Abstract

We study the oritatami model for molecular co-transcriptional folding. In oritatami systems, the transcript (the “molecule”) folds as it is synthesized (transcribed), according to a local energy optimisation process, which is similar to how actual biomolecules such as RNA fold into complex shapes and functions as they are transcribed. We prove that there is an oritatami system embedding universal computation in the folding process itself.

Our result relies on the development of a generic toolbox, which is easily reusable for future work to design complex functions in oritatami systems. We develop “low-level” tools that allow to easily spread apart the encoding of different “functions” in the transcript, even if they are required to be applied at the same geometrical location in the folding. We build upon these low-level tools, a programming framework with increasing levels of abstraction, from encoding of instructions into the transcript to logical analysis. This framework is similar to the hardware-to-algorithm levels of abstractions in standard algorithm theory. These various levels of abstractions allow to separate the proof of correctness of the global behavior of our system, from the proof of correctness of its implementation. Thanks to this framework, we were able to computerise the proof of correctness of its implementation and produce certificates, in the form of a relatively small number of proof trees, compact and easily readable/checkable by human, while encapsulating huge case enumerations. We believe this particular type of certificates can be generalised to other discrete dynamical systems, where proofs involve large case enumerations as well.

Digital Object Identifier 10.4230/LIPIcs...1

1 Introduction

Oritatami model was introduced in [5] to try to understand the kinetics of co-transcriptional folding. This process has been shown to play an important role in the final shape of biomolecules [1], especially in the case of RNA [4]. 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 [6].

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 configuration), 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.

Previous work on oritatami includes the implementation of a binary counter [5], the Heighway dragon fractal [11], folding of shapes at small scale [3], and NP-hardness of the rule minimization [14, 8] and of the equivalence of non-deterministic oritatami systems [9].

Main result. In this paper, we construct a “universal” set of 542 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 \( \mathcal{M} \). The reduction proceeds as follows:

\[
\text{Turing machine} \xrightarrow{[15, 12]} \text{Cyclic tag system} \xrightarrow{\text{Prop. 2}} \text{Skipping cyclic tag system} \xrightarrow{\text{Thm. 6}} \text{Oritatami system}
\]

Our result relies on the development of a generic toolbox, which is easily reusable for future work to design complex functions in oritatami systems.

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 [7, 2] or tile assembly systems [10]. 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: blocks, where exactly one step of the simulation is performed, which are composed of bricks, 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.

2 Definitions and Main results

2.1 Oritatami Systems

Let \( B \) be a finite set of bead types. A configuration \( c \) of a bead type sequence \( p \in B^* \cup B^\infty \) is a directed self-avoiding path in the triangular lattice \( \mathbb{T} \),\(^1\) where for all integer \( i \), vertex \( c_i \) of \( c \) is labelled by \( p_i \). \( c_i \) is the position in \( \mathbb{T} \) of the \( (i+1) \)th bead, of type \( p_i \), in configuration \( c \). A partial configuration of a sequence \( p \) is a configuration of a prefix of \( p \).

For any partial configuration \( c \) of some sequence \( p \), an elongation of \( c \) by \( k \) beads (or \( k \)-elongation) is a partial configuration of \( p \) of length \( |c| + k \) extending by \( k \) positions the self-avoiding path \( c \). We denote by \( C_p \) the set of all partial configurations of \( p \) (the index \( p \) will be omitted when the context is clear). We denote by \( c^k \) the set of all \( k \)-elongations of a partial configuration \( c \) of sequence \( p \).

Oritatami systems. An oritatami system \( \mathcal{O} = (p, \heartsuit, \delta) \) is composed of (1) a (possibly infinite) bead type sequence \( p \), called the transcript, (2) an attraction rule, which is a symmetric relation \( \heartsuit \subseteq B^2 \), (3) a parameter \( \delta \) called the delay. \( \mathcal{O} \) is said periodic if \( p \) is infinite and periodic. Periodicity ensures that the “program” \( p \) embedded in the oritatami system is

\(^1\) The triangular lattice is defined as \( \mathbb{T} = (\mathbb{Z}^2, \sim) \), where \( (x, y) \sim (u, v) \) if and only if \( (u, v) \in \cup_{c=\pm 1} \{ (x+c, y), (x+y+c), (x+c, y+c) \} \). Every position \( (x, y) \) in \( \mathbb{T} \) is mapped in the euclidean plane to \( x\cdot \vec{E} + y\cdot \vec{SW} \) using the vector basis \( \vec{E} = (1, 0) \) and \( \vec{SW} = \text{RotateClockwise}(\vec{E}, 120^\circ) = (-\frac{1}{2}, -\frac{\sqrt{3}}{2}) \).
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 (partial) configuration $c$ of a bead type sequence $q$, we say that there is a bond between two adjacent positions $c_i$ and $c_j$ of $c$ in $\mathbb{T}$ if $q_i \heartsuit q_j$ and $|i - j| > 1$. The number of bonds of configuration $c$ of $q$ is denoted by $H(c) = |\{(i, j) : c_i \sim c_j, j > i + 1, \text{ and } q_i \heartsuit q_j\}|$.

### Oritatami dynamics

The folding of an oritatami system is controlled by the delay $\delta$. Informally, the configuration grows from a seed configuration (the input), one bead at a time. This new bead adopts the position(s) that maximise the potential number of bonds the configuration 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 [5]; it might also be considered as closer to experimental conditions such as in [6].

Formally, given an oritatami system $O = (p, \heartsuit, \delta)$ and a seed configuration $\sigma$ of a seed bead type sequence $s$, we denote by $C_{\sigma, p}$ the set of all partial configurations of the sequence $s \cdot p$ elongating the seed configuration $\sigma$. The considered dynamics $D : 2^{c^{s \cdot p}} \rightarrow 2^{c^{s \cdot p}}$ maps every subset $S$ of partial configurations of length $\ell$ elongating $\sigma$ of the sequence $s \cdot p$ to the subset $D(S)$ of partial configurations of length $\ell + 1$ of $s \cdot p$ as follows:

$$D(S) = \bigcup_{c \in S} \arg \max_{\gamma \in c^{p \cdot 1}} \left( \max_{\eta \in \gamma^{\delta^{(p - 1)}}} H(\eta) \right)$$

The possible configurations at time $t$ of the oritatami system $O$ are the elongations of the seed configuration $\sigma$ by $t$ beads in the set $D^t(\{\sigma\})$.

We say that the oritatami system is deterministic if at all time $t$, $D^t(\{\sigma\})$ is either a singleton or the empty set. In this case, we denote by $c^t$ the configuration at time $t$, such that: $c^0 = \sigma$ and $D^t(\{\sigma\}) = \{c^t\}$ for all $t > 0$; we say that the partial configuration $c^t$ folds (co-transcriptionally) into the partial configuration $c^{t+1}$ deterministically. In this case, at time $t$, the $(t + 1)$-th bead of $p$ 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 $D^t(\{\sigma\}) = \emptyset$. The folding process may only stop because of a geometric obstruction (no more elongation is possible because the configuration is trapped in a closed area).

Please refer to Fig. 1(d) and 1(e) for examples of the dynamical folding of a transcript. Observe that a given transcript may fold (deterministically) into different paths because of its interactions with its local environment (see section 2.3 for more details).

### 2.2 Main result

Our main result consists in proving the following theorem that demonstrates that oritatami systems are able to complete arbitrary Turing computation. It shows in particular that deciding whether a given oritatami system folds into a finite size shape for a given seed is undecidable.

**Theorem 1 (Main result).** There is a fixed set $B$ of 542 bead types with a fixed attraction rule $\heartsuit$ on $B$, together with two encodings:

- $\pi$ that maps in polynomial time, any single tape Turing machine $M$ to a bead type sequence $\pi_M \in B^*$;
that maps in polynomial-time, any single-tape Turing machine $M$ and any input $x$ of $M$ to a seed configuration $\sigma_M(x)$ of a bead type sequence $s_M(x)$ of length $O_M(|x|)$, linear in the size of the input $x$ (and polynomial in $|M|$);

such that: For any single tape Turing machine $M$ and every input $x$ of $M$, the deterministic and periodic oritatami system $O_M = ((\pi_M)^\infty, \spadesuit, 3)$ whose transcript has period $\pi_M$ and whose delay is $\delta = 3$, halts its folding from the seed configuration $\sigma_M(x)$ if and only if $M$ halts on input $x$. Furthermore, for all $t$ and all input $x$ of $M$, if $M$ halts on $x$ after $t$ steps, then the folding of $O_M$ from seed configuration $\sigma_M(x)$ halts after folding $O_M(t^4 \log^2 t)$ beads.

There is one Turing-universal periodic transcript. Note that if we apply this theorem to an intrinsically universal single tape Turing machine $U$ (see [13]), then we obtain one single absolutely fixed transcript $\pi_U$ such that the deterministic and periodic oritatami system $O_U = ((\pi_U)^\infty, \spadesuit, 3)$ with 542 bead types can simulate efficiently the halting of any Turing machine $M$ on any input $x$ using a suitable seed configuration obtained via the encoding of $M$ and $x$ in $U$. It follows that this absolutely fixed oritatami system consisting of one single periodic transcript is able of arbitrary Turing computation.

From now on, we only consider deterministic periodic oritatami systems with delay $\delta = 3$.

2.3 Basic design tool: Glider/Switchback

As a warm-up, let us introduce a special type of bead sequence (see Fig. 1) that, depending on the initial context of its folding, either folds as a glider (a long and thin self-supported shape heading in a fixed direction) or as switchbacks (a narrow and high shape allowing compact storage). This only requires a small number of distinct beads types (12 per switchbacks, that can be repeated every 4 switchbacks). This is achieved by designing a rule with minimum interactions ensuring minimum interferences between both folding patterns. Compatibility between the glider and the turns in switchbacks is ensured by aligning the switchback turns with the turns of the glider, exploiting thus the similarity of their finger-like shape there.

This glider/switchback sequence will be used to store (as switchbacks) and expose (as glider) specific information encoded in the transcript when needed.

2.4 Skipping Cyclic Tag Systems and Turing-Universality

Our proof of the Turing-universality of oritatami systems consists in simulating a special kind of cyclic tag systems (CTS), called skipping cyclic tag system. Cook introduced CTS in [2] and proved that they combined the tremendous advantage of simulating efficiently any Turing machines, while not requiring a random access lookup table, which makes simulation a lot easier.

A skipping cyclic tag system (SCTS) consists of a cyclic list of $n$ words $\alpha = (\alpha^0, \ldots, \alpha^{n-1}) \in \{0, 1\}^*$, called appendants, and an initial dataword $u^0 \in \{0, 1\}^*$. Intuitively, $\alpha$ encodes the program and $u^0$ encodes the input. Its configuration at time $t$ consists of a marker $m^t$, recording the index of the current appendant at time $t$, and a dataword $u^t$. Initially, $m^0 = 0$ and the dataword is $u^0$. At each time step $t$, the SCTS acts deterministically on configuration $(m^t, u^t)$ in one of three ways:

(Halt step) If $u^t$ is the empty word $\epsilon$, then the SCTS halts;\(^2\)

\(^2\) Note that SCTS halting condition requires the dataword to be empty as opposed to [2, 15] where the
Moreover, the number of appendants of $S$ that for all input $u$ with a linear-time encoding there is a polynomial algorithm that computes a skipping cyclic tag system efficiently in the following sense: (proof deferred to appendix on page 17)

**Turing universality.** A sequence of articles and thesis by Cook [2], and Neary and Woods [15, 12], allows to show that SCTS are able to simulate any Turing machine efficiently in the following sense: (proof deferred to appendix on page 17)

> **Proposition 2** ([15, 12]). Let $M$ be a deterministic Turing machine using a single tape. There is a polynomial algorithm that computes a skipping cyclic tag system $S_M$, together with a linear-time encoding $u_M(x)$ of the input $x$ of $M$ as an input dataword for $S_M$, such that for all input $x$: $S_M$ halts on input dataword $u_M(x)$ if and only if $M$ halts on input $x$. Furthermore, for all $t$, if $M$ halts after $t$ steps, then $S$ halts after $O_M(t^2 \log t)$ steps. Moreover, the number of appendants of $S$ is a multiple of 4.

---

**Figure 1 Glider/switchback subsequence.** The folding path of the transcript is represented as the thick colorful line and the $\heartsuit$-bonds between beads are represented as dashed lines. The bond-maximizing path for the $\delta = 3$ lastly produced beads is represented by a thick black line, possibly terminated by several colorful paths if several paths realize the maximum of number of bonds.

**(Nop step)** If the first letter $u^t_0$ of $u^t$ is 0, then $u^t_0$ is deleted and the marker moves to the next appendant cyclically: i.e., $m^{t+1} = (m^t + 1) \mod n$ and $u^{t+1} = u^t_1 \cdots u^{t+1}_{|u^t|} - 1$;

**(Skip-append step)** If $u^t_0 = 1$, then $u^t_0$ is deleted, the next appendant $a^{(m^t + 1) \mod n}$ is appended onto the right end of $u^t$, and the marker moves to the second next appendant: i.e., $u^{t+1} = u^t_1 \cdots u^{t+1}_{|u^t|} \cdot a^{(m^t + 1) \mod n}$ and $m^{t+1} = (m^t + 2) \mod n$.

For example, consider the SCTS $E = ([110, e, 11, 0]; u^0 = 010)$. Its execution $(m^t u^t)_t$ is:

\[
[0]010 \rightarrow [1]10 \rightarrow [2;11]3011 \rightarrow [0]11 \rightarrow [1;e]21 \rightarrow [2;1]00 \rightarrow [1]Halt
\]
In order to prove Theorem 1, we are thus left with proving that there is an oritatami system that simulates in quadratic time any SCTS system (see Theorem 6 in appendix for a precise statement).

### 3 The block simulation of SCTS: Proving the correctness of local folding is enough

Given a SCTS $S$, we design an oritatami system $O_S$ that folds into a version, at a larger scale, of the annotated trimmed space-time diagram of $S$ (or trimmed diagram for short) defined as follows:

**Trimmed diagram of SCTS.** Any SCTS proceeds as follows: it trims all the leading $0$s in the data word and then appends the currently marked appendant when it reads the first $1$ (if any; otherwise it halts). It is thus natural to group all these steps (trim leading $0$s and process the leading $1$) as one single macro step. This motivates the following representation. Given a SCTS $(α_0, ..., α^{n-1}; u_0)$, we denote by $t_0 = -1$ by convention. Let us now group all deletion steps occurring during steps $t_i + 1$ to $t_{i+1} - 1$ by simply indicating in exponent the marker $m^t$ before each letter read. In the case of our STCS $E$, we have $t_0 = -1, t_1 = 1, t_2 = 3, t_3 = 4$ and its execution is now represented as:

\[
\begin{align*}
[0]^1_0 &\xrightarrow{\text{Append } [2:11]} [3]_0^1 11 \\
&\xrightarrow{\text{Append } [1:ε]} [2]_1 1 \\
&\xrightarrow{\text{Append } [3:0]} [0]_0^1 0 \\
&\xrightarrow{\text{Halt } [1:ε]}
\end{align*}
\]

Now, let’s align the resulting datawords in a 2D diagram according to their common parts:

\[
\begin{array}{c|c|c|c}
\hline
&t_0 & t_1 & t_2 & t_3 \\
\hline
\text{Read } 0 & 0 & 1 & 0 & \rightarrow \text{Append } [2:11] \\
\text{Read } 1 & [3]_0^1 & 1 & [2]_1 & \rightarrow \text{Append } [1:ε] \\
\text{Halt } & [0]_0^1 & \rightarrow \text{Append } [3:0] \\
\hline
\end{array}
\]

This defines the *annotated trimmed space-time diagram* for the SCTS $E$. Lemma 4 in appendix provides the formal definition for an arbitrary SCTS.

**The transcript.** The proof of Theorem 6 relies on constructing a transcript (and a fixed rule) that will reproduce faithfully the trimmed diagram of the simulated STCS. Figure 2 illustrates the folded configuration of the transcript corresponding to SCTS $E$. Macroscopically, the transcript folds into a zig-zag sequence of blocks, each performing a specific operation.

**The current dataword** is encoded at the bottom of each row of blocks: $0$s are encoded by a spike, and $1$s are encoded by a flat surface.

**The seed configuration** encodes the initial dataword and opens the first zig row at which the folding of the transcript starts. Letters $0$ and $1$ are encoded by a spike (see Fig. 3(a)) and a flat surface (see Fig. 3(b)) respectively.

**In each zig row (left to right),** the transcript folds into a series of **Read** blocks (trimming the leading $0$s from the dataword encoded above), then into a **Read** block, if the dataword contains a $1$, or into a **Halt** block terminating the folding, otherwise; this is the **zig-up phase**. Then, the transcript starts the **zig-down phase** which consists in folding
into Copy block copying the letters encoded above to the bottom of the row; once the end of the dataword is reached, the transcript folds into an Append Return block which encodes, at the bottom of the row, the currently marked appendant, and finally, opens the next zag row.

**in each zag row (right to left)**, the transcript folds into Copy blocks copying the dataword encoded above to the bottom of the row. For the leftmost letter, the transcript folds into the special Copy LineFeed block which also opens the next zig row.

The transcript is a periodic sequence whose period is the concatenation of \( n \) bead type sequences \( \text{Appending} \alpha^0, \ldots, \text{Appending} \alpha^{n-1} \) called segments, each encoding one appendant.

**Encoding of the marker.** Read and Append Return blocks consist of the folding of exactly one segment, whereas Copy, Copy and Copy LineFeed consist of the folding of exactly \( n \) segments. It follows that the appendant encoded in the leading segment folded inside each block corresponds to the currently marked appendant in the simulated SCTS. As a consequence, the appendant contained in the folded Append Return block is indeed the appendant to be appended to the dataword.

**The segment sequence.** Each segment \( \text{Appending} \alpha^i \) encodes the appendant \( \alpha^i \) as a sequence of \( 6 + |\alpha^i| \) modules: one of each module \( [A], [B], \) and \( [C] \), then \( |\alpha^i| \) of module \( [D] \), then one of each module \( [E], [F] \) and \( [G] \). Each module is a bead type sequence that plays a particular role in the design:

Module \( [A] \) folds into the initial scaffold upon which the next modules rely.

