Universality in assembly models

Nicolas Schabanel
CNRS
LIP & IXXI - ÉNS de Lyon
Universality in algorithmic self-assembly

• Can we decide if an assembly terminates?
• Can we design a tile system for a given shape?
• Do I need more than one tileset?
• Do I need more than one tile?
• Do I need more than one molecule?
Algorithmic self-assembly simulates any Turing machine at $T^2$ in 2D.

<table>
<thead>
<tr>
<th>Initial configuration</th>
<th>Positions of the head</th>
<th>letters written by the head</th>
</tr>
</thead>
<tbody>
<tr>
<td>t=1</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>t=2</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>t=3</td>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td>t=4</td>
<td>1 1 0 0 1</td>
<td></td>
</tr>
</tbody>
</table>

Order of assembly and movements of the head.
Algorithmic self-assembly simulates any Turing machine at $T^{°2}$ in 2D

Position and state of the head

<table>
<thead>
<tr>
<th>t=4</th>
<th>0</th>
<th>0</th>
<th>1,Halt</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>★</td>
</tr>
<tr>
<td>t=3</td>
<td>0</td>
<td>0</td>
<td>0,q'''</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>★</td>
</tr>
<tr>
<td>t=2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0,q''</td>
</tr>
<tr>
<td>t=1</td>
<td>0</td>
<td>0</td>
<td>1,q'</td>
<td></td>
</tr>
<tr>
<td>t=0</td>
<td>0</td>
<td>1,q</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transitions:

$(0,q''') \leftrightarrow (1,Halt,\bullet)$

$(0,q'') \leftrightarrow (1,q''',\leftarrow)$

$(1,q') \leftrightarrow (0,q'',\rightarrow)$

$(1,q) \leftrightarrow (0,q',\rightarrow)$
Algorithmic self-assembly simulates any Turing machine at $T^2$ in 2D

Tiles for $\langle a, q \rangle \rightarrow \langle b, q', \leftarrow \rangle$
Algorithmic self-assembly simulates any Turing machine at $T^2$ in 2D

1) Organize the information flow passing through the glues on the sides

<table>
<thead>
<tr>
<th>t+1</th>
<th>a-L</th>
<th>L</th>
<th>a-i</th>
<th>...</th>
<th>L</th>
<th>a-2</th>
<th>L</th>
<th>a-1, q'</th>
<th>q'</th>
<th>b</th>
<th>a-2</th>
<th>L</th>
<th>a-1</th>
<th>q'</th>
<th>b</th>
<th>a-1, q'</th>
<th>q'</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a-L</td>
<td>L</td>
<td>a-i</td>
<td>a-2</td>
<td>L</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
<td>a-2</td>
<td>L</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
<td>a-1</td>
<td>q'</td>
</tr>
<tr>
<td></td>
<td>a-L</td>
<td>L</td>
<td>a-i</td>
<td>a-2</td>
<td>L</td>
<td>a-1</td>
<td>(b,q',←)</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
<td>a-2</td>
<td>L</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>a-L</td>
<td>L</td>
<td>a-i</td>
<td>a-2</td>
<td>L</td>
<td>a-1</td>
<td>prv trans</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
<td>a-2</td>
<td>L</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
<td>a-1</td>
<td>q'</td>
<td>b</td>
</tr>
</tbody>
</table>

Tiles for $(a,q) \rightarrow (b,q',\leftarrow)$
Algorithmic self-assembly simulates any Turing machine at $T^\circ 2$ in 2D

1) Organize the information flow passing through the glues on the sides

Tiles for $(a,q) \mapsto (b,q',\leftarrow)$
Algorithmic self-assembly simulates any Turing machine at $T^{\circ}2$ in 2D

1) Organize the information flow passing through the glues on the sides

2) Identify each type of tiles

<table>
<thead>
<tr>
<th>t+1</th>
<th>a_L</th>
<th>a_i</th>
<th>a_2</th>
<th>nxt trans.</th>
<th>a_1</th>
<th>a_i</th>
<th>b</th>
<th>R</th>
<th>a_1</th>
<th>a_i</th>
<th>a_1</th>
<th>a_i</th>
<th>ar</th>
<th>ar</th>
<th>0</th>
<th>★</th>
<th>★</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>a_L</td>
<td>L</td>
<td>L</td>
<td>...</td>
<td>L</td>
<td>a_2</td>
<td>q'</td>
<td>b</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>0</td>
<td>★</td>
<td>★</td>
<td></td>
</tr>
<tr>
<td>a_L</td>
<td>a_i</td>
<td>a_2</td>
<td>a_1</td>
<td>(b,q',←)</td>
<td>a_1</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>a_1</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>ar</td>
<td>ar</td>
<td>0</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>L</td>
<td>a_L</td>
<td>a_i</td>
<td>a_2</td>
<td>prv trans</td>
<td>a_1</td>
<td>a_i</td>
<td>q</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>ar</td>
<td>ar</td>
<td>0</td>
<td>★</td>
<td>★</td>
</tr>
<tr>
<td>t</td>
<td>a_L</td>
<td>a_i</td>
<td>a_2</td>
<td>a_1</td>
<td>a_2</td>
<td>a_1</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>a_i</td>
<td>ar</td>
<td>ar</td>
<td>0</td>
<td>★</td>
<td>★</td>
</tr>
</tbody>
</table>

Tiles for $(a,q) \rightarrow (b,q',\leftarrow)$
Algorithmic self-assembly simulates any Turing machine at $T^2$ in 2D

1) Organize the information flow passing through the glues on the sides

2) Identify each type of tiles

The skeleton

Tiles for $(a,q) \rightarrow (b,q',\leftarrow)$
Algorithmic self-assembly simulates any Turing machine at $T^2$ in 2D

1) Organize the information flow passing through the glues on the sides

2) Identify each type of tiles

The carriers

Tiles for $(a,q) \rightarrow (b,q',\leftarrow)$
Algorithmic self-assembly simulates any Turing machine at $T^2$ in 2D.

