom of the zag part for the corresponding row. In this figure, 1 is pictured as a flat border with a red rectangle and 0 is pictured as a red spike with a red circle. Green and cyan ovals mark the presence of a letter, on zig and zag rows respectively.

### 4.1.2 Types of blocks

There are 10 different types of blocks, shown in Figure 31. All of them are used in our example in Figure 2:

- Seed encodes the initial word into a conformation beads where 0s are represented as red-dot spikes and 1s as red rectangles.
- Read0▶, Read1▶, Copy0▶, Copy1▶ are responsible for reading, and copying the letters of the current word during the zig phase (left to right). When clear from the context, we will sometimes refer to both Read0▶ and Read1▶ collectively as Read▶, and to both Copy0▶ and Copy1▶ as Copy▶.
- ◀Copy0, ◀Copy1 are responsible for copying the letters of the new word over to the next step, during the zag pass (right to left). When clear from the context, we will sometimes refer to both ◀Copy0 and ◀Copy1 collectively as ◀Copy.
- Append & CR⚡ and ◀CopyLineFeed are responsible for appending a production, carriage return and line-feed.
- Halt is the last block produced corresponding to the halt of the simulated tag system.

### 4.1.3 The block automaton

A visual depiction of the logic is shown in Figure 3 in the form of an automaton: starting from the seed block, blocks attach at the orange anchors (▶ and ◀) one next to the other as described by the block automaton in Figure 3. Each block is labelled with the current production index of the tag system which determines the production to be appended. An example of execution of the block automaton for the SCTS ( $p = \langle 110, \epsilon, 11, 0 \rangle; u^0 = 010$ ) is illustrated in Figure 2.

### 4.1.4 Simulating tag systems with blocks

Let  $\mathcal{S} = (p_0, p_1 \dots p_{n-1}; u^0)$  be a skipping cyclic tag system, and for all integer  $i \geq 0$ , let  $t_i$  be the  $i^{\text{th}}$  step where  $u^t$  starts with 1 (starting from 0, i.e.  $t_0$  is the first step where  $u^{t_0}$  starts with 1). The following lemma describes the encoding of  $\mathcal{S}$  into blocks (i.e. generalises Figure 2 to arbitrary tag systems).

► **Lemma 4.** *The (possibly infinite) final block configuration consists of: (see illustration on Figure 6)*

- The seed row consists of the block Seed( $u^0$ ) anchored at its end point at coordinates  $(0, 1)$ .
- For  $i \geq 0$ , the  $i$ -th row consists of a zig row anchored at height  $Y = 2i$ , and a zag row anchored at height  $Y = 2i + 1$  defined as follows:
  - (Compute) if  $u^{1+t_i} = 0^r 1 \cdot s$  and if  $s$  and  $p_{1+i+t_{i+1}}$  are not both  $\epsilon$ : then  $r = t_{i+1} - t_i - 1$  and, as illustrated in Figure 4(b) and Figure 4(c):

- the  $i$ -th zig-row, growing from left to right, contains the sequence of annotated blocks located at the following coordinates (with respect to their anchor point, shown in Figure 31):

Y	2i + 1				2i			
X	$i + 1 + t_i$	⋯	$i + r + t_i$	$i + t_{i+1}$	$i + 1 + t_{i+1}$	⋯	$i +  s  + t_{i+1}$	$i +  s  + 1 + t_{i+1}$
Blocks	Read0▶	⋯	Read0▶	Read1▶	Copy( $s_0$ )▶	⋯	Copy( $s_{ s -1}$ )▶	Append & CR⚡( $p_{1+i+t_{i+1}}$ )
[q]	$[i + 1 + t_i]$	⋯	$[i + r + t_i]$	$[i + t_{i+1}]$	$[i + 1 + t_{i+1}]$	⋯	$[i + 1 + t_{i+1}]$	$[i + 1 + t_{i+1}]$

- the  $i$ -th zag-row consists from left to right of the sequence of blocks located at the following coordinates (with respect to their anchor point, see Figure 31):

Y	2i + 1			
X	$i + 3 + t_{i+1} - \Delta$	$i + 4 + t_{i+1} - \Delta$	⋯	$i + 2 +  v  + t_{i+1} - \Delta$
Blocks	◀CopyLineFeed( $v_0$ )	◀Copy( $v_1$ )	⋯	◀Copy( $v_{ v -1}$ )
[q]	$[i + 2 + t_{i+1}]$	$[i + 2 + t_{i+1}]$	⋯	$[i + 2 + t_{i+1}]$

where  $v = u^{1+t_{i+1}} = s \cdot p_{1+i+t_{i+1}} \neq \epsilon$  (as  $s$  and  $p_{1+i+t_{i+1}}$  are not both  $\epsilon$ ).

- (Halt 1) if  $u^{1+t_i} = 0^r 1$  and  $p_{1+i+t_{i+1}} = \epsilon$ : then  $r = t_{i+1} - t_i - 1$  and the last rows of the block configuration consist from left to right in the sequence of annotated blocks located at the following

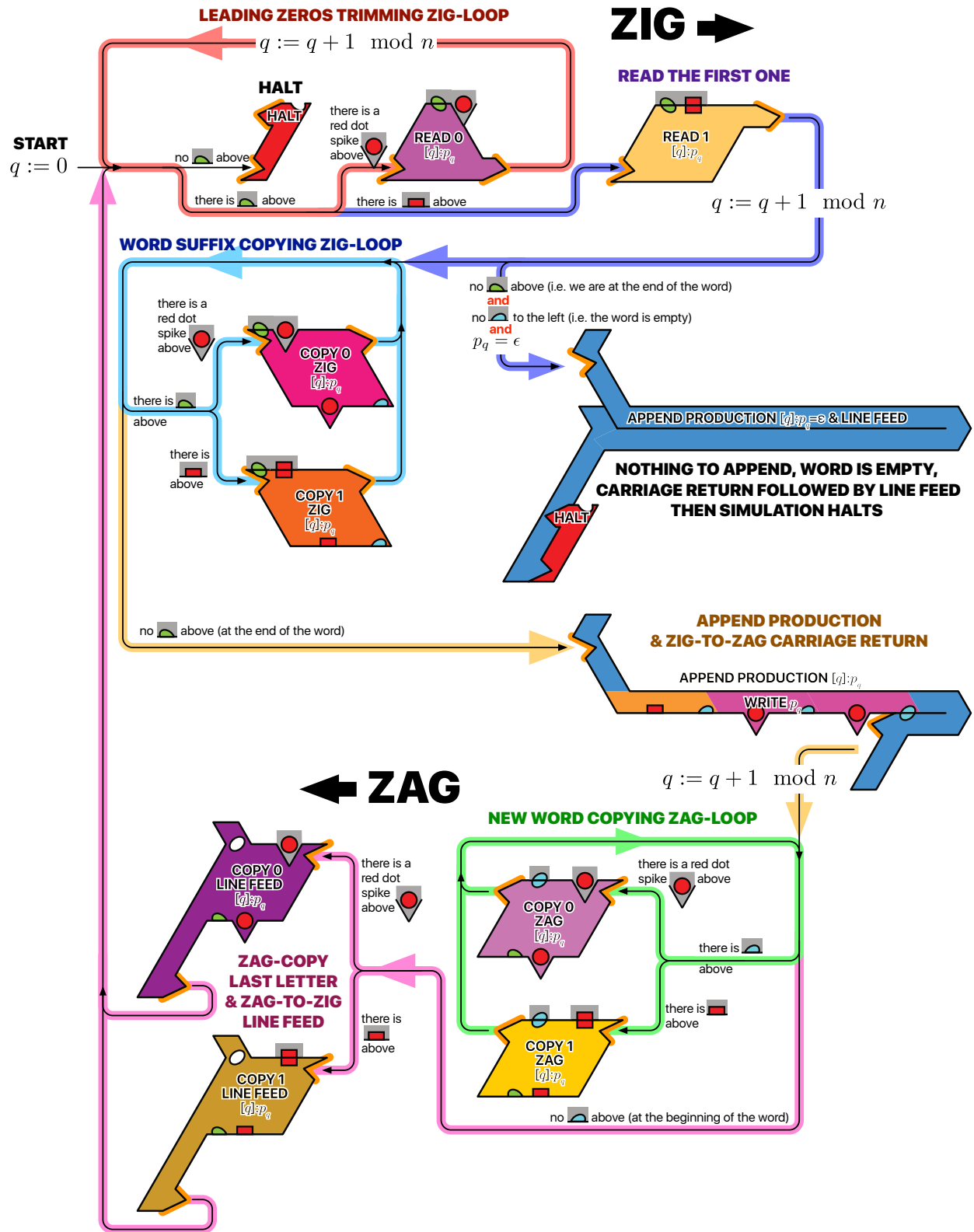


Figure 3 The block automaton.

coordinates, as illustrated in Figure 4(d):

	2i + 1			2i		2i + 3
Y						
X	i + 1 + t <sub>i</sub>	⋯	i + r + t <sub>i</sub>	i + t <sub>i+1</sub>	i + 1 + t <sub>i+1</sub>	i + 2 + t <sub>i+1</sub>
Blocks	Read0▶	⋯	Read0▶	Read1▶	CarriageReturn&LineFeed	Halt
[q]	[i + 1 + t <sub>i</sub> ]	⋯	[i + r + t <sub>i</sub> ]	[i + t <sub>i+1</sub> ]	[i + 1 + t <sub>i+1</sub> ]	[i + 2 + t <sub>i+1</sub> ]

- **finally, (Halt 2) if  $u^{1+t_i} = 0^r$  for some  $r \geq 0$ : then the  $i$ -th zig-row is last row of the block con-figuration and consists of the sequence of annotated blocks located at the following coordinates, as illustrated in Figure 4(e):**

	2i + 1			
Y				
X	i + 1 + t <sub>i</sub>	⋯	i + r + t <sub>i</sub>	i + r + 1 + t <sub>i</sub>
Blocks	Read0▶	⋯	Read0▶	Halt
[q]	[i + 1 + t <sub>i</sub> ]	⋯	[i + r + t <sub>i</sub> ]	[i + r + 1 + t <sub>i</sub> ]

**Proof.** This follows from an easy induction on the number of rows. The induction hypothesis at step  $i$  is that the word encoded by the blocks at the bottom of the zag-row  $(i - 1)$  is  $u^{1+t_i}$  and the production index in this zag row is  $q^{1+t_i} = 1 + i + t_i \pmod n$ .

First, since we choose the conformation of the seed, we choose the encoding of the initial word in the tag system. Then, showing that if the induction hypothesis holds at step  $i$ , it also holds at step  $i + 1$ , follows from the case enumeration in Figure 6 and the block automaton in Figure 3. ◀

## 4.2 General geometry of the Blocks

The precise geometry of each block is given by the figures 5 and 32–39. We begin by introducing a number of parameters we will use to align bricks properly<sup>3</sup> for all possible tag systems and inputs.

We first define the *write position* of a block the position on its border where its value is “written”, i.e. as either a spike (0, red circles on Figure 4) or a dent (1, red squares on Figure 4). Similarly, *read positions* are positions where the shape of the folding depends on whether there is a spike or a dent on the adjacent block. See Figure 5 for an illustration.

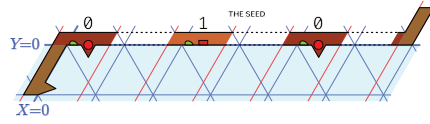
Starting from a skipping cyclic tag system  $\mathcal{S}$ , we first build a tag system  $\mathcal{T}$  by turning  $\mathcal{S}$  into a skipping cyclic tag system such that  $n$ , the number of productions of  $\mathcal{T}$ , is a multiple of 4, and moreover  $n \geq 8$ . We build  $\mathcal{T}$  by duplicating all the productions of  $\mathcal{S}$  and all the 0s in all productions of  $\mathcal{S}$ , until  $4|n$  and  $n \geq 8$ .

- $n$  is the number of productions in  $\mathcal{T}$ , hence  $n$  is a multiple of 4, and  $n \geq 8$ .
- $L$  is the length of the longest production in  $\mathcal{T}$ .
- $P$  is the length of an extra padding on each production. We let  $P = 11 + (L \pmod 2)$ , hence  $L + P$  is even.
- $w$  is an atomic width we need to define other constants  $W$  and  $h$ . For now, let  $w = 6(L + P) + 18$ . We will later use the fact that  $w \pmod{12} = 6$ .
- $W$  is the width of the Copy▶ and ◀Copy blocks.  
Let  $W = n \cdot (w + 6)$ . We will later use the fact that  $W \pmod{48} = 0$  (because  $n$  is a multiple of 4 and  $w + 6$  is a multiple of 12).
- $h$  is the height of the Read▶, Copy▶ and ◀Copy blocks, not counting the small bumps.  
Let  $h = W - (w + 3)$ . Note that  $h \pmod{12} = 3$ .

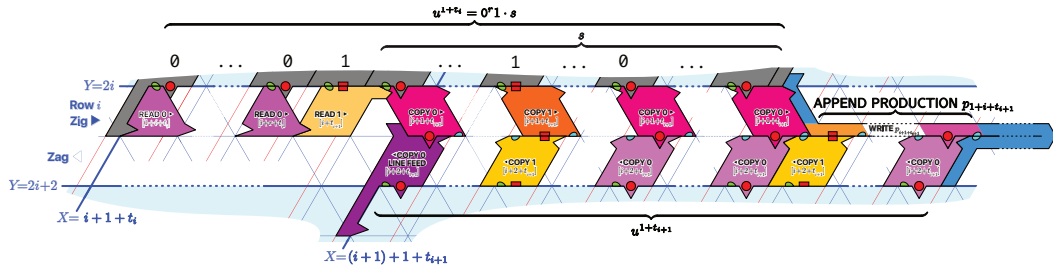
We can now translate Lemma 4, to give blocks their actual coordinates in the simulation:

<sup>3</sup> Here, we understand “align” both as “align in the plane” and “adjust the length of sequences to match modulo common parameters”.

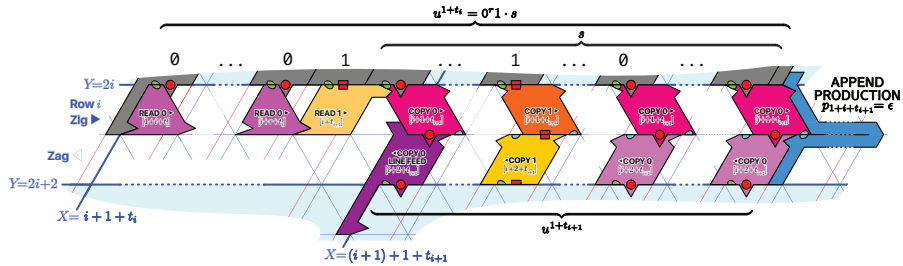




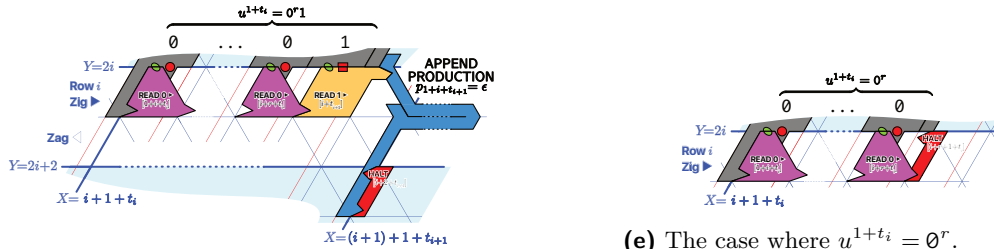
(a) The seed row anchored at coordinates  $(0, 0)$ .



(b) The case where  $u^{1+t_i} = 0^r 1 \cdot s$  and the production to be appended is  $p_{1+i+t_{i+1}} \neq \epsilon$ .



(c) The case where  $u^{1+t_i} = 0^r 1 \cdot s$  with  $s \neq \epsilon$  and the production to be appended is  $p_{1+i+t_{i+1}} = \epsilon$ .



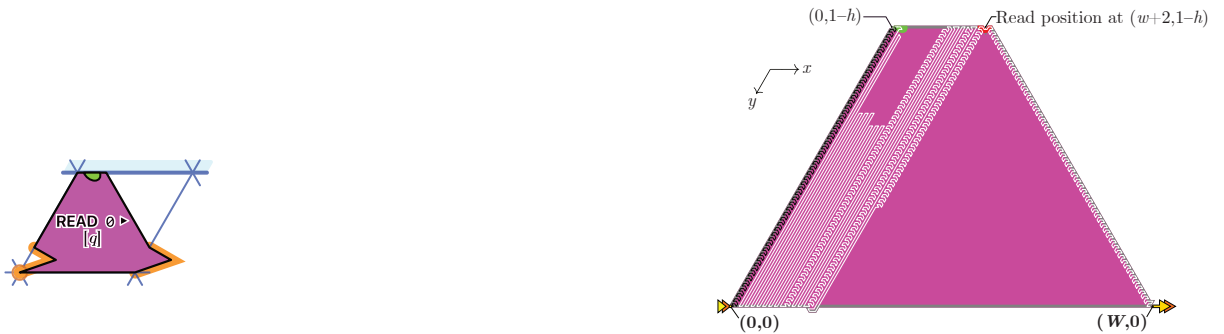
(d) The case where  $u^{1+t_i} = 0^r 1$  and the production to be appended is  $p_{1+i+t_{i+1}} = \epsilon$ .

(e) The case where  $u^{1+t_i} = 0^r$ .

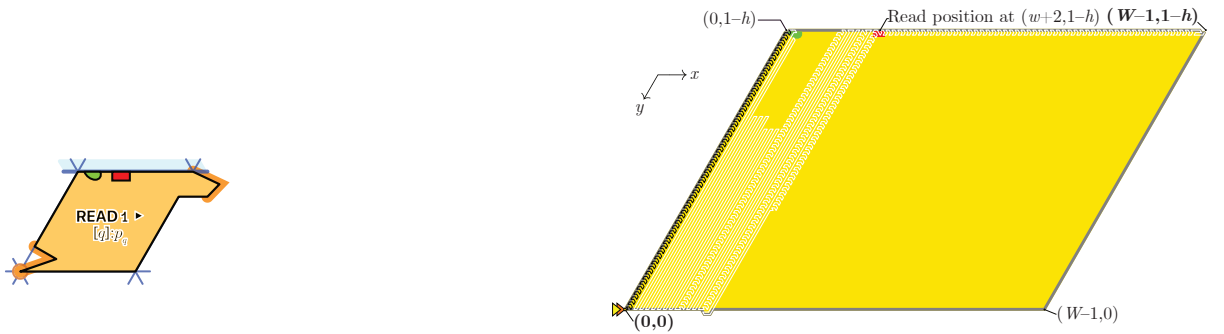
**Figure 4** The  $i$ th row of the final block configuration (the previous and next rows are shaded in blue). Production index in the label are computed modulo  $n$ . Observe that the Read and Copy in the  $i$ -th zig row correspond readily the  $i$ -th line in the annotated trimmed space-time diagram of the simulated SCTS.



■ **Figure 5 Geometry of the Read► blocks.** Note that the internal structures (the lines in white) of both blocks Read0► and Read1► agree until position  $(w + 2, -h + 1)$  where the presence or absence of a spike, encoding a  $\emptyset$ , at the bottom of the row above forces the block to adopt the shape Read0► or Read1► respectively.



(a) The Read0► block has the shape of a trapezium whose bottom basis has length  $W$  and top basis has length  $w + 5$ , with height  $h$ . It has a dent (an empty position) located at  $(w + 2, -h + 1)$  (w.r.t. to its origin at the bottom left corner), in which plugs the spike of the block from the row above it, encoding the letter  $\emptyset$ . The next block will start folding at the bottom right corner, at  $(W, 0)$ .



(b) The Read1► block has the shape of a parallelogram with horizontal side length  $W$  and vertical side length  $h$ . The red rectangle area at position  $(w + 2, -h + 1)$  (w.r.t. its origin at the bottom left corner) aligns with the flat bottom block above encoding the letter 1 (as opposed to a spiked-block encoding a  $\emptyset$ ). The next block will start folding at the top right corner, at  $(W - 1, -h + 1)$ .

► **Lemma 5.** *The (possibly infinite) final block configuration consists of: (see illustration on Figure 6)*

- *The seed row, i.e. the block  $\text{Seed}\langle u^0 \rangle$  ending at coordinates  $(-1, h)$ .*
- *For  $i \geq 0$ , the  $i$ -th row consists of a zig row located between  $y = 2ih + 1$  and  $y = (2i + 1)h$ , and a zag row located between  $y = (2i + 1)h + 1$  and  $y = 2(i + 1)$ , defined as follows:*

- **(Compute)** *if  $u^{1+t_i} = \emptyset^r 1 \cdot s$  and if  $s$  and  $p_{1+i+t_{i+1}}$  are not both  $\epsilon$ : then  $r = t_{i+1} - t_i - 1$  and, as illustrated in Figure 6(a) and Figure 6(b):*

- *the  $i$ -th zig-row consists from left to right of the sequence of geometrical blocks whose origin is located at the following coordinates:*

$\swarrow y$	$(2i + 1)h$			$2ih + 1$				
$\rightarrow x$	$ih + (1 + t_i)W$	$\cdots$	$ih + (r + t_i)W$	$ih + t_{i+1}W$	$ih + (1 + t_{i+1})W - 1$	$\cdots$	$ih + ( s  + t_{i+1})W - 1$	$ih + (1 +  s  + t_{i+1})W - 1$
Blocks	Read0►	$\cdots$	Read0►	Read1►	Copy( $s_0$ )►	$\cdots$	Copy( $s_{ s -1}$ )►	Append & CR◄( $p_{1+i+t_{i+1}}$ )

*This row ends at position  $((i + 1)h + (1 + |s| + |p_{1+i+t_{i+1}}| + t_{i+1})W - 7, (2i + 1)h + 1)$ .*

- *the  $i$ -th zag-row consists from left to right of the sequence of geometrical blocks whose origins are located at the following coordinates:*

$\swarrow y$	$(2i + 1)h + 1$			
$\rightarrow x$	$(i + 1)h + (2 + t_{i+1})W - 8$	$(i + 1)h + (3 + t_{i+1})W - 8$	$\cdots$	$(i + 1)h + (1 +  v  + t_{i+1})W - 8$
Blocks	◄CopyLineFeed( $v_0$ )	◄Copy( $v_1$ )	$\cdots$	◄Copy( $v_{ v -1}$ )

*where  $v = u^{1+t_{i+1}} = s \cdot p_{i+1+t_{i+1}} \neq \epsilon$  (as  $s$  and  $p_{i+1+t_{i+1}}$  are not both  $\epsilon$ ). This row ends at position  $((i + 1)h + (1 + t_{i+1})W - 1, (2(i + 1) + 1)h)$ .*

- **(Halt 1)** *if  $u^{1+t_i} = \emptyset^r 1$  and  $p_{1+i+t_{i+1}} = \epsilon$ : then  $r = t_{i+1} - t_i - 1$  and the last rows of the geometrical block configuration consist from left to right of the sequence of geometrical blocks located at the following coordinates, as illustrated in Figure 6(c):*

$\swarrow y$	$(2i + 1)h$			$2ih + 1$		$(2i + 3)h$	
$\rightarrow x$	$ih + (1 + t_i)W$	$\cdots$	$ih + (r + t_i)W$	$ih + t_{i+1}W$	$ih + (1 + t_{i+1})W - 1$	$(i + 1)h + (1 + t_{i+1})W$	
Blocks	Read0►	$\cdots$	Read0►	Read1►	CarriageReturn & LineFeed	Halt	

- **finally, (Halt 2)** *if  $u^{1+t_i} = \emptyset^r$  for some  $r \geq 0$ : then the  $i$ -th zig-row is last row of the geometrical block configuration and consists of the sequence of geometrical blocks located at the following coordinates, as illustrated in Figure 6(d):*

$\swarrow y$	$(2i + 1)h$			
$\rightarrow x$	$ih + (1 + t_i)W$	$\cdots$	$ih + (r + t_i)W$	$ih + (1 + r + t_i)W$
Blocks	Read0►	$\cdots$	Read0►	Halt

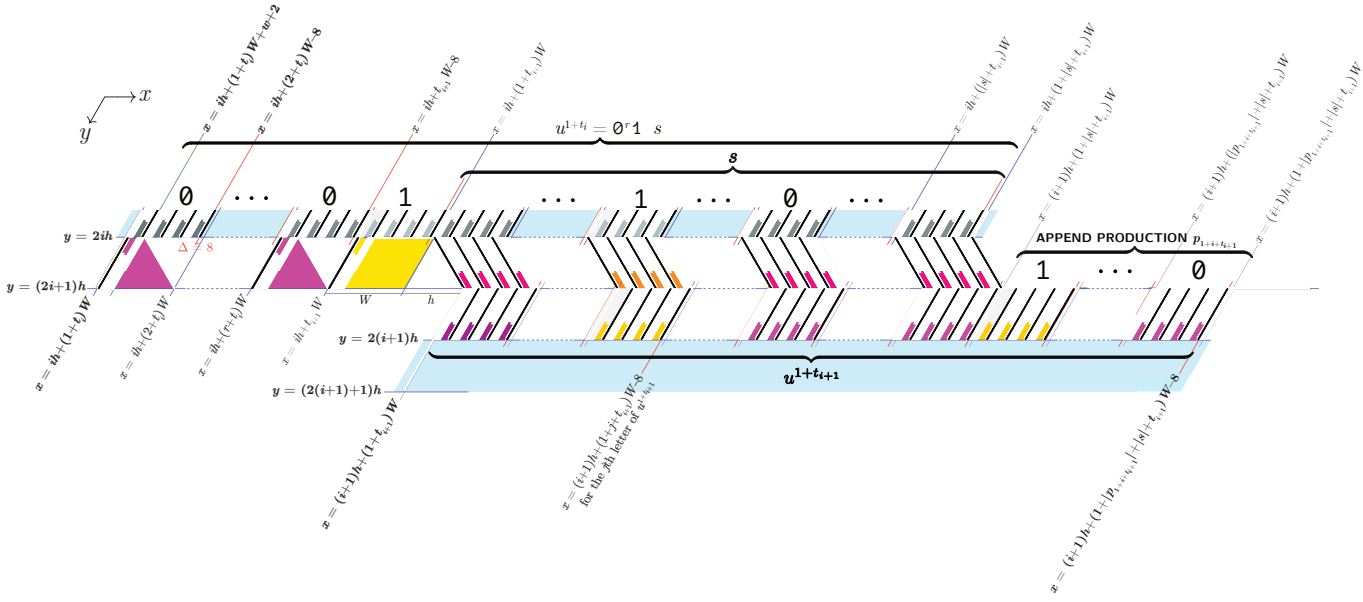
*Hence, the read positions and write positions of blocks in consecutive rows are adjacent.*

**Proof.** We map each block from Lemma 4 to its actual position, using the following table to compute the space taken by each block:

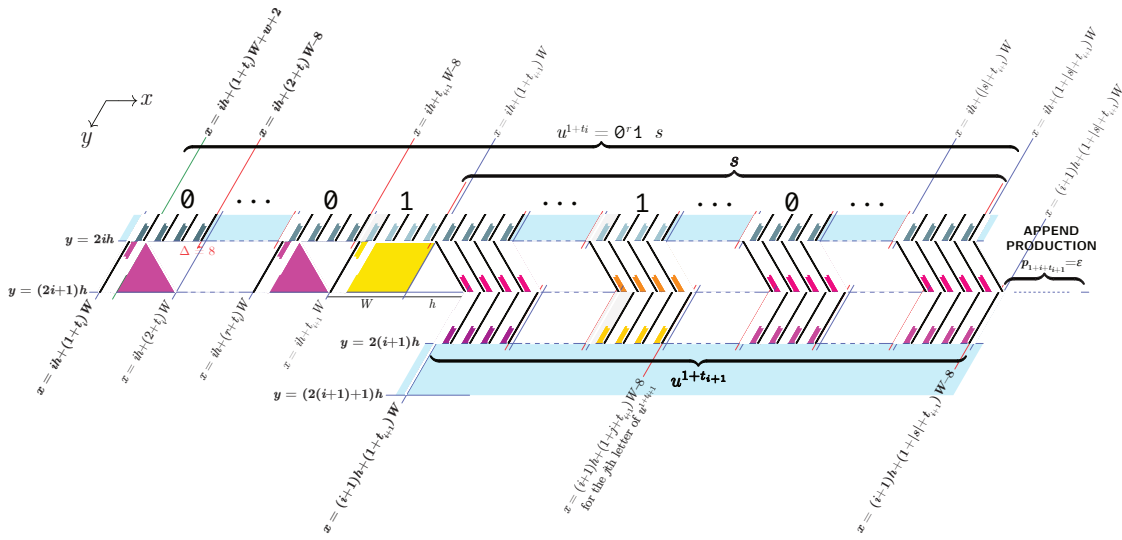
Block	$\Delta x$	$\Delta y$
Read0►	$W$	$0$
Read1►	$W - 1$	$1 - h$
Copy0► and Copy1►	$W$	$0$
Append & CR◄( $u$ )	$ u  \cdot W + h - 7$	$h$
◄Copy0 and ◄Copy1	$-W$	$0$
◄CopyLineFeed0 and ◄CopyLineFeed1	$-W + 8$	$2h - 1$

► **Corollary 6.** *The geometrical blocks simulate the associated skipping cyclic tag system.*

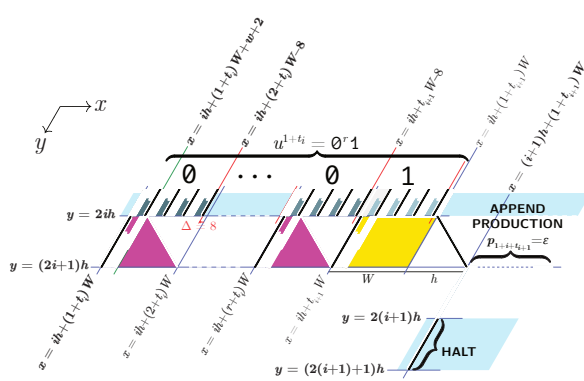
**XX:14 Proving the Turing Universality of Oritatami Co-Transcriptional Folding**



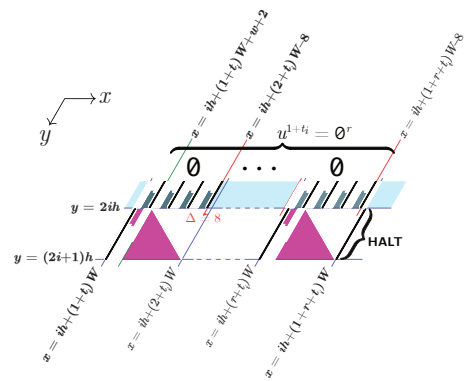
(a) The case where  $u^{1+t_i} = 0^r 1 \cdot s$  and the production to be appended is  $p_{1+i+t_{i+1}} \neq \epsilon$ .



(b) The case where  $u^{1+t_i} = 0^r 1 \cdot s$  with  $s \neq \epsilon$  and the production to be appended is  $p_{1+i+t_{i+1}} = \epsilon$ .



(c) The case where  $u^{1+t_i} = 0^r 1$  and the production to be appended is  $p_{1+i+t_{i+1}} = \epsilon$ .



(d) The case where  $u^{1+t_i} = 0^r$ .

**Figure 6** The  $i$ th row of the final geometrical block configuration (the previous and next rows are shaded in blue). Production index in the label are computed modulo  $n$ . Observe that the Read and Copy in the  $i$ -th zig row correspond readily the  $i$ -th line in the annotated trimmed space-time diagram of the simulated SCTS.

### 4.3 Production segments: encoding the production index

The primary structure we use to simulate a skipping cyclic tag system with  $n$  productions  $p_0, p_1, \dots, p_{n-1}$ , is a periodic sequence of  $n$  strings of beads of equal length called *production segments*  $\llbracket p_0 \rrbracket, \dots, \llbracket p_{n-1} \rrbracket$ , where for all  $i$ ,  $\llbracket p_i \rrbracket$  encodes production  $p_i$ . The first module of production segments is written as black lines on the figures in Section B.

The primary sequence of the oritatami system corresponding to the skipping cyclic tag system with productions  $\langle p_0, \dots, p_{n-1} \rangle$  is the infinite sequence with period  $\llbracket p_0 \rrbracket \cdot \llbracket p_1 \rrbracket \cdots \llbracket p_{n-1} \rrbracket$ .

Each block is the result of folding a number of production segments (depending on the block type):

- **Read** and **Append & CR**( $u$ ) take one production segment each,
- **Hal t** stops before folding one full production segment,
- all other blocks take  $n$  production segments each.

We call the *internal state* of a block  $B$  the production index  $q$  of the first (and possibly only, for **Read**, **Append & CR**( $u$ ) and **Hal t** blocks) production segment  $\llbracket p_q \rrbracket$  of  $B$ .

► **Lemma 7.** *At each step, the internal state of every block is equal to the state variable  $q$  in the block automaton in Figure 3.*

*Therefore, the block automaton simulates exactly the SCTS.*

**Proof.** In the construction of our blocks, the internal state is increased by one (modulo  $n$ ) each time a block consisting of one production segment is folded (**Read** or **Append & CR**), and is unchanged (modulo  $n$ ) otherwise (**Copy**, **◀Copy** or **◀CopyLineFeed**). The case of **Hal t**, which stops the entire simulation, is special.

This is exactly the same as in the block automaton (Figure 3).

Moreover, the zag phase contains only blocks of  $n$  productions segments (i.e. of width  $W$ ), hence does not change the internal state, again as in the block automaton. ◀

## 5 The Structure of the Sequence: the Modules and the Bricks

### 5.1 Modules

Each production segment is split into seven *modules* **A**,  $\dots$ , **G**, each serving one or several purposes:

**Module A** ( $3h - 2$  beads long) is the initial scaffold upon which the other modules fold.

**Module B** (5 beads long) is responsible for the detection of an empty tape word: if it is empty, it folds to the left and the molecule gets trapped in a closed space and the computation halts; otherwise, it folds to the right and the computation continues.

**Module C** ( $3h - 10$  beads long) is responsible for the detection of the end of the tape word to start appending to it the production word.

**Modules D<sub>0/1</sub>** ( $3W + 30$  beads long each) encodes each letter of the production word inside the production segment. It adopts two shapes: compact inside reading and copying blocks, or expanded in appending blocks.

**Module E<sub>a</sub>** ( $3W(L - a + P) + 8h - 1$  beads long) ensures by padding that all production segment have the same length (even if the production word have different length). It serves two other purposes: its presence indicates to **C** and **B** that the end of tape is not yet reached; and it accomplishes the carriage return initiating the Zag-phase once the current production word has been appended.

**Module F** ( $4h$  beads long) is the scaffold upon which **G** folds. It is specially designed to induce two very distinct shapes on **G** depending on the initial shift of **G**.