Module \( [B] \) detects if the dataword is empty: if so, it folds to the left and the folding gets trapped in a closed space and halts; otherwise, it folds to the right and the folding continues.

Module \( [C] \) detects the end of the dataword and triggers the appending of the marked appendant accordingly.

Module \( [D] \) encodes each letter of the appendant.

Module \( [E] \) ensures by padding that all appendant sequences have the same length when folded (even if the appendant have different length). It serves two other purposes: Module \( [B] \) senses its presence to detect if the dataword is empty; and its folding initiates the opening the zag row once the marked appendant has been appended to the dataword.

Module \( [F] \) is the scaffold upon which Module \( [G] \) folds. It is specially designed to induce two very distinct shapes on \( [G] \) depending on the initial shift of \( [G] \). Furthermore, when Module \( [F] \) is exposed, Module \( [C] \) folds along \( [F] \) which triggers the appending of the marked appendant encoded by the modules \( [D] \) following \( [C] \).

Module \( [G] \) is the “logical unit” of the transcript. It implements three distinct functions which are triggered by its interactions with its environment: Reading the leading letter of the dataword, Copying a letter of the dataword, and Opening the next zig row at the leftmost end of a zag row.

We call bricks the folding of each of these modules. The blocks into which the transcript folds, depend on the bricks in which its modules fold, as illustrated in Fig. 2(b). Please refer to sections C to F in appendix for the description of blocks in terms of bricks and of how they articulate with each other to produce the desired macroscopic folding pattern.

The full description of each of these sequences is given in Section F in appendix.

Let \( S = (\alpha^0, \alpha^1, \ldots, \alpha^{n-1}; u^0) \) be a skipping cyclic tag system, and, as before, let for all integer \( i \geq 0 \), \( t_i \) be the \( i^{th} \) step where \( u^i \) starts with 1 (starting from 0, i.e. \( t_0 \) is the
(a) Folding of the oritatami system simulating the STCS $E$.

(b) Exploded view of the bricks and modules inside the blocks involved in the simulation above.

**Figure 2** Folding of the transcript simulating the STCS $E$, and some block internal structures.
first step where \( u^0 \) starts with 1). The following lemma shows that the transcript described above folds indeed into blocks that simulates the trimmed diagram of \( S \). Proposition 2 and Theorem 6 are direct corollaries of this lemma.

**Lemma 3 (Key lemma).** There is a bead type set \( B \) and a rule \( \heartsuit \) such that: for every SCTS \( S \), there are \( \pi_S \) and \( (\sigma_S, S) \) defined as in Theorem 1 such that, for every initial dataword \( u^0 \), the (possibly infinite) final folded path of the oritatami system \( O_S = ((\pi_S)^\infty, \heartsuit, \delta = 3) \) from the seed configuration \((\sigma(u^0), s(u^0))\) is exactly structured as the following sequence of blocks organized in zig and zag rows as follows: (recall Fig. 2(a))

- First, the block \( \text{Seed}(u^0) \) ending at coordinates \((-1,0)\).
- Then, for \( i \geq 0 \), the \( i \)-th row consists of a zig row located between \( y = 2(i - 1)h + 1 \) and \( y = 2i + 1 \), and a zag row located between \( y = 2i + 1 \) and \( y = 2(i + 1)h \), composed as follows:
  - **(Compute)** if \( u^{1+i} = 0^{r+1} \cdot s \) and if \( s \neq \epsilon \) or \( \alpha^{1+i+t+1} \neq \epsilon \); then
    \( r = t_{i+1} - t_i - 1 \) and:
    - the \( i \)-th zig-row consists from left to right of the following sequence of blocks whose origins are located at the following coordinates:
      \[
      \begin{array}{c|c|c|c}
      \checkmark y & 2ih & (2i - 1)h + 1 \\
      \hline
      \text{Blocks} & \text{Read}\text{0} & \cdots & \text{Read}\text{2} \\
      \text{Marker} & i + 1 + t_i & \cdots & i + r + t_i & i + t_{i+1} & i + 1 + t_{i+1} \\
      \end{array}
      \]
      This row ends at position \(((i + 1)h + (1 + |s| + |\alpha^{1+i+t+1} + t_{i+1}|)W - 7, 2ih + 2)\).
    - the \( i \)-th zag-row consists from right to left of the following sequence of blocks whose origins are located at the following coordinates:
      \[
      \begin{array}{c|c|c|c}
      \checkmark y & 2ih + 1 & (2i - 1)h + 1 \\
      \hline
      \text{Blocks} & \text{Copy}(s) & \text{LineFeed} & \text{Copy}(\pi) \\
      \text{Marker} & i + 2 + t_{i+1} & \cdots & i + 2 + t_{i+1} \\
      \end{array}
      \]
      where \( v = u^{1+i+t+1} = s \cdot \pi_{i+1} + t_{i+1} \neq \epsilon \) (as \( s \) and \( \alpha^{1+i+t+1} \) are not both \( \epsilon \)). This row ends at position \(((i + 1)h + (1 + t_{i+1})W - 1, 2(i + 1)h)\).
  - **(Halt)** if \( u^{1+i} = 0^{r} \) and \( \alpha^{1+i+t+1} = \epsilon \); then \( r = t_{i+1} - t_i - 1 \) and the last rows of the configuration consists from left to right of the following sequence of blocks located at the following coordinates:
    \[
    \begin{array}{c|c|c|c}
    \checkmark y & 2ih & (2i - 1)h + 1 & 2(i + 1)h \\
    \hline
    \text{Blocks} & \text{Read}\text{0} & \cdots & \text{CarriageReturn} & \text{Halt} \\
    \text{Marker} & i + 1 + t_i & \cdots & i + t_{i+1} & i + t_{i+1} \\
    \end{array}
    \]
  - **finally, (Halt 2)** if \( u^{1+i} = 0^r \) for some \( r \geq 0 \); then the \( i \)-th zig-row is last row of the configuration and consists of the following sequence of blocks located at the following coordinates:
    \[
    \begin{array}{c|c}
    \checkmark y & 2ih \\
    \hline
    \text{Blocks} & \text{Read}\text{0} & \text{Read}\text{0} \\
    \text{Marker} & i + 1 + t_i & i + r + t_i & i + r + 1 + t_i \\
    \end{array}
    \]

The following sections are dedicated to the proof of Key Lemma 3.
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.

4.1 Implementing the logic

As in [5], the internal state of our “molecular computing machinery” consists essentially of two parameters: 1) the position inside the transcript of the part currently folding; and 2) the entry point of transcript inside the environment. Indeed, depending on the entry point or the position inside the transcript, different beads will be in contact with the environment and thus different functions will be applied as a result of their interactions. This happens during the zig phase: in the first (zig-up) part, the transcript starts folding at the bottom, forcing the modules $G$ to fold into $\text{G\text{\text{\char'13}}}\text{Read}$ bricks; whereas during the second (zig-down) part, the transcript starts folding at the top, forcing the modules $G$ to fold into $\text{G\text{\text{\char'13}}}\text{Copy}$ bricks instead. Similarly, the memory of the system consists of the beads already placed on the surrounding of the area currently visited (the environment). This happens in every row of the folding: depending on the letter encoded at the bottom of the row above, the modules $G$ fold into $\text{G\text{\text{\char'13}}}\text{Read0}$ or $\text{G\text{\text{\char'13}}}\text{Read1}$ bricks (zig-up phase), $\text{G\text{\text{\char'13}}}\text{Copy0}$ or $\text{G\text{\text{\char'13}}}\text{Copy1}$ bricks (zig-down phase), and $\text{G\text{\text{\char'13}}}\text{Copy0}$ or $\text{G\text{\text{\char'13}}}\text{Copy1}$ bricks (zag phase).

At different places, we need the transcript 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 by $F$ 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 27 in appendix).

**Geometry obstruction.** An typical example is illustrated by $G$. During the zig-up phase where the absence of environment below the block $\text{Read\text{\text{\char'13}}}\text{Read}$ allows $G$ to fold downward at the beginning (see Figure 41) which shift the transcript 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 (see Figure 43).

**Geometry of the environment.** Figure 3 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 by going back to the same entry point (Fig. 3(a)), switch from zip-up to zig-down phase when reading a 1 by opening the next block from the top (Fig. 3(b)), and from zag to zig-up phase when it has rewind to the beginning of the tape word, by getting down to the bottom of the next zig row (Fig. 3(c)).

4.2 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 faced is that the different functions one wants to implement tend to concentrate at the same “hot-spots” in the transcript. A typical example is the midpoint of $G$ where one wants to implement all the functions: Read, Copy and Line Feed. The following powerful tools allow to overcome these difficulties:
Socks work by letting a glider/switchback module fold into a switchback turn conformation for some time when it would otherwise fold into a glider. Examples are given in Figure 4. They are easy to implement: indeed, the socks naturally adopt the same shape as the corresponding switchback turn and require thus no extra interfering bonds. They allow 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 configuration for that part, as shown in Figure 4(a).
- 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 (see Fig. 4(b)) and therefore to realign them. We used that feature repeatedly to “shift” some modules: starting the folding at an initial speed-1 (i.e., straight line) and then compensating for that speed later on by introducing socks (see Fig. 4(b)). This is a key point in our design, as it allowed us to spread apart the Read and Copy functions into different subsequences of module $G$, and therefore to get less constraints on our 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! (see section F.10 for full details)
- Finally, socks allow to prevent unwanted interactions between beads by concealing potentially harmful beads in unreachable area as in Figure 4(c).

Exponential bead type 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. The problem it solves is that in order for $G$ to fold into switchbacks, we need strong interactions between $G$ and its neighboring module $F$ (see Fig. 41), whereas in order for $G$ to fold as glider, we want to avoid those interactions (see Fig. 43). This is made possible because gliders progress at speed 1/3 while switchbacks progress at speed 1. Using a power-of-3 coloring, we manage to easily achieve these contradicting goals altogether (the construction is analysed in Lemma 11 in Section G.1).
Correctness of local folding: Proof tree certificates

The goal of this section is to conclude the proof of our design by proving Key Lemma 3. The proof works by induction, assuming that the preceding beads of the transcript fold at the locations claimed by the lemma. We proceed in 3 steps:

- We first enumerate all the possible environments for every part of the transcript. As, we carefully aligned our design, most of the beads only see a small number of different environments.
- For the few cases (three in total) where the number of environments is unbounded, we give an explicit proof of correctness of their folding (Lemmas 9, 10, and 11 in section G.1). This is where the concealing feature of socks and the exponential bead type coloring play a crucial role.
- For all the other cases, we designed human-checkable computer-generated certificates, called proof trees. It consists in listing in a compact but readable manner all the possible paths for the transcript in every possible environment. In order to match human readability, paths with identical bonding patterns are grouped into one single ball. Balls containing the paths maximizing the number of bonds are highlighted in bold and organized in a tree. This reduces the number of cases to less than 5 balls in most of the levels of the tree, achieving human-checkability of the computed certificate (see Fig. 5 in appendix). Proof trees are available at https://www.irif.fr/~nschaban/oritatami/

This ensures the highest level of certification of the correctness of our design.

References

Proving the Turing Universality of oritatami Co-Transcriptional Folding

Please find next the omitted part of article due to space constraints.
(a) Proof tree for the glider turn in G ». Read 0.

(b) Proof tree for the glider turn in G ». Read 1.

Figure 5 Two examples of proof trees for the same subsequence in two different environments. The number at the upper-left corner of every ball stands for the number of bonds for the path inside the ball. The number at the lower right corner of each ball stands for the number of paths grouped in the ball, allowing to check that no path was omitted. Balls highlighted in black bold contain the bonds-maximizing paths. Balls highlighted in grey bold contain the paths that places the bead at the same location as the bonds-maximizing paths, and which must thus be considered in the next level as well.
A Skipping Cyclic Tag Systems

Notations. We index the letters of every word \( u = u_0 \ldots u_{|u|−1} \) from 0 to \(|u|−1\). Given two words \( u \) and \( v \), we denote by \( u \cdot v \) their concatenation: \( u \cdot v = u_0 \ldots u_{|u|−1}v_0 \ldots v_{|v|−1} \). We denote by \( u^\infty \) the one-way infinite periodic word \( u \cdot u \cdot \ldots \). For all \( i \leq j \), we denote by \( u_{i..j} \) the (possibly empty) factor \( u_{\max(0,i)} \ldots u_{\min(j,|u|−1)} \). The empty word is denoted by \( \epsilon \). The indices in the notation \( O_L(\cdot) \) where \( L \) is a list of variables (for instance \( L = A, B \)) indicates that the constant in the \( O(\cdot) \) only depends on the variable in \( L \) (for instance \( A \) and \( B \)) and on no other values.

A.1 Trimmed diagram

The following lemma gives the formal description of the trimmed diagram of a SCTS \( (\alpha; u^0) \) with marker \( m^t \). Recall that \( t_i \) is the \( i \)-th time \( t \) such that the dataword \( u^t \) starts with letter \( 1 \) (\( t_0 = −1 \) by convention).

Lemma 4. The annotated word on the row \( i \) (indexed from \( i = 0 \)) of the trimmed diagram is: (the markers in exponent are computed modulo \( n \))

- if \( u^{t_i+t_i} = 0^1 \cdot 1 \cdot s \) for some \( r \geq 0 \) and \( s \in \{0,1\}^* \): then, \( r = t_i+1 - t_i - 1 \) and the annotated word on row \( i \) is \([i+1+t_i]0^\ldots[i−1+t_i+1]0^{i+t_i+1}1^* \cdot 1 \cdot s \) whose first letter is placed in column \( t_i + 1 \) (assuming the leftmost column is indexed by \( 0 \));

- if \( u^{1+t_i} = 0^r \) for some \( r > 0 \): then, row \( i \) is the last row of the diagram and its annotated word is \([i+1+t_i]0^\ldots[i+t_i+r]0^{i+t_i+r+1}1\cdot1 \cdot s \) and starts at column \( t_i + 1 \).

Proof sketch. Simply observe that \( m^t = i + t_i \mod n \), as indeed exactly \( t_i \) letters have been read and exactly \( i \) appending steps have occurred before reading the \( i \)-th 1.

A.2 Turing-universality of Skipping Cyclic Tag Systems

This proof makes use of the time-efficient reduction from Turing machines to cyclic tag systems (CTS) designed in [12, Theorem 4.3.2 p. 65], improving on [2, 15].

A cyclic tag system \( C = (\alpha^0, \ldots, \alpha^{n−1}; u^0) \) consists of a list of \( n \) appendants \( \alpha^0, \ldots, \alpha^{n−1} \in \{0,1\}^* \) and an initial dataword \( v^0 \in \{0,1\}^* \). Its configuration at time \( t \) consists of a marker \( m^t \equiv t \mod n \), recording the index of the current appendant at time \( t \), and a dataword \( v^t \). Initially, \( m^0 = 0 \) and the dataword is \( v^0 \). At each time step \( t \), the CTS acts deterministically on configuration \( (m^t, v^t) \) in one of three ways:

-Halt step If \( v^t \) is the empty word \( \epsilon \), then the CTS halts;

-Nop step If the first letter \( v^t_0 \) of \( v^t \) is 0, then \( v^t_0 \) is deleted and the marker moves to the next appendant cyclically: i.e., \( m^{t+1} = (m^t + 1) \mod n \) and \( v^{t+1} = v^t \cdot v^{n−1} \);

-Append step If \( v^t_0 = 1 \), then \( v^t_0 \) is deleted, the currently marked appendant \( \alpha^{(m^t \mod n)} \) is appended onto the right end of \( v^t \): i.e., \( v^{t+1} = v^t \cdot v^{n−1} \cdot \alpha^{(m^t \mod n)} \) and \( m^{t+1} = (m^t + 1) \mod n \) (no skipping).

According to the definition in [12, 2, 15], the computation of a CTS is said to end if either the dataword is \( \epsilon \) or if it repeats a configuration. This relaxed definition of termination was introduced for the purpose of reducing any Turing machine to cellular automaton rule 110, whose computation never halts. Precisely, [12, Theorem 4.3.2, p. 65] states the following: let \( M \) be any deterministic Turing machine using a single tape; there is a cyclic tag system \( C_M \) with appendants \( \alpha^0, \ldots, \alpha^{n−1} \) and a linear-time encoding \( v_M \) of the input \( x \) of \( M \), such that for all input \( x \): (1) \( C_M \) halts from initial dataword \( v_M(x) \) if and only if \( M \) halts from
input \( x \); and (2) for all \( i \), if \( \mathcal{M} \) halts after \( t \) steps on \( x \), then \( T_{\mathcal{M}} \) halts after \( O_M(t^2 \log t) \) steps on \( u_M(x) \). For our purpose, we need the computation of the CTS to \textit{stop with an empty dataword} (and not to enter a cycle) if the simulated Turing machine stops, precisely:

\textbf{Lemma 5} \textbf{(Corollary of Theorem 4.3.2 in [12])}. For every Turing machine \( \mathcal{M} \), there is CTS \( \hat{\mathcal{C}}_{\mathcal{M}} \) and a linear time encoding \( \hat{\mathcal{v}}_{\mathcal{M}}(x) = \chi(v_{\mathcal{M}}(x)) \) that encodes any input \( x \) of \( \mathcal{M} \) into an initial dataword \( \hat{\ell}\hat{v}^0 = \hat{\mathcal{v}}_{\mathcal{M}}(x) \) such that: (1) \( \hat{\mathcal{C}}_{\mathcal{M}} \) halts from \( \hat{\ell}\hat{v}^0 \) with an empty dataword iff \( \mathcal{M} \) halts from input \( x \); and (2) if \( \mathcal{M} \) halts after \( t \) steps from \( x \), then \( \hat{\mathcal{C}}_{\mathcal{M}} \) after \( O(t^2 \log t) \) steps from \( \hat{\ell}\hat{v}^0 \).