1) Organize the information flow passing through the glues on the sides.

2) Identify each type of tiles.

Tiles for $(a,q) \mapsto (b,q',\downarrow)$
Algorithmic self-assembly simulates any Turing machine at $T^2$ in 2D

1) Organize the information flow passing through the glues on the sides

2) Identify each type of tiles

3) Set the order, glue strength & colors to match the flow

Tiles for $(a,q) \rightarrow (b,q',\leftarrow)$
The tileset with size $O(|Q|^2|\Sigma|^2)$

The tiles for each transition $(a,q) \rightarrow (b,q',\subseteq?)$
Encoding the input as a seed

- **Solution 1: hardcoding**
  Uses $n$ tiles for $n$ bits

  - Each tile requires $4 \log n$ bits to encode its glues, thus this encoding uses $4n \log n$ bits in total
  - Can we do better, with less tiles?
Kolmogorov complexity

• Given a universal Turing machine U:
  $K_U(x) = \min \{ |p| : U(p,\varepsilon) = x \}$
  is the size of the smallest program in U that outputs x

• **Fact.** $\forall U, \exists A \text{ s.t. } \forall x, K_U(x) \leq |x| + A$
  
  *Proof.* A is the size of the program "print" in U.

• **Theorem.** $K_U(x)$ is independent of U, indeed:
  $\forall U, U' \exists A, B \text{ s.t. } \forall x, K_{U'}(x) - A \leq K_U(x) \leq K_{U'}(x) + B$
  
  *Proof.* A and B are the sizes of the programs that execute U and U' respectively in U' and U.
Lower bounding the required number of tiles to encode the seed

• The Kolmogorov complexity $K(x)$ is the size of the smallest program that outputs $x$

• Bit size of the encoding with $T$ tiles is $\leq 4T \log_2 T$

• A tileset that self-assembles $x$ is a program that outputs $x$,
  Thus: $4T \log T \geq K(x)$, i.e. $T \geq K(x) / 4 \log K(x)$
Unpacking a binary string

- Cut string $x$ in $n/b$ chunks of $b$ bits and uncompress it

- Example: $x = 001 \ 101 \ 110 \ 011$ and $b = 3$

  $\begin{array}{cccc}
    \text{A} & \text{001} & \text{B} & \text{101} \\
    \text{B} & \text{101} & \text{C} & \text{110} \\
    \text{C} & \text{110} & \text{D} & \text{011} \\
    \text{D} & \text{011} & \star & \\
  \end{array}$

  $n/b$ tiles
## Unpacking a binary string

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>unpacked</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Diagram

- **Unpacked**
- **Packed**

- **Pack**
- **Unpack**

19
Determine the various tile types
Add tiles at the boundary for continuation
The tiles
Total number of tiles: $\sim \frac{n}{b} + 2^{2b}$

Seed: $\frac{n}{b} + 2$ tiles

$2 \leq |w| \leq b$
$|w'| = b$

$2 \leq |w| \leq b$
$|w| = b$

$2 \leq |w| < b$

$2 \times 2^{2b}$
$\times 2^b$
$\times 2^{b+1}$
$\times 2^b$
$\times 4$
Total number of tiles: 
\[ \sim \frac{n}{b} + 2^{2b} \]

Minimized when 
\[ \frac{n}{b} = 2^{2b} \]

i.e. for \[ b = \log(n/\log n)/2 \]

The total number of tiles is: 
\[ \sim \frac{n}{\log n} \] (optimal !)
What about $T^\circ = 1$?
Does algorithmic self-assembly simulate Turing machine at $T^1$ in 2D?

- How to read and propagate the position of the head?

Initial configuration

Crucial to rely on 2 sides!
Does algorithmic self-assembly simulate Turing machine at $T^\circ 1$ in 2D?

• **Theorem.** [Meunier, Regnault, 2015]
  Any deterministic $T^\circ 1$ tile set can be pumped outside a fixed radius in 2D

• **Corollary.** No $T^\circ 1$ tile system is Turing complete in 2D
Algorithmic self-assembly simulates any Turing machine at \( T^\circ 1 \) in 3D

- **Theorem.** [Cook, Fu, Schweller, 2011] There is a \( T^\circ 1 \) tile set that simulates any Turing machine with 2-layers in 3D.

- *Key beautiful idea:* Blocking probe crystal
Algorithmic self-assembly simulates any Turing machine at $T^\circ 1$ in 3D

- Key beautiful idea: Blocking probe crystal
Algorithmic self-assembly simulates any Turing machine at $T^°1$ in 3D

- *Key beautiful idea:* Blocking probe crystal

*Building the 0-blocking bridge*
Algorithmic self-assembly simulates any Turing machine at $T^\circ 1$ in 3D

- *Key beautiful idea*: Blocking probe crystal

We can thus read and write 0 and 1 on the "next tape"!

**Crystals are Turing complete !!!**
An experimental realization of a universal computer
Conclusion

• A lot of new models
• A new kind of geometric algorithmic
• A lot of implication in biology and bio-engineering
• Lots of extensions: mixed dynamics, errors,…

More on fixing errors at the last lecture