Grammatical Self Assembly for Planar Tiles

Robert Ghrist and David Lipsky
Department of Mathematics, University of Illinois, Urbana, IL 61801 USA
{ghrist,dlipsky}@math.uiuc.edu

Abstract

We introduce a formal grammatical process for planar self-assembling systems with conformal switching which instantiates the geometry of the tiles. This extends prior work which gave a grammatical structure that models only the topology of the assembly. The addition of geometric data leads to the problem of determining which sets of grammatical rules preserve the planar constraint. We illustrate global obstructions while generating examples of rule sets which intrinsically force the preservation of planar geometry.

1. Introduction

The collective behavior of mechanical devices at micro- and nano- scales increasingly resembles that of chemical or biological reactions, in which individual particles endowed with intrinsic geometric data (the shape of the particle) and algorithmic data (reactivity, bonding proclivities, repulsions) yield assemblies of complex devices through purely local means. The challenge of being able to control these processes for purposes of design in the mechanical realm is two-fold. First is the *physical* challenge: to be able to understand and exploit the behavior of mechanical devices at micro- (or smaller) scales. Second is the *algorithmic* challenge: how does one 'program' a collection of parts under physical constraints to assemble into a desired device. At small scales, one exchanges the ability to fabricate very intricate components for the ability to produce huge numbers of simple components. A significant algorithmic question is therefore which complex devices can be built out of simple pieces with simple assembly rules?

Not surprisingly, experimentalists have been able to build objects which have complete regularity with respect to the local geometry: planar hex and square lattices [17], cylindrical sheets [6], and spheres [1]. These have the property that the component 'tiles' are regularly shaped and the assembly 'rules' are determined by passive physical means, e.g., via surface tension effects on a liquid surface. Such geometrically regular assemblies can and are having significant impact on manufacturing at small scales [10, 11].

We are particularly concerned with 'programmable' self-assembly of mechanical devices at small scales. A recent NASA-IAC report on *Kinematic Cellular Automata* [15] contains extensive information on small-scale programmable mechanical assembly, mostly in the context of self-replication. We note in particular the claim presented there that self-assembly requires "a new mathematics" to be effective in programming devices.

This note continues an ongoing project with E. Klavins to approach the algorithmic problems of mechanical self-assembly using new mathematical tools. The goal of this project is to ascertain when and how one can 'program' system features into a grammatical assembly structure. In this note, we keep formal definitions to a minimum.

1.1. A grammatical approach

We approach the problem of modeling self-assembling systems from the point of view of grammars and computation. There are four principal components to a comprehensive model for self-assembly: these are the

1. **algorithmic**; what type of abstract computation is generated by an assembly sequence of parts?
2. **topological**; in which type of network are the various pieces connected?
3. **geometric**; how do the pieces fill up their environment?
4. **physical**; to what laws are pieces and bonds between pieces subject?

Several authors have approached self-assembling systems from the point of view of algorithms and computation (e.g., [18]), especially in those systems which use DNA strands as a bonding protocol. The work in [4] introduced a grammatical formalism (see §2 below) which captures both algorithmic and topological aspects of self-assembling systems. Geometric and physical constraints are significant challenges to encode.

This note works toward developing a formal procedure for appending geometric content to an abstract self-assembling system. This takes the form of a mapping from abstract graph assemblies to geometric cell complexes. This leads to several interesting problems.

It is well-known that the problem of assembling tiles in the Euclidean plane via compatibility rules on the edges (*Wang tiles*) can perform computations. Indeed, this has recently been implemented using DNA strands as the edge markers (e.g., [18]. This paper considers an inverse of this situation: Which abstract computations (in the form of graph-replacement rule sets) involving marked planar tiles necessarily construct *only* Euclidean planar outputs?

1.2. Prior and proximate work

The assembly model we formalize uses *conformational switching* of components — the ability of an assembled pair to register a change in conformation after an assembly step. Conformational switching was first described as a symbolic process for self assembly by Saitou [12], who considered the assembly of (linear, 1-d) strings. Self assembly as a graph process has been described in [8, 9], with the formalism of graph grammars being introduced in [4]. Graph grammars were introduced in theoretical computer science decades ago [5, 3] and have been used to describe numerous systems, from data structure maintenance to mechanical system synthesis. Graph grammars are a generalization of the standard "linear" grammars used in automata theory and linguistics.

There are many other models of self assembly besides graph grammars. As a brief example, several groups [18, 2] have explored self assembly using passive *tiles* floating in liquid. The tiles attach along complimentary edges (due, for example, to capillary forces or the assembly of complimentary strands of DNA) upon random collisions. A mathematical approach via combinatorial assemblies has been explored in [13]. There is also preliminary work on supplying tile systems with conformal-like state information [7]. Such systems can in fact be used to perform arbitrary computations [16] and are best understood as two or three dimensional symbolic processes. Other authors have used geometric constraints on part-part interactions to model, for example, the assembly of proteins into spherical shells called *capsids* [1]. The addition of simple processing to each part is considered in models of the assembly of the T4 bacteriophage [14]. We note that the ability to program mechanical devices at this scale holds applications beyond assembly, extending to, e.g., mechanical information security.