\textbf{Proof}. We proceed as follows by defining a CTS \( \hat{\mathcal{C}}_{\mathcal{M}} \) “with two processing modes”: the first mode emulates \( \mathcal{C}_{\mathcal{M}} \), the second mode just erases the data word; switching from one mode to the other just requires inserting a single letter \( 0 \) in the dataword. Consider \( \chi \) the homomorphism on \( \{0,1\}^* \) such that \( \chi(0) = 00 \) and \( \chi(1) = 01 \), i.e. \( \chi \) inserts a \( 0 \) before every letter of a word. Then, consider the CTS \( \hat{\mathcal{C}}_{\mathcal{M}} \) with \( 2n \) appendants: \( \hat{\nu}^{2i+1} = \chi(\hat{\nu}^i) \) and \( \hat{\alpha}^n = \epsilon \), for \( 0 \leq i < n \). An immediate induction shows that \( \hat{\mathcal{C}}_{\mathcal{M}} \) simulates \( \mathcal{C}_{\mathcal{M}} \) exactly twice slower, indeed: if \( \nu^t \) and \( \hat{\nu}^t \) denote the datawords of \( \mathcal{C}_{\mathcal{M}} \) and \( \hat{\mathcal{C}}_{\mathcal{M}} \) starting from the initial datawords \( \nu^0 \) and \( \hat{\nu}^0 = \chi(\nu^0) \) respectively, then for all time \( t \), \( \hat{\nu}^{2t} = \chi(\nu^t) \). Now, if we shift the dataword of \( \hat{\mathcal{C}}_{\mathcal{M}} \) by one letter, the appendants that will be appended next, are all the empty word \( \epsilon \), and the dataword will be completely erased, yielding to the desired terminaison. Without loss of generality, we assume that \( \mathcal{M} \) has an unique final state \( q_\text{F} \).

According to the design of \( \mathcal{C}_{\mathcal{M}} \) in the proof Theorem 4.3.2 p. 65 in [12], for every step \( t \) of \( \mathcal{M} \) where the configuration is \( \cdots B \cdots B \sigma_1 \cdots \sigma_j q d \sigma_{j+1} \sigma_{j+2} \cdots \sigma_{s-2} B \cdots B \cdots \) (i.e., where the head is over position \( \sigma_{j+1} \), the current state is \( q \) and the next head movement is \( d \), \( B \) denotes the blank symbol), the dataword of \( \mathcal{C}_{\mathcal{M}} \) is, at the step \( O(t^2 \log t) \) corresponding to first stage of the processing of this configuration by \( \mathcal{C}_{\mathcal{M}} \):

\[
\langle 1, q, d \rangle \langle \sigma_{j+1} \rangle \cdots \langle \sigma_{s-2} \rangle \langle B \rangle \mu^s \langle \sigma_1 \rangle \cdots \langle \sigma_j \rangle
\]

and the marker is \( 0 \). Each \( \langle \cdot \rangle \) and \( \mu \) stands for a binary encoding containing a single \( 1 \). Furthermore, there is at most one pattern \( \langle 1, q, d \rangle \) in the dataword of \( \mathcal{C}_{\mathcal{M}} \) at all time. Let \( i \) be the index of the only \( 1 \) in \( \langle 1, q, d \rangle \). We then change the appendant \( \hat{\nu}^{2i+1} \) to \( 0 \). It follows that, the first time the pattern \( \chi(1, q, d \rangle \) appears, i.e. the first time the simulated Turing machine \( \mathcal{M} \) enters the final state, the CTS \( \hat{\mathcal{C}}_{\mathcal{M}} \) switches to the even-indexed empty-appendants-mode, then erases the whole dataword, and halts with an empty dataword, as desired. Furthermore, if \( \mathcal{M} \) halts after \( t \) steps, \( \hat{\mathcal{C}}_{\mathcal{M}} \) halts after \( O(t^2 \log t) \) steps. \hfill \Box

We now show how to simulate the CTS \( \hat{\mathcal{C}}_{\mathcal{M}} \) with a SCTS.

\textbf{Proof of Proposition 2}. The original cyclic tag system by Cook [2] 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. Consider the CTS \( \hat{\mathcal{C}}_{\mathcal{M}} \), given by the lemma above, with \( 2n \) appendants \( \hat{\nu}^0, \ldots, \hat{\nu}^{2n-1} \), together with its linear-time input encoding \( \hat{\nu}_{\mathcal{M}} \). Let \( \chi' \) be the homomorphism over \( \{0,1\}^* \) defined as \( \chi'(0) = 00 \) and \( \chi'(1) = 1 \). Let \( S_{\mathcal{M}} \) be the SCTS with \( 4n \) appendants: \( \beta^{2i} = \epsilon \) and \( \beta^{2i+1} = \chi'(\hat{\nu}^i) \) for \( 0 \leq i < 2n \). An immediate recurrence shows that \( S_{\mathcal{M}} \) simulates \( \hat{\mathcal{C}}_{\mathcal{M}} \), precisely: if \( \nu^t \) and \( \hat{\nu}^t \) denote respectively the datawords of \( \hat{\mathcal{C}}_{\mathcal{M}} \) and \( S_{\mathcal{M}} \) with initial datawords \( v^0 \in \{0,1\}^* \) and \( \hat{\nu}^0 = \chi'(\nu^0) \) then, for all time \( t \), \( \nu^{t+r_1} = \hat{\nu}^t \) where \( r_1 \) is the number of \( 0 \)s read by \( \hat{\mathcal{C}}_{\mathcal{M}} \) up to time \( t \) (note that \( r_1 \leq t \)). Let \( u_{\mathcal{M}}(x) = \chi'(\hat{\nu}_{\mathcal{M}}(x)) \) denote the linear time encoding of the input \( x \) of \( \mathcal{M} \) as the initial dataword of \( S_{\mathcal{M}} \). It follows that (1) \( S_{\mathcal{M}} \) halts from input dataword \( u_{\mathcal{M}}(x) \) iff \( \mathcal{M} \) halts from input \( x \); and (2) if \( \mathcal{M} \) halts from input \( x \) after \( t \) steps, then \( S_{\mathcal{M}} \) halts from \( u_{\mathcal{M}}(x) \) with an empty dataword.
after $O(t^2 \log t)$ steps. Note that moreover, the number of appendants of $S_M$ is a multiple of 4.

\section*{B Proof of main Theorem 1 as a consequence of key Theorem 6}

The remaining of this article is dedicated to prove the following theorem which implies Theorem 1 by the proposition above (see appendix on the current page).

\textbf{Theorem 6 (Key theorem).} There is a fixed set $B$ of 542 bead types and a fixed attraction rule $\heartsuit$ on $B$ together with two polynomial-time encodings:

\begin{enumerate}
  \item $\pi$ that maps any SCTS $S$ with $n \geq 8$ appendants $\alpha = (\alpha^0, \ldots, \alpha^{n-1})$ where $n$ is a multiple of 4, to a bead-type sequence $\pi_S \in B^*$ of exact length:
    \[ |\pi_S| = 18Kn(Kn + 12n - 8) + 3n(192n - 171) + 30\sum_{i=0}^{n-1} |\alpha^i| = O(|\alpha|^4) \]
    where $K = L + 12 - (L \mod 2) \leq L + 12$ with $L = \max_i |\alpha^i|$ being the length of the longest appendant, and $|\alpha| = \sum_{i=0}^{n-1} |\alpha^i|$. Note that $\pi_S$ only depends on the appendants of $S$.
  \item $(s_S, \sigma_S)$ that maps any input dataword $u$ of $S$ to a seed configuration $\sigma_S(u)$ of a bead type sequence $s_S(u)$ of length $O_S(|u|)$, precisely:
    \[ |\sigma_S(u)| = 2|u|(3K + 16) + |u_0| + 9K(n - 1) + 36n - 21 = O_S(|u|) \]
    where $|u_0| = \#\{i : u_i = 0\}$ is the number of 0s in $u$,
\end{enumerate}

such that: For any SCTS $S$ with $n \geq 8$ appendants, where $n$ is a multiple of 4, and every input dataword $u$ of $S$, the deterministic and periodic oritatami system $O_S = ((\pi_S)\infty, \heartsuit, 3)$ with bead type sequence $\pi_S(\infty)$ and delay $\delta = 3$, halts when folding from seed configuration $\sigma_S(u)$ if and only if $S$ halts on input dataword $u$. Furthermore, for all $t$, if $S$ halts on $u$ after $t$ steps, then the folding $O_S$ from seed configuration $\sigma_S(s)$ halts after folding $O_S(t^2)$ beads.

Note that requiring that $n \geq 8$ and $n$ being a multiple of 4 does not restrict this result.
Indeed, repeating the appendants sequence $k$ times in a SCTS, yields a strictly identical SCTS with $k$ times the number of appendants. These requirements are however necessary to ensure the proper folding alignment in the design of our oritatami system.

\textbf{Proof of Theorem 1.} Consider a universal Turing machine $M$ and the skipping cyclic tag system $S_M$ provided by Proposition 2 together with its linear-time input encoder $u_M$. Consider the set of 542 bead types $B$, the rule $\heartsuit$, the oritatami system $O_M = ((\pi_M)\infty, \heartsuit, 3)$ whose primary structure has period $\pi_M = \pi_{S_M}$, and the linear-time seed encodings $(s_{S_M}, \sigma_{S_M})$ provided by Theorem 6 when applied to $S_M$. Let us define for short $s_M(x) = s_{S_M}(u_M(x))$ and $\sigma_M(x) = \sigma_{S_M}(u_M(x))$, the seed bead types sequence and the seed conformation of $O_M$ corresponding to the input $x$ of $M$. Then, by construction:

1. For all input $x$ of $M$, $M$ halts on input $x$, if and only if $S_M$ halts on input dataword $u_M(x)$, if and only if $O_M$ halts its folding from seed conformation $\sigma_M(x) = \sigma_{S_M}(u_M(x))$.
2. For all input $x$ of $M$ and all time $t$, if $M$ halts on $x$ after $t$ steps, then $S_M$ halts on input dataword $u_M(x)$ after $T = O_M(t^2 \log t)$ steps, and thus $O_M$ halts its folding from seed conformation $\sigma_M(x)$ after $O_{S_M}(T^2) = O_M(t^4 \log^2 t)$ steps.
3. For all input $x$ of $M$, the length of the seed conformation encoding $x$ in $O_M$ is $O_{S_M}(|u_M(x)|) = O_M(|x|)$, linear in $|x|$.
4. Finally, the oritatami system $O_M$ and seed encoding $(s_M, \sigma_M)$ are obtained in polynomial time from the skipping tag system $S_M$, which is also obtained in polynomial time from $M$. The reduction is thus computed in polynomial time.
C Folding paths of all the bricks of our design

This section presents all the bricks, i.e. all the folding paths of the 7 modules (i.e. subsequences of the transcript) composing each unit of the transcript. The folding of each module into one of these bricks depending on the context, is the key to the correctness of the folding of our transcript design into the shape of the trimmed diagram the simulated STCS as stated in Lemma 3. This section just presents the shape of each brick for each module together with an illustration to scale. Its purpose is to provide a guideline to the description of the blocks in the next section. The full description of each brick will be given in section F.

To ease the reading, the brick name contains a specific symbol indicating to which kind of phase of the folding the brick belongs:

- Zig-up bricks are annotated by ▶
- Zig-down bricks are annotated by ▼
- Appending bricks are annotated by ◐
- Carriage-return bricks (spanning from a zig row to the next zag row) are annotated by ◐ ◐
- Zag bricks are annotated by ◐
- Line-feed bricks (spanning from a zag row to the next zig row) are annotated by ◐ ◐
- Halting bricks are annotated by ■

The subsequences corresponding to each of the 7 modules are referred by \( A, B, C, D, E, F, G \). Some modules (\( D \) and \( E \)) have parameters that will be explained later, in the next sections. Their bricks, i.e. their possible folding depending on the context are referred by:

Module \( A \) (see section C.1): \( A \) (zig-up), \( A \) (zig-down), \( A \) (zag).
Module \( B \) (see section C.2): \( B \) (zig-up), \( B \) (zig-down), \( B \) (zag), \( B \) (halt).
Module \( C \) (see section C.3): \( C \) (zig-up), \( C \) (zig-down), \( C \) (zag), \( C \) (zag).
Module \( D(x)_{r,t} \) (see section C.4): \( D(x)_{r,t} \) (zig-up), \( D(x)_{r,t} \) (zig-down), \( D(x)_{r,t} \) (zag), \( D(x)_{r,t} \) (append).
Module \( E_{a} \) (see section C.5): \( E_{a} \) (zig-up), \( E_{a} \) (zig-down), \( E_{a} \) (zag), \( E_{a} \) (carriage return).
Module \( F \) (see section C.6): \( F \) (zig-up), \( F \) (zig-down), \( F \) (zag).
Module \( G \) (see section C.7): \( G \) (read 0), \( G \) (read 1), \( G \) (read 1), \( G \) (read 0), \( G \) (zig-down), \( G \) (zig-down), \( G \) (zig-down), \( G \) (zag), \( G \) (zag), \( G \) (zag), \( G \) (line feed).

In the following figures, a lighter and a darker grey arrow indicates the beginning and the end of the folding path of each brick respectively. The parameter \( h \) will be defined later and refers to the height of the blocks composing the folded shape of our design.

C.1 All bricks for Module A

Module \( A \) always folds as a glider of height \( h \) and width 3, pointing to NE in zig-up phase, SE in zig-down phase and SW in zag phase. The folding of module \( A \) serves as a scaffold for the folding of the next modules in the zig-up and zig-down phases.

All its possible bricks are displayed in Fig. 6 on the facing page.
(a) The brick $A\triangleright$.  

(b) The brick $A\frown = \text{HorizontalMirror}(A\triangleright)$.  

(c) The brick $A\frown = \text{Rotate}_{180}(A\triangleright)$.  

Figure 6 Folding paths to scale of all the bricks for Module $A$ (see section F.4 for full description).
C.2 All bricks for Module B

Module $B$ is 5 beads long. It folds along the preceding brick of Module $A$:

- to the right in zig-down;
- to the left in zag phase;
- to right in zig-up phase if the dataword encoded in the zag-row above is not empty;
- but to the left in the zig-up phase if the dataword encoded above is empty (terminating the folding as it is now trapped in a closed area).

All its possible bricks are displayed in Fig. 7. Note that $B ▶ = \text{HorizontalMirror}(B ▶)$ and $B ▼ = \text{Rotate}_{180°}(B ▶)$.

![Folding paths to scale of all the bricks for Module $B$](image-url)

- (a) The brick $B ▶$.
- (b) The brick $B ▼$.
- (c) The brick $B ▼$.
- (d) The brick $B ▼$.

**Figure 7** Folding paths to scale of all the bricks for Module $B$ (see section F.5 for full description).
C.3 All bricks for Module C

Module $C$ folds in switchbacks of height almost $h$, along the brick of the preceding module $A$:

- in 3 switchbacks in the zag phases;
- in 3 switchbacks in the zig-up or zig-down phases if the current folding did not reach the end of the dataword encoded in the zag-row above yet;
- but in 2 switchbacks ($C\cdot\text{End}$) if this end is reached, creating the initial condition for folding the next modules as the encoding of the letters of the appendant to be appended at this stage of the simulation.

All its possible bricks are displayed in Fig. 8 on the following page.
(a) The brick $\text{C} \triangleright$

(b) The brick $\text{C} \triangleright = \text{HorizontalMirror}(\text{C} \triangleright)$.

(c) The brick $\text{C} \triangleright = \text{Rotate}_{180^\circ}(\text{C} \triangleright)$.

(d) The brick $\text{C} \triangleright \text{End}$.

**Figure 8** Folding paths to scale of all the bricks for Module $\text{C}$ (see section F.6 for full description).
C.4 All bricks for Module D

Module $D(x)_{r,t}$ is used to encode each letter of the appendant stored in each block unit. Its parameters $x, r, t$ stands for the letter $x \in \{0, 1\}$ and the position index of that letter in the encoded appendant ($r$ says if it is either at the first, odd or even, and $t$ if it is the last or not). All these variants of module $D$ fold slightly differently. Module $D(x)_{r,t}$ folds either:

- in 6 switchbacks of height approximatively $h/2$, along the preceding brick of Module $C$ in the zig-up, zig-down and zag phases;
- or as a glider in the append phase where it is forced by the preceding brick of module $B$ to adopt this shape.

All its possible bricks are displayed in Fig. 9 on the next page.

Note that: $D\rightarrow = \text{HorizontalMirror}(D\leftarrow)$ and $D\leftarrow = \text{Rotate}_{180}(D\rightarrow)$. 
(a) The brick \( D_k \)  
(b) The brick \( D_k \)  
(c) The brick \( D_k \)  
(d) The bricks \( D_{0,r,t} \) and \( D_{1,r,t} \).

\textbf{Figure 9} Folding paths to scale of all the bricks for Module \( D \) (see section F.7 for full description).
C.5 All bricks for Module E

Module $E_a$ is used for padding and carriage return. Its parameter $a$ stands for the number of letter of the appendant it needs to pad so as the block units of all appendants have the same dimensions. The length of module $E_a$ decreases with $a$ accordingly. Module $E_a$ folds either:

- in many short switchbacks of height approximatively $h/2$ followed by a glider and 5 large switchbacks of height $h$ in zig-up, zig-down and zag phases. The many short switchbacks are used to pad the appendants to a fixed length in the block units.
- or as a glider in append phase, achieving the carriage return from zig to zag phase. The glider folding is triggered as for module $D$, either by the preceding brick of module $B$ or the preceding glider append brick of a module $D$.

All its possible bricks are displayed in Fig. 10 on the following page.

Note that: $E_a^\rightarrow = \text{HorizontalMirror}(E_a^\rightarrow)$ and $E_a^\leftarrow = \text{Rotate}_{180^\circ}(E_a^\rightarrow)$. 

Figure 10 Folding paths to scale of all the bricks for Module $E_a$ (see section F.8 for full description).
C.6 All bricks for Module F

Module $F$ always folds as a glider of height $h$ and width 4, pointing to NE in zig-up phase, SE in zig-down phase and SW in zag phase. As for module $A$ in the zig phases, the folding of module $F$ serves as a scaffold for the folding of the next modules in the zag phases.

All its possible bricks are displayed in Fig. 11.

(a) The brick $F\rightharpoonup$.

(b) The brick $F\rightharpoonup = \text{HorizontalMirror}(F\rightharpoonup)$.