**Module G** ( $6h - 1$  beads long) is the real “brain” of the molecule. It implements three distinction functions which are triggered by its interaction with its environments: in the zig-up phase, it reads the current letter of the tape word, ignoring the 0s and moves to the zig-down phase when it reads a 1; it copies

## XX:16 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

the 0 and 1 in the zig-down and zag phases; it accomplishes the line feed when the molecule reaches the beginning of the tape word at the end of each zag-phase.

The bead-by-bead description of each of these sequences will be given in Section 7.

Each production segment  $[[p_i]]$  is split into a sequence of modules:  $[[p_i]] = \mathbf{A} \cdot \mathbf{B} \cdot \mathbf{C} \cdot \mathbf{D}_{(p_i)_0} \cdot \mathbf{D}_{(p_i)_1} \cdot \dots \cdot \mathbf{D}_{(p_i)_{|p_i|-1}} \cdot \mathbf{E}_{L=|p_i|} \cdot \mathbf{F} \cdot \mathbf{G}$  where  $\cdot$  denotes the concatenation of two bead sequences.

### 5.2 Bricks

► **Definition 8** (Brick). Each module adopts different conformations to accomplish each of its tasks. We call *brick* every conformation that a given module adopts when folded in a valid environment.

Figure 7 lists all the bricks that adopts the modules in our design and how they are organized inside each block. The exact geometry of the bricks will be given together with their beads sequence in Section 7. Bricks are the lowest design level we will consider in this article before going to the beads level. Figure 8 presents the brick automaton which details how the bricks articulates with eachother. The lemma below shows that if the modules folds into bricks according to this automaton, then our design simulates indeed the block automaton and thus the SCTS. We are left with proving that each module folds into the expected brick for every possible environment to complete the proof of our main theorem.

► **Lemma 9.** *Starting from a wellformed seed (see section 7), the brick automaton in Figure 8 simulates the Skipping Cyclic Tag System.*

**Proof sketch.** Starting from a wellformed seed, we prove by induction that the brick automaton implements precisely the block automaton which simulates the SCTS by Lemma 7. ◀

We are left with designing sequences implementing the bricks. We can forget about the simulation itself and focus on the local folding of each module in every possible environment.

## 6 Design Toolbox

In this section, we present several key tools to program Oritatami design and which we believe to be generic as they allowed us to get a lot of freedom in our design.

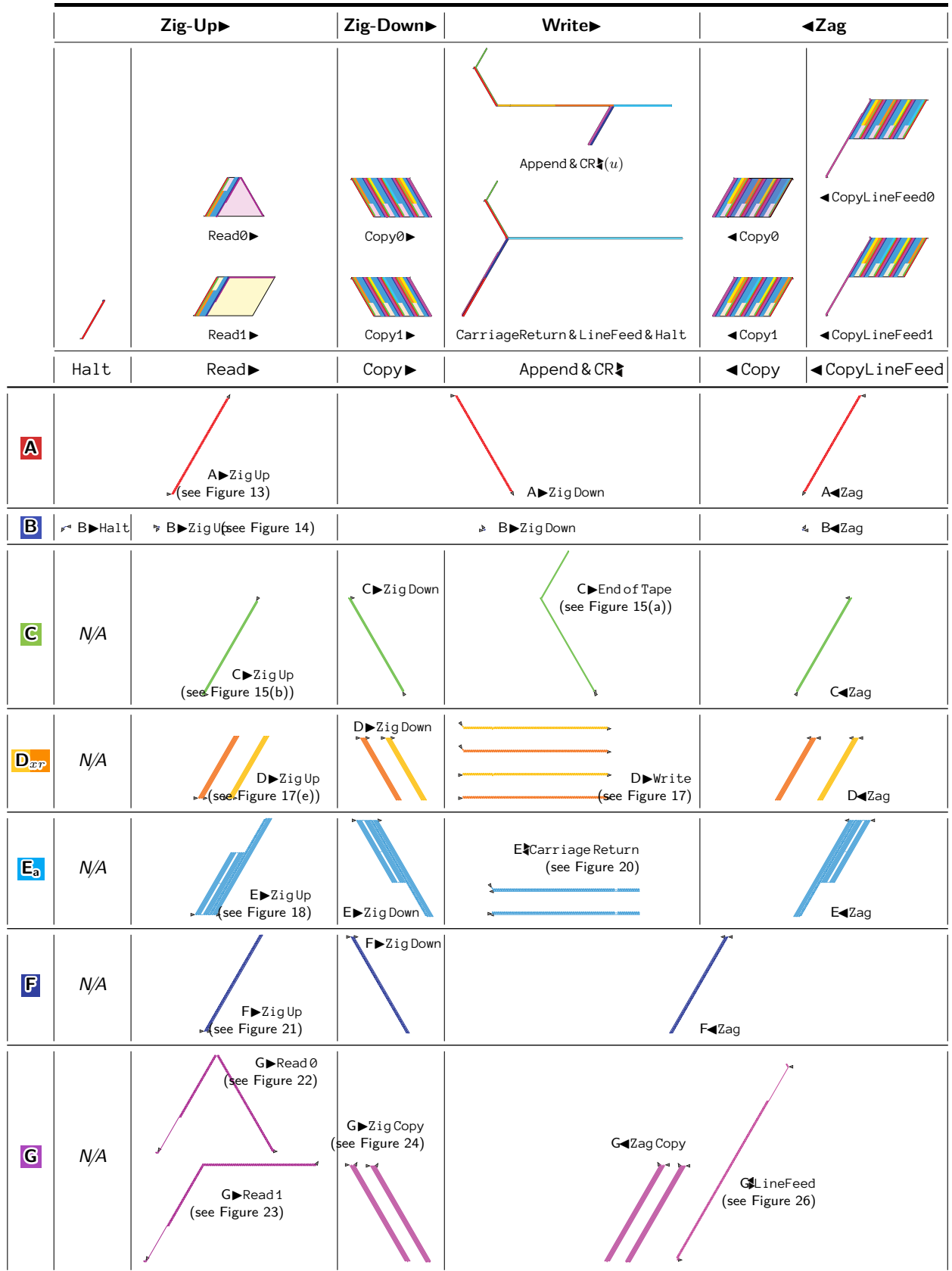
### 6.1 Expanding shapes: Glider and Switchback

In our design we need to store many letters in a very compact space inside the blocks Read►, Copy► and ◀Copy, and to expand each of them to the width of a block in the Append&CR◄ blocks. This is achieved using the glider/switchback device illustrated in Figure 9. The key in this design is that both shapes use a small enough number of bonds so that they don't interfere once the beginning of the molecule is folded in one way, it keeps folding that way. The design of modules **D**, **E** and **G** is based on this bonding pattern (see Section 7). This behavior is best observed in the proof-trees (see Section 8.2 or the companion website of this paper<sup>4</sup>).

### 6.2 Implementing the logic

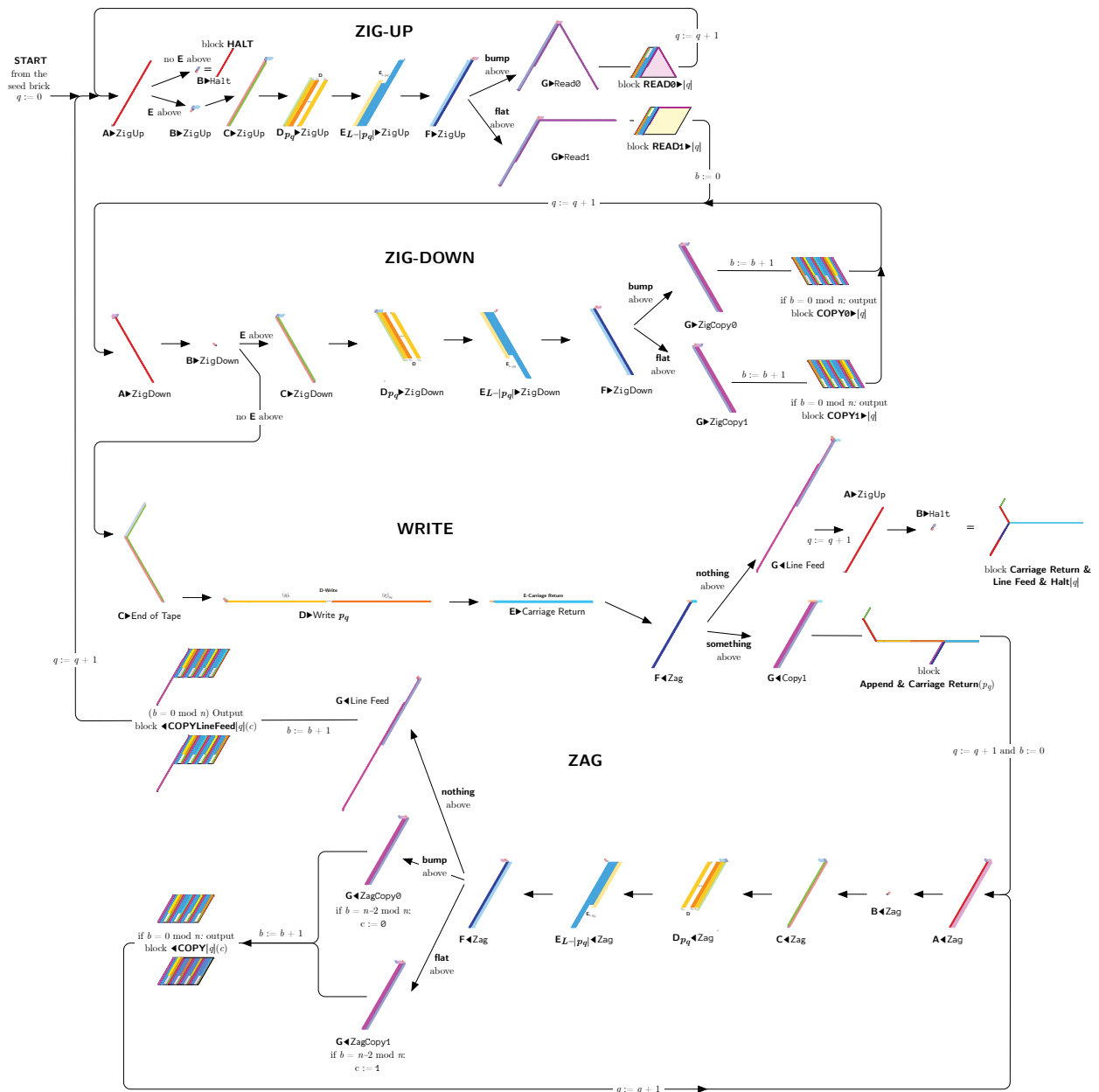
As in [11], the internal state of our “molecular computing machinery” consists essentially of two parameters: 1) the *position inside the primary structure* of the part currently folding; and 2) the *entry point* of molecule inside the environment. Indeed, depending on the entry point or the position inside the primary structure, different beads will be in contact with the environment and thus different “functions” will be applied as a

<sup>4</sup> <https://www.irif.fr/~nschaban/oritatami/prooftrees/>

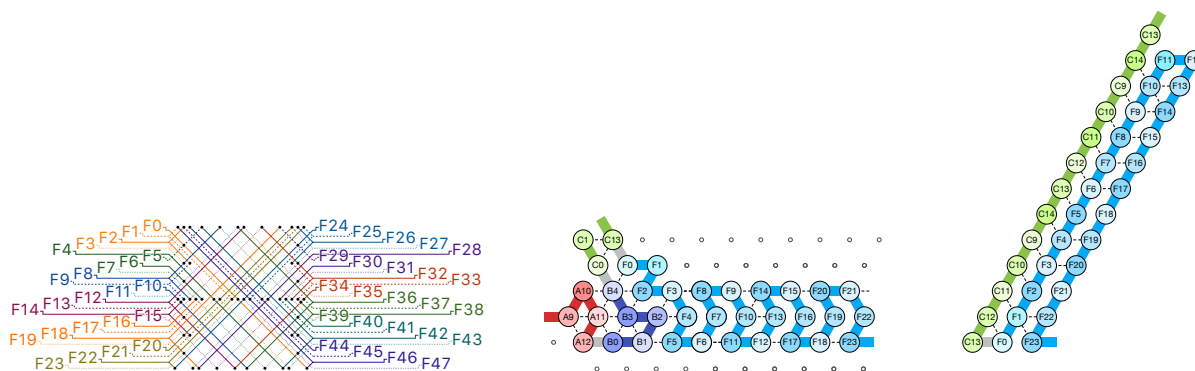


■ **Figure 7** The bricks inside each block.

XX:18 Proving the Turing Universality of Oritatami Co-Transcriptional Folding



■ **Figure 8 The brick automaton** implementing the block automaton. Note that in the Zig Down-phase, each letter of the word above is copied by the first module **G** of the Copy $\blacktriangleright$  block and the end of the word is detected by the first module **C** of the block. In the Zag-phase however, each letter of the word above is copied by the penultimate module **G** of the  $\blacktriangleleft$ Copy block and the beginning of the word is detected by the last **G** of the block. Note that this automaton is presented as a “transducer” producing the block diagram: the variables  $c$  and  $b$ , which counts up to  $n$ , are introduced only to output the right module at the right time during the zig-down and zag phases (assuming the seed is wellformed).



■ **Figure 9** The glider/switchback bond pattern. To the left: the ♥-rule for the F-beads. In the middle: the strong binding of the F-beads with the B-beads triangular shape imposes the glider shape. To the right: the strong bonding of the F-beads to the C-beads imposes the switchback pattern forever.

result of their interactions. Similarly, the memory of the system consists of the beads already placed in the area currently visited (the environment).

At different places, we need the molecule to read information from the environment and trigger the appropriate folding. This is obtained through different mechanisms.

**Default folding.** By default, during the zig-up phase, **B** is attracted to the left and folds to the right only in presence of **E** above. This allows to continue the folding only if the tape word is not empty or to halt it otherwise (see Figure 14).

**Geometry obstruction.** An typical example is illustrated by **G**. During the zig-up phase where the absence of environment below the block **Read**▶ allows **G** to fold downward at the beginning (see Figure 22) which shift the molecule by 7 beads along **F** resulting in **G** to adopt the glider-shape (more details on this mechanism in the next section). Whereas during the zig-down phase, **G** cannot make this loop because it is occupied by a previously placed **G**. This results in a perfect alignment of **G** with **F** whose strong attraction forces **G** to adopt the switchback shape.

**Geometry of the environment.** Figure 10 shows how the shape of the environment is used to change the direction of **G** in glider-shape. This results in modifying the entry point in the environment and allows the Oritatami system to trim the leading 0s in the tape word, switch from zip-up to zig-down phase when reading a 1 and from zag- to zig-up phase when it has rewind to the beginning of the tape word.

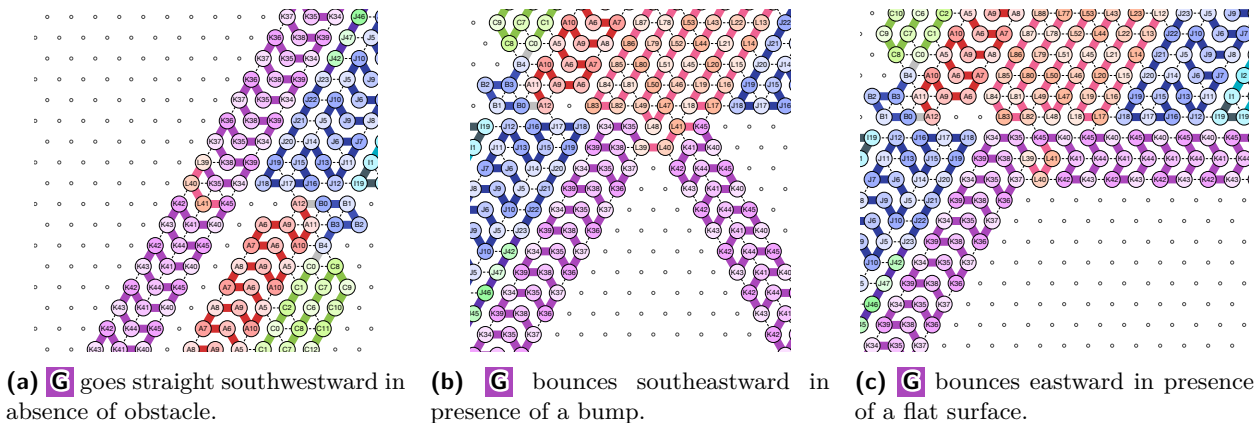
### 6.3 Easing the design: getting the freedom you need

Several key tools allowed to ease considerably our design, and even in some cases to make it feasible. These tools are generic enough to be considered as *programming paradigms*. One main difficulty we had to face is that the different functions one wants to implement tend to concentrate at the same “hot-spots” in the molecule. A typical example is the center of **G** which is the place where one wants to implement all the functions: Read, Copy, Line Feed. The following powerful tools allowed to overcome these difficulties:

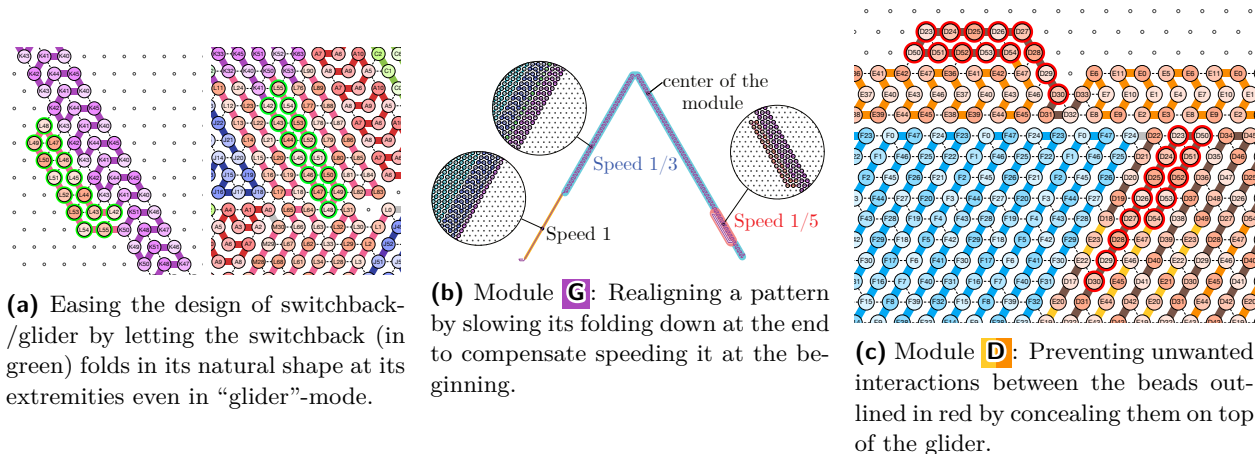
**Socks** work by letting a glider/switchback module fold into a switchback conformation for some time when it would otherwise fold into a glider. Examples are shown in Figure 11. They are easy to implement, since the socks naturally adopt the same shape as turn that part of the module has in the switchback conformation. They offer a lot of freedom in the design, for several reasons:

- First, they simplify the design of important switchback part by *lifting the need for implementing the glider conformation* for that part, as shown in Figure 11(a).





**Figure 10** The interactions of module **G** in “glider”-mode with different environments result in heading to different entry points to the next area of the folding.



**Figure 11** Different uses of socks: (a) Easing bond design; (b) Delaying; (c) Preventing unwanted interactions.

- Second, a glider naturally progresses at speed 1/3. Adding a sock allows us to *slow its progression down* to speed 1/5 for some time, as in Figure 11(b), and therefore realign them. We used that feature repeatedly to “shift” some modules, by starting them with an initial speed-1 (i.e. straight line) progression, as in Figure 11(b), and then compensate for that speed by introducing a socket later on, and realign the brick with others. This is a key point in the design, as it allowed us to separate the Read and Copy functions into different parts of module **G**, and therefore to get less constraints on rule design. In the specific case of module **G**, the Copy-function occurs at the center of the module, while the Read-function is implemented earlier in module!
- Finally, socks allow to prevent unwanted interactions between beads by *concealing* potentially harmful beads in unreachable area as in Figure 11(c).

**Exponential coloring** is a key tool to allow module **G** to fold into different shapes, glider or switchback, along module **F**, when folding in the Read► configuration. This trick is described in greater detail in Section 7.10. The problem it solves is that in order for **G** to fold into its switchback shape, we need strong interactions between **G** and neighboring module **F**, whereas in order for **G** to adopt the glider shape, we want to avoid those interactions. This is made possible because gliders progress at speed 1/3 while switchbacks progress at speed 1. Using a power-of-3 coloring allows to realise these contradicting goals altogether (precise construction is analysed in Lemma 15 in Section 8).

## 7 The Sequences for the Bricks

We now define the primary structure we use to simulate a skipping cyclic tag system. The complete rule  $\heartsuit$  is given in the appendix in Section C.

### 7.1 Extra notations for sequences

In order to do so, we need a few extra notations to manipulate sequences: if  $u$  and  $v$  are finite sequences, we write their concatenation as  $u \cdot v$ . For any two integers  $0 \leq i \leq j < |u|$ , we also write  $u_{[i..j]}$  for  $u_i u_{i+1} \dots u_j$ . The *reverse sequence* of  $u$ , written as  $u^R$ , is  $u_{|u|-1} u_{|u|-2} \dots u_1 u_0$ .

Finally, given a sequence  $u$ , we write  $u \langle \langle a_1 @ i_1, \dots, a_k @ i_k \rangle \rangle$  for the sequence  $w$  where for all  $j \in \{1, 2, \dots, k\}$ , bead  $i_j$  of  $u$  has been replaced by  $a_j$ :

$$w_i = \begin{cases} a_j & \text{if } i = i_j \text{ for some } j \\ u_i & \text{otherwise} \end{cases}$$

By extension, we write  $u \langle \langle v @ k..l \rangle \rangle$  for the sequence  $w$  where for all  $i \in \{k, k+1, \dots, l\}$ , the beads at indices  $k$  to  $l$  of  $u$  have been replaced by the word  $v$  (of length  $l - k + 1$ ):

$$w_i = \begin{cases} v_{i-k} & \text{if } k \leq i \leq l \\ u_i & \text{otherwise} \end{cases}$$

For an infinite sequence of (finite) words  $(u_i)_{i \geq 1}$ , we denote by  $\odot_{i \geq 1} u_i$  the infinite word  $u_1 u_2 \dots u_i \dots$  obtained by containing all the words  $u_1, \dots$ .

### 7.2 More constants: $k$ , $\lambda$ and $\kappa$

We also define three new constants as helpers for the module sequences:

- $k = \frac{h-3}{6}$ . Note that by the definition of  $h$  in Section 4.2,  $k$  is even.
- $\lambda = W/2$ . By the definition of  $W$  in Section 4.2,  $\lambda \bmod 24 = 0$ .
- $\kappa = W/24$ . By the definition of  $W$  in Section 4.2,  $\kappa$  is even.

### 7.3 $\text{Seed}_w$ : Seed for input $u$ .

We first describe the seed  $\text{Seed}$ , which is essentially an encoding of the input word  $u$  to the skipping cyclic tag system we are simulating. As per the definition of oritatami systems, this is a conformation, thus a sequence of beads *together with positions* (i.e. all other sequences have their positions defined by the folding dynamics). These positions will be encoded incrementally, using the following notation, relative to the axes define in Figure 6:

- $a \swarrow b$  means a bead of type  $a$ , followed by a bead of type  $b$ , such that  $\text{pos } b = \text{pos } a + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ .
- $a \searrow b$  means a bead of type  $a$ , followed by a bead of type  $b$ , such that  $\text{pos } b = \text{pos } a + \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ .
- $a \overrightarrow{E} b$  means a bead of type  $a$ , followed by a bead of type  $b$ , such that  $\text{pos } b = \text{pos } a + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ .
- $a \nearrow b$  means a bead of type  $a$ , followed by a bead of type  $b$ , such that  $\text{pos } b = \text{pos } a + \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ .
- $a \nwarrow b$  means a bead of type  $a$ , followed by a bead of type  $b$ , such that  $\text{pos } b = \text{pos } a + \begin{pmatrix} -1 \\ -1 \end{pmatrix}$ .
- $a \overleftarrow{W} b$  means a bead of type  $a$ , followed by a bead of type  $b$ , such that  $\text{pos } b = \text{pos } a + \begin{pmatrix} -1 \\ 0 \end{pmatrix}$ .

## XX:22 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

For example, a sequence such as  $a \swarrow_{SW} b \rightarrow_E c \swarrow_{SW} d \leftarrow_W e$ , starting at  $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$  is a sequence of five beads: a bead of type  $a$  at  $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ , a bead of type  $b$  at  $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ , a bead of type  $c$  at  $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ , a bead of type  $d$  at  $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ , and a bead of type  $e$  at  $\begin{pmatrix} 0 \\ 2 \end{pmatrix}$ .

We are now ready to describe module **Seed**, built by combining 4 types of conformation segments, see Figure 12:

- **SegSeedTerm** =  $\left( \text{J8} \swarrow_{SW} \text{J7} \swarrow_{SW} \right)^{2+\frac{h-3}{4}} \text{J11} \swarrow_{SW} \text{J12} \leftarrow_W \text{J16} \leftarrow_W \text{J17} \leftarrow_W \text{J18}$
- **SegSeedPrefix** =  $\text{A9} \swarrow_{SW} \text{A12} \leftarrow_W \text{B0} \leftarrow_W \text{B1} \swarrow_{NW} \text{B2} \rightarrow_E \text{B3} \swarrow_{NE} \text{B4} \swarrow_{NW} \text{C4} \swarrow_{NW} \text{A0} \leftarrow_W \text{A0} \right)^{w-17} \swarrow_{SW} \text{H8} \swarrow_{SW} \text{H19} \swarrow_{SW} \text{H20} \leftarrow_W \text{H21} \swarrow_{SW} \text{H24} \leftarrow_W \text{I15} \leftarrow_W \text{I15} \leftarrow_W \text{I16} \leftarrow_W \text{I17} \leftarrow_W \text{I19} \leftarrow_W \text{I19} \leftarrow_W \text{J12} \leftarrow_W \text{J16} \leftarrow_W \text{J17} \leftarrow_W \text{J18}$
- **SegSeed(0)** =  $\text{L17} \leftarrow_W \text{L18} \leftarrow_W \text{L47} \swarrow_{SW} \text{L48} \swarrow_{NW} \text{L49} \leftarrow_W \text{L82} \leftarrow_W \text{L83} \swarrow_{NW} \text{A6}$
- **SegSeed(1)** =  $\text{L17} \leftarrow_W \text{L18} \leftarrow_W \text{L48} \leftarrow_W \text{L82} \leftarrow_W \text{L83} \swarrow_{NW} \text{L84} \leftarrow_W \text{A6}$
- **SegSeedLineFeed** =  $\text{K34} \swarrow_{SW} \left( \text{K45} \swarrow_{SW} \text{K40} \swarrow_{SW} \right)^{11} \left( \text{K46} \swarrow_{SW} \text{K47} \swarrow_{SW} \right)^{k-2} \left( \text{K57} \swarrow_{SW} \text{K52} \swarrow_{SW} \right)^{k-2} \left( \text{K59} \swarrow_{SW} \text{K60} \swarrow_{SW} \right)^{k-18} \left( \text{K69} \swarrow_{SW} \text{K64} \swarrow_{SW} \right)^{10} \text{K69} \swarrow_{SW} \text{M20} \searrow_{SE} \text{M26} \swarrow_{SW} \text{M27} \leftarrow_W \text{M28} \swarrow_{SW} \text{M29} \rightarrow_E \text{M30}$ .

Each letter  $a \in \{\emptyset, 1\}$  is encoded in the seed by the conformation:

$$\text{SegSeedLetter}(a) = \left( \text{SegSeed}(1) \leftarrow_W \text{SegSeedPrefix} \leftarrow_W \right)^{n-1} \text{SegSeed}(a) \leftarrow_W \text{SegSeedPrefix}$$

Then, the module **Seed<sub>w</sub>** is:

$$\text{Seed}_w = \text{SegSeedTerm} \leftarrow_W \left( \bigcirc_{i=1}^{|w|} \text{SegSeedLetter}(u_{|w|-i}) \leftarrow_W \right) \text{SegSeedLineFeed}$$

### 7.4 **A**: Zig-Init.

The first module, **A**, is defined as:

$$\mathbf{A} = \mathbf{A0..4} \cdot (\mathbf{A5..10})^{3k-1} \cdot \mathbf{A5..7} \cdot \mathbf{A6} \cdot \mathbf{A9..10} \cdot \mathbf{A11..12}.$$

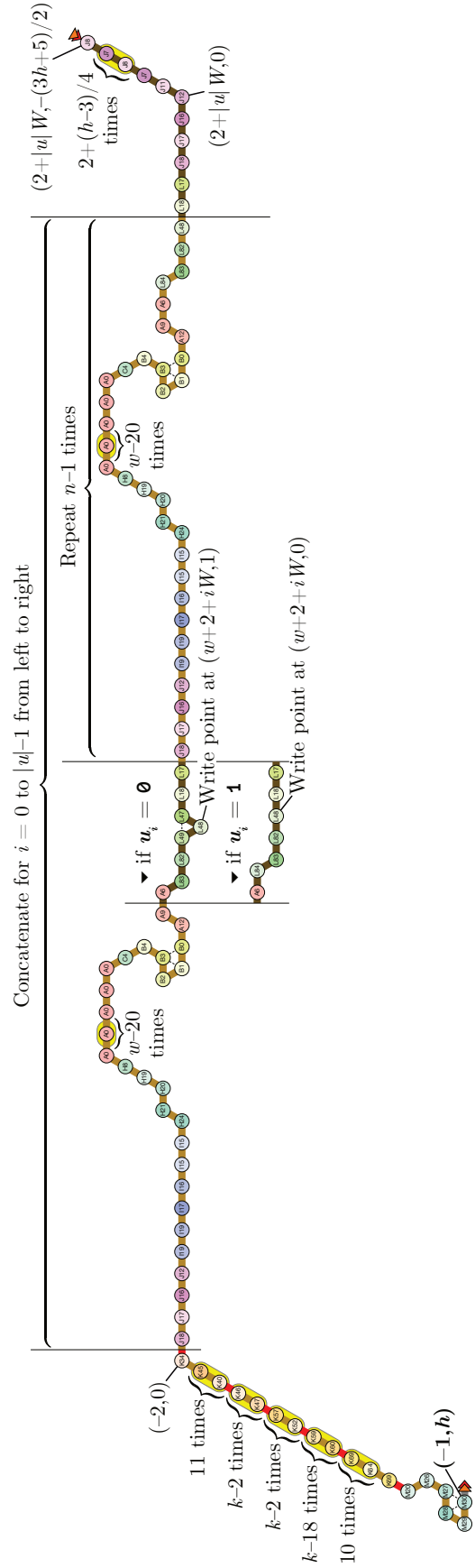
The length of **A** is therefore  $5 + 6(3k - 1) + 3 + 1 + 2 + 2 = 3h - 2$ . The proof trees in Section 8.2 prove that **A** always has height  $H = 2 + 2(3k - 1) + 3 = h$ , and width 3, and folds as in Figure 13.

### 7.5 **B**: Empty word detector.

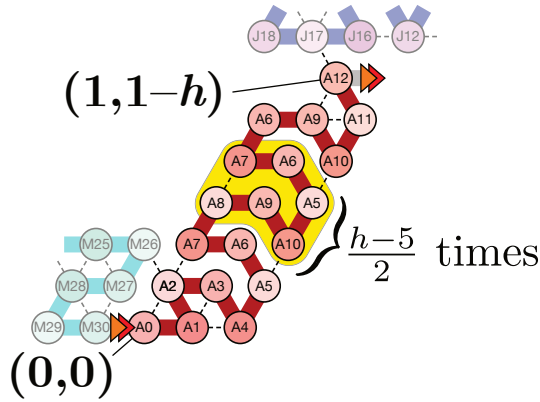
The next module is **B**, whose purpose is to test whether the word is empty, and orient the folding either into a closed connected component of the place, if the word is empty, or to the outside of that connected component. This module is defined as **B** = **B0..4**, which is of length 5, and its two possible functions are shown in Figure 14.

### 7.6 **C**: End of word detector.

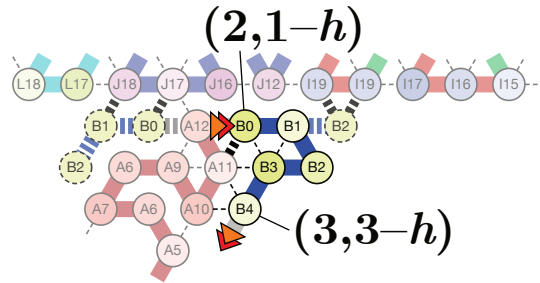
If **B** does not detect an empty word, the folding goes on to **C**, whose purpose is to detect the end of the word: if the current position is at the end of the current word, a production segment (encoded by a sequence of **D<sub>0</sub>** and **D<sub>1</sub>**) needs to fold into a word appended at the end of the current word. Else, **C** folds into a switchback conformation.



■ Figure 12 The brick Seed  $\langle u \rangle$ .



■ **Figure 13** The folding of **A** in the zig phase



■ **Figure 14** The folding of **B** in the zig phase. The two possible outcomes of the text are shown in this figure: the one in dashed lines is what happens when the word is empty, i.e. when the two *I*19 beads are absent, and the one in full lines when the word is not empty

This module is defined as  $\mathbf{C} = (\mathbf{C0..2})^{2k} \cdot (\mathbf{C3..5})^k \cdot \mathbf{C3} \cdot \mathbf{C7} \cdot \mathbf{C8} \cdot (\mathbf{C6..8})^{k-1} \cdot (\mathbf{C9..14})^{k-1} \cdot \mathbf{C9..10} \cdot \mathbf{C15..16} \cdot \mathbf{C13}$ .

The length of  $\mathbf{C}$  is  $3h - 10 = 3 \times 2k + 3k + 3 + 3(k - 1) + 6(k - 1) + 5$ .

