Effective Tiling Part 2: Self-assembly

Florent Becker

September 3, 2015
Introduction

The Natural Computing point of view
The Tiling point of view
DNA computing

The aTAM (abstract Tile Assembly Model)
Definition
Examples
Productions

Measuring Efficiency
Markov Approach
Order approach

Universality (Computation)
Counting
Computing

Model parameters
Self-assembly and the aTAM

We can approach *self-assembly* from three directions:

- A natural computing point of view: how to compute as Nature does?
- A tiling algorithm point of view: what is a local tiling algorithm?
- A nano-engineering point of view: how to do cool stuff with DNA or other materials?

From these three directions, the same model appears: the aTAM, so it must have some relevance.
Outline

Introduction
  The Natural Computing point of view
  The Tiling point of view
  DNA computing

The aTAM (abstract Tile Assembly Model)
  Definition
  Examples
  Productions

Measuring Efficiency
  Markov Approach
  Order approach

Universality (Computation)
  Counting
  Computing

Model parameters
Introduction
  The Natural Computing point of view
    The Tiling point of view
    DNA computing

The aTAM (abstract Tile Assembly Model)
  Definition
  Examples
  Productions

Measuring Efficiency
  Markov Approach
  Order approach

Universality (Computation)
  Counting
  Computing

Model parameters
What is self-assembly

- A large number of particles
What is self-assembly

- A large number of particles
- each particle is simple,
What is self-assembly

- A large number of particles
- each particle is simple,
- whose simple local interactions
What is self-assembly

- A large number of particles
- each particle is simple,
- whose simple local interactions
- yield an interesting result
Natural examples

- Crystal growth
Natural examples

- Crystal growth
- Corals
Natural examples

- Crystal growth
- Corals
- Human settlements
Let’s mathematize

- A large number of particles,
Let’s mathematize

- A large number of particles, infinitely many
Let’s mathematize

- A large number of particles, infinitely many
- each particle is simple,
Let’s mathematize

- A large number of particles, \textit{infinitely many}
- each particle is simple, \textit{taken from a finite set}
Let’s mathematize

- A large number of particles, infinitely many
- each particle is simple, taken from a finite set
- whose simple local interactions
Let’s mathematize

- A large number of particles, infinitely many
- each particle is simple, taken from a finite set
- whose simple local interactions on $\mathbb{Z}^2$, at distance 1
Let’s mathematize

- A large number of particles, infinitely many
- each particle is simple, taken from a finite set
- whose simple local interactions on $\mathbb{Z}^2$, at distance 1
- yield an interesting result
Let’s mathematize

- A large number of particles, infinitely many
- each particle is simple, taken from a finite set
- whose simple local interactions on $\mathbb{Z}^2$, at distance 1
- yield an interesting result used as an algorithmic system
Introduction
  The Natural Computing point of view
  The Tiling point of view
  DNA computing

The aTAM (abstract Tile Assembly Model)
  Definition
  Examples
  Productions

Measuring Efficiency
  Markov Approach
  Order approach

Universality (Computation)
  Counting
  Computing

Model parameters
Beyond local rules

The simplest tiling algorithms are greedy application of local rules, but they do not always work. How to add a little computing power while keeping locality? Add the option to say “I don’t know”. This gives the power to synchronize.
Introduction
- The Natural Computing point of view
- The Tiling point of view
- DNA computing

The aTAM (abstract Tile Assembly Model)
- Definition
- Examples
- Productions

Measuring Efficiency
- Markov Approach
- Order approach

Universality (Computation)
- Counting
- Computing

Model parameters
Nano-inspiration

We want to assemble nano-artefacts.

- without the need for nano-manipulations,
- with particles which assemble by local interactions,
- this means that we need molecules with selective attractions
We want to assemble nano-artefacts.

- without the need for nano-manipulations,
- with particles which assemble by local interactions,
- this means that we need molecules with selective attractions
- and planar dynamics
We want to assemble nano-artefacts.

▶ without the need for nano-manipulations,
▶ with particles which assemble by local interactions,
▶ this means that we need molecules with selective attractions
▶ and planar dynamics
▶ DNA is perfect for that
DNA-computing

Silicium machines are slow but not very parallel. Why not rather use $10^{23}$ trivial processors. Again, we need a molecule with programmable interactions.
With 4 single strands of DNA, it is possible to create objects with selective interactions with a $\mathbb{Z}^2$ topology.
DNA tiles

With 4 single strands of DNA, it is possible to create objects with selective interactions with a $\mathbb{Z}^2$ topology.
Introduction
  The Natural Computing point of view
  The Tiling point of view
  DNA computing