(c) The brick $F\leftarrow = \text{Rotate}_{180^\circ}(F\rightharpoonup)$.

Figure 11 Folding paths to scale of all the bricks for Module $F$ (see section F.9 for full description).
C.7 All bricks for Module G

Module $\text{G}$ is the most sophisticated module as it can fold, depending of the context, into very different shapes:

**Read bricks (Fig. 12):** In zig-up phase, module $\text{G}$ folds as a glider of total length approximately $2h$, heading first to NE and then bouncing to SE ($\text{G} \rightarrow \text{Read0}$) or E ($\text{G} \rightarrow \text{Read1}$) depending on whether it hits the encoding of letter 0 or 1 in the zag row above respectively.

**Copy bricks (Fig. 13):** In zig-down and zag phases, module $\text{G}$ folds into 6 switchbacks of height $h$, copying the letter, 0 or 1, encoded at the bottom of the row above, to the top of the row below.

**Line feed brick (Fig. 14):** At the end of the zag row, module $\text{G}$ folds into a glider of length $2h$ heading SW opening the next zig row.

The design of module $\text{G}$ requires a lot of care and was made possible thanks to the advanced design tools we developed for that purpose.

All its possible bricks are displayed in Fig. 12, 13 and 14.
(a) The brick $G_{\text{Read0}}$.

(b) The brick $G_{\text{Read1}}$.

Figure 12: Folding paths to scale of all the Read bricks for Module $G$ (see section F.10 for full description).
Figure 13 Folding paths to scale of all the Copy bricks for Module $G$ (see section F.10 for full description).
Figure 14 Folding paths to scale of all the Line Feed brick for Module G (see section F.10 for full description).
D Inside of the blocks

This section presents the exact content of each block, i.e. the bricks they are composed of.

Notation. We describe conformations using the following notations: given a conformation \( \sigma \) and a bead type \( b \), we denote by \( \sigma \rightarrow_e b \), \( \sigma \nearrow b \), \( \sigma \swarrow b \), \( \sigma \nwarrow b \), and \( \sigma \nearrow b \) the elongations of the conformation \( \sigma \) by one bead of bead type \( b \) located respectively to the east, south-east, south-west, west, north-west, and north-east of the last bead of \( \sigma \). We refer by \( b \) the conformation that consists of a single bead of bead type \( b \) located at \((0,0)\). Fig. 15 illustrates this notation.

![Figure 15 The conformation encoded by \( A_0 \rightarrow_e A_1 \nearrow A_2 \swarrow A_3 \nwarrow A_4 \rightarrow_e A_5 \nearrow A_6 \nearrow A_7 \swarrow A_8 \nwarrow A_9 \).](image)

We extend naturally this notation to two conformations: for instance, \( \sigma_1 \rightarrow_e \sigma_2 \) is the conformation beginning with conformation \( \sigma_1 \) followed by conformation \( \sigma_2 \) whose origin has been translated to the vertex at the east of the last bead of \( \sigma_1 \).

We denote by \( \text{HorizontalMirror}(\sigma) \), \( \text{VerticalMirror}(\sigma) \) and \( \text{Rotate}_{180^\circ}(\sigma) \) the conformations obtained by respectively mirroring horizontally, vertically and rotating by \( 180^\circ \) the conformation \( \sigma \).

Tables 1 and 2 on pages 37 and 38, present an exploded view of the bricks inside each zig- and zag-block respectively.

The folded paths of the bricks composing the blocks in the definitions bellow have been presented in Section C. Their full description will be given in section F.

These blocks are composed of one or \( n \) block units where \( n \) is the number of appendants in the simulated STCS. Each block unit encodes an appendant \( \alpha \) inside and consists of a sequence of \( 6 + |\alpha| \) bricks: one of each module \( A \), module \( B \), and module \( C \), then \( |\alpha| \) of module \( D \), then one of each module \( E \), module \( F \) and module \( G \). Read and Append blocks are composed of one block unit, whereas Copy blocks are composed of \( n \) block units.

We annotate each block by \([i]\) where \( i \) is the index of the appendant \( \alpha \) it contains. A block composed of \( n \) block units is annotated by the index of its leading unit (i.e., the leftmost block unit of zig-block and the rightmost block unit of a zag-block). These indices are computed modulo \( n \), thus in the following \([i]\) refers to \([i \mod n]\).

As for the bricks before, specific symbols indicate to which kind of phase the blocks belong:

- Zig-up blocks are annotated by \( ◁\)
- Zig-down blocks are annotated by \( ◁\)
- Appending blocks are annotated by \( ◁\)
- Carriage-return blocks (spanning from a zig row to the next zag row) are annotated by \( ◁\)
- Zag blocks are annotated by \( ◁\)
- Line-feed blocks (spanning from a zag row to the next zig row) are annotated by \( ◁\)
- Halting blocks are annotated by \( ■\)

This section presents the exact composition in terms of bricks of each block. It is more like a reference section. The next section presents with illustration the exact geometry of
each block. We encourage the reader to use the figures of the next section to picture each block when reading the block definition. We will refer to the appropriate figure after each block definition.

### D.1 The bricks encoding the appendants

Let us first describe how the appendants are encoded by $D$- and $E$-bricks inside the blocks. Each letter of an appendant $\alpha^i$ is encoded by a module $D_i$. There are 8 variants $D(x)_{r,t}$ of modules $D_i$ depending on:

- the letter $x \in \{0, 1\}$
- the rank $r \in \{0, 1, 2\}$ of the letter in $\alpha^i$: $r = 0$ for the first letter, $r = 1$ if the index of letter is odd, and $r = 2$ if the index of the letter is positive and even.
- $t \in \{0, 1\}$ where $t = 1$ iff it is the last letter of $\alpha^i$.

Each of these variant folds into a slightly different brick. This is why we must take the index of the letter into account in the definitions below.

\[
\text{Appendant}(\alpha) = \begin{cases} 
E_i & \text{for letter } x \in \{0, 1\} \\
D(x)_{r,t} \rightarrow E_{L-1} & \text{for all } x \in \{0, 1\}^* \text{ with } |v| \geq 2
\end{cases}
\]

\[
\begin{align*}
\text{Appendant}(\alpha) & = \text{HorizontalMirror}(\text{Appendant}(\alpha)) \quad \text{for all } v \in \{0, 1\}^* \\
\text{Appendant}(\alpha) & = \text{Rotate}_{180^\circ}(\text{Appendant}(\alpha)) \quad \text{for all } v \in \{0, 1\}^*
\end{align*}
\]

### D.2 The Zig-Blocks

Let us now describe the bricks inside each of the blocks present on a zig row, that is to say: Read$\triangleright$ and Copy$\triangleright$ blocks.

\[
\text{Read0}[i] = \begin{cases} 
A & \text{for all } i \geq 0 \\
B & \text{for all } i \geq 0 \\
C & \text{for all } i \geq 0 \\
\text{Appendant}(\alpha) & \text{for all } i \geq 0 \\
F & \text{for all } i \geq 0 \\
G & \text{Read0}
\end{cases}
\]

(See fig. 16(a) p. 40)

\[
\text{Read1}[i] = \begin{cases} 
A & \text{for all } i \geq 0 \\
B & \text{for all } i \geq 0 \\
C & \text{for all } i \geq 0 \\
\text{Appendant}(\alpha) & \text{for all } i \geq 0 \\
F & \text{for all } i \geq 0 \\
G & \text{Read1}
\end{cases}
\]

(See fig. 16(b) p. 40)

\[
\begin{align*}
\text{Copy0}[i] & = \begin{cases} 
A & \text{for all } i \geq 0 \\
B & \text{for all } i \geq 0 \\
C & \text{for all } i \geq 0 \\
\text{Appendant}(\alpha) & \text{for all } i \geq 0 \\
F & \text{for all } i \geq 0 \\
G & \text{Copy0}
\end{cases} \\
\text{Copy1}[i] & = \begin{cases} 
A & \text{for all } i \geq 0 \\
B & \text{for all } i \geq 0 \\
C & \text{for all } i \geq 0 \\
\text{Appendant}(\alpha) & \text{for all } i \geq 0 \\
F & \text{for all } i \geq 0 \\
G & \text{Copy1}
\end{cases}
\end{align*}
\]

(See fig. 17(a) p. 41)

\[
\begin{align*}
\text{Copy0}[i] & = \text{Copy0}[i] \text{Unit} \bigoplus_{j=1}^{n-1} \left( \text{Copy1}[i+j] \text{Unit} \right) \\
\text{Copy1}[i] & = \bigoplus_{j=0}^{n-1} \left( \text{Copy1}[i+j] \text{Unit} \right)
\end{align*}
\]

(See fig. 17(b) p. 41)

Note that the Copy$\triangleright$ blocks are composed of $n$ copy block units with consecutive appendant indices modulo $n$. They are indexed by the index of their leading copy block unit. Recall that the index $[i+j]$ is computed modulo $n$ and refers thus to $[(i+j) \mod n]$. 

D.3 Append and Carriage return blocks

Let us define the following convenience block:

\[
\text{AppendAppendant}(\epsilon) \xrightarrow{\text{E}} \text{E}
\]

\[
\text{AppendAppendant}(x) \xrightarrow{\text{E}} \text{E} \xrightarrow{\text{E}} \text{E}
\]

for letter \( x \in \{0, 1\} \)

\[
\text{AppendAppendant}(\omega) \xrightarrow{\text{E}} \text{E} \xrightarrow{\text{E}} \text{E} \xrightarrow{\text{E}} \text{E}
\]

\[
\text{for } \omega \in \{0, 1\}^* \text{ with } |\omega| \geq 2
\]

Now,

\[
\text{Append}[\omega] \text{Return} = \xrightarrow{\text{A} B \text{C End} \text{SE}} \text{AppendAppendant}(\omega) \xrightarrow{\text{F G LineFeed}} \text{Copy1}
\]

(See fig. 21 p. 46)

D.4 Zag row

Naturally, by symmetry, let:

\[
\text{Copy0}[i] \text{Unit} = \text{VerticalMirror } \text{Copy0}[i] \text{Unit}
\]

\[
\text{Copy1}[i] \text{Unit} = \text{VerticalMirror } \text{Copy1}[i] \text{Unit}
\]

\[
\text{LineFeed}[i] \text{Unit} = \xrightarrow{\text{A} B \text{C End} \text{SE}} \text{AppendAppendant}(\alpha) \xrightarrow{\text{F G LineFeed}} \text{LineFeed}
\]

Then,

\[
\text{Copy0}[i] \xrightarrow{n-3} (\text{Copy1}[i+j] \text{Unit}) \text{Copy0}[i+n-2] \xrightarrow{n-2} \text{Copy1}[i+n-1] \text{Unit}
\]

(See fig. 19(a) p. 43)

\[
\text{Copy1}[i] \xrightarrow{n-1} (\text{Copy1}[i+j] \text{Unit})
\]

(See fig. 19(b) p. 43)

\[
\text{Copy0}[i] \text{LineFeed} \xrightarrow{n-3} (\text{Copy1}[i+j] \text{Unit}) \text{Copy0}[i+n-2] \xrightarrow{n-2} \text{LineFeed}[i+n-1] \text{Unit}
\]

(See fig. 20(a) p. 44)

\[
\text{Copy1}[i] \text{LineFeed} \xrightarrow{n-2} (\text{Copy1}[i+j] \text{Unit}) \xrightarrow{n-2} \text{LineFeed}[i+n-1] \text{Unit}
\]

(See fig. 20(b) p. 45)

D.5 The special case: appending \( \epsilon \) to an empty dataword

\[
\text{CarriageReturn} \text{LineFeed} = \xrightarrow{\text{A} B \text{C End} \text{SE}} \text{AppendAppendant}(\epsilon) \xrightarrow{\text{F G LineFeed}} \text{LineFeed}
\]

\[
\text{CarriageReturn} \text{LineFeed} \text{Halt} = \text{CarriageReturn} \text{LineFeed} \xrightarrow{\text{A B} \text{SE}} \xrightarrow{\text{E}} \xrightarrow{\text{E}} \text{B}
\]

(See fig. 23 p. 48)
Table 1 The bricks inside the zig-blocks.
Table 2: The bricks inside the zag-blocks.
E Geometry of the blocks

We give here the geometrical description of each block. We describe in particular the positions of the important features of each block, such as the positions where a letter is read, copied or written.

Convenience variables. Let \( \mathcal{S} = (\alpha; u^0) \) be the SCTS to be simulated. In order to simplify the description, we introduce the following variables which correspond to key geometrical parameters of the different blocks:

\[
\begin{align*}
L &= \max_{0 \leq i < n} |\alpha_i| \quad \text{the maximum length of an appendant in } \mathcal{S} \\
P &= 12 - (L \mod 2) \quad \text{the padding constant, such that } L + P \text{ is even and at least 12} \\
w &= 6(L + P) + 18 \quad \text{the width of the appendant module excluding its read/copy/linefeed part} \\
W &= n \cdot (w + 6) \quad \text{the width of the Read\(\uparrow\), Copy\(\uparrow\), and Copy\(\downarrow\) blocks} \\
h &= W - (w + 3) \quad \text{the height of the zig and zag rows} \\
c(a) &= (6\lambda(L - a + P) + 8h - 16)/4 \quad \text{the width of the brick } \langle E, \delta \rangle \\

\end{align*}
\]

As \( n \) is a multiple of 4 and as \( L + P \) is even, we have:

\[\textbf{Fact 7.} \text{ All convenience variables } L, P, w, W, h, c(a) \text{ are integers. Furthermore,} \]

\[ w = 6 \mod 12, \quad W = 0 \mod 48, \quad h = 3 \mod 12, \quad c(a) = 2 \mod 12, \text{ for all } 0 \leq a \leq L. \]

These relations ensure for instance that all gliders finish in the correct position and that all patterns are properly aligned.

- Read\(\uparrow\) blocks are described in figure 16
- Copy\(\uparrow\) blocks are described in figure 17
- Seed blocks are described in figure 18
- Copy\(\downarrow\) blocks are described in figure 19
- Copy\(\downarrow\)LineFeed blocks are described in figure 20
- Append\(\downarrow\)Return blocks are described in figure 21
- CarriageReturn\(\downarrow\)LineFeed\(\downarrow\)Halt\(\uparrow\) and Halt blocks are described in figures 23 and 24

Intuitively, each zig block corresponds to one cell of the trimmed diagram of the simulated SCTS, upscaled to a parallelogram of width \( W \) and height \( h \). Zag-blocks are just used to copy these cells while returning at the beginning of the current datawork.

Recall that the coordinates are expressed according to the east and south-west axis: every position \((x, y)\) in \( \mathbb{T} \) is mapped in the euclidean plane to \( x \cdot \vec{E} + y \cdot \vec{SW} \) using the vector basis \( \vec{E} = (1, 0) \) and \( \vec{SW} = \text{RotateClockwise} \left( \vec{E}, 120^\circ \right) = (-\frac{1}{2}, -\frac{\sqrt{3}}{2}). \)
**Figure 16 Geometry of the Read blocks.** Note that the internal structures (the lines in white) of both blocks $\text{Read}_0\uparrow$ and $\text{Read}_1\uparrow$ agree until position $(w+2, 1-h)$ where the presence or absence of a spike, encoding a 0, at the bottom of the row above forces the block to adopt the shape $\text{Read}_0\uparrow$ or $\text{Read}_1\uparrow$ respectively.

(a) The $\text{Read}_0\uparrow$ 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. its origin at the bottom left corner), in which plugs the spike of the block from the row above it, encoding the letter 0. The next block will start folding at the bottom right corner, at $(W, 0)$.

(b) The $\text{Read}_1\uparrow$ block has the shape of a parallelogram with horizontal side length $W$ and vertical side length $h$. The red rectangle area at position $(w+2, 1-h)$ (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 0). The next block will start folding at the top right corner, at $(W-1, 1-h)$. 
Figure 17 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 $(W - 1, 1 - h)$. Note that the Copy0 and Copy1 blocks have identical internal structure apart from the line joining the two purple areas at $(w + 2, 1 - h)$ and $(h + w + 1, 1)$. Indeed, when folding, the part of the transcript 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). Furthermore, this block is made of $n$ CopyUnit blocks (from left to right), each of width $w + 6$ and height $h$.

(a) The Copy0 block has a dent (an empty position) located at $(w + 2, 1 - h)$, in which plugs the spike of the block from the row above it, and which induces (when folding) a spike at the bottom at $(h + w + 1, 1)$, copying the letter 0 from the top to the bottom of the Zig-row. Note that this block is made of $n$ CopyUnit blocks (from left to right): one CopyUnit followed by $n - 1$ CopyUnit, each of width $w + 6$ and height $h$.

(b) The Copy1 block is flat at $(w + 2, 1 - h)$, which, aligned with a flat block above (encoding a 1), induces (when folding) a flat bottom at $(h + w, 0)$, copying the letter 1 from the top to the bottom of the Zig-row. Note that this block is made of $n$ CopyUnit blocks (from left to right), each of width $w + 6$ and height $h$. 
Figure 18 Geometry of the Seed block. This block encodes the initial word so that the oritatami 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 \((-1, 0)\) where the transcript will start folding the first zig-row.
Figure 19 Geometry of the Copy\(\downarrow\) blocks. The Copy\(0\downarrow\) and Copy\(1\downarrow\) blocks are the horizontal mirror images of the Copy\(0\uparrow\) and Copy\(1\uparrow\) blocks (see Figure 17).

(a) The Copy\(0\downarrow\) block is the horizontal mirror image of the Copy\(0\uparrow\) block (see Figure 17(a)).

(b) The Copy\(1\downarrow\) block is the horizontal mirror image of the Copy\(1\uparrow\) block (see Figure 17(b)).
Figure 20 Geometry of the Copy\textsuperscript{\textregistered}LineFeed 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 Copy\textsuperscript{\textregistered}LineFeed blocks are identical to the Copy\textsuperscript{\textregistered} blocks until position \((h - W - 2, 0)\) 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 horizontal offset of 7 beads between the zig- and zag-rows). Then, instead of completing a parallelogram, the end of the Copy\textsuperscript{\textregistered}LineFeed 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 \((h - W, 2h)\). Furthermore, this block is made (from right to left) of \(n - 2\) Copy\textsuperscript{1}\textsuperscript{\textregistered}Unit blocks followed by a Copy\textsuperscript{(x)}\textsuperscript{\textregistered}Unit and a LineFeed\textsuperscript{\textregistered}Unit, where \(x\) is the letter copied.