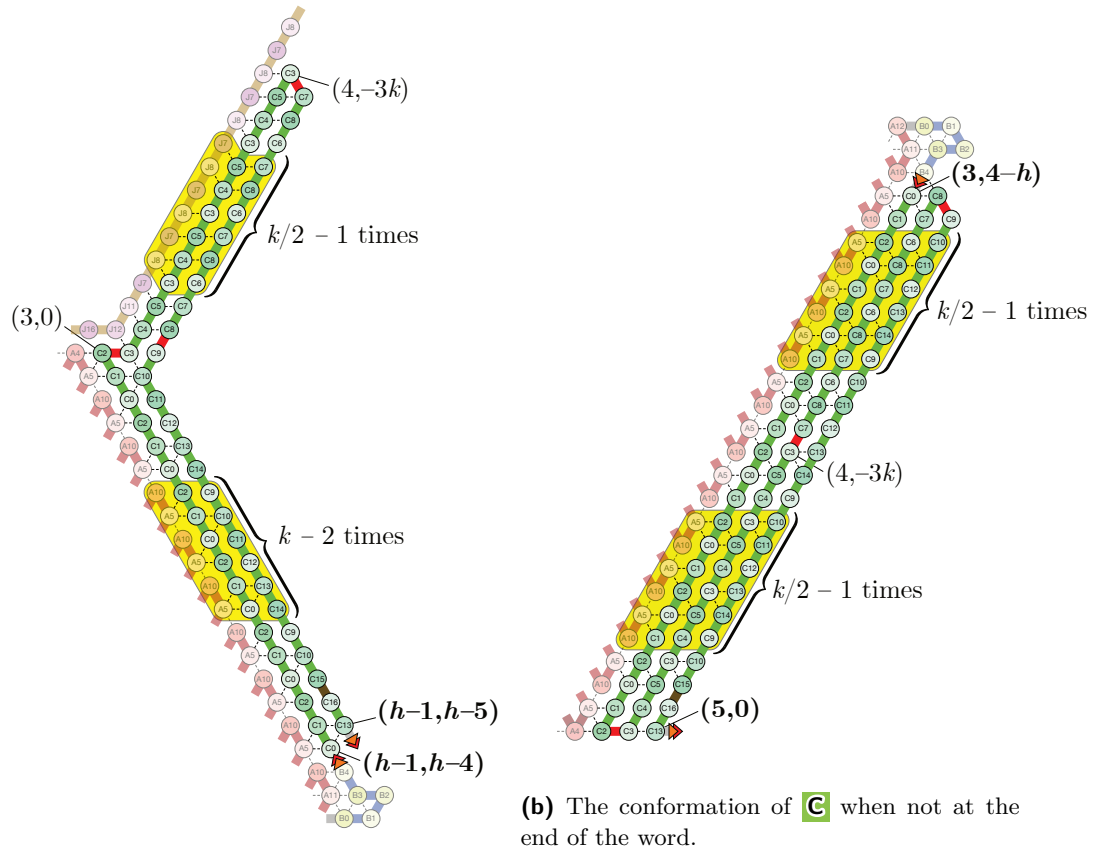
Its two possible conformations in the zig phase are shown in Figure 15:

- The left-hand side figure shows the conformation at the end of the word. Its height is  $H_{exp} = \frac{3h-9}{2} - 2$ , and its width is  $W_{exp} = 2$ .
- The right-hand side figure shows the conformation in the other case. Height (upright):  $H_{up} = h - 3$ . Width (upright):  $W_{up} = 3$ .

## 7.7 **D**: Letters

Module **D** defines the encoding of the letters in productions of the skipping cyclic tag system. It takes three parameters:

- a letter  $x \in \{0, 1\}$ ,
- a parameter  $r \in \{0, 1, 2\}$  to indicate whether this letter is the first letter in the production word (in which case  $r = 0$ ), at an odd position in the production word (in which case  $r = 1$ ), or at an even position in the production word (in which case  $r = 2$ ),



■ **Figure 15** The folding of **C** in the zig phase.

- and a parameter  $t$  indicates whether that letter is the last letter of the word ( $t = 1$ ) or not ( $t = 0$ ).

This module therefore comes in twelve different versions, all of the same length  $3W + 6 \times 5 = 6(\lambda + 5) = 6 \times (12\kappa + 5)$ .

We first describe four helper sequences, each of length  $\lambda + 5 = 12\kappa + 5$ :

- **SegD0** = **D23..33** · **E6..11** · **(E0..11) $\kappa-1$** .
- **SegD1** = **(E12..23) $\kappa$**  · **D49..45**.
- **SegD2** = **D34..44** · **E30..35** · **(E24..35) $\kappa-1$** .
- **SegD3** = **(E36..47) $\kappa$**  · **D54..50**.

### 7.7.1 Encoding of $x = 1$

Then, we define two particular versions of  $\mathbf{D}_1$ , from which the other versions are derived by replacing a few beads:

$$\mathbf{D}_{120} = \text{SegD0} \cdot \text{SegD1} \cdot \text{SegD2} \cdot \text{SegD3} \cdot \text{SegD0} \cdot \text{SegD1}.$$

$$\mathbf{D}_{110} = \text{SegD2} \cdot \text{SegD3} \cdot \text{SegD0} \cdot \text{SegD1} \cdot \text{SegD2} \cdot \text{SegD3}.$$

Module  $\mathbf{D}_{100}$  is obtained by modifying the 17 first beads of  $\mathbf{D}_{120}$ :

$$\mathbf{D}_{100} = \mathbf{D}_{120} \langle\langle \mathbf{D0..16@0..16} \rangle\rangle$$

Then, for all  $r \in \{0, 1, 2\}$ , the trailing versions  $\mathbf{D}_{1r1}$  of  $\mathbf{D}_1$  are obtained by modifying the 8th and 5 last beads of  $\mathbf{D}_{1r0}$ , as follows:

$$\mathbf{D}_{1r1} = \mathbf{D}_{1r0} \langle\langle \mathbf{D17@}(3W + 22), \mathbf{D18..22@}(3W + 25)..(3W + 29) \rangle\rangle$$

### 7.7.2 Encoding of $x = 0$

For all  $r \in \{0, 1, 2\}$  and  $t \in \{0, 1\}$ , module  $\mathbf{D}_{0rt}$  is obtained by replacing most of the beads in the range  $3w + 1..3w + 13$  as follows:

$$\mathbf{D}_{0rt} = \mathbf{D}_{1rt} \langle\langle \mathbf{L17@}(3w + 1), \mathbf{L18@}(3w + 2), \mathbf{D55..62@}(3w + 6)..(3w + 13) \rangle\rangle$$

### 7.7.3 Possible conformations

The possible conformations of  $\mathbf{D}$  are shown in Figures 16 and 17.

### 7.7.4 Size and alignment of the module

First note that the height of module  $\mathbf{D}$  (i.e. the encoding of a single letter of a production), when folded into its switchback conformation, is  $H_{up} = L/6 = W/2 + 5$ , and its width is  $W_{up} = 6$ .

We will now prove a small lemma to make the proofs of a claim in Section 8 easier:

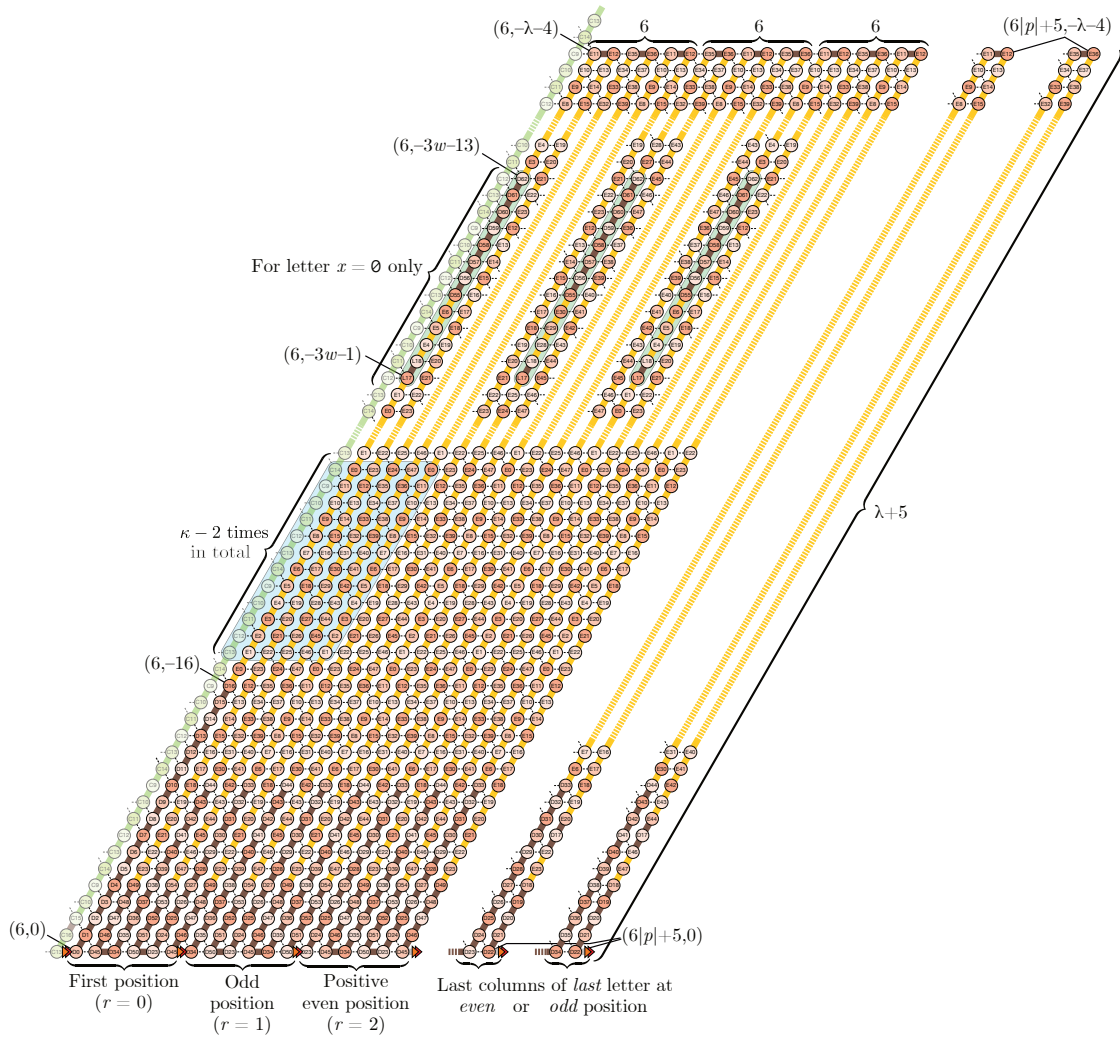
► **Lemma 10.** *The segment of  $\mathbf{D}_{1rt}$  encoding the “bump” in the expanded conformation is always adjacent to the same beads of  $\mathbf{C}$ , when both  $\mathbf{C}$  and  $\mathbf{D}_{1rt}$  are in their switchback conformation.*

**Proof.** Note that  $w = 6 \pmod{12}$ , hence index  $i = 3w + 1$ , the first index of the bump, is such that  $i \pmod{12} = 7$ . This corresponds to index  $j = (11 - i) + 5 = 9 \pmod{12}$  in the previous column, and hence we get the following table:

$i \pmod{12}$	$j = (11 - i) + 5 \pmod{12}$	Neighboring bead in previous and next columns
$3w + 1 = 7$	9	$36 + 9 = \mathbf{E45} \neq \mathbf{L17} \neq \mathbf{E21} = 12 + 9$
$3w + 2 = 8$	8	$36 + 8 = \mathbf{E44} - \mathbf{L18} - \mathbf{E20} = 12 + 8$
$3w + 6 = 0$	4	$36 + 4 = \mathbf{E40} - \mathbf{D55} - \mathbf{E16} = 12 + 4$
$3w + 13 = 7$	9	$36 + 9 = \mathbf{E45} - \mathbf{D62} - \mathbf{E21} = 12 + 9$

This proves the lemma statement.





■ **Figure 16** Module  $\mathbf{D}_0$ , during the zig phase, folded in its switchback/upright conformation.

### 7.8 $\mathbf{E}_a$ : Padding with $L - a$ extra blanks for $0 \leq a \leq L$ .

The purpose of module  $\mathbf{E}_a$  is to make sure that all production segments are of the same length, independently from the length of production words in the simulated skipping cyclic tag system  $\mathcal{S}$ . Recall from Section 4.2 that  $L$  is the length of the longest production word of  $\mathcal{S}$ .

This module has two possible conformations: one in switchback, as shown in Figure 18, and one expanded at the end of the appended production. An outline of the latter conformation is shown in Figure 19.

We will now define the different parts of module  $\mathbf{E}$ , composed of 4 parts:

- the two first parts are based on the two infinite sequences:
  - $\mathbf{SegEA} = ((\mathbf{F0..11})^\kappa \cdot (\mathbf{F12..23})^\kappa \cdot (\mathbf{F24..35})^\kappa \cdot (\mathbf{F36..47})^\kappa)^\infty$
  - $\mathbf{SegEB} = ((\mathbf{G0..11})^\kappa \cdot (\mathbf{G12..23})^\kappa \cdot (\mathbf{G24..35})^\kappa \cdot (\mathbf{G36..47})^\kappa)^\infty$
- $\mathbf{SegEC} = \mathbf{H0..4} \cdot (\mathbf{H5..16})^{q-1} \cdot \mathbf{H5..10} \cdot \mathbf{H17..24}$  of length  $5 + 12(q - 1) + 6 + 8 = 3h - 2$ , where  $q = \frac{h-3}{4} = 0 \pmod{3}$ .



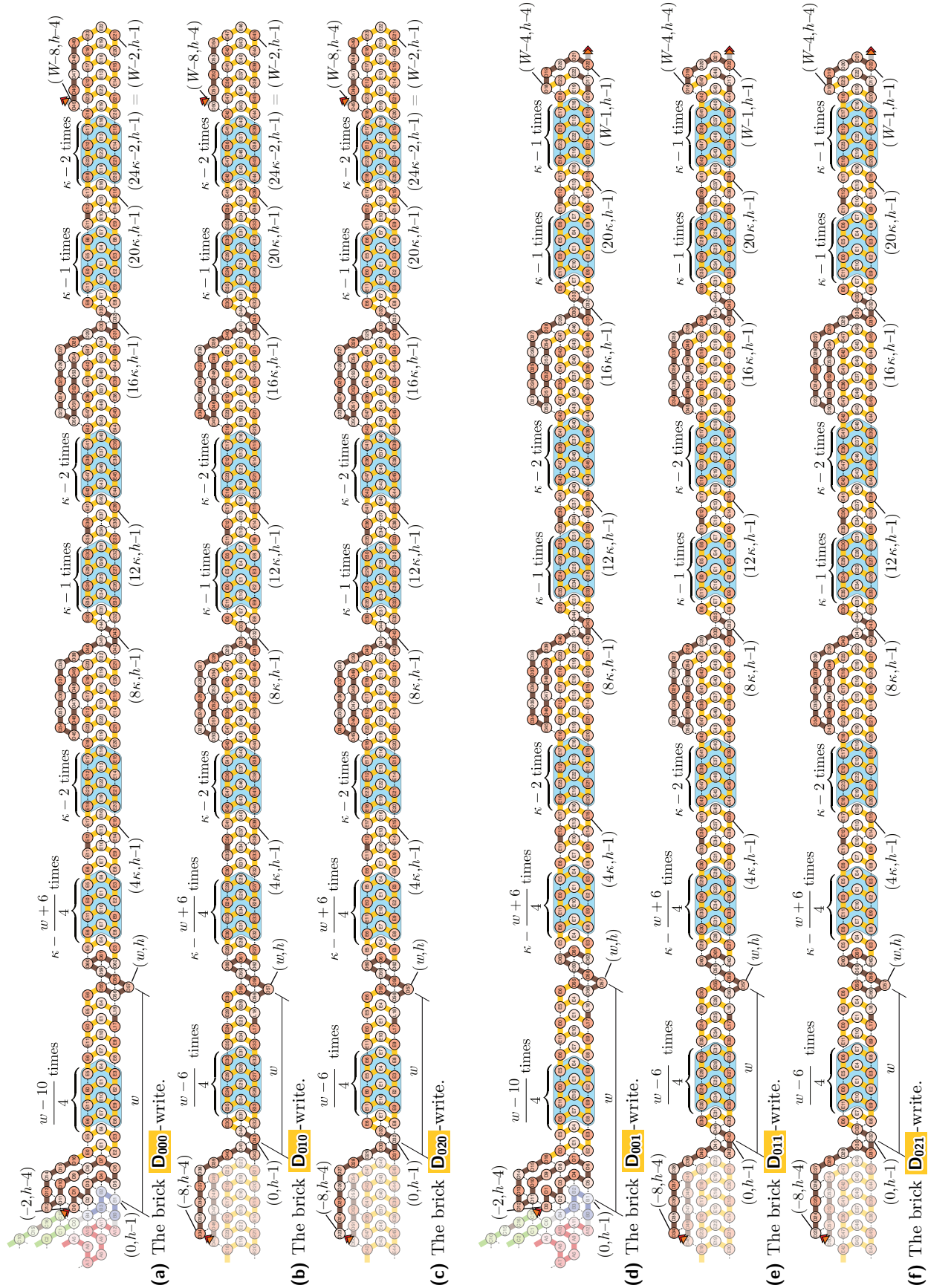
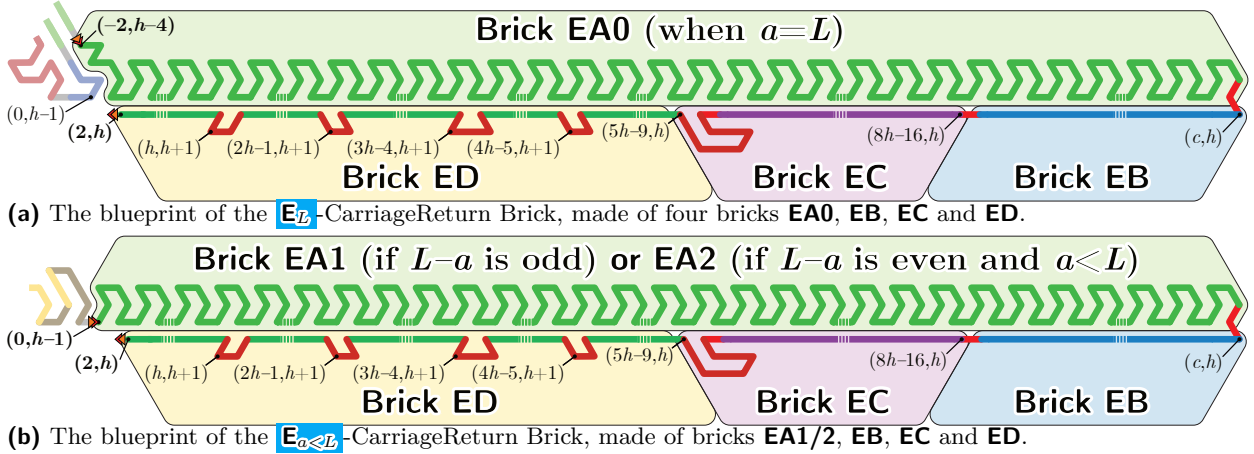


Figure 17 Module **D** expanded to append the new production word at the end of the current word.





■ **Figure 19** Outline of the four different parts of module  $\mathbf{E}_a$ , when folded at the end of the appended production word. See Figure 20 for the detailed beads of each part.

- the last part is composed of five sequences:
  - $\text{SegED}_0 = \mathbf{I15} \cdot \mathbf{I1..5} \cdot (\mathbf{I0..5})^{k-1} \cdot \mathbf{I0} \cdot \mathbf{I1} \cdot \mathbf{I18}$  of length  $6 + 6(k-1) + 3 = h$
  - $\text{SegED}_1 = \mathbf{I19} \cdot \mathbf{I7} \cdot \mathbf{I8} \cdot (\mathbf{I6..8})^{2k-1} \cdot \mathbf{I6} \cdot \mathbf{I7} \cdot \mathbf{I15} \cdot \mathbf{I16}$  of length  $3 + 3(2k-1) + 4 = h+1$
  - $\text{SegED}_2 = \mathbf{I17} \cdot \mathbf{I10} \cdot \mathbf{I11} \cdot (\mathbf{I9..11})^{2k-1} \cdot \mathbf{I9} \cdot \mathbf{I10} \cdot \mathbf{I19}$  of length  $3 + 3(2k-1) + 3 = h$
  - $\text{SegED}_3 = \mathbf{I18} \cdot \mathbf{I13} \cdot \mathbf{I14} \cdot (\mathbf{I12..14})^{2k-1} \cdot \mathbf{I12} \cdot \mathbf{I13} \cdot \mathbf{I19}$  of length  $3 + 3(2k-1) + 3 = h$
  - and  $\text{SegED}_4 = \mathbf{I19} \cdot \mathbf{I1} \cdot \mathbf{I2} \cdot (\mathbf{I0..2})^{2k}$  of length  $3 + 3 \times 2k = h$

We may now define the sequence for the module  $\mathbf{E}_a$  for  $0 \leq a \leq L$  by letting  $K = 3W(L-a+P)$ , and:

- $\text{HeadE}_a = (\text{SegEA})_{[b..b+3c-2]} \cdot \mathbf{F51} \cdot (\text{SegEB})_{[b+3c..b+K-2]}$ , of length  $K-1$ , where  $b = 0$  if  $a$  is even, and  $b = 2\lambda$  if  $a$  is odd.
- $\text{TailE} = \text{SegEC} \cdot \text{SegED}_0 \cdot \text{SegED}_1 \cdot \text{SegED}_2 \cdot \text{SegED}_3 \cdot \text{SegED}_4$ .

Then the module  $\mathbf{E}_a$  is:

- $\mathbf{E}_0 = (\text{HeadE}_0) \langle \langle \mathbf{F48..49@0..1}, \mathbf{F50@11} \rangle \rangle \cdot \mathbf{G48} \cdot \text{TailE}$
- and for  $a > 0$ :  $\mathbf{E}_a = (\text{HeadE}_a) \cdot \mathbf{G48} \cdot \text{TailE}$ .

The length of module  $\mathbf{E}_a$  is:

$$\begin{aligned} |\mathbf{E}_a| &= K + 8h - 1 = 6\lambda(L-a+P) + 8h - 1 \\ &= \underbrace{6 \times 12\kappa \times (L-a+P)}_{=0 \pmod{3 \times 48}} + \underbrace{8h}_{=24 \pmod{2 \times 48}} - 1 \end{aligned}$$

Therefore, for any word  $a$ ,  $|\mathbf{E}_a| \pmod{48} = 23$ .

We will now prove a lemma, used later in Section 8 to prove that the two conformations (switchback and expanded) can be obtained at the same time:

► **Lemma 11.** *When folded in the switchback conformation (i.e. as in Figure 18), all beads of  $\text{SegEC}$  are far enough from  $\text{SegEA}$  to be attracted by them.*

*Hence, the attractions between these two parts can be freely chosen to “force”  $\text{SegEC}$  to fold into a straight line instead of a glider in the expanded conformation.*

**Proof.** Let  $c = \frac{|\mathbf{E}_a| - 15}{4}$ , and note that  $c \bmod 12 = 2$ . Now, the width of **SegEB**, when folded in the switchback conformation is  $K - (3c - 1)$ . Moreover,  $3c - 1 = 5 \pmod{36}$ . Therefore, for any padding length  $a$ :

$$\begin{aligned} K - (3c - 1) &= \frac{6}{4}\lambda(L - a + P) - 6h + 13 = \frac{3}{2}\lambda(L - a + P) - 12\lambda + 6(w + 3) + 13 \\ &> \lambda\left(\frac{3P}{2} - 12\right) \geq 4.5\lambda, \quad \text{since } P \geq 11. \end{aligned}$$

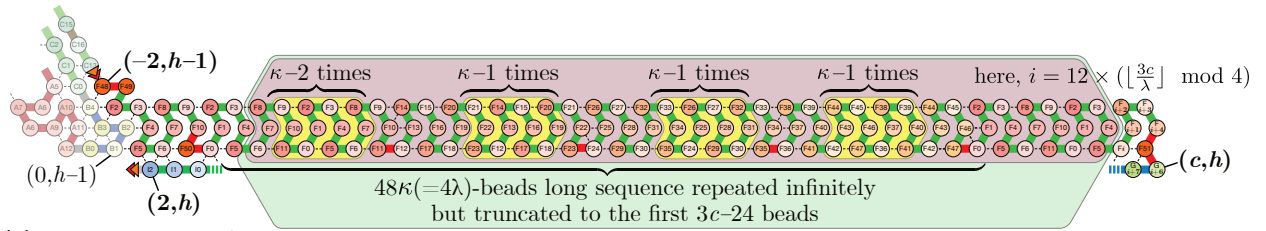
Therefore, the width of **SegEB** is at least 5 columns, and since the delay of our simulation is 3, no bead of **SegEC** can ever be attracted to a bead of **SegEA** in the switchback conformation. ◀

► **Lemma 12.** *Bead F51 is never on an edge of brick  $\mathbf{E}_L$ -CarriageReturn.*

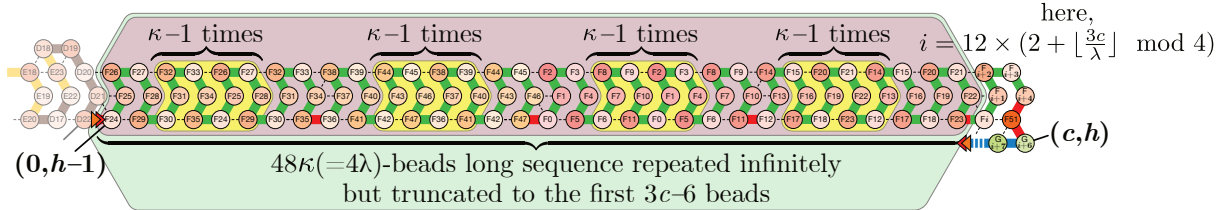
*Therefore, that bead is not involved in a turn in that brick, which means that the attraction rule can be decided mostly based on its EA0 brick, to initiate the turn at the end of the padding, in the expanded conformation of  $\mathbf{E}$ .*

**Proof.** That bead is at index  $3c - 1$  from the beginning of the module, and moreover between the **SegEA** and **SegEB** segments, which are both folded into switchbacks of height  $\lambda$  in that brick. Now, note that  $\lambda \bmod 12 = 0$ , and that  $c \bmod 12 = 2$ . Therefore,  $(3c - 1) \bmod 12 = 5$ , which means that bead **F51** cannot be on the edge of the switchback. ◀

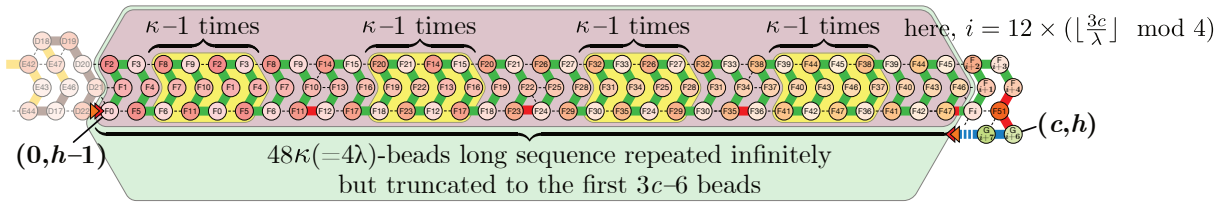
**XX:32 Proving the Turing Universality of Oritatami Co-Transcriptional Folding**



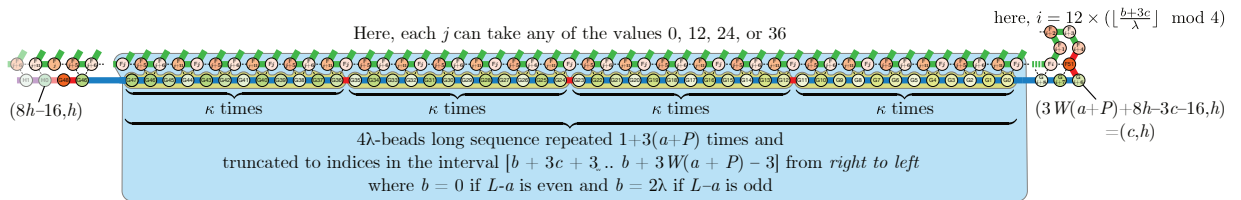
(a) The subbrick **EA0**, when  $a = L$ .



(b) The subbrick **EA1**, when  $L - a$  is odd.

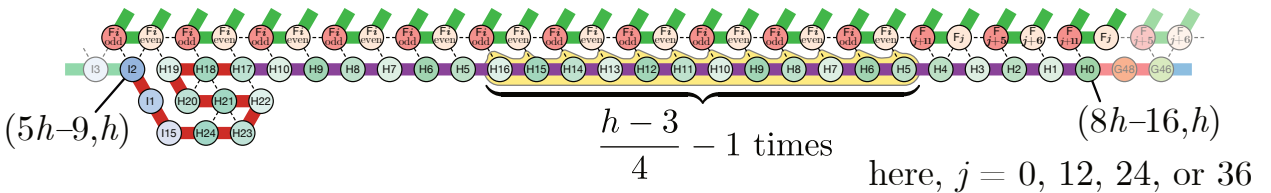


(c) The subbrick **EA2**, when  $L - a$  is even and  $a < L$ .

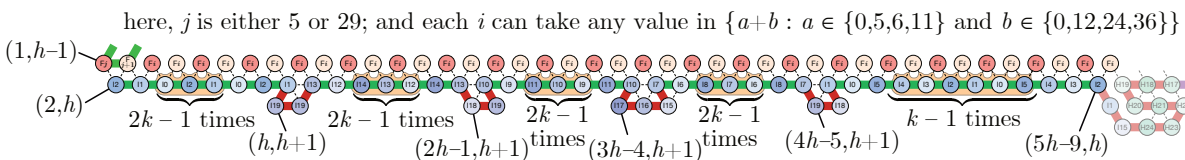


(d) The subbrick **EB**.

here, each  $i$  can take any value with suitable parity  
in  $\{a+b : a \in \{0,5,6,11\} \text{ and } b \in \{0,12,24,36\}\}$



(e) The subbrick **EC**.



(f) The subbrick **ED**.

**Figure 20** The  $\mathbf{E}_a$ -carriageReturn bricks.



### 7.9 $\mathbb{F}$ : Zag-Init.

Finally, after the end of the padding, module  $\mathbb{F}$  is used to start the copying phase. Module  $\mathbb{F}$  has the some conformation in the zig and zag phases, up to a rotations of 180 degrees. The conformation of  $\mathbb{F}$  in the zig phase is shown in Figure 21.

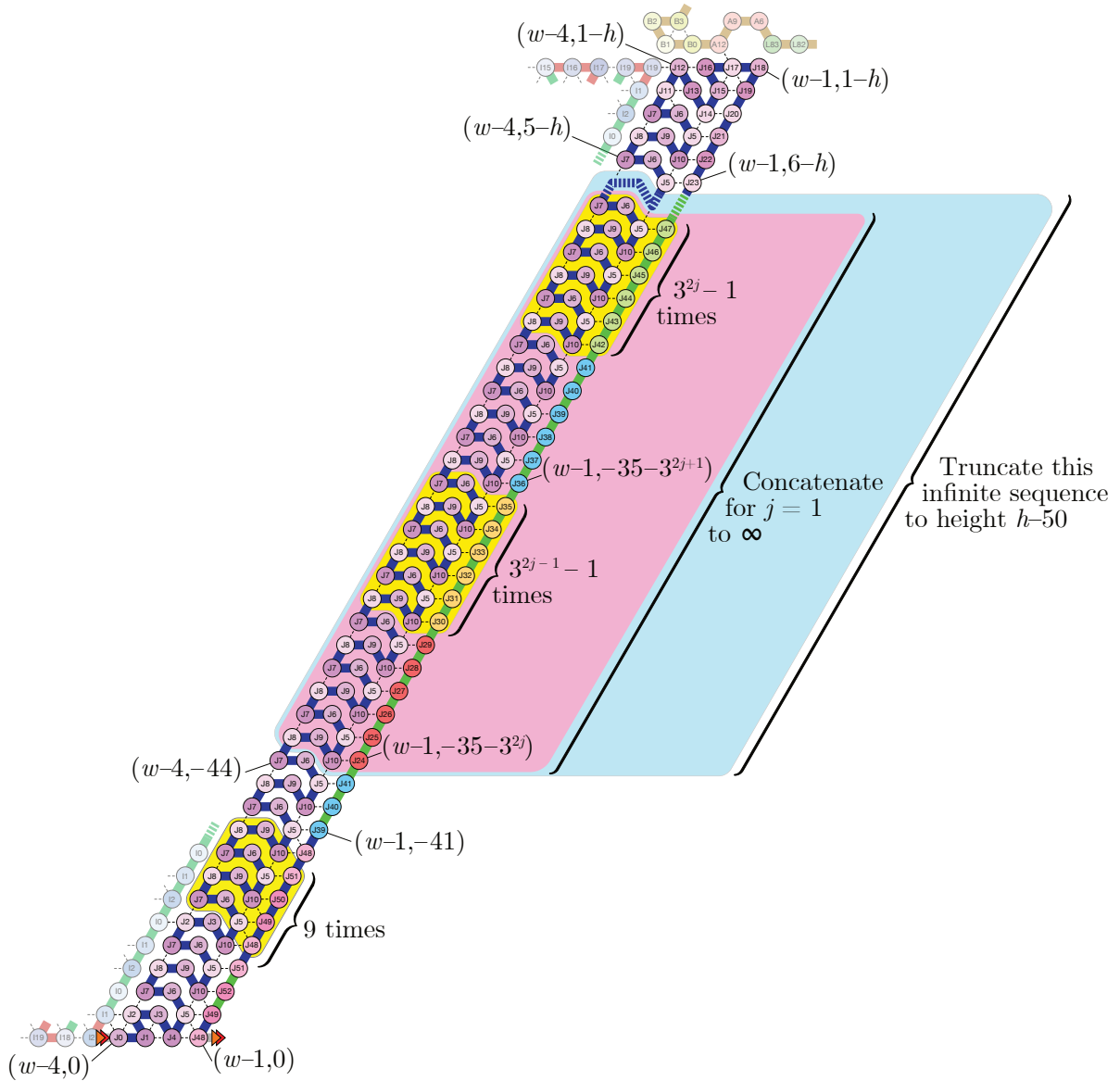


Figure 21

Module  $\mathbb{F}$  is composed of three parts. The beginning is **HeadF** and the end is **TailF**, defined as:

- **HeadF** =  $\mathbf{J0..4} \cdot (\mathbf{J5..10})^{3k-1} \cdot \mathbf{J5..7} \cdot \mathbf{J11..23}$  of length  $5 + 6(3k - 1) + 3 + 13 = 3h + 6$
- **TailF** =  $\mathbf{J48} \cdot (\mathbf{J51..48})^9 \cdot \mathbf{J51} \cdot \mathbf{J52} \cdot \mathbf{J49..48}$  of length  $1 + 4 \times 10 = 41$

The middle part is made of the following “exponential” sequence:

- for even  $i \geq 2$ , let **SegExp**( $i$ ) =  $\mathbf{J24..29} \cdot (\mathbf{J30..35})^{3^{i-1}-1}$  of length  $6 \cdot 3^{i-1}$ ;

## XX:34 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

- and for odd  $i \geq 3$ , let  $\mathbf{SegExp}(i) = \mathbf{J36..41} \cdot (\mathbf{J42..47})^{3^{i-1}-1}$  of length  $6 \cdot 3^{i-1}$ ;
- so as to define the following infinite sequence, which is the concatenation of  $\mathbf{SegExp}(2)$ ,  $\mathbf{SegExp}(3)$ ,  $\mathbf{SegExp}(4)$ ...

$$\mathbf{SegExpF} = \bigcirc_{i \geq 2} \mathbf{SegExp}(i)$$

Finally, the beads sequence for  $\mathbf{F}$  is:

$$\mathbf{F} = \mathbf{HeadF} \cdot (\mathbf{J39..41} \cdot \mathbf{SegExpF}_{[0..(h-51)]})^R \cdot \mathbf{TailF}.$$

Therefore, the length of module  $\mathbf{F}$  is  $3h + 6 + (h - 50) + 3 + 41 = 4h$ .