The aTAM (abstract Tile Assembly Model)
  Definition
  Examples
  Productions

Measuring Efficiency
  Markov Approach
  Order approach

Universality (Computation)
  Counting
  Computing

Model parameters
Introduction

The Natural Computing point of view
The Tiling point of view
DNA computing

The aTAM (abstract Tile Assembly Model)

Definition
Examples
Productions

Measuring Efficiency

Markov Approach
Order approach

Universality (Computation)

Counting
Computing

Model parameters
Tiles

Definition (Self-assembling tileset)

A *self-assembling tileset* $S$ is given by:

- A finite alphabet $G$
- a *strength* function: $s : G \rightarrow \mathbb{N}$
- A Wang Tile-Set on alphabet $G$
Definition
Let $t_1, t_2$ be two prototiles, and $d$ a direction in \( \{ N = (0, 1), S = -N, E = (1, 0), W = -E \} \) The \textit{link} between $t_1$ and $t_2$ in direction $d$ is:

- $\text{link}(t_1, d, t_2) = 0$ if $d(t_1) \neq (-d)(t_2)$
- $\text{link}(t_1, d, t_2) = f$ if $d(t_1) = d(t_2)$ and $s(d(t_1)) = f$
Definition
Let $S$ be a self-assembling tileset, and let $P$ be a pattern. Let $C$ be a cut of $\text{dom}(M)$, the link along $C$ is defined by:

$$\text{link}(C) = \sum_{e \in C} \text{link}(M(e^-), \text{dir}(e), M(e^+))$$
Definition (Stability)
A pattern $M$ of $S$ is stable at temperature $\tau$ if for any cut $C$ of $M$,

$$\tau \leq \text{link}(C)$$
Dynamics

Definition

$S$ has a transition from $M$ to $M'$ at temperature $\tau$ if:

- $M$ and $M'$ are stable at temperature $\tau$,
- $\text{dom}(M') = \text{dom}(M) \cup \{(x, y)\}$, $(x, y) \notin \text{dom}(M)$ and $M$ and $M'$ coincide on $\text{dom}(M)$.

Possible additions at $\tau = 2$. 
Self-assembling system

Definition

A *self-assembling system* is given by:

- A self-assembling tileset
- an integer $\tau$, the *temperature*
- a pattern $\sigma$, the seed
Introduction
  The Natural Computing point of view
  The Tiling point of view
  DNA computing

The aTAM (abstract Tile Assembly Model)
  Definition
  Examples
  Productions

Measuring Efficiency
  Markov Approach
  Order approach

Universality (Computation)
  Counting
  Computing

Model parameters
Assembly of a rectangle

Seed = tile with a star, Temperature = 2
Assembly of a rectangle

Seed = tile with a star, Temperature = 2
Assembly of a square

Seed = lower left tile, Temperature = 2
Assembly of a square

Seed = lower left tile, Temperature = 2
Introduction
The Natural Computing point of view
The Tiling point of view
DNA computing

The aTAM (abstract Tile Assembly Model)
Definition
Examples
Productions

Measuring Efficiency
Markov Approach
Order approach

Universality (Computation)
Counting
Computing

Model parameters
A *production* of a self-assembling system $S$ is a pattern $p$ such that there is a sequence of transitions from the seed $\sigma(S)$ to $p$ at temperature $\tau(S)$.

An assembly sequence of $p$ is a sequence of transitions from $\sigma(S)$ to $p$.

$p$ is a *final production* of $S$ if there is no transition of $S$ starting from $p$.

The set $S_F$ is the set of all final productions of $S$. 
Assembling a set of shapes

Let $X$ be a set of finite subsets of $\mathbb{Z}^2$.
The most studied problem in self-assembly is the following: given a set $X$, is there a system such that $X = \{\text{dom}(f) | f \in S_F\}$. When it is the case, we say that $S$ assembles $X$. 
Introduction
The Natural Computing point of view
The Tiling point of view
DNA computing

The aTAM (abstract Tile Assembly Model)
Definition
Examples
Productions

Measuring Efficiency
Markov Approach
Order approach

Universality (Computation)
Counting
Computing

Model parameters
Efficiency

How to get an evaluation of the efficiency of a Self-assembling system?

- Number of transitions = surface...
- Number of tiles in the system = Kolmogorov

Is there a notion of time complexity for self-assembly?
Time Efficiency?

- Turing Machine Time Complexity $= \text{scale factor}$
- Physical Assembly Time $= ??$
Parallelism

- Far away transitions are made in parallel
- Each should have its own clock
- More independent transitions $\Rightarrow$ faster assembly
It is possible to refine the Abstract Tile Assembly Model into a kinetic Tile Assembly Model. Given $S$, $P_S$ is a Continuous Time Markov Process defined as follows:

- Each tile of $S$ is given a real number, its concentration;
- The states of $P_S$ are the productions of $S$;
- There is a (Markov) transition from $p$ to $p'$ with rate $\kappa$ if there is an aTAM transition from $p$ to $p'$ and the added tile has concentration $\kappa$.