(a) The Copy\textsuperscript{\textregistered}LineFeed block proceeds as Copy\textsuperscript{\textregistered} to copy the spike encoding a 0 from the row above to the row below. It has a dent (an empty position) at \((h - W + w + 1, 1)\) in which plugs the spike (encoding a 0) of the block above. When folding, this dent induces a spike at the bottom at position \((h - W + w + 2, 1 + h)\). Note that the spike below is at position \((h - W + w + 2, 1 + h)\), which is consistent with the position of the dent in the Read\textsuperscript{\textregistered} block that will fold from \((h - W, 2h)\) (see Figure 16(a)).
Figure 20 Geometry of the CopyLineFeed blocks. (Continued)

(a) \( (h-8,1) \)

(b) \( (h-W,2) \)

(c) \( (h-W+w+1,1) \)

(d) \( (h-W+w+2,1) \)

(e) \( (h-W-2,1) \)

(f) \( (h-W-6,-2) \)

(g) \( (h-W-4,1) \)

(h) \( (h-W+h,2h) \)

(i) \( (h-W-4,1) \)

(j) \( (h-W-2,1) \)

(k) \( (h-W,-2) \)

(l) \( (h-W,2h) \)

(m) \( (h-W+h,2h) \)

(n) \( (h-W-4,1) \)

(o) \( (h-W-2,1) \)

(p) \( (h-W,-2) \)

(q) \( (h-W+2,1) \)

(r) \( (h-W+w+2,1) \)

(s) \( (h-W+w+1,1) \)

(t) \( (h-W,1) \)

(u) \( (h-W-1,1) \)

(v) \( (h-W-2,1) \)

(w) \( (h-W-3,1) \)

(x) \( (h-W-4,1) \)

(y) \( (h-W-5,1) \)

(z) \( (h-W-6,1) \)

(b) The CopyLineFeed block.
Figure 21 Geometry of the Append(\(u\))Return 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 high \(|u|W\)-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 \(u_j = 0\) at position \((jW + w + h + 1, 1)\) (1s are encoded by the absence of spike). It then expands upto position \((|u|W + h + c(|u|), 0)\) 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 \((|u|W + h - 9, 2)\), at the top right corner of the upcoming zig-row. The next block will start at \((|u|W + h - 8, 1)\).
Figure 22 Geometry of the Append block. This block is the special case of Figure 21 where \( u = \epsilon \). It is given for clarity.
Figure 23 Geometry of the CarriageReturn-LineFeed-Halt block. This block is identical to the Append(\(\varepsilon\))\(\rightarrow\)Return block until it reaches position \((h - 1, 1)\). 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 Copy\(\rightarrow\)LineFeed blocks (see Figure 20) to open a new zig-row at \((h - 1, 2h)\). It then goes up to \((h, h - 1)\). And as there are no block on the zag-row above, it is attracted inside itself and gets blocked at \((h - 1, 2h - 3)\).
Figure 24 Geometry of the Halt block. This block appears at the end of the computation. It starts as a Read block with a 3-beads wide $h$-beads high southeastbound glider until it reaches position $(2, 1-h)$. But, as there are no block in the zag-row above, the next beads are attracted to the left and the construction stops there.
Full description of the SCTS oritatami simulator

Consider a SCTS $S$ with $n$ appendants $\alpha_0, \ldots, \alpha_{n-1} \in \{0, 1\}^*$, where $n$ is at least 8 and a multiple of 4, together with an input dataword $u \in \{0, 1\}^*$.

We give here the full description of the primary structure $\pi_S$ of the oritatami systems $\mathcal{O}_S = ((\pi_S)^\infty, \heartsuit, 3)$ together with its seed conformation $\sigma_S(u)$, which simulates step by step the computation of $S$ on input dataword $u$.

Convenience variables. In order to simplify the description, we introduce the following variables which correspond to key geometrical parameters of the different modules: (note that some of the variables were already introduced in Section E)

\[
L = \max_{0 \leq i < n} |\alpha_i| \quad \text{the maximum length of an appendant in } S
\]

\[
P = 12 - (L \mod 2) \quad \text{the padding constant, such that } L + P \text{ is even and at least 12}
\]

\[
w = 6(L + P) + 18 \quad \text{the width of the appendant module excluding its read/copy/linefeed part}
\]

\[
W = n \cdot (w + 6) \quad \text{the width of the Read\textsuperscript{\triangleright}, Copy\textsuperscript{\triangleright}, and Copy\textsuperscript{\triangleright} blocks}
\]

\[
h = W - (w + 3) \quad \text{the height of the zig and zag rows}
\]

\[
k = (h - 3)/6 \quad \text{the number of periods of a glider of length } h
\]

\[
\lambda = W/2 \quad \text{the height } (\lambda + 5) \text{ of the letter modules inside the Read\textsuperscript{\triangleright}, Copy\textsuperscript{\triangleright}, \text{ and Copy\textsuperscript{\triangleleft} blocks}
\]

\[
\kappa = W/24 \quad \text{the number of periods of the glider/switchback pattern in the letter and padding modules}
\]

\[
q = (h - 3)/4 \quad \text{the number of periods of the glider in the backbone of module [E]}
\]

\[
c(a) = (6\lambda(L - a + P) + 8h - 16)/4 \quad \text{the width of the brick [E, [3]}
\]

As $n$ is a multiple of 4 and as $L + P$ is even, we have:

\begin{itemize}
  \item [\textbf{Fact 8.}] All convenience variables $L, P, w, W, h, k, \lambda, \kappa, q, c(a)$ are integers. Furthermore,
  \[
  w = 6 \mod 12, \quad W = 0 \mod 48, \quad h = 3 \mod 12, \quad k = 0 \mod 2,
  \lambda = 0 \mod 24, \quad \kappa = 0 \mod 2, \quad q = 0 \mod 3, \quad c(a) = 2 \mod 12, \text{ for all } 0 \leq a \leq L.
  \]
\end{itemize}

Notations for describing of the bead type sequences. If $u$ and $v$ are two finite bead type sequences, we write their concatenation as $u \cdot v$. For any two integers $0 \leq i \leq j < |u|$, we write $u_{i..j}$ for $u_iu_{i+1}\ldots u_j$. The reverse sequence of $u$, written as $u^R$, is $u_{|u|-1}u_{|u|-2}\ldots u_0$.

Finally, given a sequence $u$, we write $u\langle a_i@i_1, \ldots, a_k@i_k\rangle$ for the sequence $w$ where the bead type indexed by $i_j$ in $u$ has been replaced by $a_j$ for $j = 1, \ldots, k$:

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

By extension, we write $u\langle v@k..l\rangle$ for the sequence $w$ where for all $i \in \{k, k+1, \ldots, l\}$, 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 $\bigcup_{i \geq 1} u_i$ the infinite word $u_1u_2\cdots u_i \cdots$ obtained by containing all the words $u_1, u_2, \ldots$
Notation for describing conformations. Given an infinite sequence of directions \( \left( d_i \right) \in \left\{ \text{NE}, \text{E}, \text{S}, \text{SW}, \text{W}, \text{NW} \right\}^\infty \), and a finite bead type sequence \( b \in B^* \), we denote by \( \text{Conformation}(b, d) \) the configuration \( b_0d_0b_1d_1 \cdots d_{|b|-2}b_{|b|-1} \) that maps \( b \) along the path \( d \). We will use the following conventions functions which will ease the description of the bricks:

- \( \text{E-path}(b) = \text{Conformation}(b, \left( \frac{\pi}{4} \right)^\infty) \) and similarly, \( \text{SE-path}, \text{SW-path}, \text{W-path}, \text{NW-path} \) and \( \text{NE-path} \) that map a bead type sequence along the paths \( \left( \frac{\pi}{8} \right)^\infty, \left( \frac{3\pi}{8} \right)^\infty, \left( \frac{5\pi}{8} \right)^\infty, \left( \frac{7\pi}{8} \right)^\infty \), and \( \left( \frac{\pi}{4} \right)^\infty \) respectively.

- \( \text{E-glider}(b) = \text{Conformation}(b, \left( \frac{\pi}{8} \right)^\infty) \) and \( \text{E-glider}'(b) = \text{Conformation}(b, \left( \frac{3\pi}{8} \right)^\infty) \)

- \( \text{SE-rev-gliding}(b) = \text{Conformation}(b, \left( \frac{\pi}{8} \right)^\infty) \) for each appendant:

Recall that the coordinates are expressed according to the east and south-west axis: every position \( (x, y) \) in \( \mathbb{T} \) is mapped in the Euclidean plane to \( x \cdot \vec{E} + y \cdot \vec{SW} \) using the vector basis \( \vec{E} = (1, 0) \) and \( \vec{SW} = \text{RotateClockwise}(\vec{E}, 120^\circ) = (-\frac{1}{2}, -\frac{\sqrt{3}}{2}) \).

F.1 The periodic primary structure \((\pi_\mathbb{S})^\infty\)

The period of the primary structure consists in the concatenation of one appendant bead type sequence for each appendant:

\[
\pi_\mathbb{S} = \text{Appendant } \alpha^0 \cdot \text{Appendant } \alpha^1 \cdot \cdots \cdot \text{Appendant } \alpha^{n-1}
\]

F.2 The appendant bead type sequences

Each appendant bead type sequence \( \text{Appendant } \alpha \) has the exact same structure: it is the concatenation of 6 sequences: Modules \( A, B, \) and \( C \), followed by a sequence \( \text{word}(\alpha) \) encoding the appendant \( \alpha \) itself, followed by modules \( F \) and \( G \):

\[
\text{Appendant } \alpha^i = A \cdot B \cdot C \cdot \text{word}(\alpha) \cdot F \cdot G
\]

For each word \( v \in \{0, 1\}^* \) with \( |v| \leq L \), \( \text{word}(v) \) encodes each letter of \( v \) using one of the 6 variants of module \( D \) and terminates with a padding sequence \( E_{L-|v|} \) which ensures that the folded size of \( \text{word}(v) \) is independent of the length of \( v \). The 6 variants of the module \( D \) are \( D(x)_{x+1} \), where:

- \( x \in \{0, 1\} \) is the encoded letter;
- \( r \in \{0, 1, 2\} \) is the rank of the letter inside the encoded word \( v \): \( r = 0 \) if it is the first letter of \( v \); \( r = 1 \) if its index in \( v \) is odd; and \( r = 2 \) if its index in \( v \) is even but not 0;
- \( t \in \{0, 1\} \) is 1 if the encoded letter is the last letter of \( v \), and 0 otherwise.

The definition of the sequence \( \text{word}(v) \) follows as:

\[
\text{word}(\epsilon) = E_{L-1}
\]

\[
\text{word}(0) = D_{0,1} \cdot E_{L-1}
\]

\[
\text{word}(1) = D_{1,1} \cdot E_{L-1}
\]

and for \( |v| \geq 2 \),

\[
\text{word}(v) = D_{0,0} \cdot \left( \left( \begin{array}{c} |v| - 2 \\ i = 1 \end{array} \right) \cdot D_{v_1 - 2 \cdot i (|v| - 1) \text{mod } 2} \right) \cdot E_{L-1}
\]
The next section concludes the full description of the primary structure by giving the sequences for modules \( A, B, C, D(x)_P, E, F, \) and \( G. \)

### F.3 Modules bead type sequences and brick conformations

The modules are given using 546 bead types. However, 5 of them, \( A_6, C_{16}, J_9, L_{86}, \) and \( L_{89}, \) are neutral, i.e. do not have any interaction with any other bead types (see Section H) and can thus be all substituted by one single neutral bead type \( N_0. \) The total number of distinct bead types is thus 542 (541 + 1 neutral bead type \( N_0). \) However, in the description below, we prefer to use \( A_6, C_{16}, J_9, L_{86}, \) and \( L_{89}, \) (and not \( N_0), \) as it keeps the bead types homogenous and continuously numbered in each module.

### F.4 Module A

#### F.4.1 Bead type sequence of Module A

Length: \( 3h - 2; \) 13 bead types used: \( A_{0..12} \)

\[
A = A_0 \cdot A_{4} \cdot (A_{5..10})^{3k-1} \cdot A_{5..7} \cdot A_6 \cdot A_{9..10} \cdot A_{11..12}
\]

#### F.4.2 The bricks for module A

Module \( A \) adopts three brick conformations: \( A \uparrow, \ A \downarrow, \) and \( A \leftarrow, \) where the two last ones are just obtained by mirroring and rotating the first:

\[
A \uparrow = A_0 \rightarrow E A_1 \nwarrow A_2 \rightarrow E A_3 \searrow A_4 \nearrow (A_5 \nwarrow A_6 \rightarrow E A_7 \nearrow A_8 \rightarrow E A_9 \searrow A_{10} \nearrow (A_{11} \nwarrow A_{12} \nearrow A_{5} \nwarrow
\]

\[
A \downarrow = \text{HorizontalMirror}(A \uparrow)
\]

\[
A \leftarrow = \text{Rotate}_{180}(A \uparrow)
\]

Figure 25 displays brick \( A \uparrow. \)
F.4.3 Subrule for Module A

Module A interacts with A, B, C, D, F and G. Figure 26 presents the subrule for the interactions between the beads of A and the beads of the other modules.

F.5 Module B

F.5.1 Bead type sequence for Module B

Length: 5; 5 bead types used: B0..4

\[ B = B0..4 \]

F.5.2 The bricks for module B

Module B adopts four brick conformations plus some incomplete ones if the folding stops because the dataword in the simulated SCTS is empty.

\[ B \rightharpoonup = B0 \rightharpoonup B1 \rightharpoonup B2 \rightharpoonup B3 \rightharpoonup B4 \]

\[ B \triangleright = \text{HorizontalMirror}(B \rightharpoonup) \]

\[ B \lefttriangleleft = \text{Rotate}_{180}(B \rightharpoonup) \]

\[ B \triangleleft = B0 \lefttriangleleft B1 \lefttriangleleft B2 \lefttriangleleft B3 \lefttriangleleft B4 \]

Figure 27 displays bricks \( B \rightharpoonup \) and \( B \lefttriangleleft \) (shaded).

F.5.3 Subrule for Module B

Module B interacts with A, C, D, E and F. Figure 28 presents the subrule for the interactions between the beads of B and the beads of the other modules.
Figure 27 Module $B$: Brick $B \searrow$ to the right, and brick $B \swarrow$ shaded to the left.

Figure 28 Subrule for Module $B$. 
F.6 Module C

F.6.1 Bead type sequence for Module C

Length: 3h – 10; 17 bead types used: \textbf{C0..16}

\[ |C| = (C0..2)^{2k} \cdot (C3..5)^k \cdot C3 \cdot (C7..8)^{k-1} \cdot (C9..14)^{k-1} \cdot C9..10 \cdot C15..16 \cdot C13 \]

F.6.2 The bricks for module C

Module $C$ adopts four brick conformations:

\begin{align*}
\text{C⃗} &= \text{HorizontalMirror}(\text{C⃗}) \\
\text{C⃗} &= \text{Rotate}_{180°}(\text{C⃗}) \\
\text{C⃗End} &= \text{HorizontalMirror}(\text{C⃗}) \\
\text{C⃗End} &= \text{HorizontalMirror}(\text{C⃗})
\end{align*}

Figures 29 and Figure 30 display bricks $\text{C⃗}$ and $\text{C⃗End}$ respectively.