We claim that for all  $i$ , the pattern  $\mathbf{SegExp}(i)$  starts at index  $3^i - 9$  of  $\mathbf{SegExpF}$ . Indeed:

$$\sum_{j=2}^{i-1} 6 \cdot 3^{j-1} = 2 \cdot \frac{3^i - 9}{3 - 1} = 3^i - 9$$

Finally, notice  $\left| \mathbf{SegExpF}_{[0..(h-51)]} \right| \bmod 12 = (h - 50) \bmod 12 = 1$ .

### 7.10 $\mathbf{G}$ : Read-Copy-Line Feed module.

Module  $\mathbf{G}$  is the last module, and the one in charge of reading and copying the information around. This module can fold into a seven different conformations:

- Figures 22 and 23 show the module reading the encodings of 0 and 1, respectively. These conformations only happen during the zig phase.
- Figures 24 and 25 show the module copying the encodings of 0 and 1, respectively. These conformations are shown on these figures in the zig phase, but are the same in the zag phase, rotated by 180 degrees.
- Figures 26 only happens at the end of the zag phase, after copying the word, and before starting the next step.

Module  $\mathbf{G}$  is of length exactly  $6h - 1$ , and consists of six parts, each of length approximately  $h$ . We described these parts now:

The first part is the most sophisticated since it can be folded either in a straight line, and hence “progress vertically” at speed 1 (i.e. one row per bead), or in a glider, which progresses vertically at speed 1/3 (i.e. two rows every six beads).

The other parts just contains a small delay loop (a sock) that allow to separate crucial sensing function from basic geometry, as explained in Section 6.3. For the rest of this section, let  $k = \frac{h-3}{6} = 0 \pmod 2$ .

**Part 1:** As for Module  $\mathbf{F}$  we define the following exponential pattern:

- for even  $i \geq 2$ , let  $\mathbf{SegExp}'(i) = \mathbf{K4..9} \cdot (\mathbf{K10..15})^{3^{i-1}-1}$  of length  $6 \cdot 3^{i-1}$ ;
- and for odd  $i \geq 3$ , let  $\mathbf{SegExp}'(i) = \mathbf{K16..21} \cdot (\mathbf{K22..27})^{3^{i-1}-1}$  of length  $6 \cdot 3^{i-1}$ ;
- so as to define the infinite sequence:

$$\mathbf{SegExpG} = \bigcirc_{i \geq 2} \mathbf{SegExp}'(i)$$

As for  $\mathbf{SegExpF}$ , the pattern  $\mathbf{SegExp}'(i)$  starts at index  $3^i - 9$  exactly in  $\mathbf{SegExpG}$ .

The first part of  $\mathbf{G}$  is:

$$\mathbf{SegG1} = \mathbf{L0..6} \cdot \mathbf{K3} \cdot (\mathbf{K0..3})^9 \cdot \mathbf{K0..2} \cdot \mathbf{L7..10} \cdot \mathbf{SegExpG}_{[8..h-51]}$$

of length  $7 + 1 + 9 \times 4 + 3 + 4 + h - 51 - 7 = h - 7$ . Note that  $h - 51 = 0 \pmod 12$  and thus the index of the last bead of the exponential part is a multiple of 12.

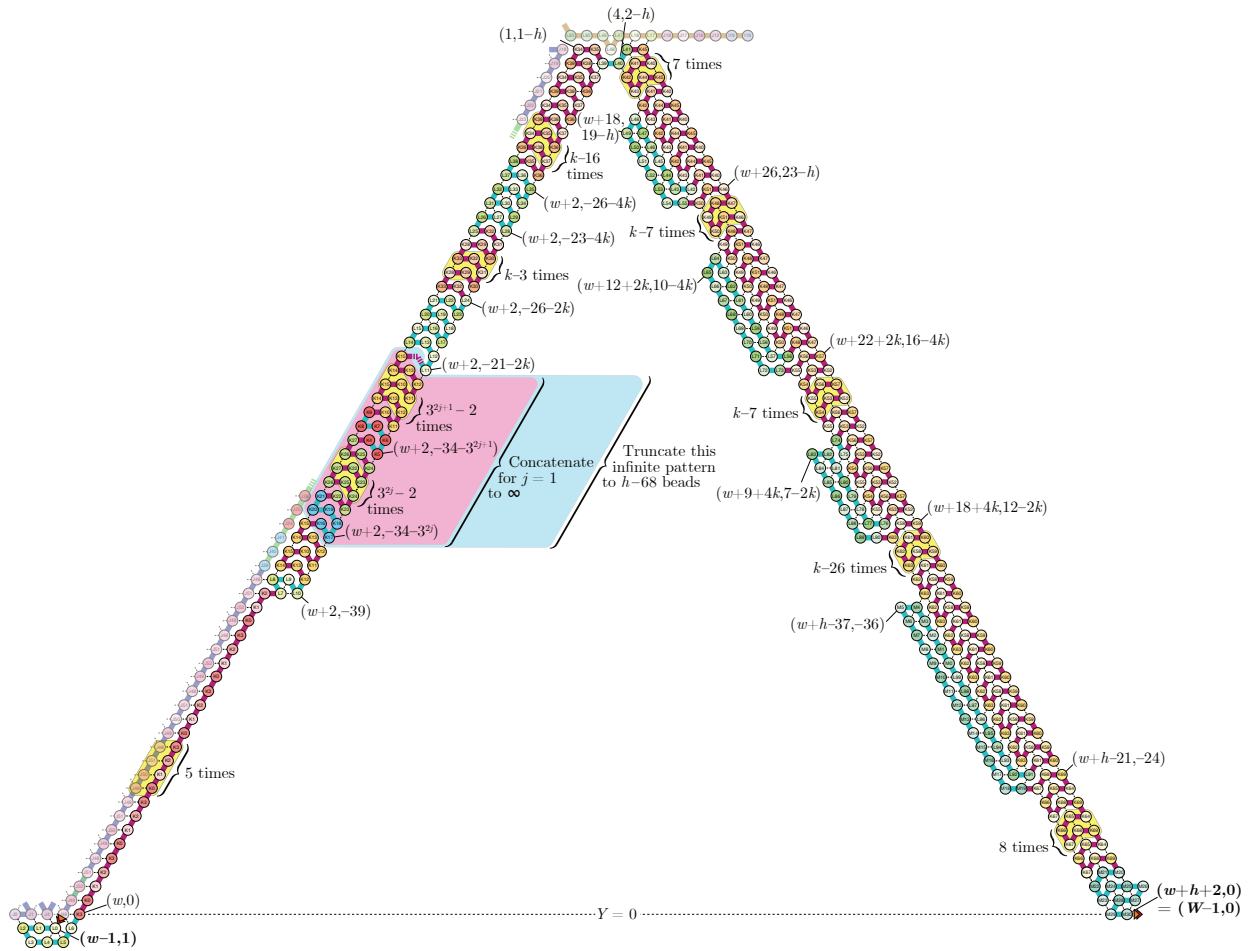


Figure 22 The brick  $\mathbb{G}$ -read $\emptyset$ .



## XX:36 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

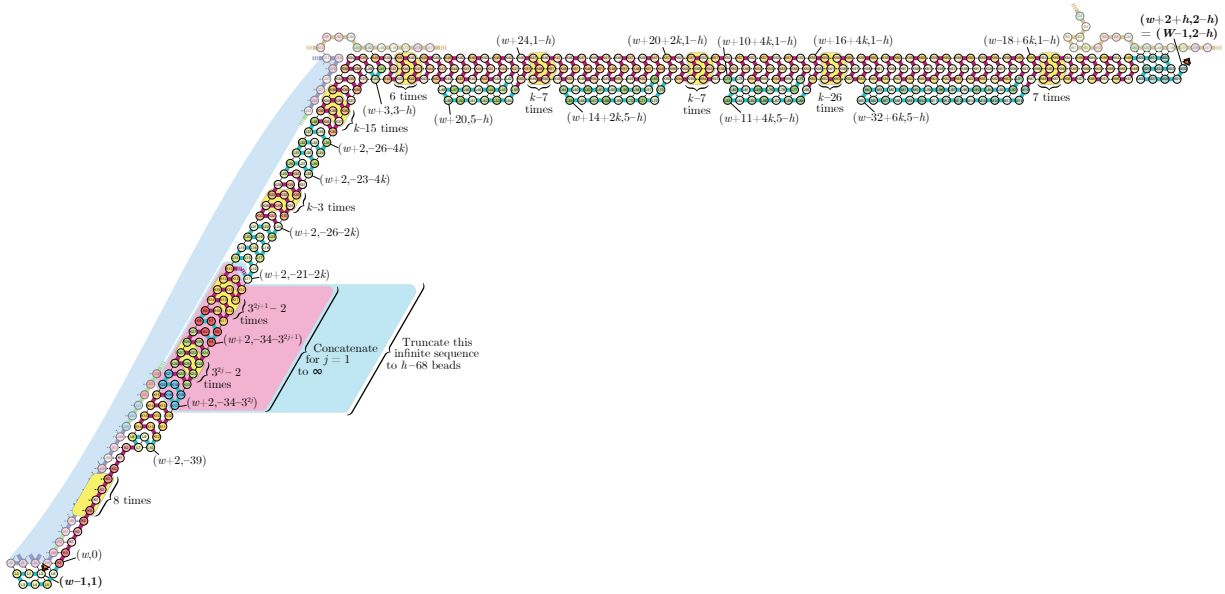


Figure 23 The brick  $\mathbf{G}$ -read1.

**Part 2:**  $\text{SegG2} = \mathbf{K32} \cdot \mathbf{K33} \cdot (\mathbf{K28..33})^{k-3} \cdot \mathbf{K28..32}$  of length  $2 + 6(k-3) + 5 = h - 14 = 1 \pmod{12}$ .

**Part 3:**  $\text{SegG3} = \mathbf{K35..39} \cdot (\mathbf{K34..39})^{k-14} \cdot \mathbf{K34} \cdot \mathbf{K35} \cdot \mathbf{L39..41} \cdot \mathbf{K45} \cdot (\mathbf{K40..45})^{10} \cdot \mathbf{K40} \cdot \mathbf{K41}$  of length  $5 + 6(k-14) + 6 + 10 \times 6 + 2 = h - 14 = 1 \pmod{12}$ .

**Part 4:**  $\text{SegG4} = \mathbf{K50..51} \cdot (\mathbf{K46..51})^{k-3} \cdot \mathbf{K46..48}$  of length  $2 + 6(k-3) + 3 = h - 13 - 3 = h - 16 = 5 \pmod{6}$ .

**Part 5:**  $\text{SegG5} = \mathbf{K55..57} \cdot (\mathbf{K52..57})^{k-6} \cdot \mathbf{K52..53} \cdot \mathbf{L74} \cdot \mathbf{L75} \cdot \mathbf{K56..57} \cdot (\mathbf{K52..57})^2 \cdot \mathbf{K52..53}$  of length  $3 + 6(k-6) + 2 + 2 + 2 + 2 \times 6 + 2 = h - 16 = 5 \pmod{6}$ .

**Part 6:**  $\text{SegG6} = \mathbf{K63} \cdot (\mathbf{K58..63})^{k-19} \cdot \mathbf{K58..61} \cdot \mathbf{L91..99} \cdot \mathbf{M0..19} \cdot \mathbf{K67..69} \cdot (\mathbf{K64..69})^{10} \cdot \mathbf{M20..30}$  of length  $1 + 6(k-19) + 4 + 9 + 20 + 3 + 60 + 11 = h - 9 = 6 \pmod{12}$ .

Finally,

$$\mathbf{G} = \text{SegG1} \cdot \mathbf{L11..24} \cdot \text{SegG2} \cdot \mathbf{L25..38} \cdot \text{SegG3} \cdot \mathbf{L42..55} \cdot \text{SegG4} \cdot \mathbf{L56..73} \\ \cdot \text{SegG5} \cdot \mathbf{L76..90} \cdot \text{SegG6}$$

of total length  $= h - 7 + 14 + h - 14 + 14 + h - 14 + 14 + h - 16 + 18 + h - 16 + 15 + h - 9 = 6h - 1$ .

### 8 Correctness of the folding

We will now resume and expand the explanation give in Section 3. Here is how we proceeded to ensure the correctness of our design:

1. Enumerate all the surrounding for each brick of each module
2. Enumerate all possible modules following the module
3. Generate automatically human-readable certificate of the correctness of the folding for each possibility, in the form of *proof trees*.
4. In the few cases where the surrounding may vary, prove that it has no incidence on the folding of the brick. This happens only for three bricks exactly: when the brick  $\mathbf{G} \blacktriangleright \text{Read}$  zig-folds along  $\mathbf{F} \blacktriangleright \text{ZigUp}$ , when the top of the brick  $\mathbf{G} \blacktriangleright \text{Read1}$  folds, and when the zag-bricks folds under  $\mathbf{D} \blacktriangleright \text{Write}$ .

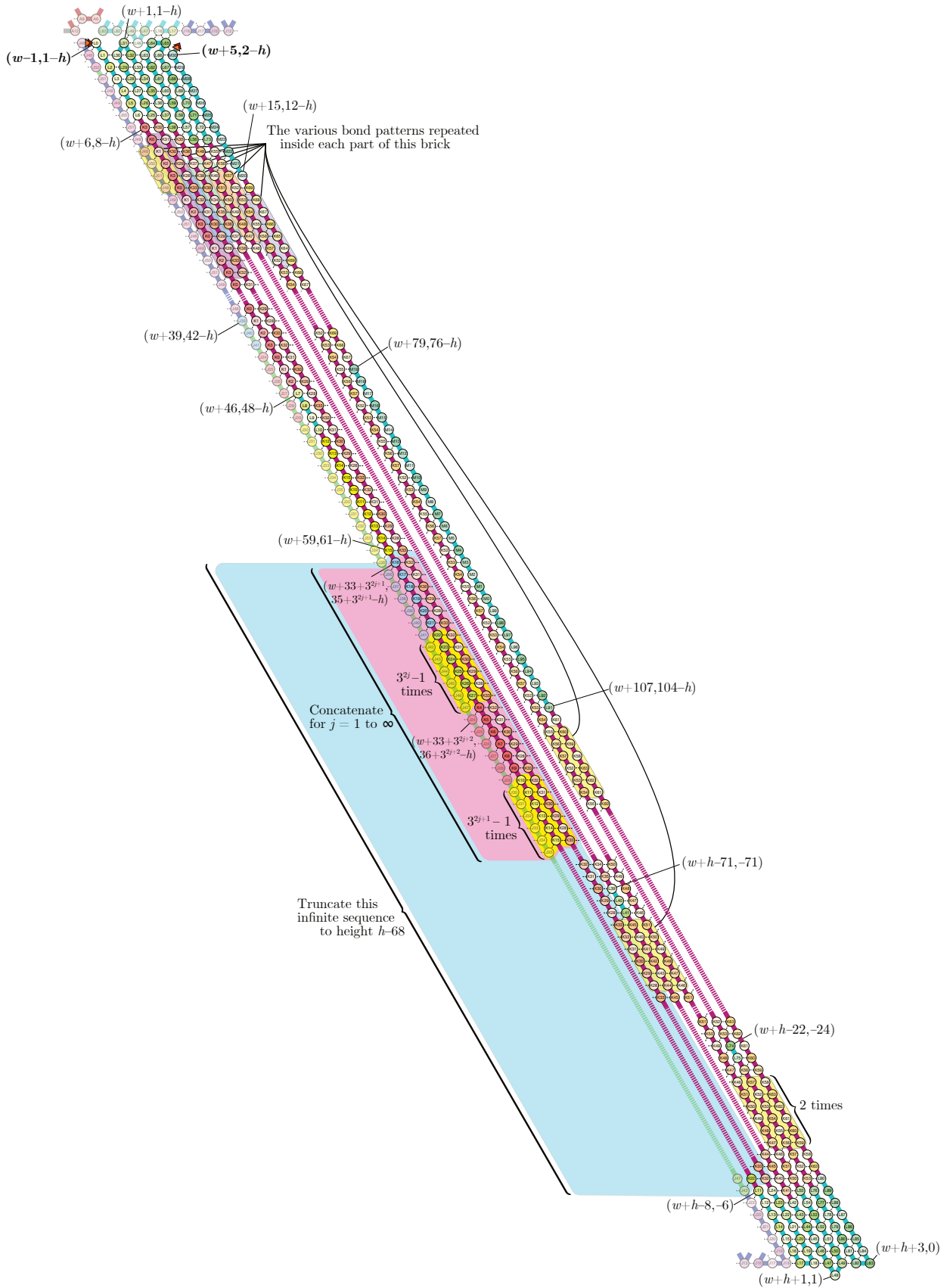


Figure 24 The brick  $\mathbb{G}$ -copy $\emptyset$ .

XX:38 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

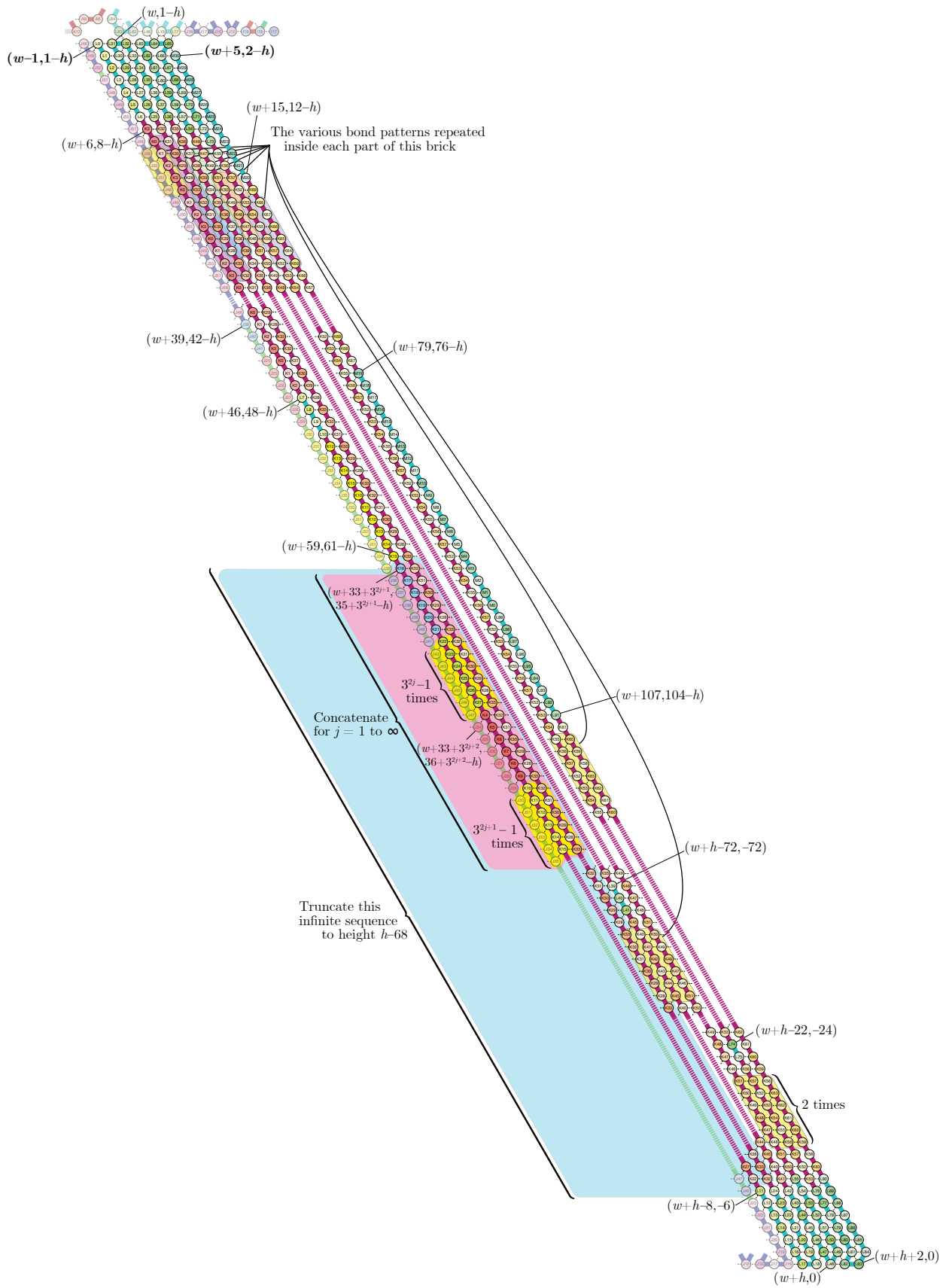


Figure 25 The brick  $\mathbb{G}$ -copy1.

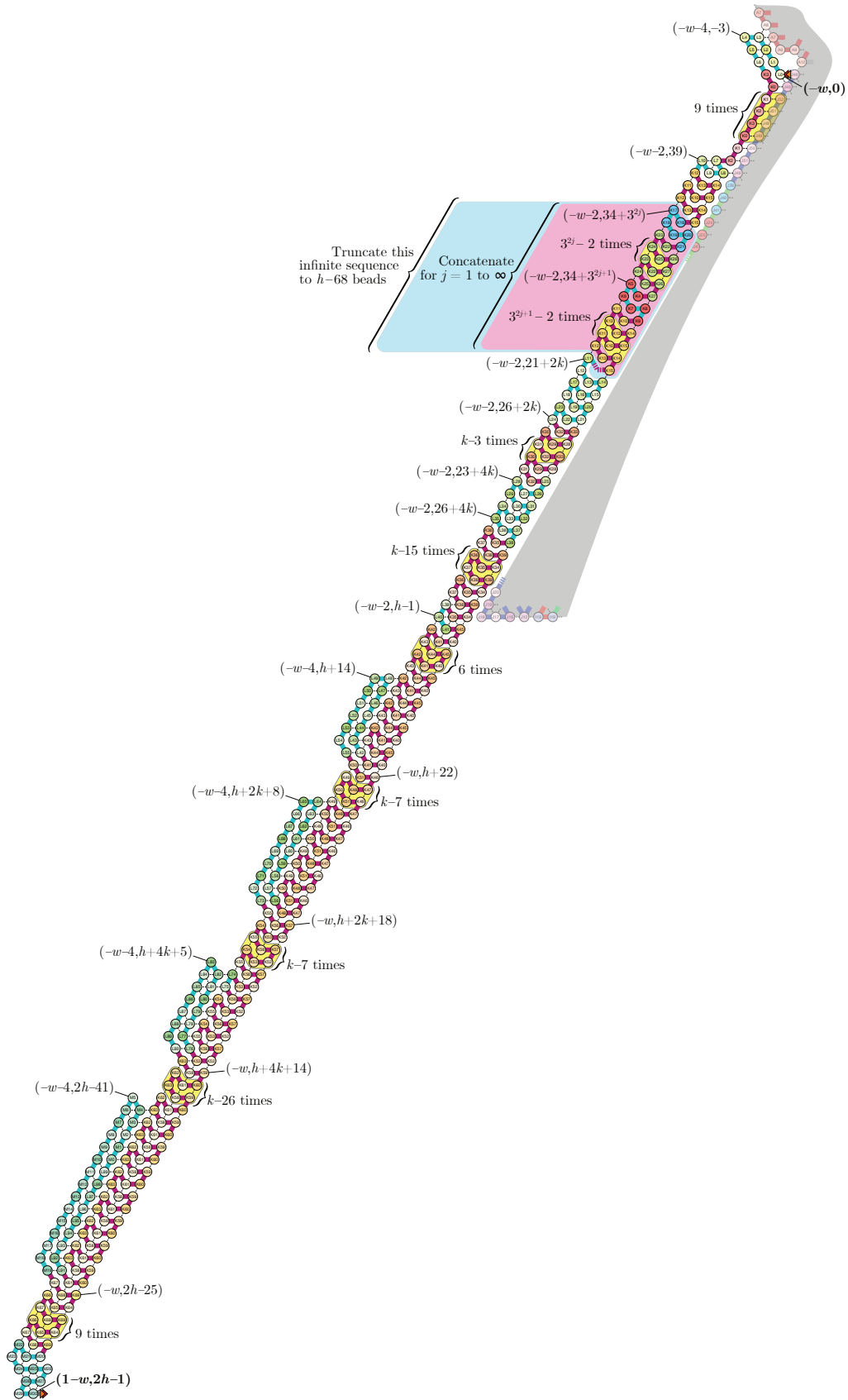


Figure 26 The brick  $\mathbb{G}$ -lineFeed.

The lemmas in Section 7 have proved that the bead alignment in each brick does not change when  $n$  and  $L$  vary. This implies that the figures of the bricks are indeed generic. It follows that with the exception of the three cases listed in point 4 above, and handled in Section 8.1, it is enough to prove the folding of each brick only once. And as most of them are made of repeating patterns, only a finite number of environments have to be considered. That last case will be treated in Section 8.2 using an automatic procedure which produces human-readable certificates called *proof-trees*.

### 8.1 The three bricks with varying environments

The following lemma show that it is enough to proof one folding of the bricks under a  $D \blacktriangleright \text{Write}$ , all the other are the same since there are no interaction between the  $D$ -write brick and any brick folding immediately below it.

► **Lemma 13** (Zag-folding under  $D \blacktriangleright \text{Write}$ ). *The modules ZAG-folding under the bricks  $D \blacktriangleright \text{Write}$  have no interaction with  $D \blacktriangleright \text{Write}$ , with the only exceptions of:*

- the beads **A0** and **A1** of module **A** which have bonds with beads **E(2+12i)** and **L17** for **A0**, and **E(9+12i)** for **A1**, for all  $0 \leq i \leq 3$ .
- the beads **L17, L18, D57, D58** (the bump in module **D0**) which bond with the beads **L65, L64, L31** so that the corresponding module **G** folds into the expected brick  $G \blacktriangleleft \text{Zag Copy } \emptyset$ .

**Proof.** Figure 27 lists all the possible  $\heartsuit$ -interactions between the beads accessible from below the  $D \blacktriangleright \text{Write}$  bricks (to the left) with the beads at the top the modules zag-folding below it that can interact with them (to the right).

The only possible bonds are thus:

**with beads D17 and D22:** (in green on Figure 27) these are only present at the junction between the bricks  $D \blacktriangleright \text{Write}$  and  $E \blacktriangleright \text{CarriageReturn}$ , at the end of the rightmost  $D \blacktriangleright \text{Write}$  brick. The correctness of the zag-folding of the  $F \blacktriangleleft \text{Zag}$  brick below is given next in the proof-trees section.

**with beads L17, L18, D56, D57, D58, D62:** (in blue on Figure 27) these beads are only present in the spike encoding a  $\emptyset$  in the brick  $D \blacktriangleright \text{Write}$ , and these interactions are the one expected to ensure the copy of the encoding of  $\emptyset$  by the module **G** that will Zag-fold below.

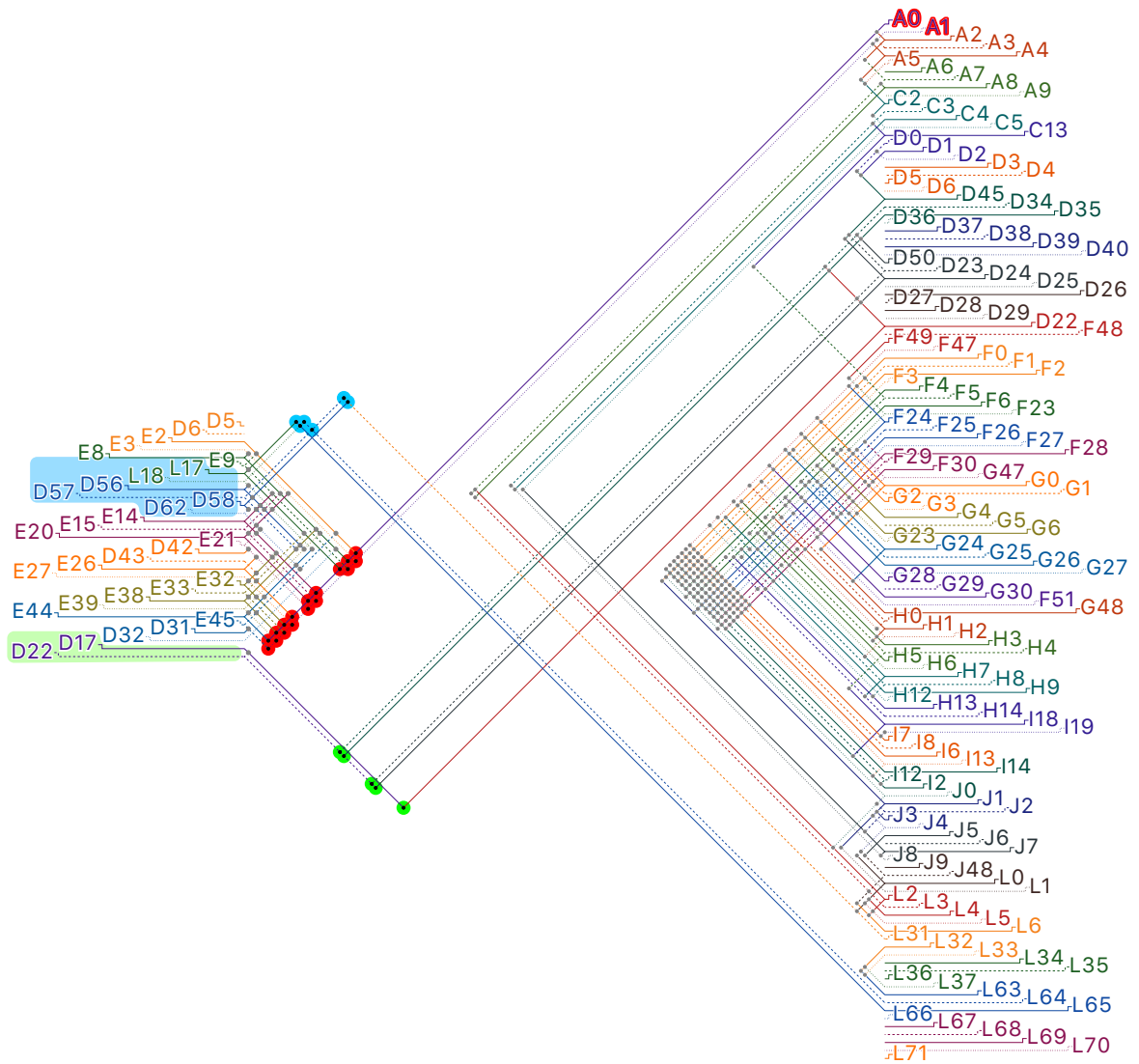
**and finally between beads A0 and A1, and 4 groups of beads:** **E2, E3, E8, E9**, then **E14, E15, E20, E21**, then **E26, E27, E32, E33**, and finally **E38, E39, E44, E45** (in red on Figure 27). As the width of a zag-folded production segment is  $w + 6 = 0 \pmod{12}$ , the beads **A0** and **A1** are always aligned with the same beads within each of these groups (see Figure 17), namely **A0** with **E2, E14, E26** and **E38**, and **A1** with **E9, E21, E33** and **E45**. Furthermore as the interactions of **A0** and **A1** are the same with each of them, it is enough to prove that the module **A** zag-folds correctly between *one* of these groups only, which is done next in the proof-trees section.

It follows that outside these three cases (each handled by a proof-tree, see later), no interactions are possible and the modules will zag-fold below the  $D \blacktriangleright \text{Write}$  bricks independently of the exact beads that are present inside. It is thus enough to show that each module zag-folds correctly at any location to ensure that it zag-folds correctly anywhere below the  $D \blacktriangleright \text{Write}$  brick. ◀

► **Lemma 14** (Top of  $G \blacktriangleright \text{Read 1}$ ). *During the folding of the brick  $G \blacktriangleright \text{Read 1}$ , no bead in **G** interacts with the row above but at its two extremities, i.e. the 82 top-leftmost beads and the 11 last (**K34..L55** and **M20..M30** resp. in Figure 23).*

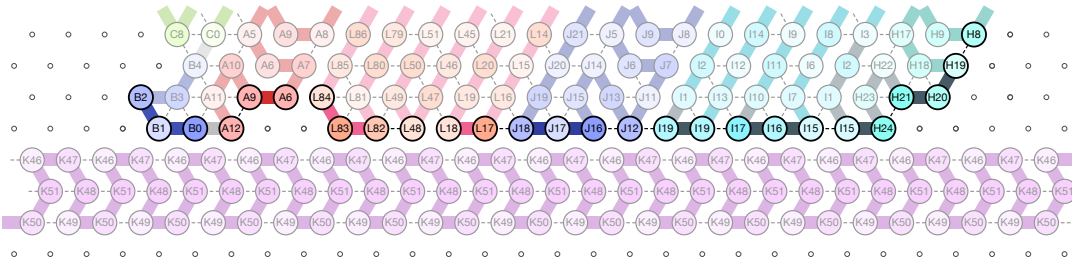
**Proof.** Figure 28(a) lists the only beads exposed and accessible from below above  $G \blacktriangleright \text{Read 1}$ . And Figure 28(b) lists all the possible  $\heartsuit$ -interactions between them (to the left) and the beads of the brick  $G \blacktriangleright \text{Read 1}$  zig-folding below (to the right).