2. Assembly via Graph Grammars

The majority of this description is derived from [4], where graph grammars were introduced as a language for computational self-assembling systems. Formal definitions will be kept to a minimum.

2.1. Definitions

Let G denote a labeled graph: $G = (V, E, \ell)$, where V is a set of *vertices*, $E \subset V \times V$ is a set of *edges*, and $\ell : V \to \Sigma$ is a labeling function over some alphabet Σ. We denote by V_G, E_G and ℓ_G the vertex set, edge set and labeling function of the graph G or by V, E and ℓ when there is no danger of confusion. For obvious reasons, we prefer a finite alphabet Σ. We assume basic definitions from graph theory such as connectivity, isomorphism, and embedding.

Roughly speaking, a graph assembly system consists of an initial graph G_0 together with a collection of local rules. These rules act as local replacement operations on graphs.

More specifically, a *rule* is a pair of graphs $r = (L, R)$ where $V_L = V_R$. The graphs L and R are called the *left hand side* and *right hand side* of r respectively. The *size* of r is $|V_L| = |V_R|$. Most of the examples in this paper will involve rules which are *binary* — the size of each rule is two. We will assume for this paper that all rules are *constructive* ($E_L \subset E_R$). A rule is *acyclic* if its right hand side contains no cycles (the left hand side may contain cycles).

A rule r is *applicable* to a graph G if there exists a label-preserving embedding $h : L \to G$. An *action* of a rule r on a graph G is a pair (r, h) such that r is applicable to G via h. The *application* of (r, h) to G yields a new graph $G' = (V', E', l')$ defined by replacing the image of L in G with a copy of R. More formally, G' is defined thus:

$$\begin{aligned} V' &= V \\ E' &= (E - \{\{h(x), h(y)\} \mid \{x, y\} \in L\}) \\ &\quad \cup \{\{h(x), h(y)\} \mid \{x, y\} \in R\} \\ l'(x) &= \begin{cases} l(x) \text{ if } x \notin h(V) \\ l_R \circ h^{-1}(x) \text{ otherwise.} \end{cases} \end{aligned}$$

We write $G \xrightarrow{r,h} G'$ to denote that G' was obtained from G by the application of (r, h).

A *graph assembly system* is a pair (G_0, Φ) where G_0 is the *initial graph* and Φ is a set of rules (called the *rule set*). We may refer to a system simply by its rule set Φ and assume that the initial graph G_0 is an unbounded supply of isolated vertices labeled with an initial symbol. An *assembly sequence* for a system (G_0, Φ) is a finite sequence of pairs of graphs $\{G_i\}_{i=0}^k$ and actions $\{(r_i, h_i)\}_{i=1}^k$ where $r_i \in \Phi$ and

$$G_i \xrightarrow{r_i,h_i} G_{i+1}$$

for $i \in \{0, ..., k-1\}$. A graph assembly system (G_0, Φ) defines a non-deterministic system since, at any step, many rules in Φ may be simultaneously applicable.

2.2. Examples and properties

Example 2.1 *Consider the assembly system with initial graph consisting of isolated vertices labeled A_0 and rule set*

$$\Phi = \left\{ \begin{array}{lllll} A_0 & A_0 & \Rightarrow & A_1 - A_1, \\ A_0 & A_1 & \Rightarrow & A_1 - A_2 \end{array} \right. \quad (1)$$

where the vertices are relabeled according to position. This rule set generates arbitrarily long chains of vertices labeled A_2.

Example 2.2 *If we augment to the rule set of the previous example the single rule $(A_1 \quad A_1 \quad \Rightarrow \quad A_2 - A_2)$, then we obtain a graph grammar that assembles cycles of length greater than or equal to three, as well as long open chains.*

There are highly non-trivial examples which build a variety of devices. In the general case, [4] gives an algorithm for generating a rule set (with constructive and destructive rules) which will build any graph in such a way that it is *stable*, in the following sense.

The pair (G_0, Φ) provides a 'seed' from which may grow a variety of outputs. These are called the *reachable* states for the graph assembly system. More formally, a connected graph G is *reachable* in a system (G_0, Φ) if there exists an assembly sequence $\{G_i\}_{i=0}^k$ of (G_0, Φ) such that G is isomorphic to some component of G_k. The set of all such reachable graphs is denoted $\mathcal{R}(G_0, \Phi)$.

Likewise, one cares about those assemblies in the reachable set which are mechanically 'inert' and not subject to further change. A graph $G \in \mathcal{R}(G_0, \Phi)$ is *stable* if for all $G' \in \mathcal{R}(G_0, \Phi)$ there does not exist an action (r, h) on the disjoint union $G \amalg G'$ such that $r = (L, R) \in \Phi$ and $h(L) \cap V_G$ is nonempty. The set of all such stable graphs is denoted $\mathcal{S}(G_0, \Phi)$.