F.6.3 Subrule for Module C

Module $C$ interacts with $A$, $B$, $D$, $E$ and $F$ and might interact with $G$. Figure 31 presents the subrule for the interactions between the beads of $C$ and the beads of the other modules.
Figure 31 Subrule for Module C.
Figure 29 Module C: Brick
Proving the Turing Universality of oritatami Co-Transcriptional Folding

Figure 30: Module C: Brick C End.
F.7 Modules D

F.7.1 Bead type sequence for Modules D

Length: $3W + 30 = 6(\lambda + 5)$; uses: 111 proper bead types, $\text{D0..62}$ and $\text{E0..47}$; plus 2 special bead types from $\text{G, L17..18}$

$$\text{SegD}_0 = D23..33 \cdot E6..11 \cdot (E0..11)^{\kappa-1}$$
$$\text{SegD}_1 = (E12..23)^\omega \cdot D49..45$$
$$\text{SegD}_2 = D34..44 \cdot E30..35 \cdot (E24..35)^{\kappa-1}$$
$$\text{SegD}_3 = (E36..47)^\omega \cdot D54..50$$

$$\text{D1}_{1,0} = \text{SegD}_0 \cdot \text{SegD}_1 \cdot \text{SegD}_2 \cdot \text{SegD}_3 \cdot \text{SegD}_0 \cdot \text{SegD}_1$$

$$\text{D1}_{1,0} = \text{D1}_{2,0} \langle \langle \text{D0..16@0..16} \rangle \rangle$$

$$\text{D1}_{r,1} = \text{D1}_{1,0} \langle \langle \text{D17@}(3W + 22), \text{D18..22@}(3W + 25) .. (3W + 29) \rangle \rangle$$

$$\text{D0}_{r,t} = \text{D1}_{r,1} \langle \langle \text{L17@}(3w + 1), \text{L18@}(3w + 2), \text{D55..62@}(3w + 6) .. (3w + 13) \rangle \rangle$$

F.7.2 The bricks for the modules D

F.7.2.1 The zig and zag brick conformations for Module D

$$\text{SegD}_0 \xrightarrow[\kappa-1]{\kappa-1} \text{NE-path}(D23..33 \cdot E6..11 \cdot (E0..11))$$

$$\text{SegD}_1 \xrightarrow[\kappa-1]{\kappa-1} \text{SW-path}((E12..23)^\omega \cdot D49..45)$$

$$\text{SegD}_2 \xrightarrow[\kappa-1]{\kappa-1} \text{NE-path}(D34..44 \cdot E30..35 \cdot (E24..35))$$

$$\text{SegD}_3 \xrightarrow[\kappa-1]{\kappa-1} \text{SW-path}((E36..47)^\omega \cdot D54..50)$$

$$\text{D1}_{1,0} = \text{SegD}_0 \xrightarrow[\kappa-1]{\kappa-1} \text{SegD}_1 \xrightarrow[\kappa-1]{\kappa-1} \text{SegD}_2 \xrightarrow[\kappa-1]{\kappa-1} \text{SegD}_3 \xrightarrow[\kappa-1]{\kappa-1} \text{SegD}_0 \xrightarrow[\kappa-1]{\kappa-1} \text{SegD}_1$$

$$\text{D1}_{1,0} = \text{D1}_{2,0} \langle \langle \text{D0..16@0..16} \rangle \rangle$$

$$\text{D1}_{r,1} = \text{D1}_{1,0} \langle \langle \text{D17@}(3W + 22), \text{SW-path(D18..22@}(3W + 25) .. (3W + 29) \rangle \rangle$$

$$\text{D0}_{r,t} = \text{D1}_{r,1} \langle \langle \text{L17@}(3w + 1), \text{L18@}(3w + 2), \text{D55..62@}(3w + 6) .. (3w + 13) \rangle \rangle$$

The zig-down and zag brick conformations are obtained by mirroring and rotating the zig-up brick conformation: for $x \in \{0, 1\}$, $r \in \{0, 1, 2\}$ and $t \in \{0, 1\}$, we have

$$\text{D}(x)_{r,t} = \text{HorizontalMirror}(\text{D}(x)_{r,t})$$
$$\text{D}(x)_{r,t} = \text{Rotate}_{180}(\text{D}(x)_{r,t})$$

Figure 32 displays the bricks $\text{D0}_{r,t}$. 
Figure 32 Modules $D_{0,r}$: Bricks $D_{0,r} \uparrow$. 
F.7.2.2  The append brick conformations for Module D

\[ \text{SegD}_0 = \text{E-path}(D23..27) \bowtie \text{SE-path}(D28..30) \bowtie D31 \rightarrow D32 \searrow D33 \nearrow \]
\[ \text{E-glider}'(E6..11 \cdot (E0..11)^{κ-1}) \]
\[ \text{SegD}_1 = \text{E-glider}'((E12..23)^{κ}) \nearrow \text{W-path}(D49..45) \]
\[ \text{SegD}_2 = \text{E-path}(D34..38) \bowtie \text{SE-path}(D39..41) \bowtie D42 \rightarrow D43 \nearrow D44 \searrow \]
\[ \text{E-glider}'((E30..35) \cdot (E24..35)^{κ-1}) \]
\[ \text{SegD}_3 = \text{E-glider}'((E36..47)^{κ}) \nearrow \text{W-path}(D54..50) \]
\[ \text{SegD}_{\text{start}} = D0 \rightarrow D1 \swarrow D2 \rightarrow D3 \searrow D4 \rightarrow D5 \rightarrow D6 \rightarrow D7 \rightarrow D8 \rightarrow D9 \rightarrow D10 \rightarrow \]
\[ \text{D11 \nearrow D12 \searrow D13 \rightarrow D14 \searrow D15 \searrow D16} \]
\[ \text{SegD}_{\text{end}} = D18 \rightarrow D19 \searrow D20 \rightarrow D21 \rightarrow D22 \]
\[ \text{SegD}_{\text{spike}} = D55 \swarrow D56 \searrow D57 \rightarrow D58 \rightarrow D59 \rightarrow D60 \searrow D61 \swarrow D62 \]

\[ \text{D}_{1,0} = \left( \text{SegD}_0 \right) \searrow \left( \text{SegD}_1 \right) \searrow \left( \text{SegD}_2 \right) \searrow \left( \text{SegD}_3 \right) \searrow \left( \text{SegD}_{\text{end}} \right) \searrow \left( \text{SegD}_{\text{spike}} \right) \searrow \left( \text{SegD}_{\text{start}} \right) \]
\[ \text{D}_{1,0} = \left( \text{SegD}_0 \right) \searrow \left( \text{SegD}_1 \right) \searrow \left( \text{SegD}_2 \right) \searrow \left( \text{SegD}_3 \right) \searrow \left( \text{SegD}_{\text{end}} \right) \searrow \left( \text{SegD}_{\text{spike}} \right) \searrow \left( \text{SegD}_{\text{start}} \right) \]

\[ \text{D}_{1,1} = \left( \text{D}_{1,2,0} \right) \searrow \left( \text{SegD}_{\text{end}} \right) \searrow \left( \text{SegD}_{\text{spike}} \right) \searrow \left( \text{SegD}_{\text{start}} \right) \]
\[ \text{D}_{1,1} = \left( \text{D}_{1,2,0} \right) \searrow \left( \text{SegD}_{\text{end}} \right) \searrow \left( \text{SegD}_{\text{spike}} \right) \searrow \left( \text{SegD}_{\text{start}} \right) \]

\[ \text{D}_{0,0} = \left( \text{L}_{17} \right) \searrow \left( \text{L}_{18} \right) \searrow \left( \text{SegD}_{\text{spike}} \right) \searrow \left( \text{SegD}_{\text{end}} \right) \searrow \left( \text{SegD}_{\text{start}} \right) \]
\[ \text{D}_{0,0} = \left( \text{L}_{17} \right) \searrow \left( \text{L}_{18} \right) \searrow \left( \text{SegD}_{\text{spike}} \right) \searrow \left( \text{SegD}_{\text{end}} \right) \searrow \left( \text{SegD}_{\text{start}} \right) \]

Figure 33 displays the bricks \( \text{D}_{0,r,t} \).

F.7.3  Subrule for Module D

Module \( D \) interacts with \( A, B, C, E \) and \( G \). Figure 34 presents the subrule for the interactions between the beads of \( D \) and the beads of the other modules.
Figure 33: Module $D_{0,0}$: Bricks $D_{0,0}$. 

(a) The brick $D_{0,0}$.

(b) The brick $D_{0,0}$.

(c) The brick $D_{0,0}$.

(d) The brick $D_{0,0}$.

(e) The brick $D_{0,0}$.

(f) The brick $D_{0,0}$.
Figure 34: Subrule for Module D1.
F.8 Modules E

F.8.1 Bead type sequences for Modules E

Length: \(6\lambda(L - a + P) + 8h - 1\); 
146 bead types used: \(F_{0..51}, G_{0..48}, H_{0..24}, \text{and} I_{0..19}\).

\[
\begin{align*}
\overline{\text{SegEA}} &= (F_{0..11})^\kappa \cdot (F_{12..23})^\kappa \cdot (F_{24..35})^\kappa \cdot (F_{36..47})^\kappa \cdot \infty \\
\overline{\text{SegEB}} &= (G_{0..11})^\kappa \cdot (G_{12..23})^\kappa \cdot (G_{24..35})^\kappa \cdot (G_{36..47})^\kappa \cdot \infty \\
\text{Head}_{E_n} &= \overline{\text{SegEA}}_{b..b+3c(a)−2} \cdot F_{51} \cdot \overline{\text{SegEB}}_{b+3c(a)..b+K−2}
\end{align*}
\]

where \(b = 0\) if \(a\) is even, 
and \(b = 2\lambda\) if \(a\) is odd.

\[
\begin{align*}
\overline{\text{SegEC}} &= H_{0..4} \cdot (H_{5..16})^{q−1} \cdot H_{5..10} \cdot H_{17..24} \\
\overline{\text{SegED}}_0 &= I_{15} \cdot I_{11..5} \cdot (I_{0..5})^{k−1} \cdot I_{0..1} \cdot I_{18} \\
\overline{\text{SegED}}_1 &= I_{19} \cdot I_{7..8} \cdot (I_{6..8})^{2k−1} \cdot I_{6..7} \cdot I_{15..16} \\
\overline{\text{SegED}}_2 &= I_{17} \cdot I_{10..11} \cdot (I_{9..11})^{2k−1} \cdot I_{9..10} \cdot I_{19} \\
\overline{\text{SegED}}_3 &= I_{18} \cdot I_{13..14} \cdot (I_{12..14})^{2k−1} \cdot I_{12..13} \cdot I_{19} \\
\overline{\text{SegED}}_4 &= I_{19} \cdot I_{11..2} \cdot (I_{0..2})^{2k} \\
\text{TailE} &= \overline{\text{SegEC}} \cdot \overline{\text{SegED}}_0 \cdot \overline{\text{SegED}}_1 \cdot \overline{\text{SegED}}_2 \cdot \overline{\text{SegED}}_3 \cdot \overline{\text{SegED}}_4
\end{align*}
\]

\[
\begin{align*}
E_0 &= \text{Head}_{E_0} \langle [F_{48..49}@0..1, F_{50}@11] \rangle \cdot G_{48} \cdot \text{TailE} \\
E_{>0} &= \text{Head}_{E_{>0}} \cdot G_{48} \cdot \text{TailE}
\end{align*}
\]
F.8.2 The bricks for the modules E

F.8.2.1 Zig-up, zig-down and zag bricks for Module E

$$\text{SegEA} = \left(\text{NE-path} \left( F_{0..11} \right) \right) \rightarrow \text{SW-path} \left( F_{12..23} \right) \rightarrow \text{NE-path} \left( F_{24..35} \right) \rightarrow \text{SW-path} \left( F_{36..47} \right)$$

$$\text{SegEB} = \left(\text{NE-path} \left( G_{0..11} \right) \right) \rightarrow \text{SW-path} \left( G_{12..23} \right) \rightarrow \text{NE-path} \left( G_{24..35} \right) \rightarrow \text{SW-path} \left( G_{36..47} \right)$$

$$\text{HeadE} = \text{SegEA}_{b..b+3c(a)−2} d \ F_{51} \ d \ \text{SegEB}_{b+3c(a)..b+K−2}$$

where $b = 0$ if $a$ is even, and $b = 2\lambda$ if $a$ is odd; and $d = \text{NE}$ if $\frac{3c^2}{λ}$ is even, else $d = \text{SW}$

$$\text{SegEC} = \left(\text{H0} \rightarrow \text{H1} \rightarrow \text{H2} \rightarrow \text{H3} \rightarrow \text{H4} \right)$$

$$\text{SegED}_0 = \left(\text{I15} \rightarrow \text{I16} \rightarrow \text{I17} \rightarrow \text{I18} \rightarrow \text{I19} \rightarrow \text{I20} \right)$$

$$\text{SegED}_1 = \left(\text{I19} \rightarrow \text{I20} \rightarrow \text{I21} \rightarrow \text{I22} \rightarrow \text{I23} \rightarrow \text{I24} \right)$$

$$\text{SegED}_2 = \left(\text{I19} \rightarrow \text{I20} \rightarrow \text{I21} \rightarrow \text{I22} \rightarrow \text{I23} \rightarrow \text{I24} \right)$$

$$\text{SegED}_3 = \left(\text{I19} \rightarrow \text{I20} \rightarrow \text{I21} \rightarrow \text{I22} \rightarrow \text{I23} \rightarrow \text{I24} \right)$$

Figure 35 displays the bricks $E_i$. 
Proving the Turing Universality of oritatami Co-Transcriptional Folding

(a) Module \( \text{E}_1 \): Blueprint of the brick \( \text{E}_1 \).

(b) Precise description of the brick \( \text{E}_1 \).

\( \text{Figure 35} \) Module \( \text{E}_1 \): Bricks \( \text{E}_1 \).
F.8.2.2 Carriage-return bricks for Module E

\[
\text{SegEA} = \text{E-glider}(\text{SegEA})
\]
\[
\text{SegEB} = \text{W-path}(\text{SegEB})
\]
\[
\text{HeadE} = \text{SegEA}_{b..b+3c(a)-2} \uparrow \text{SegEB}_{b+3c(a).b+K-2}
\]
where \( b = 0 \) if \( a \) is even, and \( b = 2\lambda \) if \( a \) is odd;

\[
\text{SegEC} = \text{W-path}(H0..4 \cdot (H5..16)^{q-1} \cdot H5..10 \cdot H17..19) \searrow H20 \rightarrow H21 \rightarrow H22 \rightarrow H23 \rightarrow H24
\]
\[
\text{SegED} = \text{W-path}(H12..5 \cdot (I0..5)^{k-1} \cdot I0..1) \searrow I15 \rightarrow I16
\]
\[
\text{SegED} = \text{W-path}(I17..8 \cdot (16..8)^{2k-1} \cdot 16..7) \searrow I15 \rightarrow I16
\]
\[
\text{SegED} = \text{W-path}(I11..11 \cdot (I9..11)^{2k-1} \cdot I9..10) \searrow I19
\]
\[
\text{SegED} = \text{W-path}(I13..14 \cdot (I12..14)^{2k-1} \cdot I12..13) \searrow I19
\]
\[
\text{TailE} = \text{SegEC} \uparrow \text{SegED} \uparrow \text{SegED} \uparrow \text{SegED} \uparrow \text{SegED}
\]
\[
\text{E} = \text{HeadE} \Leftarrow (F48 \rightarrow F49 \rightarrow F50 \rightarrow 11) \searrow G48 \rightarrow \text{TailE}
\]
\[
\text{E}_{a>0} = \text{HeadE} \Uparrow G48 \leftarrow \text{TailE}
\]

Figures 36 and 37 display the blueprints and the description of bricks \( \text{E} \) respectively.

\[\text{(a) Blueprint of brick } \text{E}_{a=0} \text{, made of four bricks } \text{SegEA}, \text{SegEB}, \text{SegEC} \text{ and } \text{SegED}.\]

\[\text{(b) Blueprint of brick } \text{E}_{a>0} \text{, made of four bricks } \text{SegEA}, \text{SegEB}, \text{SegEC} \text{ and } \text{SegED}.\]

\[\text{Figure 36 Outline of the four different parts of module } \text{E}, \text{ when folded at the end of the appended appendant. See Figure 37 for the detailed beads of each part.}\]

F.8.3 Subrule for Module E

Module \( \text{E} \) interacts with \( \text{B}, \text{C}, \text{D} \) and \( \text{F} \). Figure 38 presents the subrule for the interactions between the beads of \( \text{E} \) and the beads of the other modules.
Figure 38 Subrule for Module E.
(a) The \textbf{SegEA} part of brick \textbf{HeadE}_a, when \( a = 0 \).

\[ \kappa - 1 \text{ times} \quad \kappa - 1 \text{ times} \quad \kappa - 1 \text{ times} \quad \kappa - 1 \text{ times} \quad \text{here, } i = \lfloor \frac{3c(a) - 6}{3} \rfloor \mod 4 \]

\[ (h+1,0) \quad 48\kappa(-4\lambda)\text{-bead long sequence repeated infinitely but truncated to the first } 3c(a) - 6 \text{ beads} \quad (h+1+c(a),1) \]

(b) The \textbf{SegEA} part of brick \textbf{HeadE}_a, when \( a \) is odd.

\[ \kappa - 1 \text{ times} \quad \kappa - 1 \text{ times} \quad \kappa - 1 \text{ times} \quad \kappa - 1 \text{ times} \quad \text{here, } i = \lfloor \frac{3c(a)}{3} \rfloor \mod 4 \]

\[ (h+1,0) \quad 48\kappa(-4\lambda)\text{-bead long sequence repeated infinitely but truncated to the first } 3c(a) - 6 \text{ beads} \quad (h+1+c(a),1) \]

(c) The \textbf{SegEA} part of brick \textbf{HeadE}_a, when \( a \) is even and positive.

\[ \text{here, each } j \text{ can take any of the values } 0, 12, 24, \text{ or } 36 \]

\[ (h+1,0) \quad \kappa \text{ times} \quad \kappa \text{ times} \quad \kappa \text{ times} \quad \kappa \text{ times} \quad \text{here, } i = \lfloor \frac{h}{3}\rfloor \mod 4 \]

\[ (6h-8,1) \quad 4\lambda\text{-bead long sequence repeated } 1:3(a+b) \text{ times and } \text{truncated to indices in the interval } [b + 3e + 3a \cdot b - 3W(a) - 3] \text{ from right to left} \]

\[ (9h-15,1) \quad \frac{h - 3}{4} - 1 \text{ times} \quad (9h-15,1) \]

(d) The \textbf{SegEB} part of brick \textbf{HeadE}_a.

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

\[ (6h-8,1) \]

(e) The brick \textbf{SegEC}_a.

\[ \text{here, } j \text{ is either } 5 \text{ or } 29; \text{ and each } i \text{ can take any value in } \{a+b: a \in \{0,5,6,11\} \text{ and } b \in \{0,12,24,36\}\} \]

\[ (h+2,0) \quad \text{here, } j = \{5,29\} \]

(f) The brick \textbf{SegEd}_a.

\[ \text{Figure 37 The bricks } E_a \]
F.9 Module F

F.9.1 Bead type sequence for Module F

Length: 4h; 53 bead types used: \textbf{J0..52}.

\[
\begin{align*}
\text{Head}_F &= J0_\cdot (J5..10)^{3k-1} \cdot J5 \cdot J11 \cdot J23 \\
\text{Tail}_F &= J48 \cdot (J51..48)^9 \cdot J51 \cdot J52 \cdot J49 \cdot 48
\end{align*}
\]

\[
\begin{align*}
\text{SegExp}(2i) &= J24..29 \cdot (J30..35)^{3^{2i-1}-1} \\
\text{SegExp}(2i+1) &= J36..41 \cdot (J42..47)^{3^{2i}-1} \\
\text{SegExp}_F &= \bigcirc_{i \geq 2} \text{SegExp}(i) \\
F &= \text{Head}_F \cdot \left( J39..41 \cdot \text{SegExp}_F_{0..(h-51)} \right)^R \cdot \text{Tail}_F
\end{align*}
\]

F.9.2 The bricks for the modules F

Module \textbf{F} adopts three brick conformations: zig-up \textbf{F$$\uparrow$$}, zig-down \textbf{F$$\downarrow$$}, and zag \textbf{F$$\leftarrow$$}. The two last are obtained by mirroring and rotating the first.

\[
\begin{align*}
\text{Head}_{F$$\uparrow$$} &= J0 \rightarrow J1_{NW} J2 \leftarrow J3_{SE} J4_{NE} \text{NE-glider} \left( J5..10 \right)^{3k-1} \rightarrow J5_{NW} J6_{E} J7_{NE} \\
&\hspace{1cm} J11_{SE} J12_{NE} J13_{E} J14_{NE} J15_{W} J16_{E} J17_{E} \rightarrow \text{SW-path}(J18..23) \\
\text{Tail}_{F$$\uparrow$$} &= \text{SW-path}(J48 \cdot (J51..48)^9 \cdot J51 \cdot J52 \cdot J49 \cdot 48)
\end{align*}
\]

Recall that \textbf{SegExpF} = \bigcirc_{i \geq 2} \text{SegExp}(i)

\[
\begin{align*}
\textbf{F$$\uparrow$$} &= \text{HorizontalMirror}(\textbf{F}) \\
\textbf{F$$\downarrow$$} &= \text{Rotate}_{180}(\textbf{F$$\uparrow$$})
\end{align*}
\]

Figure 39 displays the brick \textbf{F$$\uparrow$$}.