In other words, we assume that every $1/\kappa$ in average, a tile with concentration $\kappa$ arrives at each possible attachment site, and all these arrival are independant.
Assembly time in the kTAM

- Assembly time in the kTAM = expected time to assemble in the kTAM
- Models parallelism as expected
- But computing the expected time of assembly is not easy (ask your local probabilist).
A combinatorial approach

Markov processes let us model parallelism, but they introduce (sometimes) difficult calculations. Is there a way to express parallelism directly?
Dependency order

Let $s$ be an assembly sequence of a production $p$.

- $s$ defines a total order $o(s)$ on the positions of $p$.
- $\cap_s$ assembly sequence of $p$ $o(s)$ defines a partial order $<_p$ on the positions of $p$.

A production $p$ is ordered (by $<_p$) if any total order that is compatible with $<_p$ corresponds to an assembly sequence of $p$. In an ordered production, $<_p$ captures dependencies between positions.
Theorem

Let $S$ be a $k$TAM where all concentrations are equal to 1, and where the seed is a single tile. Let $P$ be an ordered production, with $<_p$ its order. Let $d(<_P)$ be the depth of $<_P$, and $t(P)$ the assembly time of $P$. Then $t(P) = \Omega(p(<_P))$. 
Introduction
The Natural Computing point of view
The Tiling point of view
DNA computing

The aTAM (abstract Tile Assembly Model)
Definition
Examples
Productions

Measuring Efficiency
Markov Approach
Order approach

Universality (Computation)
Counting
Computing

Model parameters
Introduction
  The Natural Computing point of view
  The Tiling point of view
  DNA computing

The aTAM (abstract Tile Assembly Model)
  Definition
  Examples
  Productions

Measuring Efficiency
  Markov Approach
  Order approach

Universality (Computation)
  Counting
  Computing

Model parameters
Counter tiles

```
  1g  1c  
    0g

  0d  0c  
    0g

  0  0c  0c
    0

  1  0c  0c
    1

  1  0c  1c
    0

  0  1c  1c
    1

  0g  0  0  0
    S S S S

  0d  1c  
    1d
```

done

```
Running the Counter

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0g</td>
<td>0g</td>
<td>0g</td>
<td>0g</td>
<td>0g</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
</tr>
<tr>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0c</td>
<td>0c</td>
<td>0c</td>
<td>0c</td>
<td>0c</td>
</tr>
<tr>
<td>1c</td>
<td>1c</td>
<td>1c</td>
<td>1c</td>
<td>1c</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
</tr>
<tr>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
<td>0d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
<td>1d</td>
</tr>
</tbody>
</table>
Introduction
The Natural Computing point of view
The Tiling point of view
DNA computing

The aTAM (abstract Tile Assembly Model)
Definition
Examples
Productions

Measuring Efficiency
Markov Approach
Order approach

Universality (Computation)
Counting
Computing

Model parameters
Undecidability

Theorem

*Deciding if a self-assembly system $S$ with temperature 2 has a finite final production is undecidable.*

Proof.

By simulation of a Turing Machine: assemble the space-time diagrams of the machine.
Turing Simulation: tiles
Turing Simulation: run

\[
\begin{array}{ccc}
\theta & \mathcal{y} & \theta \\
< & < & < \\
\theta & \mathcal{q}' & \mathcal{q}\mathcal{x} \\
< & < & > \\
\mathcal{q},1 & r,1 & 1 \\
< & < & > \\
\mathcal{q},1 & 1 & d \\
< & < & > \\
\mathcal{q},1 & \mathcal{q}\mathcal{i},\theta & d \\
< & < & > \\
\mathcal{q},1 & \mathcal{q}\mathcal{i},\theta & d \\
< & < & > \\
\mathcal{q},1 & \mathcal{q}\mathcal{i},\theta & d \\
< & < & > \\
\mathcal{q},1 & \mathcal{q}\mathcal{i},\theta & d \\
\end{array}
\]
Introduction
  The Natural Computing point of view
  The Tiling point of view
  DNA computing

The aTAM (abstract Tile Assembly Model)
  Definition
  Examples
  Productions

Measuring Efficiency
  Markov Approach
  Order approach

Universality (Computation)
  Counting
  Computing

Model parameters
Temperature

- At temperature 1, 2d self-assembly is conjectured to be unable to do universal computation.
- But in $\mathbb{Z}^3$, it can simulate Turing Machines.
- In both cases, it cannot simulate temperature 2 self-assembly.
Conflicts

A mismatch is when two adjacent tiles have non-matching glues on their common side.

**Theorem**

_Some sets of shapes cannot be assembled by systems without mismatches._