According to the rule in Figure 28(b), besides the interactions at the 82 first beads at the very top-leftmost part of  $G \blacktriangleright \text{Read 1}$  (**K34..L55** in Figure 23, interactions in green in Figure 28(b)) and the 11 beads

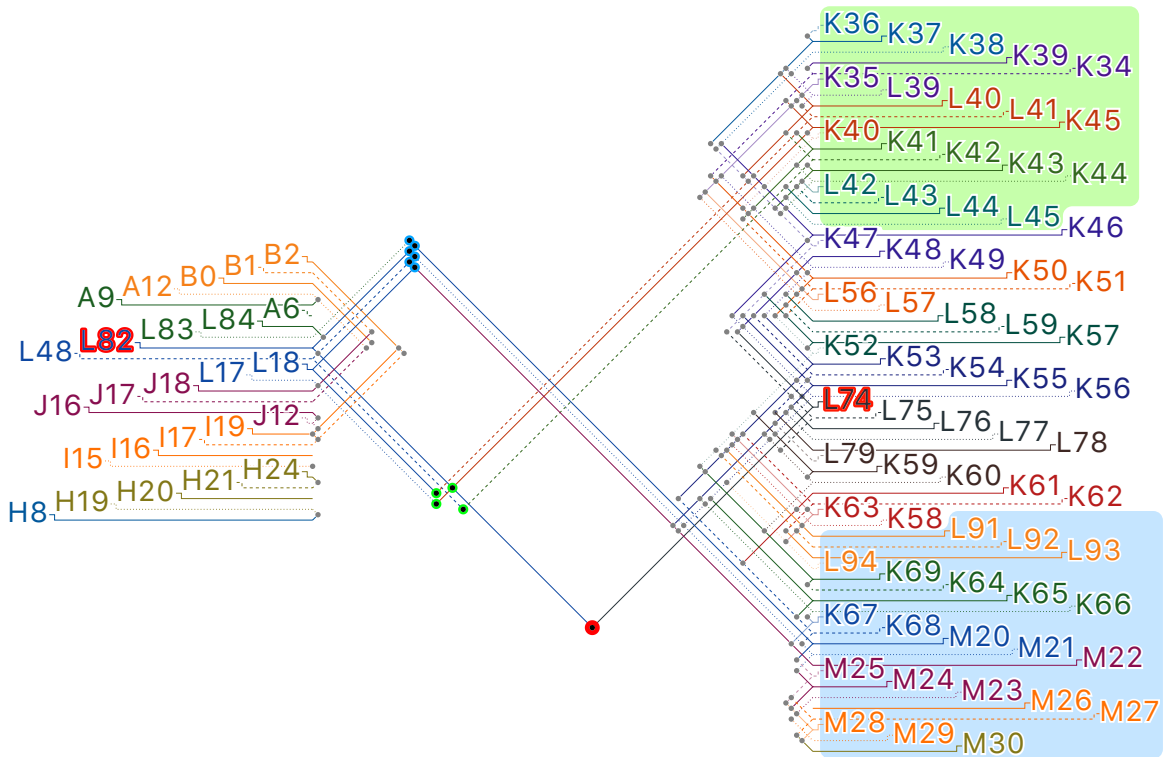


■ **Figure 27** The ♥-rule between the beads accessible from below of brick D►write and the beads that will get in touch with them from all the modules Zag-folding below.

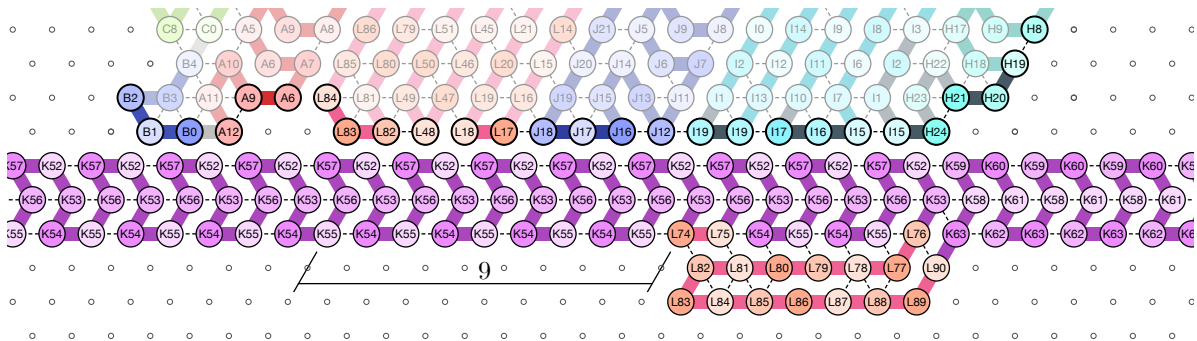
**XX:42 Proving the Turing Universality of Oritatami Co-Transcriptional Folding**



(a) The beads accessible when the brick  $G \blacktriangleright \text{Read } 1$  zig-folds itself.

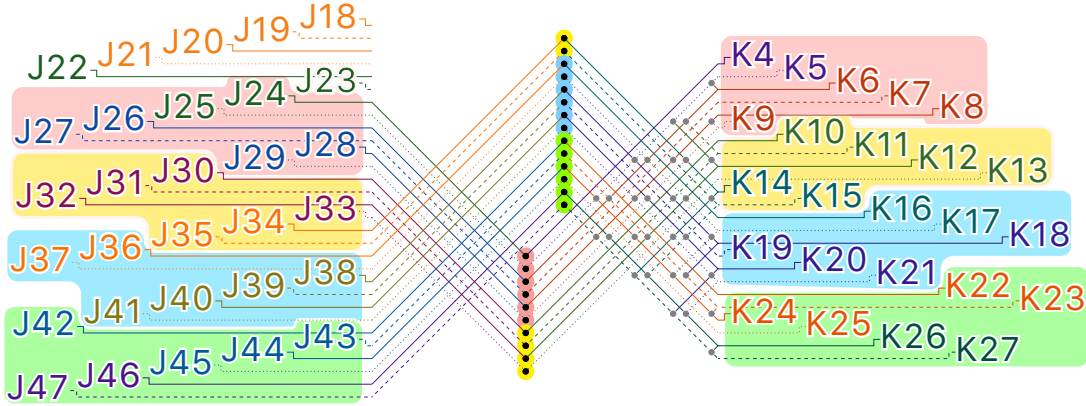


(b) The  $\heartsuit$ -rule for the beads accessible by the beads in  $G \blacktriangleright \text{Read } 1$  as it zig-folds.



(c) The closest bead L74 in brick  $G \blacktriangleright \text{Read } 1$  can get from one bead L82 above (case  $n = 1 \pmod 3$ ).





■ **Figure 29** ♥-rule between the two exponential segments in **F** and **G**. Note that each bead makes exactly one bond, with a bead of the same shade, red, blue, yellow or green (see Figure 22 and 23) and of the same rank within the shade.

at the very end of **G**►Read 1 (**M20..M30** in Figure 23, interactions in blue in Figure 28(b)), the only possible interaction between **G**►Read 1 and the already present beads above it is: **L82**♥**L74**. But **L74** appears only once in **G**►Read 1, at coordinates  $(w + 10 + 4k, 1 - h)$  (see Figure 23), while **L82** appears above **G**►Read 1 at coordinates  $(w + 1 + i(w + 6), 2 - h)$  for  $i = 0..n$ . The minimal  $x$ -distance between **L82** and **L74** is thus  $\min_{i=0..n} |9 + 4k - i(w + 6)|$ . But  $9 + 4k - i(w + 6) = 9 + 4(n - 1)(w + 6) / 6 - i(w + 6) = 9 + 2(n - 1 - 3i)(2(L + P) + 8)$ . It follows that the minimum difference in  $x$ -coordinate between **L82** and **L74** is:

- $17 + 2(L + P) \geq 41$ , if  $n = 0 \pmod 3$ ;
- $9$ , if  $n = 1 \pmod 3$ ; and
- $1 - 2(L + P) \leq -23$ , if  $n = 2 \pmod 3$ .

As a consequence, **L74** never gets close enough to interact with **L82** above (see Figure 28(c) for the closest situation). It follows that one only need to take into account the environnement for the folding of the top-leftmost and top-rightmost part of brick **G**►Read 1 (which is done next using proof-trees), the glider between them, zig-folds regardless of the beads above in the environment. ◀

► **Lemma 15** (**G**►Read 1 along **F**►Zig Up). *When **G** folds into the brick **G**►Read, no bead in **SegExpG** can make bonds with the beads in **F**►Zig Up nearby and thus folds regardless of the beads nearby (as a glider).*

**Proof.** Figure 29 lists the interactions between the beads in **SegExpG** and the beads in **SegExpF**: these are exactly **K**(4 +  $i$ )♥**J**(24 +  $i$ ) for  $i = 0..23$ ; in particular red-shaded beads **K4..K9** in **G** (resp. yellow, **K10..K15**; blue, **K16..K21**; and green, **K22..K27**) can only bond with beads of the same shade **J24..J29** in **F** (resp. **J30..J35**; **J36..J41**; **J42..J47**).

As shown on Figure 21 and 22 the  $y$ -coordinates explored by these beads are as follows when **G** zig-folds into **G**►Read 0 or **G**►Read 1:

- Red** : the  $y$ -coordinates of beads **J24..J29** in **F** belong to  $\{-40 - 3^{2j}, \dots, -35 - 3^{2j}\}$  for  $j \geq 1$ , while the corresponding beads **K4..K9** in **G** explore  $y$ -coordinates in  $\{-38 - 3^{2j'+1}, \dots, -34 - 3^{2j'+1}\}$  for  $j' \geq 1$ .
- Yellow** : the  $y$ -coordinates of beads **J30..J35** in **F** belong to  $\{-34 - 3^{2j+1}, \dots, -41 - 3^{2j}\}$  for  $j \geq 1$ , while the corresponding beads **K10..K15** in **G** explore  $y$ -coordinates in  $\{-36 - 3^{2j'+2}, \dots, -36 - 3^{2j'+1}\}$  for  $j' \geq 1$ .
- Blue** : the  $y$ -coordinates of beads **J36..J41** in **F** belong to  $\{-40 - 3^{2j+1}, \dots, -35 - 3^{2j+1}\}$  for  $j \geq 1$ , while the corresponding beads **K16..K21** in **G** explore  $y$ -coordinates in  $\{-38 - 3^{2j'}, \dots, -34 - 3^{2j'}\}$  for  $j' \geq 1$ .

**Green** : the  $y$ -coordinates of beads **J42..J47** in **F** belong to  $\{-34 - 3^{2j+2}, \dots, -41 - 3^{2j+1}\}$  for  $j \geq 1$ , while the corresponding beads **K2..K27** in **G** explore  $y$ -coordinates in  $\{-36 - 3^{2j'+1}, \dots, -36 - 3^{2j'}\}$  for  $j' \geq 1$ .

Now, as for all  $j \geq 1$  (with the notation,  $a \triangleleft b$  iff  $a \leq b - 2$ )

$$\begin{aligned} & -35 - 3^{2j+2} \triangleleft -38 - 3^{2j+1} \triangleleft -34 - 3^{2j+1} \triangleleft -40 - 3^{2j} \\ \text{and } & -36 - 3^{2j+1} \triangleleft -34 - 3^{2j+1} \triangleleft -41 - 3^{2j} \triangleleft -36 - 3^{2j} \\ \text{and } & -34 - 3^{2j+2} \triangleleft -40 - 3^{2j+1} \triangleleft -35 - 3^{2j+1} \triangleleft -38 - 3^{2j} \\ \text{and } & -41 - 3^{2j+1} \triangleleft -36 - 3^{2j+1} \triangleleft -36 - 3^{2j} \triangleleft -34 - 3^{2j} \end{aligned}$$

none of the (same-shade) interacting beads ever get close enough to each other and the beads in the segment **SegExpG** folds without making any bond (into a glider), regardless of the beads next to them in **F**►ZigUp when **G** zig-folds into brick **G**►Read. ◀

## 8.2 Proof-trees

A *proof-tree* is a compact representation of the enumeration of all the possible paths the molecule explores as it folds. Figure 30 presents the proof-tree for the folding of **G** when bouncing on a bump encoding a 0 in **G**►Read0. For the sake of readability, several paths are drawn in the same ball when they share the same beginning up to their last bond with the environment; then, as a sanity check, the grey number at the bottom left of the ball indicates how many paths are drawn in this ball. The black number in the top right corner of each ball indicates how many bonds are made by the paths with the environment. The ball(s) with the maximum number of bonds is(are) highlighted in black and go to the next round, together with the balls that place the first bead at the same position.

These proof-trees are automatically generated as the molecule folds. Each environment (surrounding + the three beads currently folding) is given a number (written #xxxx). When an already studied environment is encountered, the proof-tree is stopped, and the next (already encountered) environment number is written, allowing easy navigation in the proof — note that Figure 30 is an excerpt from a larger proof-tree and does not show its beginning nor its end, this is why the navigation tag cannot be observed in this figure.

The complete proof certificates may be found on the website:

<https://www.irif.fr/~nschaban/oritatami/prooftrees/>

---

### References

- 1 R. Agarwala, S. Batzoglou, V. Dançik, S. E. Decatur, S. Hannenhalli, M. Farach, S. Muthukrishnan, and S. Skiena. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the hp model. *Journal of Computational Biology*, 4(3):275–296, 1997.
- 2 J. Atkins and W. E. Hart. On the intractability of protein folding with a finite alphabet of amino acids. *Algorithmica*, 25(2–3):279–294, 1999.
- 3 Bonnie Berger and Tom Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 5(1):27–40, 1998.
- 4 Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.
- 5 Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding. *Journal of computational biology*, 5(3):423–465, 1998.
- 6 K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.
- 7 Kirsten L. Frieda and Steven M. Block. Direct observation of cotranscriptional folding in an adenine riboswitch. *Science*, 338(6105):397–400, 2012.
- 8 Peter Gács. Reliable cellular automata with self-organization. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 90–99. IEEE, 1997.

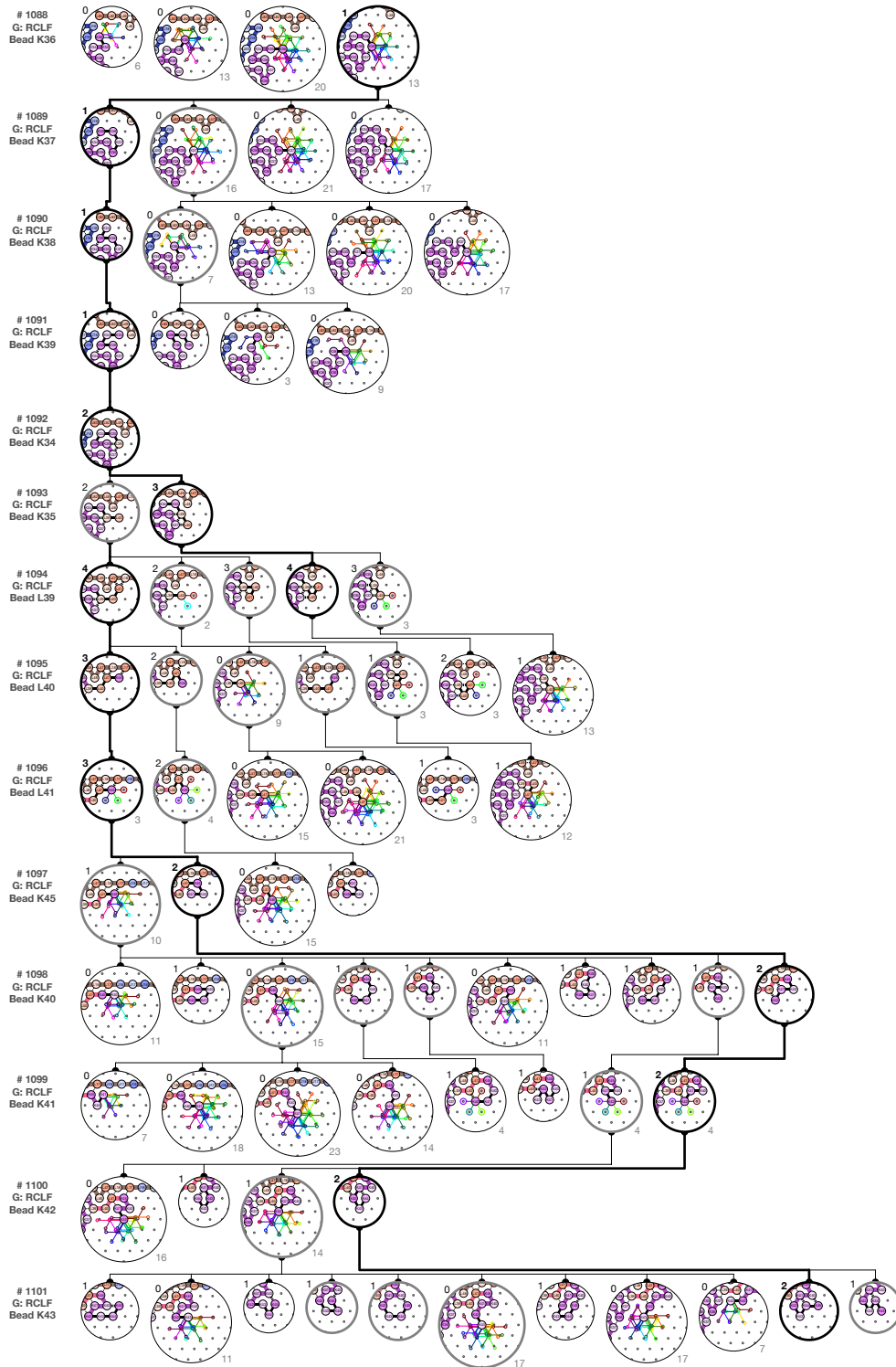
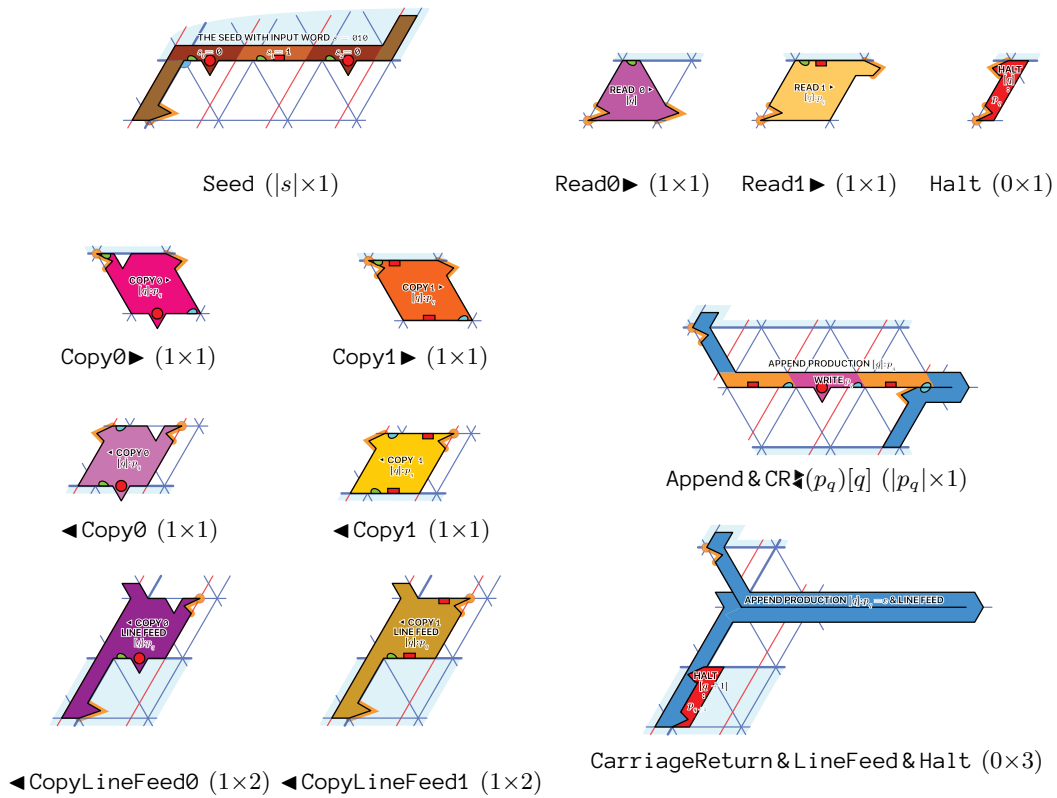


Figure 30 Excerpt from the proof-tree certificate for the folding of  $\mathbb{G}$  into  $\mathbb{G} \blacktriangleright \text{Read } 0$  when bouncing on a spike encoding a  $\emptyset$ .

- 9 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Programming biomolecules that fold greedily during transcription. In *Proc. 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *Leibniz International Proceedings in Informatics*, pages 43:1–43:14, 2016.
- 10 Cody Geary, Paul W. K. Rothmund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345:799–804, 2014.
- 11 Cody W. Geary, Pierre-Etienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Programming biomolecules that fold greedily during transcription. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, volume 58 of *LIPICs*, pages 43:1–43:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
- 12 Boyle J, Robillard G, and Kim S. Sequential folding of transfer RNA. a nuclear magnetic resonance study of successively longer tRNA fragments with a common 5' end. *J Mol Biol*, 139:601–625, 1980.
- 13 Lila Kari, Steffen Kopecki, Pierre-Étienne Meunier, Matthew J. Patitz, and Shinnosuke Seki. Binary pattern tile set synthesis is np-hard. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 1022–1034. Springer, 2015.
- 14 Turlough Neary. *Small universal Turing machines*. PhD thesis, NUI, Maynooth, 2008.
- 15 Turlough Neary and Damien Woods. P-completeness of cellular automaton rule 110. In *Proc. 33rd International Colloquium on Automata, Languages, and Programming (ICALP2006)*, LNCS 4051, pages 132–143. Springer, 2006.
- 16 A. Newman. A new algorithm for protein folding in the HP model. In *Proceedings of 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 867–884, 2002.
- 17 M. Paterson and T. Przytycka. On the complexity of string folding. In F. Meyer and B. Monien, editors, *ICALP 1996*, volume 1099 of *LNCS*, pages 658–669. Springer Berlin Heidelberg, 1996.
- 18 Paul W. K. Rothmund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *STOC*, pages 459–468, Portland, Oregon, United States, 2000. ACM.
- 19 Marcel Schmidt Am Busch, Anne Lopes, David Mignon, Thomas Gaillard, and Thomas Simonson. The inverse protein folding problem: protein design and structure prediction in the genomic era. In H. Treutlein J. Zeng, R. Zhang, editor, *Quantum Simulations of Materials and Biological Systems*, pages 121–140. Springer, 2012.
- 20 R. Unger and J. Moult. Finding the lowest free energy conformation of a protein is an NP-hard problem: proof and implications. *Bulletin of Mathematical Biology*, 55(6):1183–1198, 1993.
- 21 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, Caltech, June 1998.
- 22 Damien Woods and Turlough Neary. On the time complexity of 2-tag systems and small universal Turing machines. In *Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006)*, pages 439–448, 2006.

**A** Types of blocks

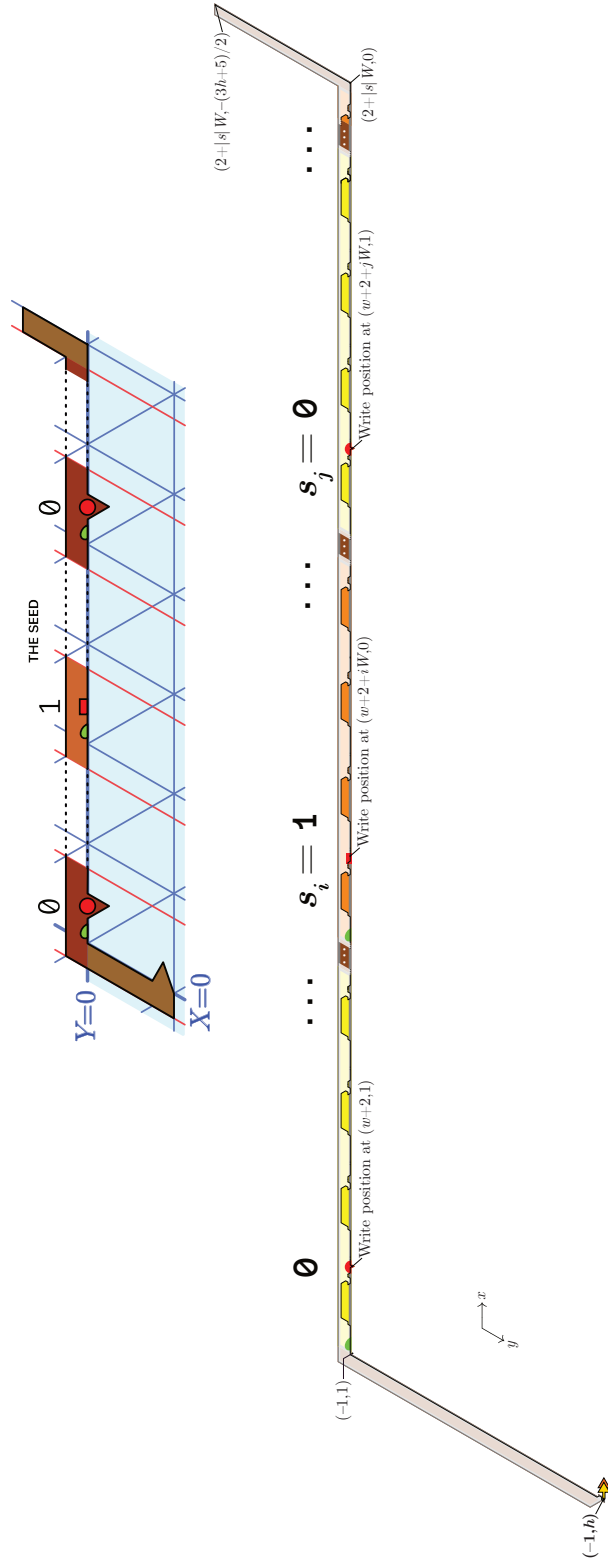


**Figure 31** The different type of blocks. The orange circles locate their anchors on the underlying triangle grid. The orange chevrons shows where they plug into each other. The current row of each block is shaded in white while the previous and the next rows are shaded in blue in the underlying triangular grid.

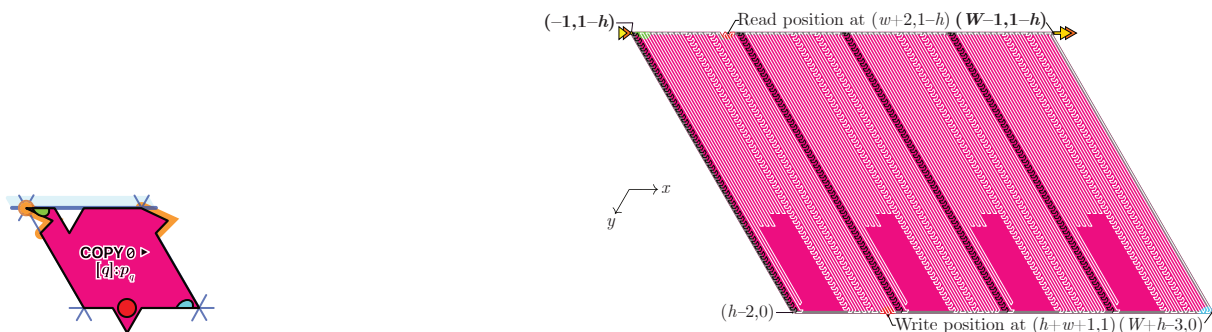
**B** Geometry of the blocks

The following figures 32–39 describe the geometry of each block (except for the Read  $\blacktriangleright$  blocks presented in Figure 5). Note that they display an idealized version of the real path inside them, omitting details (mainly, socks) that are vital for computing but irrelevant to the block general geometry – see Section 7 for the exact geometry of each brick inside each block.

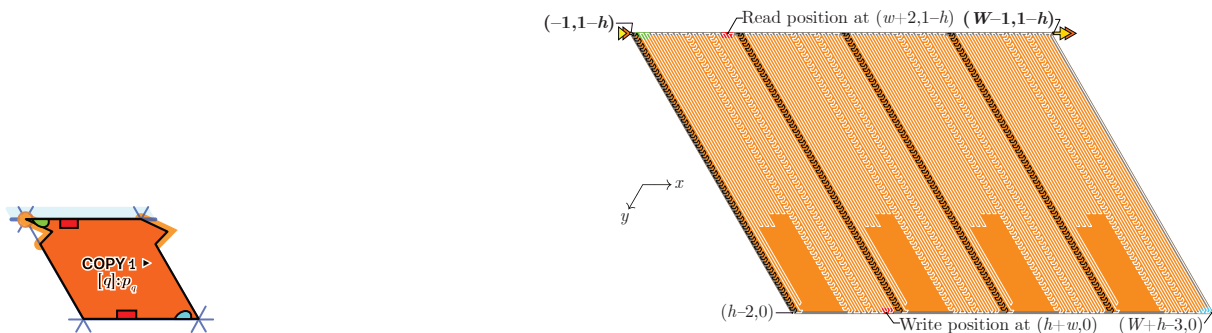
**Figure 32 Geometry of the Seed block.** This block encodes the initial word so that the oritami system simulates properly the corresponding tag system. It consists of placing the different letter at the expected Write positions. Its rightmost part consists in a northeast-bound segment signalling the end of the (initial) word. Its leftmost part ends at the position  $(0, 0)$  where the molecule will start folding the first zig-row.



■ **Figure 33 Geometry of the Copy blocks.** The Copy0 and Copy1 blocks have both the shape of a parallelogram with horizontal side length  $W$  and vertical side length  $h$ . For both, the next block will start folding at the top right corner, at  $(W, 0)$ . Note that the Copy0 and Copy1 blocks have identical internal structure apart from the line joining the two red areas at  $(w + 3, 0)$  and  $(h + w + 2, h)$ . Indeed, when folding, the part of the molecule located in the red area, either: (1) detects a spike on top (encoding a 0) and then folds into a dent on top which induces spike at the bottom (copying the 0 below, the block Copy0); or (2) folds flat (encoding a 1) on top which induces a flat folding at the bottom, copying the 1 from the top to the bottom of the Zig-row (the block Copy1).



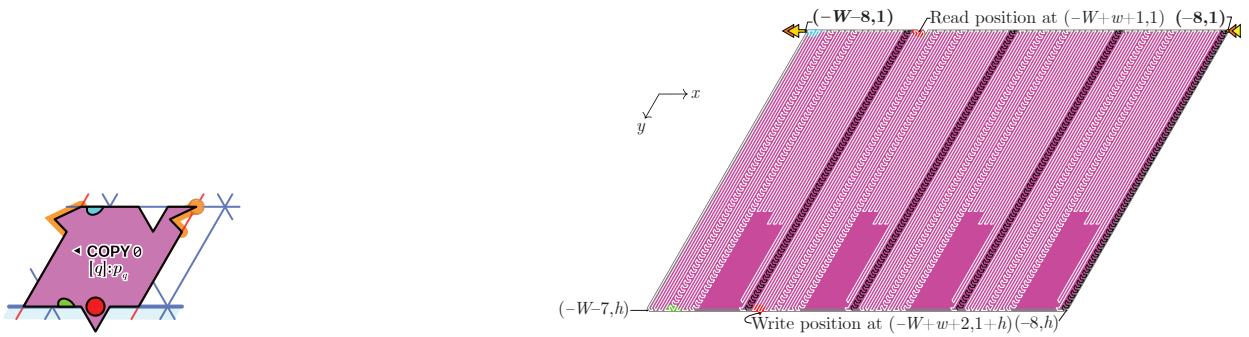
(a) The Copy0 block has a dent (an empty position) located at  $(w + 3, 0)$  (w.r.t. to its origin at the top left corner), in which plugs the spike of the block from the row above it, and which induces (when folding) a spike at the bottom at  $(h + w + 2, h)$ , copying the letter 0 from the top to the bottom of the Zig-row.



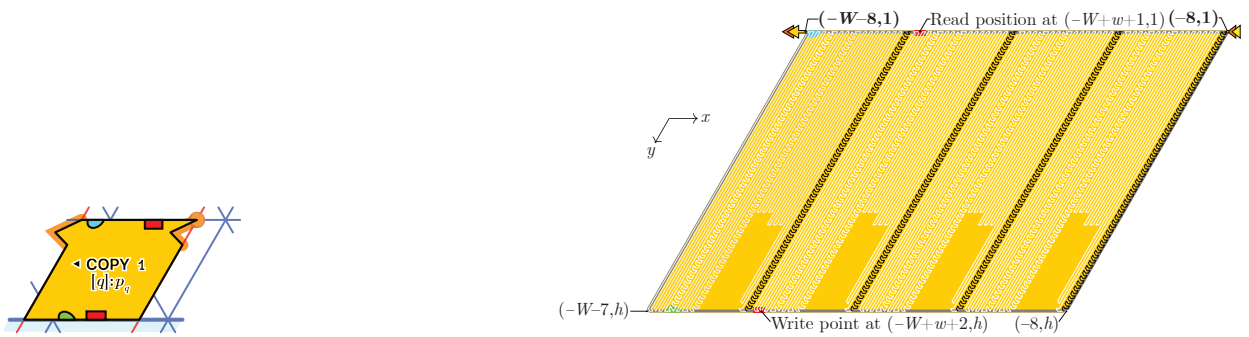
(b) The Copy1 block is flat at  $(w + 3, 0)$  (w.r.t. to its origin at the top left corner), which, aligned with a flat block above (encoding a 1), induces (when folding) a flat bottom at  $(h + w + 1, h - 1)$ , copying the letter 1 from the top to the bottom of the Zig-row.



■ **Figure 34 Geometry of the ◀Copy blocks.** The ◀Copy0 and ◀Copy1 blocks are the horizontal mirror images of the Copy0▶ and Copy1▶ blocks (see Figure 33).

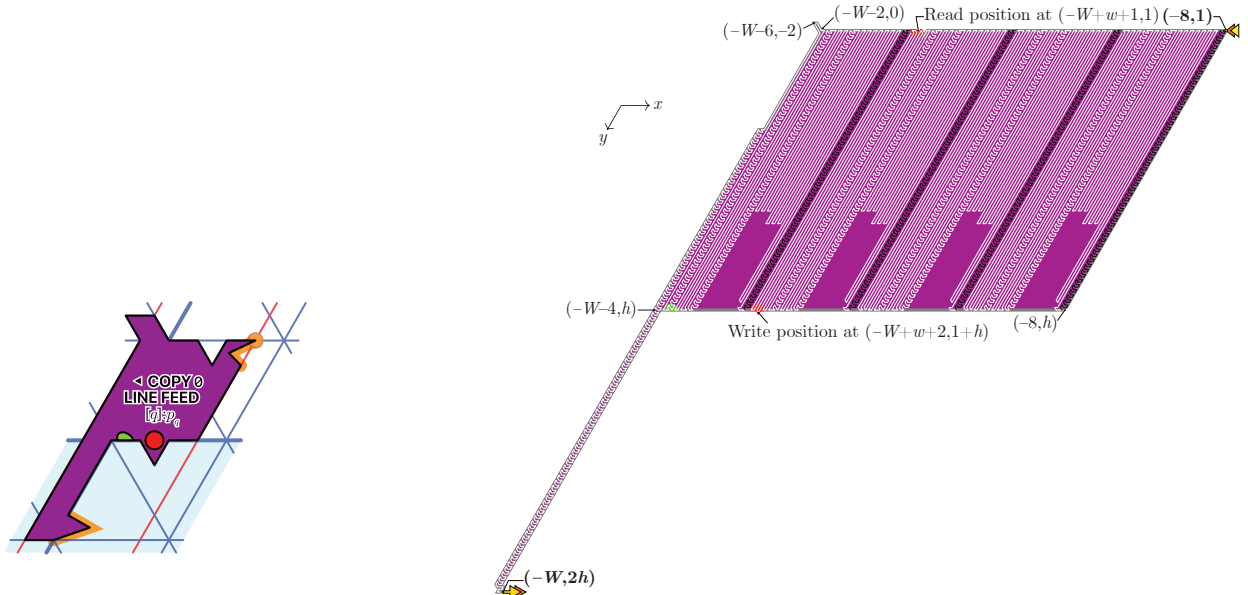


(a) The ◀Copy0 block is the horizontal mirror image of the Copy0▶ block (see Figure 33(a)).