F.9.3 Subrule for Module F

Module \textbf{F} interacts with \textbf{A}, \textbf{B}, \textbf{C}, \textbf{E}, and \textbf{G}. Figure 40 presents the subrule for the interactions between the beads of \textbf{F} and the beads of the other modules.
Truncate this infinite sequence to height $h - 50$.

Concatenate for $j = 1$ to $\infty$.

$3^{2j-1} - 1$ times.

$3^{2j} - 1$ times.

Figure 39 Module F: Brick F.
Figure 40 Subrule for Module F.
F.10 Module G

F.10.1 Bead type sequence for Module G

Length: $6h - 1$; 201 bead types used: $K0..69$, $L0..99$, and $M0..30$.

\[
\begin{align*}
\text{SegExp}'(2i) &= K4..9 \cdot (K10..15)_{2i-1-1} \quad \text{for } i \geq 1 \\
\text{SegExp}'(2i+1) &= K16..21 \cdot (K22..27)_{2i-1} \quad \text{for } i \geq 1 \\
\text{SegExpG} &= \bigodot \text{SegExp}'(i)_{i \geq 2} \\
\text{SegG1} &= L0..6 \cdot K3 \cdot (K0..3)^9 \cdot K0..2 \cdot L7..10 \cdot \text{SegExpG}_{8..h-51} \\
\text{SegG2} &= K32 \cdot K33 \cdot (K28..33)^{k-3} \cdot K28..32 \\
\text{SegG3} &= K35..39 \cdot (K34..39)^{k-14} \cdot K34 \cdot K35 \cdot L39..41 \cdot K45 \cdot (K40..45)_{10} \cdot K40 \cdot K41 \\
\text{SegG4} &= K50..51 \cdot (K46..51)^{k-3} \cdot K46..48 \\
\text{SegG5} &= K55..57 \cdot (K52..57)^{k-6} \cdot K52..53 \cdot L74 \cdot L75 \cdot K56..57 \cdot (K52..57)^2 \cdot K52..53 \\
\text{SegG6} &= K63 \cdot (K58..63)^{k-19} \cdot K58..61 \cdot L91..99 \cdot M0..19 \cdot K67..69 \cdot (K64..69)^{10} \cdot M20..30 \\
G &= \text{SegG1} \cdot L11..24 \cdot \text{SegG2} \cdot L25..38 \cdot \text{SegG3} \cdot L42..55 \cdot \text{SegG4} \cdot L56..73 \cdot \text{SegG5} \cdot L76..90 \cdot \text{SegG6}
\end{align*}
\]
F.10.2 The bricks for the modules G

F.10.2.1 Zig-up bricks for Module G

G adopts two different bricks in zig-up phase, depending on the letter encoded in the zag-phase above: \( \text{G Read}^0 \) and \( \text{G Read}^1 \).

The \( \text{G Read}^0 \) brick.

\[
\begin{align*}
\text{SegG1Start Read}^0 &= \text{L0} \uparrow \text{L1} \uparrow \text{L2} \rightarrow \text{L3} \rightarrow \text{L4} \rightarrow \text{L5} \rightarrow \text{L6} \rightarrow \text{K3} \\
\text{SegG1Glider NE Read}^0 &= \text{NE-path}(\text{K0..3})^9 \cdot \text{K0..2} \rightarrow \text{L7} \rightarrow \text{L8} \rightarrow \text{L9} \rightarrow \text{L10} \\
\text{SegG2Glider NE Read}^0 &= \text{NE-glider}(\text{L24} \cdot \text{K32..33} \cdot (\text{K28..33})^{k-3} \cdot \text{K28..32} \cdot \text{L25..34}) \\
\text{SegG3Glider NE Read}^0 &= \text{NE-glider}(\text{L35..38} \cdot \text{K35..39} \cdot (\text{K34..39})^{k-14} \cdot \text{K34} \cdot \text{K35} \cdot \text{L39}) \\
\text{SegG1..3Glider NE Read}^0 &= \text{SegG1Glider NE Read}^0 \rightarrow \text{SegG2Glider NE Read}^0 \rightarrow \text{SegG3Glider NE Read}^0 \\
\text{SegG3Glider SE Read}^0 &= \text{SE-rev-glider}(\text{K40..41} \cdot \text{K45} \cdot (\text{K40..45})^{10} \cdot \text{K40..41} \cdot \text{L42}) \\
\text{SegG5Sock NW Read}^0 &= \text{NW-path}(\text{L43..48}) \rightarrow \text{SE-path}(\text{L49..54}) \rightarrow \text{L55} \\
\text{SegG4Glider SE Read}^0 &= \text{SE-rev-glider}(\text{K50..51} \cdot (\text{K46..51})^{k-3} \cdot \text{K46..48} \cdot \text{L56}) \\
\text{SegG4Sock NW Read}^0 &= \text{NW-path}(\text{L57..64}) \rightarrow \text{SE-path}(\text{L65..72}) \rightarrow \text{L73} \\
\text{SegG5Glider SE Read}^0 &= \text{SE-rev-glider}(\text{K55..57} \cdot (\text{K52..57})^{k-6} \cdot \text{K52..53} \cdot \text{L74..75} \cdot \text{K56..57} \cdot (\text{K52..57})^2 \cdot \text{K52..53} \cdot \text{L76}) \\
\text{SegG5Sock NW Read}^0 &= \text{NW-path}(\text{L77..82}) \rightarrow \text{SE-path}(\text{L83..89}) \rightarrow \text{L90} \\
\text{SegG6AGlider SE Read}^0 &= \text{SE-rev-glider}(\text{K63} \cdot (\text{K58..63})^{k-19} \cdot \text{K58..61} \cdot \text{L91}) \\
\text{SegG6Sock NW Read}^0 &= \text{NW-path}(\text{L92..99} \cdot \text{M0..4}) \rightarrow \text{SE-path}(\text{M5..18}) \rightarrow \text{M19} \\
\text{SegG6BGlader Read}^0 &= \text{SE-rev-glider}(\text{K67..69} \cdot (\text{K64..69})^{10} \cdot \text{M20..22} \rightarrow \text{M23} \\
\text{SegG6End Read}^0 &= \text{M24} \rightarrow \text{M25} \rightarrow \text{M26} \rightarrow \text{M27} \rightarrow \text{M28} \rightarrow \text{M29} \rightarrow \text{M30} \\
\text{G Read}^0 &= \text{SegG1Start Read}^0 \rightarrow \text{SegG1..3Glider NE Read}^0 \rightarrow \text{SegG3Glider SE Read}^0 \rightarrow \text{SegG3Sock NW Read}^0 \rightarrow \text{SegG4Glider SE Read}^0 \rightarrow \text{SegG4Sock NW Read}^0 \rightarrow \text{SegG5Glider SE Read}^0 \rightarrow \text{SegG5Sock NW Read}^0 \rightarrow \text{SegG6AGlider SE Read}^0 \rightarrow \text{SegG6Sock NW Read}^0 \rightarrow \text{SegG6BGlader Read}^0 \rightarrow \text{SegG6End Read}^0
\end{align*}
\]

Figure 41 displays the brick \( \text{G Read}^0 \).
Figure 41 Module $\mathbf{G}$: Brick $\mathbf{G} \triangleright \text{Read0}$
Figure 42 displays the brick $G_{\text{Read1}}$. 

The $G_{\text{Read1}}$ brick.

\[
\begin{align*}
\text{SegG3GliderE\text{\text{\texttt{Read1}}} = L41 & \text{\text{\texttt{NW}}} K45 \rightarrow E \text{\text{-}glider'}((K40..45)^{10}) \rightarrow E K40 \rightarrow E K41 \rightarrow L42} \\
\text{SegG3SockW\text{\text{\texttt{Read1}}} = W\text{-path}(L43..48) \rightarrow E\text{-path}(L49..54) \rightarrow \text{L55}} \\
\text{SegG4GliderE\text{\text{\texttt{Read1}}} = E\text{-glider}(K50..51 \cdot (K46..51)^{k-3} \cdot (K46..48).L56)} \\
\text{SegG4SockW\text{\text{\texttt{Read1}}} = W\text{-path}(L57..64) \rightarrow E\text{-path}(L65..72) \rightarrow \text{L73}} \\
\text{SegG5GliderE\text{\text{\texttt{Read1}}} = E\text{-glider}(K55..57 \cdot (K52..57)^{k-6} \cdot (K52..53) \cdot L74..75 \cdot K56..57 \cdot (K52..57)^{2} \cdot K52..53 \cdot L76)} \\
\text{SegG5SockW\text{\text{\texttt{Read1}}} = W\text{-path}(L77..82) \rightarrow E\text{-path}(L83..89) \rightarrow \text{L90}} \\
\text{SegG6AGliderE\text{\text{\texttt{Read1}}} = E\text{-glider}(K63 \cdot (K58..63)^{k-19} \cdot (K58..61 \cdot L91)} \\
\text{SegG6SockW\text{\text{\texttt{Read1}}} = W\text{-path}(L92..99 \cdot M0..4) \rightarrow E\text{-path}(M5..18) \rightarrow M19} \\
\text{SegG6BGliderE\text{\text{\texttt{Read1}}} = E\text{-glider}(K67..69 \cdot (K64..69)^{9} \cdot (K64..66) \rightarrow E K67 \rightarrow E K68 \rightarrow L69)} \\
\text{SegG3..6GliderE\text{\text{\texttt{Read1}}} = \text{SegG3GliderE\text{\text{\texttt{Read1}}} \rightarrow \text{SegG3SockW\text{\text{\texttt{Read1}}} \rightarrow \text{SegG4GliderE\text{\text{\texttt{Read1}}} \rightarrow \text{SegG4SockW\text{\text{\texttt{Read1}}} \rightarrow \text{SegG5GliderE\text{\text{\texttt{Read1}}} \rightarrow \text{SegG5SockW\text{\text{\texttt{Read1}}} \rightarrow \text{SegG6AGliderE\text{\text{\texttt{Read1}}} \rightarrow \text{SegG6SockW\text{\text{\texttt{Read1}}} \rightarrow \text{SegG6BGliderE\text{\text{\texttt{Read1}}} \rightarrow \text{SegG6End\text{\text{\texttt{Read1}}} = E\text{-path}(M20..22) \rightarrow W\text{-path}(M23..25) \rightarrow E\text{-path}(M26..29) \rightarrow M30}} \\
\text{SegG6End\text{\text{\texttt{Read1}}} = \text{SegG1Start\text{\text{\texttt{Read}}} \rightarrow \text{SegG1..3GliderNE\text{\text{\texttt{Read}}} \rightarrow \text{L40}} \rightarrow \text{SegG6End\text{\text{\texttt{Read1}}} \rightarrow \text{SegG6End\text{\text{\texttt{Read1}}} \rightarrow \text{SegG6End\text{\text{\texttt{Read1}}}}}
\end{align*}
\]
Figure 42 Module $G$: Brick $G \rightarrow \text{Read}$.
F.10.2.2 Zig-down and Zag bricks for Module G: Copy letter 0 and 1.

\[
\text{SegG1 COPY} = \text{SE-path}(L_{0..6} \cdot K_3 \cdot (K_{0..3})^9 \cdot K_{0..2} \cdot L_{7..10} \cdot \text{SegExpG}_{k=h-51} \cdot L_{11..17})
\]

\[
\text{SegG2 COPY} = \text{NW-path}(L_{18..24} \cdot K_{32..33} \cdot (K_{28..33})^{k-3} \cdot K_{28..32} \cdot L_{25..30})
\]

\[
\text{SegG3 COPY} = \text{SE-path}(L_{32..38} \cdot K_{35..39} \cdot (K_{34..39})^{k-14} \cdot K_{34..35} \cdot L_{36..41} \cdot K_{45} \cdot (K_{40..45})^{10} \cdot K_{40..41} \cdot L_{42..48})
\]

\[
\text{SegG4 COPY} = \text{NW-path}(L_{49..55} \cdot K_{50..51} \cdot (K_{46..51})^{k-3} \cdot K_{46..48} \cdot L_{56..63})
\]

\[
\text{SegG5 COPY} = \text{SE-path}(L_{66..73} \cdot K_{55..57} \cdot (K_{52..57})^{k-6} \cdot K_{52..53} \cdot L_{74..75} \cdot K_{56..57} \cdot (K_{52..57})^2 \cdot K_{52..53} \cdot L_{76..81})
\]

\[
\text{SegG6 COPY} = \text{NW-path}(L_{84..90} \cdot K_{63} \cdot (K_{58..63})^{k-19} \cdot K_{58..61} \cdot L_{91..99} \cdot M_{0..19} \cdot K_{67..69} \cdot (K_{64..69})^{10} \cdot M_{20..30})
\]

\[
G \xrightarrow{\text{COPY0}} = G \xrightarrow{\text{SegG1 COPY}} \xrightarrow{E} G \xrightarrow{\text{SegG2 COPY}} \xrightarrow{\text{NE}} L_{31} \xrightarrow{\text{SE}} G \xrightarrow{\text{SegG3 COPY}} \xrightarrow{\text{NE}} G \xrightarrow{\text{SegG4 COPY}} \xrightarrow{\text{NE}} L_{64} \xrightarrow{E} L_{65} \xrightarrow{\text{SW}} G \xrightarrow{\text{SegG5 COPY}} \xrightarrow{\text{SE}} L_{82} \xrightarrow{E} L_{83} \xrightarrow{\text{NW}} G \xrightarrow{\text{SegG6 COPY}}
\]

\[
G \xrightarrow{\text{COPY1}} = G \xrightarrow{\text{SegG1 COPY}} \xrightarrow{E} G \xrightarrow{\text{SegG2 COPY}} \xrightarrow{\text{NW}} L_{31} \xrightarrow{E} G \xrightarrow{\text{SegG3 COPY}} \xrightarrow{\text{NE}} G \xrightarrow{\text{SegG4 COPY}} \xrightarrow{E} L_{64} \xrightarrow{E} L_{65} \xrightarrow{\text{SW}} G \xrightarrow{\text{SegG5 COPY}} \xrightarrow{\text{SE}} L_{82} \xrightarrow{E} L_{83} \xrightarrow{\text{NW}} G \xrightarrow{\text{SegG6 COPY}}
\]

\[
G \xrightarrow{\text{COPY0}} = \text{VerticalMirror}(G \xrightarrow{\text{COPY0}})
\]

\[
G \xrightarrow{\text{COPY1}} = \text{VerticalMirror}(G \xrightarrow{\text{COPY1}})
\]

Figures 43 and 44 display the bricks \( G \xrightarrow{\text{COPY0}} \) and \( G \xrightarrow{\text{COPY1}} \).
The various bond patterns repeated inside each part of this brick.
The various bond patterns repeated inside each part of this brick.

Concatenate for $j = 1$ to $\infty$

Truncate this infinite sequence to height $h - 68$

Thevariousbondpatternsrepeated insideeachpartofthisbrick
F.10.2.3 Line feed brick for Module G

\[
\text{SegG1A\#LF} = \text{NW-path}(L0..3) \xrightarrow{\text{L4}} \text{SE-path}(L4..6 \cdot K3)
\]

\[
\text{G \# LineFeed} = \text{SegG1A\#LF} \xrightarrow{\text{Rotate_{180}}} \left( \text{SegG1..3GliderNE} \equiv \text{Read} \left( \text{SegG6End} \equiv \text{Read} \right) \right) \xrightarrow{\text{L40}} \text{L40}
\]

\[
\xrightarrow{\text{M23}} \left( \text{SegG6End} \equiv \text{Read} \right)
\]

Figure 45 displays the brick \text{SegG6End} \equiv \text{Read}.

F.10.3 Subrule for Module G

Module \text{G} interacts with \text{A}, \text{C}, \text{D}, \text{F}. Figure 46 presents the subrule for the interactions between the beads of \text{G} and the beads of the other modules.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{subrule_g.png}
\caption{Subrule for Module \text{G}.}
\end{figure}
Proving the Turing Universality of oritatami Co-Transcriptional Folding

Figure 45 Module $G$ Brick $G$ LineFeed.
F.11  The seed conformation

The seed \( \text{Seed}(u) \) is a conformation encoding the input dataword \( u \) of the simulated SCTS. It is made of 4 types of conformations, (see Figure 47 for an illustration):

\[
\begin{align*}
\text{SegSeedBegin} &= \left( J_8 \downarrow J_7 \downarrow \right)^{2+\frac{2n}{2}} J_{11} \downarrow J_{12} \downarrow J_{16} \downarrow J_{17} \downarrow J_{18} \\
\text{SegSeedSuffix} &= A_9 \downarrow A_{12} \downarrow B_0 \downarrow B_1 \downarrow B_2 \downarrow B_3 \downarrow B_4 \downarrow B_5 \downarrow \left( \downarrow A_0 \right) \downarrow A_0^{-1} \downarrow H_{18} \downarrow H_{19} \downarrow H_{20} \downarrow H_{21} \downarrow H_{22} \downarrow H_{23} \downarrow H_{24} \downarrow H_{25} \downarrow H_{26} \downarrow H_{27} \downarrow H_{28} \downarrow H_{29} \downarrow H_{30} \downarrow H_{31} \downarrow H_{32} \downarrow H_{33} \downarrow H_{34} \downarrow H_{35} \downarrow H_{36} \downarrow H_{37} \downarrow H_{38} \downarrow H_{39} \downarrow H_{40} \downarrow H_{41} \downarrow H_{42} \downarrow H_{43} \downarrow H_{44} \downarrow H_{45} \downarrow H_{46} \downarrow H_{47} \downarrow H_{48} \downarrow H_{49} \downarrow H_{50} \downarrow H_{51} \downarrow H_{52} \downarrow H_{53} \downarrow H_{54} \downarrow H_{55} \downarrow H_{56} \downarrow H_{57} \downarrow H_{58} \downarrow H_{59} \downarrow H_{60} \downarrow H_{61} \downarrow H_{62} \downarrow H_{63} \downarrow H_{64} \downarrow H_{65} \downarrow H_{66} \downarrow H_{67} \downarrow H_{68} \downarrow H_{69} \downarrow H_{70} \downarrow H_{71} \downarrow H_{72} \downarrow H_{73} \downarrow H_{74} \downarrow H_{75} \downarrow H_{76} \downarrow H_{77} \downarrow H_{78} \downarrow H_{79} \downarrow H_{80} \downarrow H_{81} \downarrow H_{82} \downarrow H_{83} \downarrow A_6 \\
\text{SegSeed} (0) &= L_{17} \downarrow L_{18} \downarrow L_{47} \downarrow L_{48} \downarrow L_{49} \downarrow L_{82} \downarrow L_{83} \downarrow A_6 \\
\text{SegSeed} (1) &= L_{17} \downarrow L_{18} \downarrow L_{47} \downarrow L_{48} \downarrow L_{82} \downarrow L_{83} \downarrow L_{84} \downarrow A_6 \\
\text{SegSeedEnd} &= K_{34} \downarrow K_{45} \downarrow K_{40} \downarrow K_{41} \downarrow K_{46} \downarrow K_{47} \downarrow K_{57} \downarrow K_{52} \downarrow K_{53} \downarrow K_{54} \downarrow K_{55} \downarrow K_{56} \downarrow K_{57} \downarrow K_{58} \downarrow K_{59} \downarrow K_{60} \downarrow K_{61} \downarrow K_{62} \downarrow K_{63} \downarrow K_{64} \downarrow K_{65} \downarrow K_{66} \downarrow K_{67} \downarrow K_{68} \downarrow K_{69} \downarrow K_{70} \downarrow K_{71} \downarrow K_{72} \downarrow K_{73} \downarrow K_{74} \downarrow K_{75} \downarrow K_{76} \downarrow K_{77} \downarrow K_{78} \downarrow K_{79} \downarrow K_{80} \downarrow K_{81} \downarrow K_{82} \downarrow K_{83} \downarrow K_{84} \downarrow K_{85} \downarrow K_{86} \downarrow K_{87} \downarrow K_{88} \downarrow K_{89} \downarrow K_{90} \downarrow K_{91} \downarrow K_{92} \downarrow K_{93} \downarrow K_{94} \downarrow K_{95} \downarrow K_{96} \downarrow K_{97} \downarrow K_{98} \downarrow K_{99} \downarrow K_{100} \downarrow M_{20} \downarrow M_{26} \downarrow M_{27} \\
&\downarrow M_{28} \downarrow M_{29} \downarrow M_{30}.
\end{align*}
\]

Each letter \( a \in \{0,1\} \) is encoded in the seed by the conformation: (note that it heads westwards)

\[
\begin{align*}
\text{SegSeedLetter}(a) &= \left( \text{SegSeed} (1) \downarrow \text{SegSeedSuffix} \downarrow \right)^{n-1} \text{SegSeed} (a) \downarrow \text{SegSeedSuffix} \\
\end{align*}
\]

Then, the conformation \( \text{Seed}(u) \) is: (note that it heads southwesterly, see Fig. 47)

\[
\begin{align*}
\text{Seed}(u) &= \text{SegSeedBegin} \downarrow \left( \bigcup_{i=1}^{\left| u \right|} \text{SegSeedLetter} (u) \right) \downarrow \text{SegSeedEnd}
\end{align*}
\]

This completes the definition of the primary structure of the oritatami system \( \mathcal{O}_\mathcal{S} = (\left( \pi \right)\infty, \heartsuit, 3) \) whose folding from the seed configuration \( \text{Seed}(u) \) simulates step-by-step the computation of SCTS \( S \) starting with input dataword \( u \). Section H presents the full description of the attraction rule \( \heartsuit \), completing the description of the oritatami system \( \mathcal{O}_\mathcal{S} \).
Figure 47  The brick $\text{Seed}(w)$.  

\begin{align*}
\text{Repeat } n-1 \text{ times} & \quad \text{Concatenate for } i = 0 \text{ to } |u|-1 \text{ from left to right} \\
\text{if } u_i = \emptyset & \quad \text{Write point at } (w+2+iW,1-h) \\
\text{if } u_i = 1 & \quad \text{Write point at } (w+2+iW,-h)
\end{align*}
Computerized proof of correctness of the STCS oritatami simulation

G.1 Enumerating all possible environments for each module

We will now resume and expand the explanations given in Section 5. 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 \( \text{Read} \) zig-folds along \( \text{Read} \), when the top of the brick \( \text{Read} \) folds, and when the zag-bricks folds under \( \text{Read} \).

One can check using Fact 8 that the beads alignments in each brick do 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 G.1.1, 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 G.1.2 using an automatic procedure which produces human-readable certificates called proof-trees.

G.1.1 The three bricks with varying environments

The following lemma show that it is enough to proof one folding of the zag bricks under a \( \text{Read} \), all the other are the same since there are no interaction between the \( \text{Read} \) brick and any zag brick folding immediately below it.

\[ \text{Lemma 9} \quad (\text{Zag-folding under} \ \text{Read}) \]
\[ \text{The modules zag-folding under the bricks} \ \text{Read} \ \text{have no interaction with} \ \text{Read}, \ \text{with the only exceptions of:} \]

- the beads \( A_0 \) and \( A_1 \) of module \( A \) which have bonds with beads \( E(2 + 12i) \) and \( L_{17} \) for \( A_0 \), and \( E(9 + 12i) \) for \( A_1 \), for all \( 0 \leq i \leq 3 \).
- the beads \( L_{17}, L_{18}, D_{57}, D_{58} \) (the bump in module \( D_0 \)) which bond with the beads \( L_{65}, L_{64}, L_{31} \) so that the corresponding module \( G \) folds into the expected brick \( G \).

\[ \text{Proof.} \quad \text{Figure 48 lists all the possible \text{heart}-interactions between the beads accessible from below the \( \text{Read} \) 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 \( L_{17} \) and \( D_{22} \): (in green on Figure 48) these are only present at the junction between the bricks \( \text{Read} \) and \( \text{Read} \), at the end of the rightmost \( \text{Read} \) brick. The correctness of the zag-folding of the \( \text{Read} \) brick below is given next in the proof-trees section.

with beads \( L_{17}, L_{18}, D_{56}, D_{57}, D_{58}, D_{62} \): (in blue on Figure 48) these beads are only present in the spike encoding a \( 0 \) in the brick \( \text{Read} \), and these interactions are the one expected to ensure the copy of the encoding of \( 0 \) by the module \( G \) that will Zag-fold below.
Figure 48 The ♠-rule between the beads accessible from below of brick $D_5$ and the beads that will get in touch with them from all the modules Zag-folding below.
and finally between beads \textbf{A0} and \textbf{A1}, and 4 groups of beads: \textbf{E2}, \textbf{E3}, \textbf{E8}, \textbf{E9}, then \textbf{E14}, \textbf{E15}, \textbf{E20}, \textbf{E21}, then \textbf{E26}, \textbf{E27}, \textbf{E32}, \textbf{E33}, and finally \textbf{E38}, \textbf{E39}, \textbf{E44}, \textbf{E45} (in red on Figure 48). As the width of a zag-folded production segment is \(w + 6 = 0 \mod 12\), the beads \textbf{A0} and \textbf{A1} are always aligned with the same beads within each of these groups (see Figure 33), namely \textbf{A0} with \textbf{E2}, \textbf{E14}, \textbf{E26} and \textbf{E38}, and \textbf{A1} with \textbf{E9}, \textbf{E21}, \textbf{E33} and \textbf{E45}. Furthermore as the interactions of \textbf{A0} and \textbf{A1} are the same with each of them, it is enough to prove that the module [\textbf{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 \textbf{D} 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 \textbf{D symbol}. 

\textbf{Lemma 10} (Top of \textbf{G+\textcolor{red}{Read1}}). \textit{During the folding of the brick \textbf{G+\textcolor{red}{Read1}}, no bead in \textbf{G} interacts with the row above but at its two extremeties, i.e. the 82 top-leftmost beads and the 11 last (\textbf{K34}, \textbf{L55} and \textbf{M20}, \textbf{M30} resp. in Figure 42).}

\textbf{Proof.} Figure 49(a) lists the only beads exposed and accessible from below above \textbf{G+\textcolor{red}{Read1}}. And Figure 49(b) lists all the possible \textcolor{green}{\textbullet} interactions between them (to the left) and the beads of the brick \textbf{G+\textcolor{red}{Read1}} zig-folding below (to the right).

According to the rule in Figure 49(b), besides the interactions at the 82 first beads at the very top-leftmost part of \textbf{G+\textcolor{red}{Read1}} (\textbf{K34}, \textbf{L55} in Figure 42, interactions in green in Figure 49(b)) and the 11 beads at the very end of \textbf{G+\textcolor{red}{Read1}} (\textbf{M20}, \textbf{M30} in Figure 42, interactions in blue in Figure 49(b)), the only possible interaction between \textbf{G+\textcolor{red}{Read1}} and the already present beads above it is: \textbf{L82} \textcolor{green}{\textbullet} \textbf{L74}. But \textbf{L74} appears only once in \textbf{G+\textcolor{red}{Read1}}, at coordinates \((w + 10 + 4k, 1 − h)\) (see Figure ??), while \textbf{L82} appears above \textbf{G+\textcolor{red}{Read1}} at coordinates \((w + 1 + i(w + 6), 2 − h)\) for \(i = 0..n\). The minimal \(x\)-distance between \textbf{L82} and \textbf{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 \textbf{L82} and \textbf{L74} is:

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

As a consequence, \textbf{L74} never gets close enough to interact with \textbf{L82} above (see Figure 49(c) for the closest situation). It follows that one only need to take into account the environment for the folding of the top-leftmost and top-rightmost part of brick \textbf{G+\textcolor{red}{Read1}} (which is done next using proof-trees), the glider between them, zig-folds regardless of the beads above in the environment.

\textbf{Lemma 11} (\textbf{G+\textcolor{red}{Read1}} along \textbf{F\textcolor{green}{Read}}). \textit{When \textbf{G} folds into the brick \textbf{G+\textcolor{red}{Read1}}, no bead in \textbf{SegExpG} can make bonds with the beads in \textbf{F\textcolor{green}{Read}} nearby and thus folds regardless of the beads nearby (as a glider).}

\textbf{Proof.} Figure 50 lists the interactions between the beads in \textbf{SegExpG} and the beads in \textbf{SegExpF}: these are exactly \textbf{K}(4 + i) \textcolor{green}{\textbullet} \textbf{J}(24 + i) for \(i = 0..23\); in particular red-shaded beads \textbf{K4}, \textbf{K9} in \textbf{G} (resp. yellow, \textbf{K10}, \textbf{K15}; blue, \textbf{K16}, \textbf{K21}; and green, \textbf{K22}, \textbf{K27}) can only bond with beads of the same shade \textbf{J24}, \textbf{J29} in \textbf{F} (resp. \textbf{J30}, \textbf{J35}, \textbf{J36}, \textbf{J41}, \textbf{J42}, \textbf{J47}).
(a) The beads accessible when the brick \( \text{Read1} \) zig-folds itself.

(b) The \( \heartsuit \)-rule for the beads accessible by the beads in \( \text{Read1} \) as it zig-folds.

(c) The closest bead L74 in brick \( \text{Read1} \) can get from one bead L82 above (case \( n = 1 \mod 3 \)).
As shown on Figure 39 and 41 the $y$-coordinates explored by these beads are as follows when $G$ zig-folds into $G$ Read1 or $G$ Read2:

- **Red**: the $y$-coordinates of beads $J_{24}, J_{29}$ in $F$ belong to $\{-40 - 3^{2j}, \ldots, -35 - 3^{2j}\}$ for $j \geq 1$, while the corresponding beads $K_{4}, K_{9}$ in $G$ explore $y$-coordinates in $\{-38 - 3^{2j+1}, \ldots, -34 - 3^{2j+1}\}$ for $j' \geq 1$.

- **Yellow**: the $y$-coordinates of beads $J_{30}, J_{35}$ in $F$ belong to $\{-34 - 3^{2j+1}, \ldots, -41 - 3^{2j}\}$ for $j \geq 1$, while the corresponding beads $K_{10}, K_{15}$ in $G$ explore $y$-coordinates in $\{-36 - 3^{2j+2}, \ldots, -36 - 3^{2j+1}\}$ for $j' \geq 1$.

- **Blue**: the $y$-coordinates of beads $J_{36}, J_{41}$ in $F$ belong to $\{-40 - 3^{2j+1}, \ldots, -35 - 3^{2j+1}\}$ for $j \geq 1$, while the corresponding beads $K_{16}, K_{21}$ in $G$ explore $y$-coordinates in $\{-38 - 3^{2j'}, \ldots, -34 - 3^{2j'}\}$ for $j' \geq 1$.

- **Green**: the $y$-coordinates of beads $J_{42}, J_{47}$ in $F$ belong to $\{-34 - 3^{2j+2}, \ldots, -41 - 3^{2j+1}\}$ for $j \geq 1$, while the corresponding beads $K_{2}, K_{27}$ in $G$ explore $y$-coordinates in $\{-36 - 3^{2j+1}, \ldots, -36 - 3^{2j}\}$ for $j' \geq 1$.

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

- $-35 - 3^{2j+2} \leq -38 - 3^{2j+1} \leq -34 - 3^{2j+1} \leq -40 - 3^{2j}$
- $-36 - 3^{2j+1} \leq -34 - 3^{2j+1} \leq -41 - 3^{2j} \leq -36 - 3^{2j}$
- $-34 - 3^{2j+2} \leq -40 - 3^{2j+1} \leq -35 - 3^{2j+1} \leq -38 - 3^{2j}$
- $-41 - 3^{2j+1} \leq -36 - 3^{2j+1} \leq -36 - 3^{2j} \leq -34 - 3^{2j}$

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 $F$ when $G$ zig-folds into brick $G$ Read1.

### G.1.2 Proof-trees: An automated human-readable certificate for the correctness of oritatami system

A proof-tree is a compact representation of the enumeration of all the possible paths the molecule explores as it folds. Figure 51 presents the proof-tree for the folding of $G$ when...
Figure 51 Excerpt from the proof-tree certificate for the folding of $G$ into $G \cdot \text{Read} \cdot$ when bouncing on a spike encoding a 0.
bouncing on a bump encoding a 0 in $\text{Read}_0$. 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 51 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/
G.2 Computer-generated proof trees for each possible environment

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.

<table>
<thead>
<tr>
<th></th>
<th>ZIG-UP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>![Diagram A](#0-98 #1286-1312 #4995-4997)</td>
</tr>
<tr>
<td>B</td>
<td>![Diagram B](#99-103 #1313 #4998-5000)</td>
</tr>
<tr>
<td>C</td>
<td>![Diagram C](#104-159 #1314-1315)</td>
</tr>
<tr>
<td>D</td>
<td>![Diagram D](#160-339 #340-384)</td>
</tr>
<tr>
<td>E</td>
<td>![Diagram E](#1316-1392 #385-749)</td>
</tr>
<tr>
<td>F</td>
<td>![Diagram F](#750-856 #1393-1401)</td>
</tr>
<tr>
<td>G</td>
<td>![Diagram G](#857-1285 #1402-1853)</td>
</tr>
</tbody>
</table>
ZIG-DOWN

A
#1854-1874
#4745-4752
#2362-2578
#2745-2755
#2790-2797

B
#1875-1878
#2579-2580
#2756-

C
#1879-1889
#2798-2838
#4701-4702

D
#1890-1913
#2561-2599
#2600-2602
#1914-1932
same as previous ones

E
#1933-2011
#2603-2632
#2759-2789

F
#2012-2381
#2633-2643

G
#2042-2381
#2644-2744

WRITE

D
#2839-2999
#4753-4786

E
#4703-4733
#3000-3749
#4787-4945

F
#3750-3781
#4734-4744
#4946-4959
ZAG-WRITE

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>#4994-</td>
<td>#3806-3816, #4059-4069, #4215-4225</td>
</tr>
<tr>
<td>C</td>
<td>#3817-3827, #4070-4080, #4226-4236</td>
<td>#4321-4331, #4459-4460, #4475-4476</td>
</tr>
<tr>
<td>D</td>
<td>#3828-3851</td>
<td>#3939-3941, #4434-4439</td>
</tr>
<tr>
<td>E</td>
<td>#4081-4176, #4461-4474</td>
<td>#3852-3924, #3942-4020</td>
</tr>
<tr>
<td>F</td>
<td>#3925-3938, #4021-4034, #4177-4190</td>
<td>#4272-4285, #4429-4433, #4520-4524</td>
</tr>
<tr>
<td>G</td>
<td>#4968-4993</td>
<td>#3782-3805, #4035-4058, #4191-4214</td>
</tr>
</tbody>
</table>
ZAG

A

see Zig-Down

see Zig-Down

B

see Zig-Down

see Zig-Down

C

see Zig-Down

D

see Zig-Down

see Zig-Down

see Zig-Down

E

see Zig-Down

see Zig-Down

see Zig-Down

F

see Zig-Down

see Zig-Down

G

#4654-4700

see Zig-Down

see Zig-Down
The complete attraction rule

We first give the rule in text. Fig. 52 displays it as a matrix.
1:100 Proving the Turing Universality of oritatami Co-Transcriptional Folding
Proving the Turing Universality of oritamats Co-Transcriptional Folding
Figure 52 The rule $\heartsuit$ 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 $A0 \heartsuit A5$. 