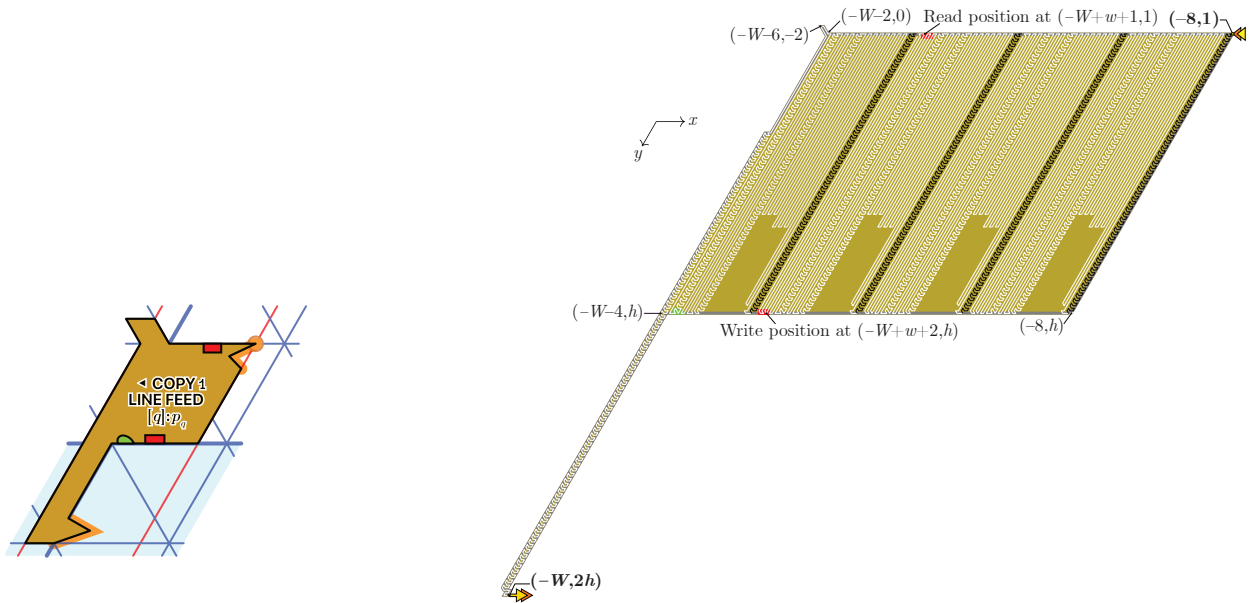


(b) The ◀Copy1 block is the horizontal mirror image of the Copy1▶ block (see Figure 33(b))

■ **Figure 35 Geometry of the ◀CopyLineFeed blocks.** These blocks adopt the shape of a  $(W - 6) \times h$ -parallelogram prolonged by an southwestbound "arm" hoping to the beginning of the next zig-row. Folding from right to left, the ◀CopyLineFeed blocks are identical to the ◀Copy blocks until position  $(-W + 6, 0)$  where it detects that there are no more blocks (encoding letter) in the row above (the detection of the absence of a block on top is made possible by the  $\Delta = 7$  horizontal offset between the zig- and zag-rows). Then, instead of completing a parallelogram, the end of the ◀CopyLineFeed blocks is attracted upwards and then folds into a southwestbound glider pattern to reach the opening position of the next zig-row. The next block will start folding at  $(-W + 8, 2h - 1)$ .

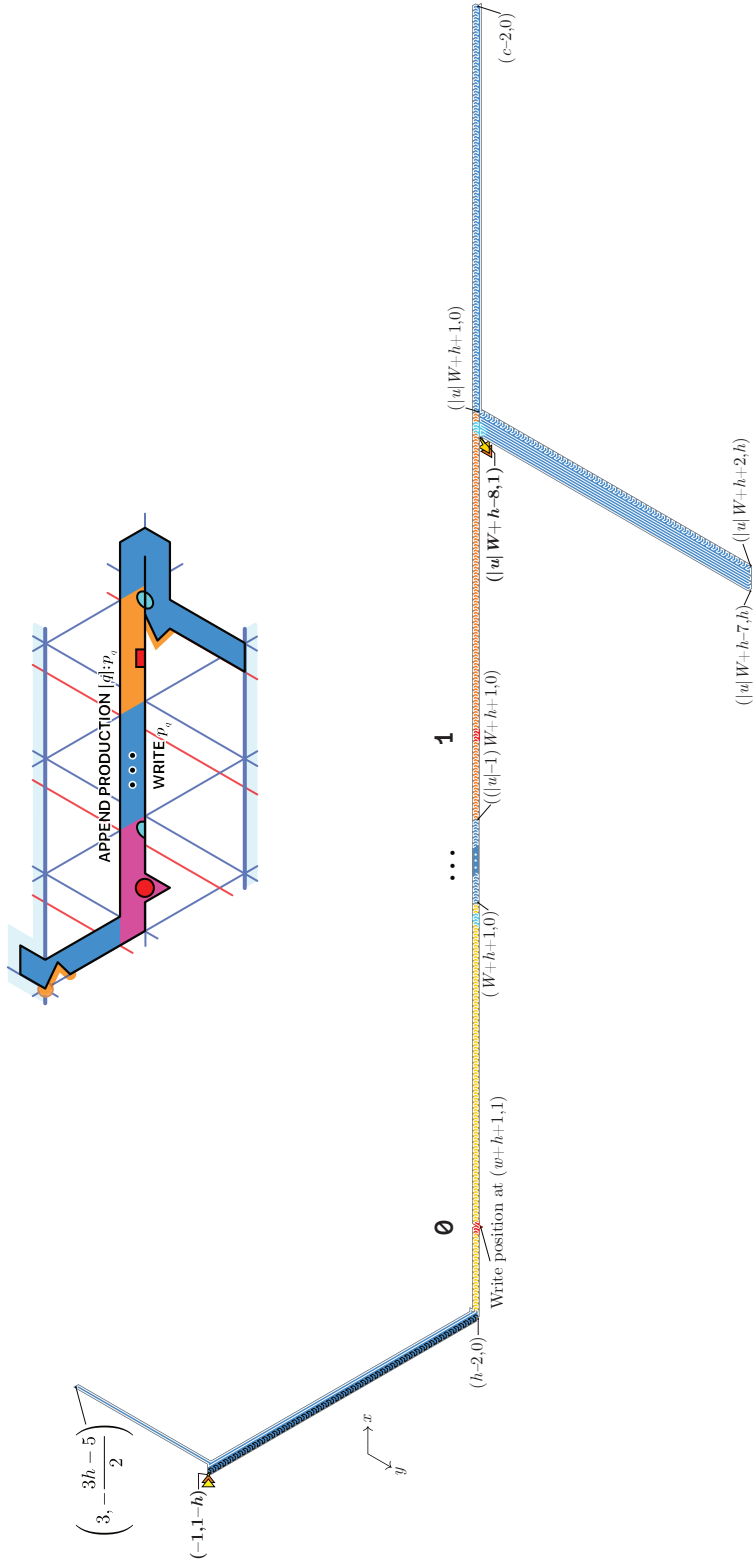


(a) The ◀CopyLineFeed0 block proceeds as ◀Copy0 to copy the spike encoding a 0 from the row above to the row below. It has a dent (an empty position) at  $(-W + w + 9, 0)$  in which plugs the spike (encoding a 0) of the block above. When folding, this dent induces a spike at the bottom at position  $(-W + w + 10, h)$  w.r.t. to the origin of the block. Note that the spike below is at position  $(w + 2, -h + 1)$  w.r.t. to the beginning of the following block, which is consistent with the position of the dent in the Read0▶ block (see Figure 5(a)).

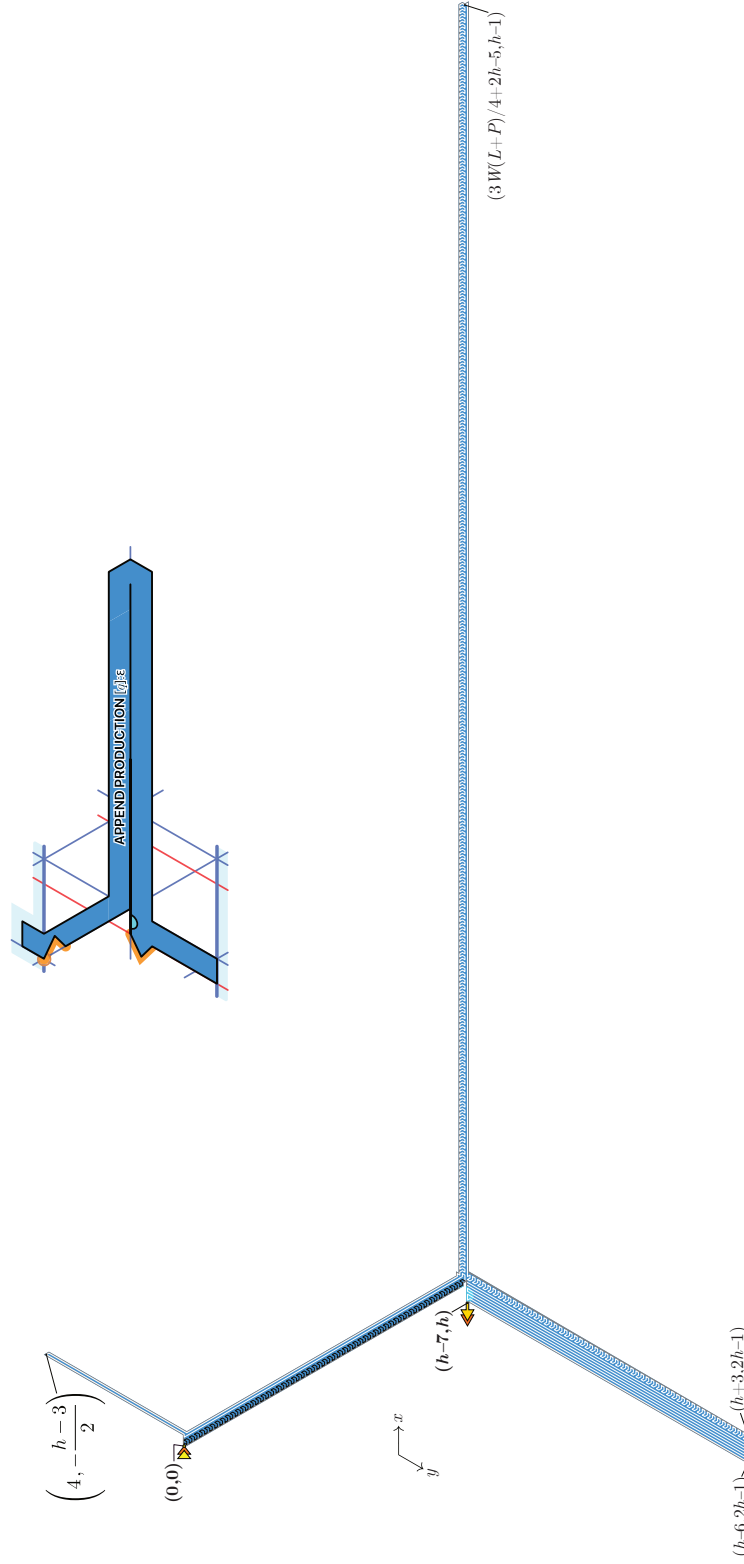


(b) The ◀CopyLineFeed1 block.

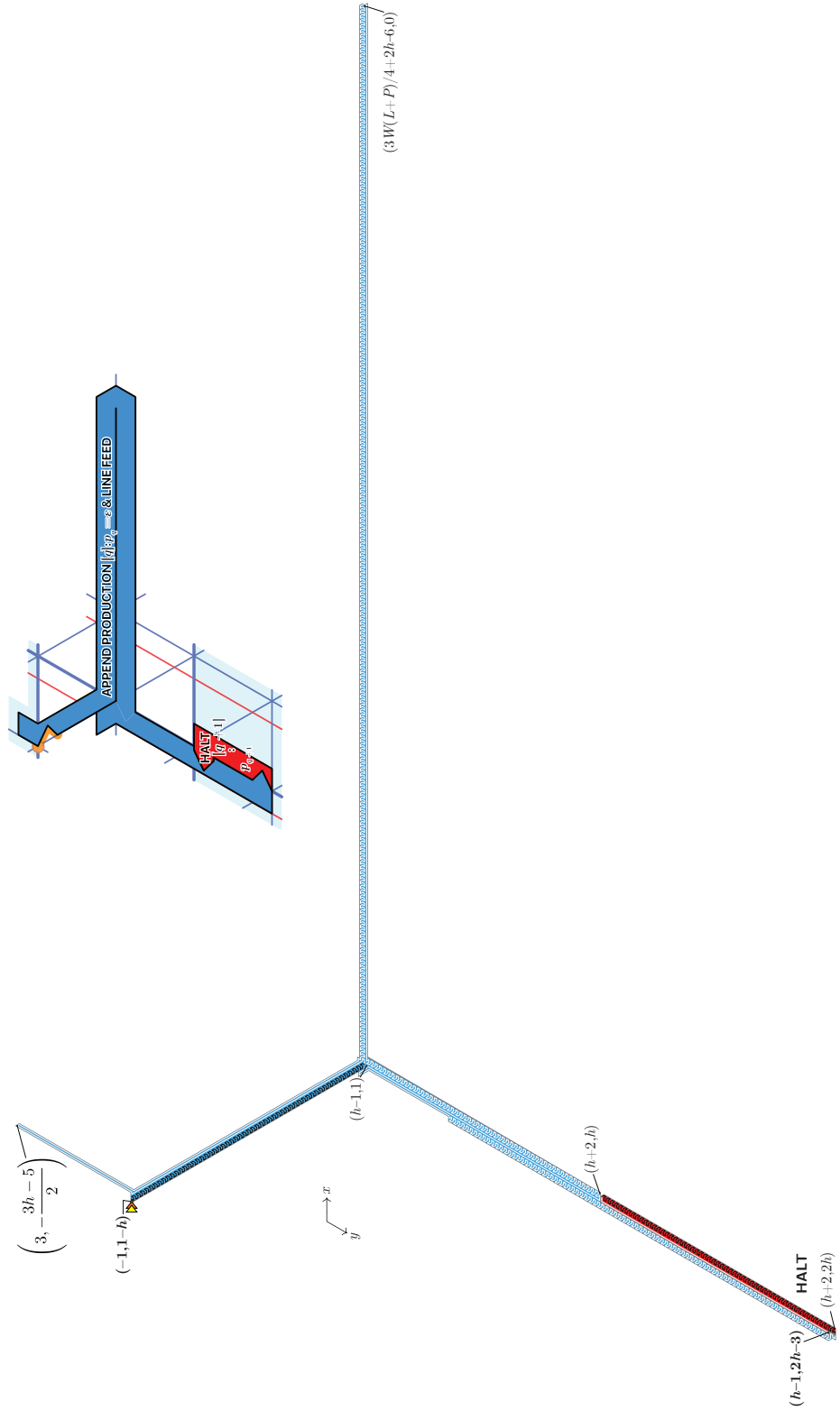
■ **Figure 36 Geometry of the Append & CR $\uparrow$ ( $u$ ) blocks.** The folding into this block is triggered by the absence of a block in row above (indicating the end of the word). It has one northeastbound 2-beads wide arm climbing along the east side of the block in the row above then a southeastbound 4-beads wide arm stopping at the bottom of the current zig-row. Then, the block consists in an 3-beads wide  $|u|W$ -beads long eastbound glider path going along the bottom of the current zig-row and encoding each letter of  $u$ : the path contains a spike (below, and a dent on top) for each  $u_j = \emptyset$  at position  $(jW + w + h + 2, h)$  (1s are encoded by the absence of spike). It then expanded upto position  $(c - 1, h - 1)$  where  $c = \frac{3}{4}W(L - |u| + P) + 2h - 4$  and go back to its origin and grows a 10-beads wide  $h$ -beads high southwestbound arm opening the next zag-row to end at the position  $(|u|W + h - 8, h)$ , at the top right corner of the upcoming zag-row. The next block will start at  $(|u|W + h - 7, h)$ .



■ **Figure 37 Geometry of the Append & CR $\uparrow$ ( $\epsilon$ ) block.** This block is the special case of Figure 36 where  $u = \epsilon$ . It is given for clarity.



■ **Figure 38 Geometry of the CarriageReturn & LineFeed & Halt block.** This block is identical to the Append & CR & LF block until it reaches position  $(h, h)$ . Then, when folding, it detects the absence of a block above which indicates that the current word is empty. It then folds as the leftmost part of the CopyLineFeed blocks (see Figure 35) to open a new zig-row at  $(h+1, 3h-1)$ . If then goes up to  $(h+3, 2h+1)$ . And as there are no block on the zag-row above, it is attracted inside itself and gets blocked at  $(h, 3h-4)$ .





# XX:56 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

E13	♥	D16	E23	♥	E00	E41	♥	E36	F00	♥	I00	F03	♥	I03	F06	♥	F17	F08	♥	I06	F11	♥	I01	F14	♥	F21	F17	♥	H10
E13	♥	D57	E23	♥	E18	E42	♥	D18	F00	♥	I01	F03	♥	I04	F06	♥	F50	F08	♥	I07	F11	♥	I02	F14	♥	G32	F17	♥	H14
E13	♥	D58	E24	♥	E22	E42	♥	D32	F00	♥	I02	F03	♥	I05	F06	♥	F51	F08	♥	I08	F11	♥	I03	F14	♥	I00	F17	♥	I00
E13	♥	E10	E24	♥	E29	E42	♥	D33	F00	♥	I03	F03	♥	I06	F06	♥	G01	F08	♥	I09	F11	♥	I04	F14	♥	I01	F17	♥	I01
E13	♥	F26	E24	♥	E47	E42	♥	D50	F00	♥	I04	F03	♥	I07	F06	♥	G02	F08	♥	I10	F11	♥	I05	F14	♥	I02	F17	♥	I02
E13	♥	F27	E25	♥	E22	E42	♥	E04	F00	♥	I05	F03	♥	I08	F06	♥	G06	F08	♥	I11	F11	♥	I06	F14	♥	I03	F17	♥	I03
E14	♥	A00	E25	♥	E46	E42	♥	E05	F00	♥	I06	F03	♥	I09	F06	♥	G10	F08	♥	I12	F11	♥	I07	F14	♥	I04	F17	♥	I04
E14	♥	D56	E26	♥	A00	E42	♥	E47	F00	♥	I07	F03	♥	I10	F06	♥	G13	F08	♥	I13	F11	♥	I08	F14	♥	I05	F17	♥	I05
E14	♥	D57	E26	♥	E21	E43	♥	D31	F00	♥	I08	F03	♥	I11	F06	♥	G14	F08	♥	I14	F11	♥	I09	F14	♥	I06	F17	♥	I06
E14	♥	E09	E26	♥	E33	E43	♥	D32	F00	♥	I09	F03	♥	I12	F06	♥	G17	F09	♥	C09	F11	♥	I10	F14	♥	I07	F17	♥	I07
E14	♥	E10	E27	♥	A01	E43	♥	D43	F00	♥	I10	F03	♥	I13	F06	♥	G18	F09	♥	E43	F11	♥	I11	F14	♥	I08	F17	♥	I08
E14	♥	E21	E27	♥	E32	E43	♥	D44	F00	♥	I11	F03	♥	I14	F06	♥	G22	F09	♥	F02	F11	♥	I12	F14	♥	I09	F17	♥	I09
E14	♥	E32	E27	♥	E44	E43	♥	E28	F00	♥	I12	F04	♥	B02	F06	♥	G25	F09	♥	F14	F11	♥	I13	F14	♥	I10	F17	♥	I10
E15	♥	A01	E28	♥	E18	E43	♥	F08	F00	♥	I13	F04	♥	C14	F06	♥	G26	F09	♥	G14	F11	♥	I14	F14	♥	I11	F17	♥	I11
E15	♥	D55	E28	♥	E43	E43	♥	F09	F00	♥	I14	F04	♥	F19	F06	♥	G30	F09	♥	I00	F12	♥	F17	F14	♥	I12	F17	♥	I12
E15	♥	D56	E29	♥	A00	E44	♥	A00	F00	♥	J00	F00	♥	F42	F06	♥	G34	F09	♥	I01	F12	♥	F34	F14	♥	I13	F17	♥	I13
E15	♥	E08	E29	♥	E18	E44	♥	D62	F00	♥	J01	F04	♥	G19	F06	♥	G37	F09	♥	I02	F12	♥	F35	F14	♥	I14	F17	♥	I14
E15	♥	E20	E29	♥	E24	E44	♥	E27	F01	♥	C11	F04	♥	I00	F06	♥	G38	F09	♥	I03	F12	♥	F51	F15	♥	F08	F18	♥	F23
E15	♥	E31	E30	♥	E35	E44	♥	E39	F01	♥	D21	F04	♥	I01	F06	♥	G42	F09	♥	I04	F12	♥	G00	F15	♥	F20	F18	♥	F28
E15	♥	E32	E30	♥	E41	E44	♥	L18	F01	♥	F22	F04	♥	I02	F06	♥	G46	F09	♥	I05	F12	♥	G04	F15	♥	F31	F18	♥	F29
E16	♥	D12	E31	♥	D44	E45	♥	A01	F01	♥	F46	F04	♥	I03	F06	♥	H01	F09	♥	I06	F12	♥	G08	F15	♥	F32	F18	♥	F51
E16	♥	D13	E31	♥	E15	E45	♥	D31	F01	♥	G22	F04	♥	I04	F06	♥	H05	F09	♥	I07	F12	♥	G12	F15	♥	G31	F18	♥	G01
E16	♥	D55	E31	♥	E16	E45	♥	D61	F01	♥	I00	F04	♥	I05	F06	♥	H09	F09	♥	I08	F12	♥	G16	F15	♥	I00	F18	♥	G02
E16	♥	E07	E31	♥	E40	E45	♥	D62	F01	♥	I01	F04	♥	I06	F06	♥	H13	F09	♥	I09	F12	♥	G20	F15	♥	I01	F18	♥	G06
E16	♥	E31	E32	♥	A00	E45	♥	E02	F01	♥	I02	F04	♥	I07	F06	♥	H19	F09	♥	I10	F12	♥	G24	F15	♥	I02	F18	♥	G10
E16	♥	F24	E32	♥	D43	E45	♥	E38	F01	♥	I03	F04	♥	I08	F06	♥	I00	F09	♥	I11	F12	♥	G28	F15	♥	I03	F18	♥	G13
E16	♥	F35	E32	♥	D44	E46	♥	D28	F01	♥	I04	F04	♥	I09	F06	♥	I01	F09	♥	I12	F12	♥	G32	F15	♥	I04	F18	♥	G14
E17	♥	E06	E32	♥	E14	E46	♥	D29	F01	♥	I05	F04	♥	I10	F06	♥	I02	F09	♥	I13	F12	♥	G34	F15	♥	I05	F18	♥	G18
E17	♥	E12	E32	♥	E15	E46	♥	D40	F01	♥	I06	F04	♥	I11	F06	♥	I03	F09	♥	I14	F12	♥	G35	F15	♥	I06	F18	♥	G22
E18	♥	D18	E32	♥	E27	E46	♥	D41	F01	♥	I07	F04	♥	I12	F06	♥	I04	F10	♥	C14	F12	♥	G36	F15	♥	I07	F18	♥	G25
E18	♥	D43	E32	♥	E39	E46	♥	D60	F01	♥	I08	F04	♥	I13	F06	♥	I05	F10	♥	F13	F12	♥	G40	F15	♥	I08	F18	♥	G26
E18	♥	D44	E33	♥	A01	E46	♥	D61	F01	♥	I09	F04	♥	I14	F06	♥	I06	F10	♥	F14	F12	♥	G44	F15	♥	I09	F18	♥	G28
E18	♥	D45	E33	♥	E26	E46	♥	E00	F01	♥	I10	F05	♥	B01	F06	♥	I07	F10	♥	F36	F12	♥	H03	F15	♥	I10	F18	♥	G30
E18	♥	E23	E33	♥	E38	E46	♥	E01	F01	♥	I11	F05	♥	C13	F06	♥	I08	F10	♥	G13	F12	♥	H04	F15	♥	I11	F18	♥	G34
E18	♥	E28	E33	♥	L17	E46	♥	E25	F01	♥	I12	F05	♥	E46	F06	♥	I09	F10	♥	I00	F12	♥	H07	F15	♥	I12	F18	♥	G37
E18	♥	E29	E34	♥	E12	E46	♥	F05	F01	♥	I13	F05	♥	F00	F06	♥	I10	F10	♥	I01	F12	♥	H11	F15	♥	I13	F18	♥	G38
E19	♥	D09	E34	♥	E37	E46	♥	F06	F01	♥	I14	F05	♥	F42	F06	♥	I11	F10	♥	I02	F12	♥	H15	F15	♥	I14	F18	♥	G42
E19	♥	D10	E34	♥	E38	E47	♥	D19	F02	♥	B03	F05	♥	G02	F06	♥	I12	F10	♥	I03	F12	♥	H17	F16	♥	F07	F18	♥	G46
E19	♥	D32	E35	♥	D59	E47	♥	D28	F02	♥	B04	F05	♥	G06	F06	♥	I13	F10	♥	I04	F12	♥	I00	F16	♥	G30	F18	♥	H01
E19	♥	D33	E35	♥	E12	E47	♥	D60	F02	♥	C10	F05	♥	G10	F06	♥	I14	F10	♥	I05	F12	♥	I01	F16	♥	I00	F18	♥	H05
E19	♥	D42	E35	♥	E30	E47	♥	E24	F02	♥	D20	F05	♥	G14	F07	♥	C11	F10	♥	I06	F12	♥	I02	F16	♥	I01	F18	♥	H09
E19	♥	D43	E36	♥	D52	E47	♥	E42	F02	♥	E37	F05	♥	G18	F07	♥	F16	F10	♥	I07	F12	♥	I03	F16	♥	I02	F18	♥	H13
E19	♥	E04	E36	♥	D58	F02	♥	F09	F05	♥	G22	F07	♥	F39	F10	♥	I08	F12	♥	I04	F16	♥	I03	F16	♥	I03	F18	♥	H19
E19	♥	F32	E36	♥	D59	F02	♥	F45	F05	♥	G26	F07	♥	F51	F10	♥	I09	F12	♥	I05	F16	♥	I04	F16	♥	I04	F18	♥	I00
E19	♥	F33	E36	♥	E10	F02	♥	G21	F05	♥	G30	F07	♥	G16	F10	♥	I10	F12	♥	I06	F16	♥	I05	F16	♥	I05	F18	♥	I01
E20	♥	A00	E36	♥	E11	F02	♥	I00	F05	♥	G34	F07	♥	I00	F10	♥	I11	F12	♥	I07	F16	♥	I06	F16	♥	I06	F18	♥	I02
E20	♥	D62	E36	♥	E41	F02	♥	I01	F05	♥	G38	F07	♥	I01	F10	♥	I12	F12	♥	I08	F16	♥	I07	F16	♥	I07	F18	♥	I03
E20	♥	E03	E37	♥	D57	F02	♥	I02	F05	♥	G42	F07	♥	I02	F10	♥	I13	F12	♥	I09	F16	♥	I08	F16	♥	I08	F18	♥	I04
E20	♥	E15	E37	♥	D58	F02	♥	I03	F05	♥	G46	F07	♥	I03	F10	♥	I14	F12	♥	I10	F16	♥	I09	F16	♥	I09	F18	♥	I05
E20	♥	L18	E37	♥	E34	F02	♥	I04	F05	♥	H02	F07	♥	I04	F11	♥	C13	F12	♥	I11	F16	♥	I10	F16	♥	I10	F18	♥	I06
E21	♥	A01	E37	♥	F02	F02	♥	I05	F05	♥	H06	F07	♥	I05	F11	♥	E40	F12	♥	I12	F16	♥	I11	F16	♥	I11	F18	♥	I07
E21	♥	D42	E37	♥	F03	F02	♥	I06	F05	♥	H10	F07	♥	I06	F11	♥	F06	F12	♥	I13	F16	♥	I12	F16	♥	I12	F18	♥	I08
E21	♥	D61	E38	♥	A00	F02	♥	I07	F05	♥	H14	F07	♥	I07	F11	♥	F36	F12	♥	I14	F16	♥	I13	F16	♥	I13	F18	♥	I09
E21	♥	D62	E38	♥	D56	F02	♥	I08	F05	♥	I00	F07	♥	I08	F11	♥	G00	F13	♥	F10	F16	♥	I14	F16	♥	I14	F18	♥	I10
E21	♥	E14	E38	♥	D57	F02	♥	I09	F05	♥	I01	F07	♥	I09	F11	♥	G04	F13	♥	G33	F17	♥	F06	F16	♥	F06	F18	♥	I11
E21	♥	E26	E38	♥	E08	F02	♥	I10	F05	♥	I02	F07	♥	I10	F11	♥	G08	F13	♥	I00	F17	♥	F12	F16	♥	F12	F18	♥	I12
E22	♥	D06	E38	♥	E33	F02	♥	I11	F05	♥	I03	F07	♥	I11	F11	♥	G12	F13	♥	I01	F17	♥	F51	F16	♥	F17	F18	♥	I13
E22	♥	D07	E38	♥	E34	F02	♥	I12	F05	♥	I04	F07	♥	I12	F11	♥	G16	F13	♥	I02	F17	♥	G02	F16	♥	G02	F18	♥	I14
E22	♥	D29	E38	♥	E45	F02	♥	I13	F05	♥	I05	F07	♥	I13	F11	♥	G20	F13	♥	I03	F17	♥	G06	F16	♥	G06	F19	♥	F04
E22	♥	D30	E39	♥	A01	F02	♥	I14	F05	♥	I06	F07	♥	I14	F11	♥	G24	F13	♥	I04	F17	♥	G10	F16	♥	G10	F19		



F19 ♡ I11	F23 ♡ G32	F25 ♡ I01	F28 ♡ I13	F30 ♡ I09	F34 ♡ I04	F36 ♡ I01	F40 ♡ I00	F42 ♡ H05	F45 ♡ I13
F19 ♡ I12	F23 ♡ G36	F25 ♡ I02	F28 ♡ I14	F30 ♡ I10	F34 ♡ I05	F36 ♡ I02	F40 ♡ I01	F42 ♡ H09	F45 ♡ I14
F19 ♡ I13	F23 ♡ G40	F25 ♡ I03	F29 ♡ E22	F30 ♡ I11	F34 ♡ I06	F36 ♡ I03	F40 ♡ I02	F42 ♡ H13	F46 ♡ F00
F19 ♡ I14	F23 ♡ G44	F25 ♡ I04	F29 ♡ F18	F30 ♡ I12	F34 ♡ I07	F36 ♡ I04	F40 ♡ I03	F42 ♡ H19	F46 ♡ F01
F20 ♡ F03	F23 ♡ H00	F25 ♡ I05	F29 ♡ F24	F30 ♡ I13	F34 ♡ I08	F36 ♡ I05	F40 ♡ I04	F42 ♡ I00	F46 ♡ F25
F20 ♡ F15	F23 ♡ H04	F25 ♡ I06	F29 ♡ G02	F30 ♡ I14	F34 ♡ I09	F36 ♡ I06	F40 ♡ I05	F42 ♡ I01	F46 ♡ G00
F20 ♡ F27	F23 ♡ H08	F25 ♡ I07	F29 ♡ G06	F31 ♡ F15	F34 ♡ I10	F36 ♡ I07	F40 ♡ I06	F42 ♡ I02	F46 ♡ I00
F20 ♡ G26	F23 ♡ H12	F25 ♡ I08	F29 ♡ G10	F31 ♡ F40	F34 ♡ I11	F36 ♡ I08	F40 ♡ I07	F42 ♡ I03	F46 ♡ I01
F20 ♡ I00	F23 ♡ H16	F25 ♡ I09	F29 ♡ G14	F31 ♡ F51	F34 ♡ I12	F36 ♡ I09	F40 ♡ I08	F42 ♡ I04	F46 ♡ I02
F20 ♡ I01	F23 ♡ H18	F25 ♡ I10	F29 ♡ G18	F31 ♡ G40	F34 ♡ I13	F36 ♡ I10	F40 ♡ I09	F42 ♡ I05	F46 ♡ I03
F20 ♡ I02	F23 ♡ I00	F25 ♡ I11	F29 ♡ G22	F31 ♡ I00	F34 ♡ I14	F36 ♡ I11	F40 ♡ I10	F42 ♡ I06	F46 ♡ I04
F20 ♡ I03	F23 ♡ I01	F25 ♡ I12	F29 ♡ G26	F31 ♡ I01	F35 ♡ E16	F36 ♡ I12	F40 ♡ I11	F42 ♡ I07	F46 ♡ I05
F20 ♡ I04	F23 ♡ I02	F25 ♡ I13	F29 ♡ G30	F31 ♡ I02	F35 ♡ F12	F36 ♡ I13	F40 ♡ I12	F42 ♡ I08	F46 ♡ I06
F20 ♡ I05	F23 ♡ I03	F25 ♡ I14	F29 ♡ G34	F31 ♡ I03	F35 ♡ F30	F36 ♡ I14	F40 ♡ I13	F42 ♡ I09	F46 ♡ I07
F20 ♡ I06	F23 ♡ I04	F26 ♡ D20	F29 ♡ G38	F31 ♡ I04	F35 ♡ G00	F37 ♡ F34	F40 ♡ I14	F42 ♡ I10	F46 ♡ I08
F20 ♡ I07	F23 ♡ I05	F26 ♡ E13	F29 ♡ G42	F31 ♡ I05	F35 ♡ G04	F37 ♡ G09	F41 ♡ F30	F42 ♡ I11	F46 ♡ I09
F20 ♡ I08	F23 ♡ I06	F26 ♡ F21	F29 ♡ G46	F31 ♡ I06	F35 ♡ G08	F37 ♡ I00	F41 ♡ F36	F42 ♡ I12	F46 ♡ I10
F20 ♡ I09	F23 ♡ I07	F26 ♡ F33	F29 ♡ H02	F31 ♡ I07	F35 ♡ G12	F37 ♡ I01	F41 ♡ F51	F42 ♡ I13	F46 ♡ I11
F20 ♡ I10	F23 ♡ I08	F26 ♡ G45	F29 ♡ H06	F31 ♡ I08	F35 ♡ G16	F37 ♡ I02	F41 ♡ G02	F42 ♡ I14	F46 ♡ I12
F20 ♡ I11	F23 ♡ I09	F26 ♡ I00	F29 ♡ H10	F31 ♡ I09	F35 ♡ G20	F37 ♡ I03	F41 ♡ G05	F43 ♡ F03	F46 ♡ I13
F20 ♡ I12	F23 ♡ I10	F26 ♡ I01	F29 ♡ H14	F31 ♡ I10	F35 ♡ G24	F37 ♡ I04	F41 ♡ G06	F43 ♡ F28	F46 ♡ I14
F20 ♡ I13	F23 ♡ I11	F26 ♡ I02	F29 ♡ I00	F31 ♡ I11	F35 ♡ G28	F37 ♡ I05	F41 ♡ G10	F43 ♡ G03	F47 ♡ F24
F20 ♡ I14	F23 ♡ I12	F26 ♡ I03	F29 ♡ I01	F31 ♡ I12	F35 ♡ G32	F37 ♡ I06	F41 ♡ G14	F43 ♡ I00	F47 ♡ F42
F21 ♡ F14	F23 ♡ I13	F26 ♡ I04	F29 ♡ I02	F31 ♡ I13	F35 ♡ G36	F37 ♡ I07	F41 ♡ G18	F43 ♡ I01	F47 ♡ G00
F21 ♡ F26	F23 ♡ I14	F26 ♡ I05	F29 ♡ I03	F31 ♡ I14	F35 ♡ G40	F37 ♡ I08	F41 ♡ G22	F43 ♡ I02	F47 ♡ G04
F21 ♡ G25	F24 ♡ D21	F26 ♡ I06	F29 ♡ I04	F32 ♡ E19	F35 ♡ G44	F37 ♡ I09	F41 ♡ G26	F43 ♡ I03	F47 ♡ G08
F21 ♡ I00	F24 ♡ E16	F26 ♡ I07	F29 ♡ I05	F32 ♡ F15	F35 ♡ H00	F37 ♡ I10	F41 ♡ G30	F43 ♡ I04	F47 ♡ G12
F21 ♡ I01	F24 ♡ F22	F26 ♡ I08	F29 ♡ I06	F32 ♡ F27	F35 ♡ H04	F37 ♡ I11	F41 ♡ G34	F43 ♡ I05	F47 ♡ G16
F21 ♡ I02	F24 ♡ F29	F26 ♡ I09	F29 ♡ I07	F32 ♡ F39	F35 ♡ H08	F37 ♡ I12	F41 ♡ G38	F43 ♡ I06	F47 ♡ G20
F21 ♡ I03	F24 ♡ F47	F26 ♡ I10	F29 ♡ I08	F32 ♡ G39	F35 ♡ H12	F37 ♡ I13	F41 ♡ G42	F43 ♡ I07	F47 ♡ G24
F21 ♡ I04	F24 ♡ F51	F26 ♡ I11	F29 ♡ I09	F32 ♡ I00	F35 ♡ H16	F37 ♡ I14	F41 ♡ G46	F43 ♡ I08	F47 ♡ G28
F21 ♡ I05	F24 ♡ G00	F26 ♡ I12	F29 ♡ I10	F32 ♡ I01	F35 ♡ H18	F38 ♡ F33	F41 ♡ H02	F43 ♡ I09	F47 ♡ G32
F21 ♡ I06	F24 ♡ G04	F26 ♡ I13	F29 ♡ I11	F32 ♡ I02	F35 ♡ I00	F38 ♡ F34	F41 ♡ H06	F43 ♡ I10	F47 ♡ G36
F21 ♡ I07	F24 ♡ G08	F26 ♡ I14	F29 ♡ I12	F32 ♡ I03	F35 ♡ I01	F38 ♡ F45	F41 ♡ H10	F43 ♡ I11	F47 ♡ G40
F21 ♡ I08	F24 ♡ G12	F27 ♡ D18	F29 ♡ I13	F32 ♡ I04	F35 ♡ I02	F38 ♡ G08	F41 ♡ H14	F43 ♡ I12	F47 ♡ G44
F21 ♡ I09	F24 ♡ G16	F27 ♡ E13	F29 ♡ I14	F32 ♡ I05	F35 ♡ I03	F38 ♡ I00	F41 ♡ I00	F43 ♡ I13	F47 ♡ H00
F21 ♡ I10	F24 ♡ G20	F27 ♡ F19	F29 ♡ J00	F32 ♡ I06	F35 ♡ I04	F38 ♡ I01	F41 ♡ I01	F43 ♡ I14	F47 ♡ H04
F21 ♡ I11	F24 ♡ G24	F27 ♡ F20	F30 ♡ E22	F32 ♡ I07	F35 ♡ I05	F38 ♡ I02	F41 ♡ I02	F44 ♡ F03	F47 ♡ H08
F21 ♡ I12	F24 ♡ G28	F27 ♡ F32	F30 ♡ F35	F32 ♡ I08	F35 ♡ I06	F38 ♡ I03	F41 ♡ I03	F44 ♡ F27	F47 ♡ H12
F21 ♡ I13	F24 ♡ G32	F27 ♡ F44	F30 ♡ F41	F32 ♡ I09	F35 ♡ I07	F38 ♡ I04	F41 ♡ I04	F44 ♡ F39	F47 ♡ H16
F21 ♡ I14	F24 ♡ G36	F27 ♡ G44	F30 ♡ F51	F32 ♡ I10	F35 ♡ I08	F38 ♡ I05	F41 ♡ I05	F44 ♡ G02	F47 ♡ H18
F22 ♡ F01	F24 ♡ G40	F27 ♡ I00	F30 ♡ G01	F32 ♡ I11	F35 ♡ I09	F38 ♡ I06	F41 ♡ I06	F44 ♡ I00	F47 ♡ I00
F22 ♡ F24	F24 ♡ G44	F27 ♡ I01	F30 ♡ G02	F32 ♡ I12	F35 ♡ I10	F38 ♡ I07	F41 ♡ I07	F44 ♡ I01	F47 ♡ I01
F22 ♡ F25	F24 ♡ G47	F27 ♡ I02	F30 ♡ G06	F32 ♡ I13	F35 ♡ I11	F38 ♡ I08	F41 ♡ I08	F44 ♡ I02	F47 ♡ I02
F22 ♡ F49	F24 ♡ H03	F27 ♡ I03	F30 ♡ G10	F32 ♡ I14	F35 ♡ I12	F38 ♡ I09	F41 ♡ I09	F44 ♡ I03	F47 ♡ I03
F22 ♡ G24	F24 ♡ H04	F27 ♡ I04	F30 ♡ G13	F33 ♡ E19	F35 ♡ I13	F38 ♡ I10	F41 ♡ I10	F44 ♡ I04	F47 ♡ I04
F22 ♡ I00	F24 ♡ H07	F27 ♡ I05	F30 ♡ G14	F33 ♡ F26	F35 ♡ I14	F38 ♡ I11	F41 ♡ I11	F44 ♡ I05	F47 ♡ I05
F22 ♡ I01	F24 ♡ H11	F27 ♡ I06	F30 ♡ G18	F33 ♡ F38	F36 ♡ F10	F38 ♡ I12	F41 ♡ I12	F44 ♡ I06	F47 ♡ I06
F22 ♡ I02	F24 ♡ H15	F27 ♡ I07	F30 ♡ G22	F33 ♡ G38	F36 ♡ F11	F38 ♡ I13	F41 ♡ I13	F44 ♡ I07	F47 ♡ I07
F22 ♡ I03	F24 ♡ H17	F27 ♡ I08	F30 ♡ G25	F33 ♡ I00	F36 ♡ F41	F38 ♡ I14	F41 ♡ I14	F44 ♡ I08	F47 ♡ I08
F22 ♡ I04	F24 ♡ I00	F27 ♡ I09	F30 ♡ G26	F33 ♡ I01	F36 ♡ F51	F39 ♡ F07	F42 ♡ F04	F44 ♡ I09	F47 ♡ I09
F22 ♡ I05	F24 ♡ I01	F27 ♡ I10	F30 ♡ G30	F33 ♡ I02	F36 ♡ G00	F39 ♡ F08	F42 ♡ F05	F44 ♡ I10	F47 ♡ I10
F22 ♡ I06	F24 ♡ I02	F27 ♡ I11	F30 ♡ G34	F33 ♡ I03	F36 ♡ G04	F39 ♡ F32	F42 ♡ F47	F44 ♡ I11	F47 ♡ I11
F22 ♡ I07	F24 ♡ I03	F27 ♡ I12	F30 ♡ G37	F33 ♡ I04	F36 ♡ G08	F39 ♡ F44	F42 ♡ F51	F44 ♡ I12	F47 ♡ I12
F22 ♡ I08	F24 ♡ I04	F27 ♡ I13	F30 ♡ G38	F33 ♡ I05	F36 ♡ G10	F39 ♡ G07	F42 ♡ G01	F44 ♡ I13	F47 ♡ I13
F22 ♡ I09	F24 ♡ I05	F27 ♡ I14	F30 ♡ G41	F33 ♡ I06	F36 ♡ G11	F39 ♡ I00	F42 ♡ G02	F44 ♡ I14	F47 ♡ I14
F22 ♡ I10	F24 ♡ I06	F28 ♡ F18	F30 ♡ G42	F33 ♡ I07	F36 ♡ G12	F39 ♡ I01	F42 ♡ G04	F45 ♡ F02	F48 ♡ B04
F22 ♡ I11	F24 ♡ I07	F28 ♡ F43	F30 ♡ G46	F33 ♡ I08	F36 ♡ G16	F39 ♡ I02	F42 ♡ G06	F45 ♡ F38	F49 ♡ F22
F22 ♡ I12	F24 ♡ I08	F28 ♡ G43	F30 ♡ H01	F33 ♡ I09	F36 ♡ G20	F39 ♡ I03	F42 ♡ G10	F45 ♡ G01	F49 ♡ F23
F22 ♡ I13	F24 ♡ I09	F28 ♡ I00	F30 ♡ H05	F33 ♡ I10	F36 ♡ G24	F39 ♡ I04	F42 ♡ G13	F45 ♡ I00	F50 ♡ F06
F22 ♡ I14	F24 ♡ I10	F28 ♡ I01	F30 ♡ H09	F33 ♡ I11	F36 ♡ G28	F39 ♡ I05	F42 ♡ G14	F45 ♡ I01	F51 ♡ F00
F23 ♡ F00	F24 ♡ I11	F28 ♡ I02	F30 ♡ H13	F33 ♡ I12	F36 ♡ G32	F39 ♡ I06	F42 ♡ G18	F45 ♡ I02	F51 ♡ F06
F23 ♡ F18	F24 ♡ I12	F28 ♡ I03	F30 ♡ H19	F33 ♡ I13	F36 ♡ G36	F39 ♡ I07	F42 ♡ G22	F45 ♡ I03	F51 ♡ F07
F23 ♡ F49	F24 ♡ I13	F28 ♡ I04	F30 ♡ I00	F33 ♡ I14	F36 ♡ G40	F39 ♡ I08	F42 ♡ G25	F45 ♡ I04	F51 ♡ F12
F23 ♡ G00	F24 ♡ I14	F28 ♡ I05	F30 ♡ I01	F34 ♡ F12	F36 ♡ G44	F39 ♡ I09	F42 ♡ G26	F45 ♡ I05	F51 ♡ F17
F23 ♡ G04	F24 ♡ J00	F28 ♡ I06	F30 ♡ I02	F34 ♡ F37	F36 ♡ H03	F39 ♡ I10	F42 ♡ G30	F45 ♡ I06	F51 ♡ F18
F23 ♡ G08	F24 ♡ J01	F28 ♡ I07	F30 ♡ I03	F34 ♡ F38	F36 ♡ H04	F39 ♡ I11	F42 ♡ G34	F45 ♡ I07	F51 ♡ F24
F23 ♡ G12	F25 ♡ D21	F28 ♡ I08	F30 ♡ I04	F34 ♡ G37	F36 ♡ H07	F39 ♡ I12	F42 ♡ G37	F45 ♡ I08	F51 ♡ F30
F23 ♡ G16	F25 ♡ F22	F28 ♡ I09	F30 ♡ I05	F34 ♡ I00	F36 ♡ H11	F39 ♡ I13	F42 ♡ G38	F45 ♡ I09	F51 ♡ F31
F23 ♡ G20	F25 ♡ F46	F28 ♡ I10	F30 ♡ I06	F34 ♡ I01	F36 ♡ H15	F39 ♡ I14	F42 ♡ G42	F45 ♡ I10	F51 ♡ F36
F23 ♡ G24	F25 ♡ G46	F28 ♡ I11	F30 ♡ I07	F34 ♡ I02	F36 ♡ H17	F40 ♡ F31	F42 ♡ G46	F45 ♡ I11	F51 ♡ F41
F23 ♡ G28	F25 ♡ I00	F28 ♡ I12	F30 ♡ I08	F34 ♡ I03	F36 ♡ I00	F40 ♡ G06	F42 ♡ H01	F45 ♡ I12	F51 ♡ F42

**XX:58 Proving the Turing Universality of Oritatami Co-Transcriptional Folding**

F51	♥	G07	G10	♥	F05	G20	♥	F23	G30	♥	F42	G41	♥	F30	H04	♥	F36	H16	♥	I02	I00	♥	F45	I02	♥	F08	I03	♥	F26
F51	♥	G19	G10	♥	F06	G20	♥	F24	G30	♥	G16	G41	♥	G30	H04	♥	F47	H16	♥	I04	I00	♥	F46	I02	♥	F09	I03	♥	F27
F51	♥	G31	G10	♥	F17	G20	♥	F35	G30	♥	G41	G42	♥	F05	H04	♥	H01	H17	♥	F00	I00	♥	F47	I02	♥	F10	I03	♥	F28
F51	♥	G43	G10	♥	F18	G20	♥	F36	G31	♥	F15	G42	♥	F06	H04	♥	I18	H17	♥	F12	I00	♥	H10	I02	♥	F11	I03	♥	F29
G00	♥	F00	G10	♥	F29	G20	♥	F47	G31	♥	F51	G42	♥	F17	H05	♥	F06	H17	♥	F24	I00	♥	H16	I02	♥	F12	I03	♥	F30
G00	♥	F11	G10	♥	F30	G20	♥	G26	G32	♥	F00	G42	♥	F18	H05	♥	F18	H17	♥	F36	I00	♥	I07	I02	♥	F13	I03	♥	F31
G00	♥	F12	G10	♥	F36	G21	♥	F02	G32	♥	F11	G42	♥	F29	H05	♥	F30	H17	♥	H22	I00	♥	I08	I02	♥	F14	I03	♥	F32
G00	♥	F23	G10	♥	F41	G21	♥	G02	G32	♥	F12	G42	♥	F30	H05	♥	F42	H17	♥	I03	I01	♥	F00	I02	♥	F15	I03	♥	F33
G00	♥	F24	G10	♥	F42	G21	♥	H02	G32	♥	F14	G42	♥	F41	H05	♥	H10	H18	♥	F11	I01	♥	F01	I02	♥	F16	I03	♥	F34
G00	♥	F35	G10	♥	G13	G22	♥	F01	G32	♥	F23	G42	♥	F42	H05	♥	I01	H18	♥	F23	I01	♥	F02	I02	♥	F17	I03	♥	F35
G00	♥	F36	G10	♥	G36	G22	♥	F05	G32	♥	F24	G42	♥	G04	H05	♥	I03	H18	♥	F35	I01	♥	F03	I02	♥	F18	I03	♥	F36
G00	♥	F46	G11	♥	F36	G22	♥	F06	G32	♥	F35	G43	♥	F28	H05	♥	I05	H18	♥	F47	I01	♥	F04	I02	♥	F19	I03	♥	F37
G00	♥	F47	G11	♥	G13	G22	♥	F17	G32	♥	F36	G43	♥	F51	H06	♥	F05	H18	♥	H20	I01	♥	F05	I02	♥	F20	I03	♥	F38
G00	♥	G23	G11	♥	G36	G22	♥	F18	G32	♥	F47	G43	♥	G28	H06	♥	F17	H18	♥	H21	I01	♥	F06	I02	♥	F21	I03	♥	F39
G00	♥	G46	G12	♥	F00	G22	♥	F29	G32	♥	G14	G44	♥	F00	H06	♥	F29	H19	♥	F06	I01	♥	F07	I02	♥	F22	I03	♥	F40
G00	♥	G48	G12	♥	F11	G22	♥	F30	G32	♥	G39	G44	♥	F11	H06	♥	F41	H19	♥	F18	I01	♥	F08	I02	♥	F23	I03	♥	F41
G01	♥	F06	G12	♥	F12	G22	♥	F41	G33	♥	F13	G44	♥	F12	H07	♥	F00	H19	♥	F30	I01	♥	F09	I02	♥	F24	I03	♥	F42
G01	♥	F18	G12	♥	F23	G22	♥	F42	G33	♥	G38	G44	♥	F23	H07	♥	F12	H19	♥	F42	I01	♥	F10	I02	♥	F25	I03	♥	F43
G01	♥	F30	G12	♥	F24	G22	♥	G24	G34	♥	F05	G44	♥	F24	H07	♥	F24	H19	♥	H08	I01	♥	F11	I02	♥	F26	I03	♥	F44
G01	♥	F42	G12	♥	F35	G22	♥	G25	G34	♥	F06	G44	♥	F27	H07	♥	F36	H20	♥	H18	I01	♥	F12	I02	♥	F27	I03	♥	F45
G01	♥	F45	G12	♥	F36	G23	♥	F00	G34	♥	F12	G44	♥	F35	H07	♥	H02	H21	♥	H18	I01	♥	F13	I02	♥	F28	I03	♥	F46
G01	♥	G46	G12	♥	F47	G23	♥	G00	G34	♥	F17	G44	♥	F36	H07	♥	H14	H21	♥	H23	I01	♥	F14	I02	♥	F29	I03	♥	F47
G02	♥	F05	G12	♥	G34	G24	♥	F00	G34	♥	F18	G44	♥	F47	H08	♥	F11	H21	♥	H24	I01	♥	F15	I02	♥	F30	I03	♥	H05
G02	♥	F06	G12	♥	G35	G24	♥	F11	G34	♥	F29	G44	♥	G02	H08	♥	F23	H22	♥	H17	I01	♥	F16	I02	♥	F31	I03	♥	H11
G02	♥	F17	G13	♥	F06	G24	♥	F12	G34	♥	F30	G45	♥	F26	H08	♥	F35	H22	♥	I02	I01	♥	F17	I02	♥	F32	I03	♥	H17
G02	♥	F18	G13	♥	F10	G24	♥	F22	G34	♥	F41	G45	♥	G26	H08	♥	F47	H23	♥	H21	I01	♥	F18	I02	♥	F33	I03	♥	I07
G02	♥	F29	G13	♥	F18	G24	♥	F23	G34	♥	F42	G45	♥	H02	H08	♥	H13	H24	♥	H21	I01	♥	F19	I02	♥	F34	I03	♥	I08
G02	♥	F30	G13	♥	F30	G24	♥	F24	G34	♥	G12	G46	♥	F05	H08	♥	H19	I00	♥	F00	I01	♥	F20	I02	♥	F35	I04	♥	F00
G02	♥	F41	G13	♥	F42	G24	♥	F35	G34	♥	G37	G46	♥	F06	H09	♥	F06	I00	♥	F01	I01	♥	F21	I02	♥	F36	I04	♥	F01
G02	♥	F42	G13	♥	G10	G24	♥	F36	G35	♥	F12	G46	♥	F17	H09	♥	F18	I00	♥	F02	I01	♥	F22	I02	♥	F37	I04	♥	F02
G02	♥	F44	G13	♥	G11	G24	♥	F47	G35	♥	G12	G46	♥	F18	H09	♥	F30	I00	♥	F03	I01	♥	F23	I02	♥	F38	I04	♥	F03
G02	♥	G21	G14	♥	F05	G24	♥	G22	G35	♥	G37	G46	♥	F25	H09	♥	F42	I00	♥	F04	I01	♥	F24	I02	♥	F39	I04	♥	F04
G02	♥	G44	G14	♥	F06	G24	♥	G47	G36	♥	F00	G46	♥	F29	H10	♥	F05	I00	♥	F05	I01	♥	F25	I02	♥	F40	I04	♥	F05
G03	♥	F43	G14	♥	F09	G24	♥	G48	G36	♥	F11	G46	♥	F30	H10	♥	F17	I00	♥	F06	I01	♥	F26	I02	♥	F41	I04	♥	F06
G04	♥	F00	G14	♥	F17	G25	♥	F06	G36	♥	F12	G46	♥	F41	H10	♥	F29	I00	♥	F07	I01	♥	F27	I02	♥	F42	I04	♥	F07
G04	♥	F11	G14	♥	F18	G25	♥	F18	G36	♥	F23	G46	♥	F42	H10	♥	F41	I00	♥	F08	I01	♥	F28	I02	♥	F43	I04	♥	F08
G04	♥	F12	G14	♥	F29	G25	♥	F21	G36	♥	F24	G46	♥	G00	H10	♥	H05	I00	♥	F09	I01	♥	F29	I02	♥	F44	I04	♥	F09
G04	♥	F23	G14	♥	F30	G25	♥	F30	G36	♥	F35	G46	♥	G01	H10	♥	I00	I00	♥	F10	I01	♥	F30	I02	♥	F45	I04	♥	F10
G04	♥	F24	G14	♥	F41	G25	♥	F42	G36	♥	F36	G47	♥	F24	H10	♥	I02	I00	♥	F11	I01	♥	F31	I02	♥	F46	I04	♥	F11
G04	♥	F35	G14	♥	F42	G25	♥	G22	G36	♥	F47	G47	♥	G24	H10	♥	I04	I00	♥	F12	I01	♥	F32	I02	♥	F47	I04	♥	F12
G04	♥	F36	G14	♥	G09	G26	♥	F05	G36	♥	G10	G48	♥	G00	H11	♥	F00	I00	♥	F13	I01	♥	F33	I02	♥	H10	I04	♥	F13
G04	♥	F42	G14	♥	G32	G26	♥	F06	G36	♥	G11	G48	♥	G24	H11	♥	F12	I00	♥	F14	I01	♥	F34	I02	♥	H16	I04	♥	F14
G04	♥	F47	G15	♥	F08	G26	♥	F17	G37	♥	F06	H00	♥	F11	H11	♥	F24	I00	♥	F15	I01	♥	F35	I02	♥	H22	I04	♥	F15
G04	♥	G19	G15	♥	G08	G26	♥	F18	G37	♥	F18	H00	♥	F23	H11	♥	F36	I00	♥	F16	I01	♥	F36	I02	♥	I12	I04	♥	F16
G04	♥	G42	G16	♥	F00	G26	♥	F20	G37	♥	F30	H00	♥	F35	H11	♥	H16	I00	♥	F17	I01	♥	F37	I02	♥	I13	I04	♥	F17
G05	♥	F41	G16	♥	F07	G26	♥	F29	G37	♥	F34	H00	♥	F47	H11	♥	I01	I00	♥	F18	I01	♥	F38	I02	♥	I18	I04	♥	F18
G06	♥	F05	G16	♥	F11	G26	♥	F30	G37	♥	F42	H00	♥	H02	H11	♥	I03	I00	♥	F19	I01	♥	F39	I03	♥	F00	I04	♥	F19
G06	♥	F06	G16	♥	F12	G26	♥	F41	G37	♥	G34	H01	♥	F06	H11	♥	I05	I00	♥	F20	I01	♥	F40	I03	♥	F01	I04	♥	F20
G06	♥	F17	G16	♥	F23	G26	♥	F42	G37	♥	G35	H01	♥	F18	H12	♥	F11	I00	♥	F21	I01	♥	F41	I03	♥	F02	I04	♥	F21
G06	♥	F18	G16	♥	F24	G26	♥	G20	G38	♥	F05	H01	♥	F30	H12	♥	F23	I00	♥	F22	I01	♥	F42	I03	♥	F03	I04	♥	F22
G06	♥	F29	G16	♥	F35	G26	♥	G45	G38	♥	F06	H01	♥	F42	H12	♥	F35	I00	♥	F23	I01	♥	F43	I03	♥	F04	I04	♥	F23
G06	♥	F30	G16	♥	F36	G27	♥	F19	G38	♥	F17	H01	♥	H03	H12	♥	F47	I00	♥	F24	I01	♥	F44	I03	♥	F05	I04	♥	F24
G06	♥	F40	G16	♥	F47	G28	♥	F00	G38	♥	F18	H01	♥	H04	H13	♥	F06	I00	♥	F25	I01	♥	F45	I03	♥	F06	I04	♥	F25
G06	♥	F41	G16	♥	G30	G28	♥	F11	G38	♥	F29	H02	♥	F05	H13	♥	F18	I00	♥	F26	I01	♥	F46	I03	♥	F07	I04	♥	F26
G06	♥	F42	G17	♥	F06	G28	♥	F12	G38	♥	F30	H02	♥	F17	H13	♥	F30	I00	♥	F27	I01	♥	F47	I03	♥	F08	I04	♥	F27
G06	♥	G17	G17	♥	G06	G28	♥	F18	G38	♥	F33	H02	♥	F29	H13	♥	F42	I00	♥	F28	I01	♥	H05	I03	♥	F09	I04	♥	F28
G06	♥	G40	G18	♥	F05	G28	♥	F23	G38	♥	F41	H02	♥	F41	H13	♥	H08	I00	♥	F29	I01	♥	H11	I03	♥	F10	I04	♥	F29
G07	♥	F39	G18	♥	F06	G28	♥	F24	G38	♥	F42	H02	♥	G21	H14	♥	F05	I00	♥	F30	I01	♥	I07	I03	♥	F11	I04	♥	F30
G07	♥	F51	G18	♥	F17	G28	♥	F35	G38	♥	G08	H02	♥	G45	H14	♥	F17	I00	♥	F31	I01	♥	I13	I03	♥	F12	I04	♥	F31
G08	♥	F00	G18	♥	F18	G28	♥	F36	G38	♥	G33	H02	♥	H00	H14	♥	F29	I00	♥	F32	I01	♥	I15	I03	♥	F13	I04		

I04 ♡ F45	I06 ♡ F17	I07 ♡ F39	I09 ♡ F06	I10 ♡ F28	I11 ♡ I06	I13 ♡ F21	I14 ♡ F40	J05 ♡ J51	J28 ♡ L07
I04 ♡ F46	I06 ♡ F18	I07 ♡ F40	I09 ♡ F07	I10 ♡ F29	I11 ♡ I07	I13 ♡ F22	I14 ♡ F41	J06 ♡ J11	J29 ♡ J05
I04 ♡ F47	I06 ♡ F19	I07 ♡ F41	I09 ♡ F08	I10 ♡ F30	I12 ♡ F00	I13 ♡ F23	I14 ♡ F42	J07 ♡ C05	J29 ♡ K09
I04 ♡ H10	I06 ♡ F20	I07 ♡ F42	I09 ♡ F09	I10 ♡ F31	I12 ♡ F01	I13 ♡ F24	I14 ♡ F43	J07 ♡ J02	J29 ♡ L08
I04 ♡ H16	I06 ♡ F21	I07 ♡ F43	I09 ♡ F10	I10 ♡ F32	I12 ♡ F02	I13 ♡ F25	I14 ♡ F44	J07 ♡ J08	J30 ♡ J10
I05 ♡ F00	I06 ♡ F22	I07 ♡ F44	I09 ♡ F11	I10 ♡ F33	I12 ♡ F03	I13 ♡ F26	I14 ♡ F45	J08 ♡ C03	J30 ♡ K10
I05 ♡ F01	I06 ♡ F23	I07 ♡ F45	I09 ♡ F12	I10 ♡ F34	I12 ♡ F04	I13 ♡ F27	I14 ♡ F46	J08 ♡ C04	J30 ♡ L09
I05 ♡ F02	I06 ♡ F24	I07 ♡ F46	I09 ♡ F13	I10 ♡ F35	I12 ♡ F05	I13 ♡ F28	I14 ♡ F47	J08 ♡ J07	J31 ♡ J05
I05 ♡ F03	I06 ♡ F25	I07 ♡ F47	I09 ♡ F14	I10 ♡ F36	I12 ♡ F06	I13 ♡ F29	I14 ♡ I09	J10 ♡ J05	J31 ♡ K11
I05 ♡ F04	I06 ♡ F26	I07 ♡ I00	I09 ♡ F15	I10 ♡ F37	I12 ♡ F07	I13 ♡ F30	I15 ♡ I01	J10 ♡ J22	J31 ♡ L10
I05 ♡ F05	I06 ♡ F27	I07 ♡ I01	I09 ♡ F16	I10 ♡ F38	I12 ♡ F08	I13 ♡ F31	I15 ♡ I06	J10 ♡ J24	J32 ♡ J10
I05 ♡ F06	I06 ♡ F28	I07 ♡ I03	I09 ♡ F17	I10 ♡ F39	I12 ♡ F09	I13 ♡ F32	I15 ♡ I15	J10 ♡ J26	J32 ♡ K12
I05 ♡ F07	I06 ♡ F29	I07 ♡ I10	I09 ♡ F18	I10 ♡ F40	I12 ♡ F10	I13 ♡ F33	I15 ♡ I15	J10 ♡ J28	J33 ♡ J05
I05 ♡ F08	I06 ♡ F30	I07 ♡ I11	I09 ♡ F19	I10 ♡ F41	I12 ♡ F11	I13 ♡ F34	I16 ♡ I01	J10 ♡ J30	J33 ♡ K13
I05 ♡ F09	I06 ♡ F31	I07 ♡ I16	I09 ♡ F20	I10 ♡ F42	I12 ♡ F12	I13 ♡ F35	I16 ♡ I07	J10 ♡ J32	J34 ♡ J10
I05 ♡ F10	I06 ♡ F32	I07 ♡ I19	I09 ♡ F21	I10 ♡ F43	I12 ♡ F13	I13 ♡ F36	I16 ♡ I10	J10 ♡ J34	J34 ♡ K14
I05 ♡ F11	I06 ♡ F33	I08 ♡ F00	I09 ♡ F22	I10 ♡ F44	I12 ♡ F14	I13 ♡ F37	I17 ♡ B02	J10 ♡ J36	J35 ♡ J05
I05 ♡ F12	I06 ♡ F34	I08 ♡ F01	I09 ♡ F23	I10 ♡ F45	I12 ♡ F15	I13 ♡ F38	I17 ♡ I13	J10 ♡ J38	J35 ♡ K15
I05 ♡ F13	I06 ♡ F35	I08 ♡ F02	I09 ♡ F24	I10 ♡ F46	I12 ♡ F16	I13 ♡ F39	I17 ♡ I19	J10 ♡ J40	J36 ♡ J10
I05 ♡ F14	I06 ♡ F36	I08 ♡ F03	I09 ♡ F25	I10 ♡ F47	I12 ♡ F17	I13 ♡ F40	I18 ♡ H04	J10 ♡ J42	J36 ♡ K16
I05 ♡ F15	I06 ♡ F37	I08 ♡ F04	I09 ♡ F26	I10 ♡ I07	I12 ♡ F18	I13 ♡ F41	I18 ♡ I02	J10 ♡ J44	J37 ♡ J05
I05 ♡ F16	I06 ♡ F38	I08 ♡ F05	I09 ♡ F27	I10 ♡ I13	I12 ♡ F19	I13 ♡ F42	I18 ♡ I10	J10 ♡ J46	J37 ♡ K17
I05 ♡ F17	I06 ♡ F39	I08 ♡ F06	I09 ♡ F28	I10 ♡ I16	I12 ♡ F20	I13 ♡ F43	I19 ♡ B02	J10 ♡ J48	J38 ♡ J10
I05 ♡ F18	I06 ♡ F40	I08 ♡ F07	I09 ♡ F29	I10 ♡ I18	I12 ♡ F21	I13 ♡ F44	I19 ♡ I01	J10 ♡ J49	J38 ♡ K18
I05 ♡ F19	I06 ♡ F41	I08 ♡ F08	I09 ♡ F30	I11 ♡ F00	I12 ♡ F22	I13 ♡ F45	I19 ♡ I07	J10 ♡ J50	J39 ♡ J05
I05 ♡ F20	I06 ♡ F42	I08 ♡ F09	I09 ♡ F31	I11 ♡ F01	I12 ♡ F23	I13 ♡ F46	I19 ♡ I17	J10 ♡ J52	J39 ♡ K00
I05 ♡ F21	I06 ♡ F43	I08 ♡ F10	I09 ♡ F32	I11 ♡ F02	I12 ♡ F24	I13 ♡ F47	I19 ♡ I19	J11 ♡ C04	J39 ♡ K19
I05 ♡ F22	I06 ♡ F44	I08 ♡ F11	I09 ♡ F33	I11 ♡ F03	I12 ♡ F25	I13 ♡ I01	I19 ♡ I19	J11 ♡ J06	J40 ♡ J10
I05 ♡ F23	I06 ♡ F45	I08 ♡ F12	I09 ♡ F34	I11 ♡ F04	I12 ♡ F26	I13 ♡ I02	I19 ♡ J12	J11 ♡ J13	J40 ♡ K01
I05 ♡ F24	I06 ♡ F46	I08 ♡ F13	I09 ♡ F35	I11 ♡ F05	I12 ♡ F27	I13 ♡ I09		J12 ♡ C03	J40 ♡ K20
I05 ♡ F25	I06 ♡ F47	I08 ♡ F14	I09 ♡ F36	I11 ♡ F06	I12 ♡ F28	I13 ♡ I10		J12 ♡ I19	J41 ♡ J05
I05 ♡ F26	I06 ♡ I11	I08 ♡ F15	I09 ♡ F37	I11 ♡ F07	I12 ♡ F29	I13 ♡ I17	<b>F</b>	J12 ♡ J16	J41 ♡ K02
I05 ♡ F27	I06 ♡ I15	I08 ♡ F16	I09 ♡ F38	I11 ♡ F08	I12 ♡ F30	I14 ♡ F00	J00 ♡ B01	J13 ♡ J11	J41 ♡ K21
I05 ♡ F28	I07 ♡ F00	I08 ♡ F17	I09 ♡ F39	I11 ♡ F09	I12 ♡ F31	I14 ♡ F01	J00 ♡ F00	J14 ♡ J05	J42 ♡ J10
I05 ♡ F29	I07 ♡ F01	I08 ♡ F18	I09 ♡ F40	I11 ♡ F10	I12 ♡ F32	I14 ♡ F02	J00 ♡ F05	J14 ♡ J20	J42 ♡ K22
I05 ♡ F30	I07 ♡ F02	I08 ♡ F19	I09 ♡ F41	I11 ♡ F11	I12 ♡ F33	I14 ♡ F03	J00 ♡ F24	J15 ♡ J17	J43 ♡ J05
I05 ♡ F31	I07 ♡ F03	I08 ♡ F20	I09 ♡ F42	I11 ♡ F12	I12 ♡ F34	I14 ♡ F04	J00 ♡ F29	J15 ♡ J19	J43 ♡ K23
I05 ♡ F32	I07 ♡ F04	I08 ♡ F21	I09 ♡ F43	I11 ♡ F13	I12 ♡ F35	I14 ♡ F05	J00 ♡ I01	J16 ♡ J12	J44 ♡ J10
I05 ♡ F33	I07 ♡ F05	I08 ♡ F22	I09 ♡ F44	I11 ♡ F14	I12 ♡ F36	I14 ♡ F06	J00 ♡ J02	J17 ♡ A12	J44 ♡ K24
I05 ♡ F34	I07 ♡ F06	I08 ♡ F23	I09 ♡ F45	I11 ♡ F15	I12 ♡ F37	I14 ♡ F07	J00 ♡ L02	J17 ♡ B00	J45 ♡ J05
I05 ♡ F35	I07 ♡ F07	I08 ♡ F24	I09 ♡ F46	I11 ♡ F16	I12 ♡ F38	I14 ♡ F08	J01 ♡ B01	J17 ♡ J15	J45 ♡ K25
I05 ♡ F36	I07 ♡ F08	I08 ♡ F25	I09 ♡ F47	I11 ♡ F17	I12 ♡ F39	I14 ♡ F09	J01 ♡ F00	J17 ♡ J19	J46 ♡ J10
I05 ♡ F37	I07 ♡ F09	I08 ♡ F26	I09 ♡ I13	I11 ♡ F18	I12 ♡ F40	I14 ♡ F10	J01 ♡ F24	J18 ♡ A00	J46 ♡ K26
I05 ♡ F38	I07 ♡ F10	I08 ♡ F27	I09 ♡ I14	I11 ♡ F19	I12 ♡ F41	I14 ♡ F11	J01 ♡ J03	J18 ♡ A01	J47 ♡ J05
I05 ♡ F39	I07 ♡ F11	I08 ♡ F28	I10 ♡ F00	I11 ♡ F20	I12 ♡ F42	I14 ♡ F12	J01 ♡ J04	J18 ♡ B01	J47 ♡ K27
I05 ♡ F40	I07 ♡ F12	I08 ♡ F29	I10 ♡ F01	I11 ♡ F21	I12 ♡ F43	I14 ♡ F13	J01 ♡ L01	J18 ♡ L16	J48 ♡ J04
I05 ♡ F41	I07 ♡ F13	I08 ♡ F30	I10 ♡ F02	I11 ♡ F22	I12 ♡ F44	I14 ♡ F14	J02 ♡ I01	J18 ♡ L17	J48 ♡ J05
I05 ♡ F42	I07 ♡ F14	I08 ♡ F31	I10 ♡ F03	I11 ♡ F23	I12 ♡ F45	I14 ♡ F15	J02 ♡ J00	J19 ♡ J15	J48 ♡ J10
I05 ♡ F43	I07 ♡ F15	I08 ♡ F32	I10 ♡ F04	I11 ♡ F24	I12 ♡ F46	I14 ♡ F16	J02 ♡ J07	J19 ♡ J17	J48 ♡ K03
I05 ♡ F44	I07 ♡ F16	I08 ♡ F33	I10 ♡ F05	I11 ♡ F25	I12 ♡ F47	I14 ♡ F17	J03 ♡ J01	J19 ♡ L15	J48 ♡ L03
I05 ♡ F45	I07 ♡ F17	I08 ♡ F34	I10 ♡ F06	I11 ♡ F26	I12 ♡ I02	I14 ♡ F18	J04 ♡ J01	J20 ♡ J14	J48 ♡ L06
I05 ♡ F46	I07 ♡ F18	I08 ♡ F35	I10 ♡ F07	I11 ♡ F27	I13 ♡ F00	I14 ♡ F19	J04 ♡ J48	J20 ♡ L14	J48 ♡ L08
I05 ♡ F47	I07 ♡ F19	I08 ♡ F36	I10 ♡ F08	I11 ♡ F28	I13 ♡ F01	I14 ♡ F20	J04 ♡ L00	J21 ♡ J05	J49 ♡ J05
I05 ♡ H05	I07 ♡ F20	I08 ♡ F37	I10 ♡ F09	I11 ♡ F29	I13 ♡ F02	I14 ♡ F21	J05 ♡ J10	J21 ♡ L16	J49 ♡ J10
I05 ♡ H11	I07 ♡ F21	I08 ♡ F38	I10 ♡ F10	I11 ♡ F30	I13 ♡ F03	I14 ♡ F22	J05 ♡ J14	J22 ♡ J10	J49 ♡ K00
I06 ♡ F00	I07 ♡ F22	I08 ♡ F39	I10 ♡ F11	I11 ♡ F31	I13 ♡ F04	I14 ♡ F23	J05 ♡ J21	J22 ♡ L12	J49 ♡ L00
I06 ♡ F01	I07 ♡ F23	I08 ♡ F40	I10 ♡ F12	I11 ♡ F32	I13 ♡ F05	I14 ♡ F24	J05 ♡ J23	J23 ♡ J05	J49 ♡ L04
I06 ♡ F02	I07 ♡ F24	I08 ♡ F41	I10 ♡ F13	I11 ♡ F33	I13 ♡ F06	I14 ♡ F25	J05 ♡ J25	J23 ♡ L11	J50 ♡ J10
I06 ♡ F03	I07 ♡ F25	I08 ♡ F42	I10 ♡ F14	I11 ♡ F34	I13 ♡ F07	I14 ♡ F26	J05 ♡ J27	J24 ♡ J10	J50 ♡ K01
I06 ♡ F04	I07 ♡ F26	I08 ♡ F43	I10 ♡ F15	I11 ♡ F35	I13 ♡ F08	I14 ♡ F27	J05 ♡ J29	J24 ♡ K03	J50 ♡ L05
I06 ♡ F05	I07 ♡ F27	I08 ♡ F44	I10 ♡ F16	I11 ♡ F36	I13 ♡ F09	I14 ♡ F28	J05 ♡ J31	J24 ♡ K04	J51 ♡ J05
I06 ♡ F06	I07 ♡ F28	I08 ♡ F45	I10 ♡ F17	I11 ♡ F37	I13 ♡ F10	I14 ♡ F29	J05 ♡ J33	J25 ♡ J05	J51 ♡ K02
I06 ♡ F07	I07 ♡ F29	I08 ♡ F46	I10 ♡ F18	I11 ♡ F38	I13 ♡ F11	I14 ♡ F30	J05 ♡ J35	J25 ♡ K00	J51 ♡ L02
I06 ♡ F08	I07 ♡ F30	I08 ♡ F47	I10 ♡ F19	I11 ♡ F39	I13 ♡ F12	I14 ♡ F31	J05 ♡ J37	J25 ♡ K05	J51 ♡ L06
I06 ♡ F09	I07 ♡ F31	I08 ♡ I00	I10 ♡ F20	I11 ♡ F40	I13 ♡ F13	I14 ♡ F32	J05 ♡ J39	J26 ♡ J10	J52 ♡ J10
I06 ♡ F10	I07 ♡ F32	I08 ♡ I03	I10 ♡ F21	I11 ♡ F41	I13 ♡ F14	I14 ♡ F33	J05 ♡ J41	J26 ♡ K01	J52 ♡ K01
I06 ♡ F11	I07 ♡ F33	I09 ♡ F00	I10 ♡ F22	I11 ♡ F42	I13 ♡ F15	I14 ♡ F34	J05 ♡ J43	J26 ♡ K06	
I06 ♡ F12	I07 ♡ F34	I09 ♡ F01	I10 ♡ F23	I11 ♡ F43	I13 ♡ F16	I14 ♡ F35	J05 ♡ J45	J27 ♡ J05	
I06 ♡ F13	I07 ♡ F35	I09 ♡ F02	I10 ♡ F24	I11 ♡ F44	I13 ♡ F17	I14 ♡ F36	J05 ♡ J47	J27 ♡ K02	<b>G</b>
I06 ♡ F14	I07 ♡ F36	I09 ♡ F03	I10 ♡ F25	I11 ♡ F45	I13 ♡ F18	I14 ♡ F37	J05 ♡ J48	J27 ♡ K07	K00 ♡ J25
I06 ♡ F15	I07 ♡ F37	I09 ♡ F04	I10 ♡ F26	I11 ♡ F46	I13 ♡ F19	I14 ♡ F38	J05 ♡ J49	J28 ♡ J10	K00 ♡ J39
I06 ♡ F16	I07 ♡ F38	I09 ♡ F05	I10 ♡ F27	I11 ♡ F47	I13 ♡ F20	I14 ♡ F39		J28 ♡ K08	

# XX:60 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

K00	♥	J49	K14	♥	K21	K28	♥	K03	K34	♥	K50	K47	♥	L40	K55	♥	L79	K68	♥	M21	L17	♥	K45	L38	♥	L57	L56	♥	L73
K00	♥	K29	K14	♥	K27	K28	♥	K08	K35	♥	K31	K47	♥	L41	K55	♥	L95	K69	♥	K52	L17	♥	L12	L39	♥	K29	L57	♥	K35
K00	♥	K31	K14	♥	K28	K28	♥	K14	K35	♥	K45	K47	♥	L75	K55	♥	M01	K69	♥	K53	L17	♥	L65	L39	♥	K30	L57	♥	K50
K00	♥	K33	K14	♥	L08	K28	♥	K20	K35	♥	K49	K48	♥	K51	K55	♥	M07	K69	♥	K60	L18	♥	D56	L39	♥	K31	L57	♥	L38
K01	♥	J26	K15	♥	J35	K28	♥	K26	K35	♥	K50	K48	♥	K54	K55	♥	M13	K69	♥	K64	L18	♥	E20	L39	♥	K37	L57	♥	L71
K01	♥	J40	K15	♥	K08	K28	♥	K33	K35	♥	L40	K48	♥	K56	K55	♥	M19	L00	♥	J04	L18	♥	E44	L39	♥	K38	L58	♥	K49
K01	♥	J50	K15	♥	K14	K28	♥	K38	K35	♥	L41	K48	♥	L72	K55	♥	M22	L00	♥	J49	L18	♥	L23	L39	♥	L41	L58	♥	L70
K01	♥	J52	K15	♥	K20	K28	♥	K44	K35	♥	L56	K48	♥	L73	K56	♥	K46	L00	♥	L04	L18	♥	L41	L40	♥	K28	L59	♥	K50
K01	♥	K28	K15	♥	K26	K28	♥	L07	K35	♥	L57	K48	♥	L74	K56	♥	K48	L00	♥	L05	L18	♥	L47	L40	♥	K29	L59	♥	L36
K01	♥	K30	K15	♥	K33	K28	♥	L25	K36	♥	K30	K49	♥	K35	K56	♥	K53	L00	♥	L31	L18	♥	L64	L40	♥	K30	L59	♥	L69
K01	♥	K32	K15	♥	L14	K28	♥	L40	K36	♥	K37	K49	♥	K41	K56	♥	K59	L01	♥	J01	L18	♥	L65	L40	♥	K35	L60	♥	K49
K02	♥	J27	K16	♥	J36	K28	♥	L41	K36	♥	L35	K49	♥	K50	K56	♥	K60	L01	♥	L30	L18	♥	M22	L40	♥	K37	L60	♥	L35
K02	♥	J41	K16	♥	K32	K29	♥	K00	K37	♥	K29	K49	♥	K53	K56	♥	K65	L02	♥	A07	L19	♥	L16	L40	♥	K42	L60	♥	L36
K02	♥	J51	K17	♥	J37	K29	♥	K02	K37	♥	K36	K49	♥	L58	K56	♥	K66	L02	♥	J00	L19	♥	L46	L40	♥	K47	L60	♥	L68
K02	♥	K29	K17	♥	K06	K29	♥	K07	K37	♥	K47	K49	♥	L60	K56	♥	L94	L02	♥	J51	L20	♥	L15	L41	♥	K28	L61	♥	K50
K02	♥	K31	K17	♥	K12	K29	♥	K13	K37	♥	L39	K49	♥	L62	K56	♥	L95	L02	♥	L05	L20	♥	L16	L41	♥	K29	L61	♥	L67
K02	♥	K33	K17	♥	K18	K29	♥	K19	K37	♥	L40	K49	♥	L64	K56	♥	M00	L02	♥	L06	L20	♥	L45	L41	♥	K33	L62	♥	K49
K03	♥	J24	K17	♥	K24	K29	♥	K25	K38	♥	K28	K50	♥	K34	K56	♥	M01	L02	♥	L29	L21	♥	K33	L41	♥	K35	L62	♥	L33
K03	♥	J48	K17	♥	K31	K29	♥	K37	K38	♥	K46	K50	♥	K35	K56	♥	M06	L03	♥	A07	L21	♥	L44	L41	♥	K41	L62	♥	L66
K03	♥	K28	K18	♥	J38	K29	♥	K43	K38	♥	K47	K50	♥	K40	K56	♥	M07	L03	♥	A08	L22	♥	L13	L41	♥	K46	L63	♥	K50
K03	♥	K30	K18	♥	K05	K29	♥	L39	K38	♥	L39	K50	♥	K41	K56	♥	M12	L03	♥	J48	L22	♥	L43	L41	♥	K47	L63	♥	L32
K03	♥	K32	K18	♥	K11	K29	♥	L40	K39	♥	K33	K50	♥	K49	K56	♥	M13	L03	♥	L28	L23	♥	L12	L41	♥	L18	L63	♥	L33
K04	♥	J24	K18	♥	K17	K29	♥	L41	K39	♥	K34	K50	♥	K52	K56	♥	M18	L04	♥	J49	L23	♥	L18	L41	♥	L39	L64	♥	D57
K04	♥	K32	K18	♥	K23	K30	♥	K01	K39	♥	L38	K50	♥	L56	K56	♥	M19	L04	♥	L00	L23	♥	L42	L41	♥	L47	L64	♥	K49
K05	♥	J25	K18	♥	K30	K30	♥	K03	K40	♥	K32	K50	♥	L57	K56	♥	M21	L04	♥	L27	L24	♥	K30	L42	♥	K43	L64	♥	L18
K05	♥	K06	K18	♥	L11	K30	♥	K06	K40	♥	K45	K50	♥	L59	K56	♥	M22	L05	♥	J50	L24	♥	K41	L42	♥	L23	L64	♥	L48
K05	♥	K12	K19	♥	J39	K30	♥	K12	K40	♥	K46	K50	♥	L61	K57	♥	K47	L05	♥	L00	L25	♥	K28	L42	♥	L55	L65	♥	L17
K05	♥	K18	K19	♥	K29	K30	♥	K18	K40	♥	K50	K50	♥	L63	K57	♥	K51	L05	♥	L02	L25	♥	L06	L43	♥	K43	L65	♥	L18
K05	♥	K24	K20	♥	J40	K30	♥	K24	K41	♥	K31	K51	♥	K41	K57	♥	K52	L06	♥	J48	L25	♥	L37	L43	♥	L22	L65	♥	M30
K05	♥	K31	K20	♥	K09	K30	♥	K31	K41	♥	K44	K51	♥	K48	K57	♥	M05	L06	♥	J51	L26	♥	L31	L43	♥	L53	L66	♥	L62
K06	♥	J26	K20	♥	K15	K30	♥	K36	K41	♥	K49	K51	♥	K57	K58	♥	K53	L06	♥	L02	L26	♥	L36	L44	♥	K42	L66	♥	M30
K06	♥	K05	K20	♥	K21	K30	♥	K42	K41	♥	K50	K52	♥	K50	K58	♥	K61	L06	♥	L25	L27	♥	L04	L44	♥	L21	L67	♥	L61
K06	♥	K11	K20	♥	K27	K30	♥	L24	K41	♥	K51	K52	♥	K57	K59	♥	K52	L07	♥	J28	L27	♥	L31	L44	♥	L52	L67	♥	M29
K06	♥	K17	K20	♥	K28	K30	♥	L39	K41	♥	L24	K52	♥	K59	K59	♥	K56	L07	♥	K28	L27	♥	L35	L44	♥	L53	L68	♥	L60
K06	♥	K23	K21	♥	J41	K30	♥	L40	K41	♥	L41	K52	♥	K63	K59	♥	K60	L07	♥	L09	L28	♥	K31	L45	♥	K43	L68	♥	M28
K06	♥	K30	K21	♥	K08	K31	♥	K00	K41	♥	L55	K52	♥	K69	K60	♥	K55	L07	♥	L10	L28	♥	L03	L45	♥	L20	L69	♥	L59
K06	♥	L11	K21	♥	K14	K31	♥	K02	K42	♥	K30	K52	♥	L92	K60	♥	K56	L08	♥	J29	L28	♥	L33	L46	♥	K42	L69	♥	M27
K07	♥	J27	K21	♥	K20	K31	♥	K05	K42	♥	K43	K52	♥	L98	K60	♥	K59	L08	♥	J48	L28	♥	L34	L46	♥	L19	L69	♥	M28
K07	♥	K29	K21	♥	K26	K31	♥	K11	K42	♥	L40	K52	♥	M04	K60	♥	K69	L08	♥	K14	L29	♥	L02	L46	♥	L50	L70	♥	L58
K08	♥	J28	K21	♥	K33	K31	♥	K17	K42	♥	L44	K52	♥	M10	K60	♥	L75	L08	♥	K33	L29	♥	L33	L46	♥	L51	L70	♥	M27
K08	♥	K09	K21	♥	L14	K31	♥	K23	K42	♥	L46	K52	♥	M16	K61	♥	K58	L09	♥	J30	L29	♥	L34	L47	♥	K43	L71	♥	L57
K08	♥	K15	K22	♥	J42	K31	♥	K30	K42	♥	L48	K53	♥	K49	K61	♥	K68	L09	♥	K32	L30	♥	L01	L47	♥	L18	L71	♥	M25
K08	♥	K21	K22	♥	K32	K31	♥	K35	K43	♥	K29	K53	♥	K56	K62	♥	K53	L09	♥	L07	L30	♥	L32	L47	♥	L41	L72	♥	K48
K08	♥	K27	K23	♥	J43	K31	♥	K41	K43	♥	K42	K53	♥	K58	K62	♥	K63	L10	♥	J31	L30	♥	L33	L47	♥	L49	L72	♥	L56
K08	♥	K28	K23	♥	K06	K31	♥	L10	K43	♥	K47	K53	♥	K62	K62	♥	L93	L10	♥	K31	L31	♥	D57	L47	♥	L50	L72	♥	M24
K09	♥	J29	K23	♥	K12	K31	♥	L28	K43	♥	L42	K53	♥	K63	K62	♥	L95	L10	♥	L07	L31	♥	D58	L48	♥	K42	L72	♥	M25
K09	♥	K08	K23	♥	K18	K31	♥	L39	K43	♥	L43	K53	♥	K68	K62	♥	L97	L11	♥	J23	L31	♥	L00	L48	♥	L31	L73	♥	K48
K09	♥	K14	K23	♥	K24	K32	♥	K01	K43	♥	L45	K53	♥	K69	K62	♥	L99	L11	♥	K06	L31	♥	L26	L48	♥	L64	L73	♥	L56
K09	♥	K20	K23	♥	K31	K32	♥	K03	K43	♥	L47	K53	♥	L55	K62	♥	M01	L11	♥	K12	L31	♥	L27	L48	♥	L82	L74	♥	K48
K09	♥	K26	K24	♥	J44	K32	♥	K04	K44	♥	K28	K53	♥	L90	K62	♥	M03	L11	♥	K18	L31	♥	L48	L48	♥	M21	L74	♥	K55
K09	♥	K33	K24	♥	K05	K32	♥	K10	K44	♥	K41	K53	♥	L91	K63	♥	K52	L11	♥	K24	L31	♥	L49	L48	♥	M22	L74	♥	L82
K09	♥	L14	K24	♥	K11	K32	♥	K16	K44	♥	K46	K53	♥	L92	K63	♥	K53	L12	♥	J22	L32	♥	L30	L49	♥	L31	L75	♥	K47
K10	♥	J30	K24	♥	K17	K32	♥	K22	K44	♥	K47	K53	♥	L97	K63	♥	K62	L12	♥	L17	L32	♥	L37	L49	♥	L47	L75	♥	K54
K10	♥	K32	K24	♥	K23	K32	♥	K34	K45	♥	K33	K53	♥	L98	K63	♥	L91	L12	♥	L23	L32	♥	L63	L49	♥	L81	L75	♥	K60
K11	♥	J31	K24	♥	K30	K32	♥	K40	K45	♥	K34	K53	♥	M03	K63	♥	L92	L13	♥	L22	L32	♥	L83	L49	♥	L82	L75	♥	L81
K11	♥	K06	K24	♥	L11	K32	♥	L09	K45	♥	K35	K53	♥	M04	K63	♥	L94	L14	♥	J20	L33	♥	L28	L50	♥	L46	L76	♥	K55
K11	♥	K12	K25	♥	J45	K32	♥	L38	K45	♥	K40	K53	♥	M09	K63	♥	L96	L14	♥	K09	L33	♥	L29	L50	♥	L47	L76	♥	L54
K11	♥	K18	K25	♥	K29	K33	♥	K00	K45	♥	L17	K53	♥	M10	K63	♥	L98	L14	♥	K15	L33	♥	L30	L50	♥	L80	L76	♥	L90
K11	♥	K24	K26	♥	J46	K33	♥	K02	K45	♥	L82	K53	♥	M15	K63	♥	M00	L14	♥	K21	L33	♥	L62	L51	♥	L46	L77	♥	K55
K11	♥	K31	K26	♥	K09	K33	♥	K09	K46	♥	K38	K53	♥	M16	K63	♥	M02	L14	♥	K27	L33	♥	L63	L51	♥	L79	L77		



L81 ♥ L83	L84 ♥ L82	L92 ♥ M17	L97 ♥ M12	M02 ♥ K63	M07 ♥ K56	M14 ♥ L95	M21 ♥ K56	M24 ♥ M28	M28 ♥ M24
L81 ♥ L84	L85 ♥ L80	L93 ♥ K62	L98 ♥ K52	M02 ♥ M07	M07 ♥ M02	M15 ♥ K53	M21 ♥ K68	M25 ♥ L71	M28 ♥ M25
L81 ♥ L85	L85 ♥ L81	L93 ♥ M16	L98 ♥ K53	M03 ♥ K53	M08 ♥ M01	M15 ♥ L94	M21 ♥ L48	M25 ♥ L72	M28 ♥ M30
L82 ♥ K45	L87 ♥ L78	L94 ♥ K56	L98 ♥ K63	M03 ♥ K62	M09 ♥ K53	M16 ♥ K52	M21 ♥ L82	M25 ♥ M20	M29 ♥ L67
L82 ♥ L48	L88 ♥ L77	L94 ♥ K63	L98 ♥ M11	M03 ♥ M06	M09 ♥ M00	M16 ♥ K53	M21 ♥ M24	M25 ♥ M27	M29 ♥ M23
L82 ♥ L49	L88 ♥ L78	L94 ♥ M15	L99 ♥ K62	M04 ♥ K52	M10 ♥ K52	M16 ♥ L93	M22 ♥ K55	M25 ♥ M28	M30 ♥ L65
L82 ♥ L74	L90 ♥ K53	L95 ♥ K55	L99 ♥ M10	M04 ♥ K53	M10 ♥ K53	M17 ♥ L92	M22 ♥ K56	M26 ♥ A02	M30 ♥ L66
L82 ♥ L84	L90 ♥ L76	L95 ♥ K56	M00 ♥ K56	M04 ♥ K63	M10 ♥ L99	M18 ♥ K56	M22 ♥ K67	M27 ♥ L69	M30 ♥ M27
L82 ♥ M20	L91 ♥ K53	L95 ♥ K62	M00 ♥ K63	M04 ♥ M06	M11 ♥ L98	M19 ♥ K55	M22 ♥ L18	M27 ♥ L70	M30 ♥ M28
L82 ♥ M21	L91 ♥ K63	L95 ♥ M14	M00 ♥ M09	M05 ♥ K57	M12 ♥ K56	M19 ♥ K56	M22 ♥ L48	M27 ♥ M25	
L83 ♥ L32	L91 ♥ M19	L96 ♥ K63	M01 ♥ K55	M06 ♥ K56	M12 ♥ L97	M19 ♥ L91	M23 ♥ M28	M27 ♥ M30	
L83 ♥ L81	L92 ♥ K52	L96 ♥ M13	M01 ♥ K56	M06 ♥ M03	M13 ♥ K55	M20 ♥ L82	M23 ♥ M29	M28 ♥ L68	
L83 ♥ M20	L92 ♥ K53	L97 ♥ K53	M01 ♥ K62	M06 ♥ M04	M13 ♥ K56	M20 ♥ L83	M24 ♥ L72	M28 ♥ L69	
L84 ♥ L81	L92 ♥ K63	L97 ♥ K62	M01 ♥ M08	M07 ♥ K55	M13 ♥ L96	M20 ♥ M25	M24 ♥ M21	M28 ♥ M23	

**D Enumeration of the environments together with their proof-trees**

The following tables refer to the proof-trees on the website:

<https://www.irif.fr/~nschaban/oritatami/prooftrees/>

proving the correctness of the folding of our design in every possible surroundings.

**ZIG-UP**

---

<b>A</b>				
	#0-98	#1286-1312	#-	#4995-4997

---

<b>B</b>				
	#99-103	#1313-	#-	#4998-5000

---

<b>C</b>	
	#104-159 #1314-1315

---

<b>D<sub>a</sub></b>		
	#160-339	#340-384

---

<b>E</b>		
	#1316-1392	#385-749

---

<b>F</b>		
	#750-856	#1393-1401

---

<b>G</b>		
	#857-1285	#1402-1853

# XX:62 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

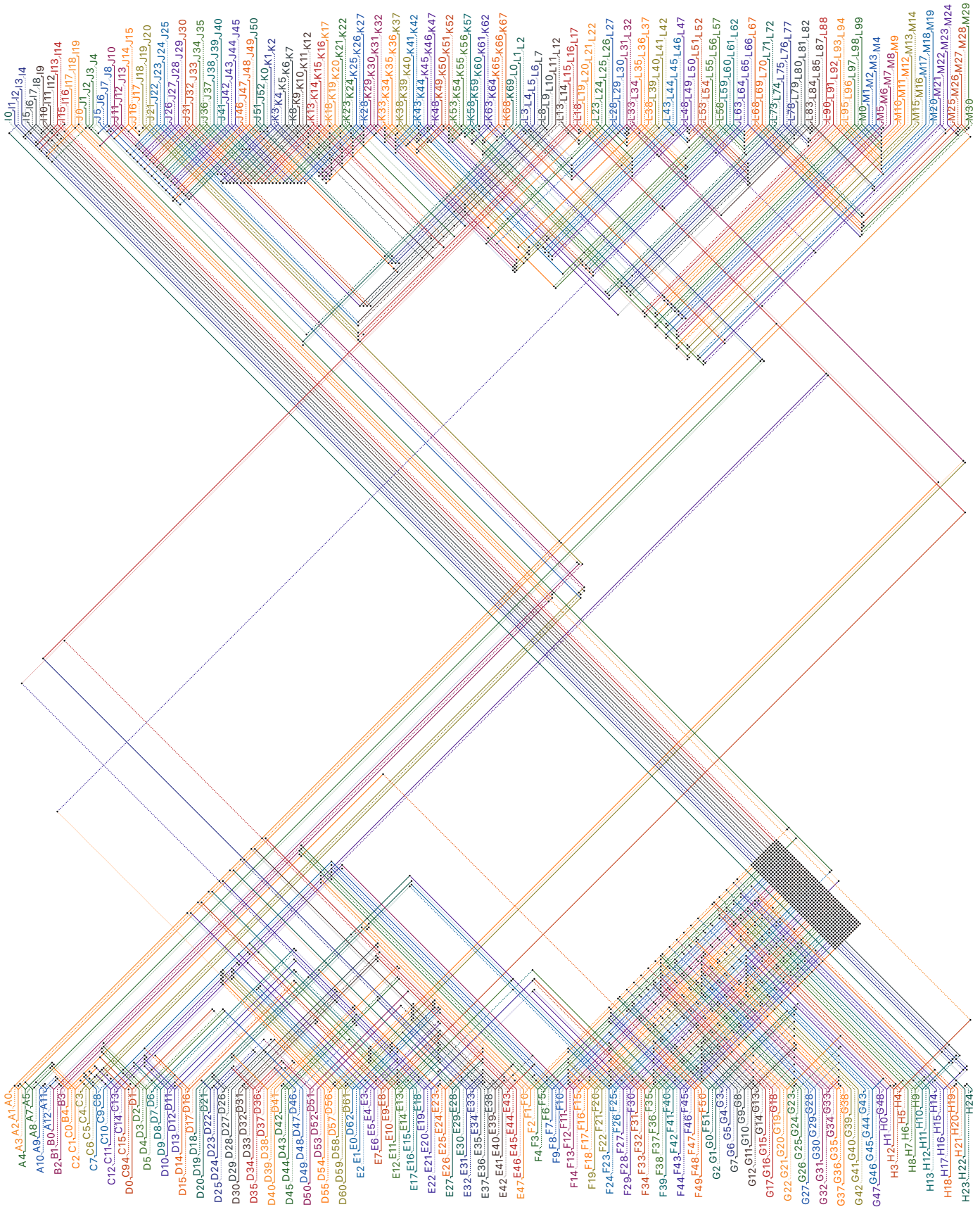
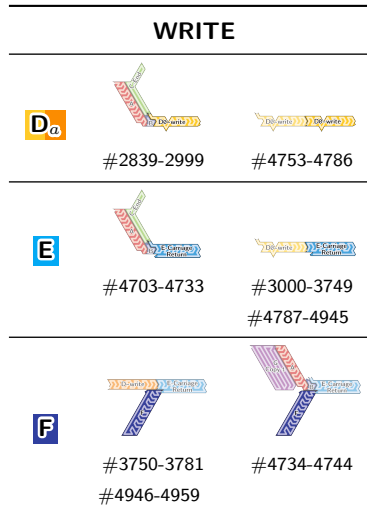
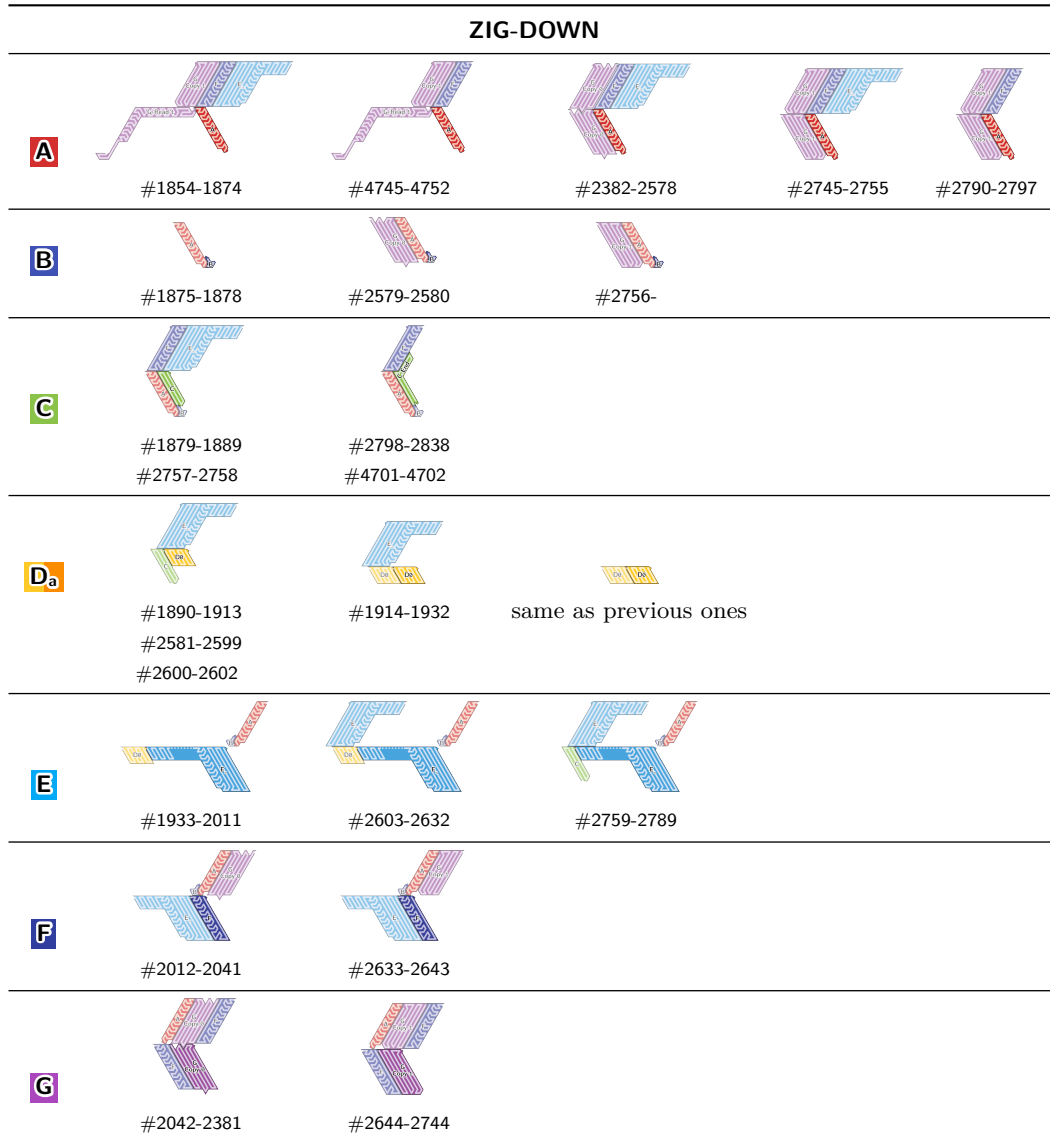


Figure 40 The rule of the SCTS Oritatami system: in this diagram, we have  $b \heartsuit b'$  iff there is a bullet  $\bullet$  at the intersection of one the two lines coming from  $b$  and from  $b'$ ; for instance, we have  $A0 \heartsuit A2$  but not  $A0 \heartsuit A5$ .





# XX:64 Proving the Turing Universality of Oritatami Co-Transcriptional Folding

## ZAG-WRITE

**A**



#4994-



#3806-3816 , #4059-4069 , #4215-4225  
#4310-4320

**C**



#3817-3827 , #4070-4080 , #4226-4236  
#4321-4331 , #4459-4460 , #4475-4476  
#4550-4551 , #4605-4606

**D<sub>a</sub>**



#3828-3851  
#4237-4263  
#4332-4355



#3939-3941 , #4434-4439 , #4525-4527  
#4571-4573 , #4587-4589 , #4595-4597  
#4618-4620

**E**



#4081-4176 , #4461-4474  
#4552-4570 , #4607-4617



#3852-3924 , #3942-4020 , #4264-4271  
#4356-4428 , #4440-4458 , #4504-4519  
#4528-4549 , #4574-4581 , #4590-4594  
#4598-4604 , #4621-4644

**F**



#3925-3938 , #4021-4034 , #4177-4190  
#4272-4285 , #4429-4433 , #4520-4524  
#4582-4586



#4645-4653



#4960-4967


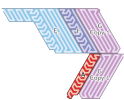


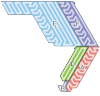





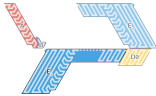






**G**



#4968-4993



#3782-3805 , #4035-4058 , #4191-4214  
#4286-4309

ZAG			
<b>A</b>			
	see Zig-Down	see Zig-Down	
<b>B</b>			
	see Zig-Down	see Zig-Down	
<b>C</b>			
	see Zig-Down		
<b>D<sub>a</sub></b>			
	see Zig-Down	see Zig-Down	see Zig-Down
<b>E</b>			
	see Zig-Down	see Zig-Down	see Zig-Down
<b>F</b>			
	see Zig-Down	see Zig-Down	
<b>G</b>			
	#4654-4700	see Zig-Down	see Zig-Down