For systems which build graphs of unbounded size (e.g., a planar lattice), we introduce a notion of stability. An assembly sequence of graphs $\{G_n\}$ is defined to be *asymptotically stable* if, given any $R > 0$ large, there exists an $M > 0$ such that for all $n > M$, the only rules applicable to G_n act outside a fixed domain in G_n of diameter R. Such systems may not have stable elements, but rather have asymptotically stable limit sets. Clearly, all finite stable assembly sequences are asymptotically stable, as is the sequence from Example 2.1 which builds an unbounded linear chain.

3. Grammatical tiles

One would like a way to assign a geometric structure to a graph assembly system in a manner that, e.g., represents the geometry of tiles whose attachments occur along sides. We give a broad definition of a geometric realization for a graph assembly system that amounts to a mapping from the reachable set to geometric objects that is defined locally on graphs. For this note, we will always assume that the 'tiles' are compact two-dimensional Euclidean domains: three-dimensional formulations follow analogously. We begin with two examples that illustrate how one goes from the geometric system to the graph-grammar system. We then present a more general formulation for how to proceed from the grammatical to the geometric.

Example 3.1 *If we wish to repeat Ex. 2.1 geometrically, we can think of each vertex as representing a rectangular tile and each bond as representing an attachment of tile faces in an end-to-end fashion. This rule set creates arbitrarily long straight chains of tile. Note that extending this same action to the augmented rule set of Ex. 2.2 leads to tile assemblies which are cylindrical and cannot fit in the plane geometrically (though they are planar as topological graphs).*

A general definition for a *tile assembly system* consists of a graph assembly system along with a *geometric realization*: a map $\mathcal{G}$ which (1) sends labeled vertices to *tiles* — compact connected subsets of the Euclidean plane; and (2) sends edges between pairs of vertices to identifications between connected subsets of the vertex tiles. These identifications may or may not occur along 'faces' of the tiles (subsets of the boundaries). The image of a connected graph G under $\mathcal{G}$ naturally has the structure of a connected cell complex, where the individual cells are Euclidean. The map $\mathcal{G}$ thus sends $\mathcal{R}(\Phi)$ to (abstract) two-dimensional piecewise Euclidean cell complexes.

4. Planar constraints

Is it possible to 'program' the geometry of the Euclidean plane into an abstract set of tile assembly rules? Otherwise said, if we write a set of grammatical rules based on a geometric assembly system, will these rules produce only planar outputs?

A tile assembly system is said to be *planar* if the image of $\mathcal{R}(\Phi, G_0)$ under $\mathcal{G}$ consists solely of complexes isometric to subsets of the Euclidean plane. Without care, it is often the case that a geometric rule set designed for a given planar output in fact produces much more than is legal. For example, a rule set which tends to construct highly non-convex shapes can create collisions between sections of the output which are far apart in the graph structure.

Question 4.1 How can one determine if a tile assembly system forces the construction of illegal outputs?

4.1. Obstructions

There exist obstructions for planar tile assembly which manifest themselves purely on the level of the topology of the aggregate. Roughly speaking, any tile assembly system which can assemble a non-simply connected object out of ‘small’ rules is necessarily non-planar: cf. Example 3.1.

Theorem 4.1 *If an acyclic graph assembly system builds a cycle, then the system cannot be planar, no matter what the geometric realization map is.*

PROOF: It is shown in [4] that for rule sets which are acyclic, the reachable set is closed under *topological covers*. Otherwise said, such a rule set cannot distinguish between building k disjoint copies of a graph containing a cycle of length N and one copy of a graph containing a multiple cycle of length kN (obtained by splitting open and ‘chaining’ the k copies together end-to-end).

Assume that for some $G \in \mathcal{R}$, there is a cycle γ. Denote the vertices of γ by $\{v_1, \ldots, v_n\}$ where v_i is connected to v_{i-1} and v_{i+1} (mod n). The image of this cycle under $\mathcal{G}$ is a 2-d cell complex which by assumption embeds in the Euclidean plane. From the result of [4], there is a graph in $\mathcal{R}$ containing a double-cover $\widetilde{\gamma}$ of the cycle γ: a cycle obtained by cutting open and joining two copies of γ. Denote the vertices of this cover $\{v_1^0, \ldots, v_n^0, v_1^1, \ldots, v_n^1\}$ with cyclic edge connections. The map $\mathcal{G}$ sends the subgraph

$$v_1^0 - v_2^0 - \cdots - v_n^0$$

to $\mathcal{G}(\gamma)$, but with one edge-bond not included. Since $\mathcal{G}$ acts locally (acting on edges of graphs), it must send $v_n^0 - v_1^1$ to the same type of cell gluing as is sends $v_n - v_1$ in γ. This forces the tile associated to v_1^1 to lie on top of the tile for v_1^0, contradicting the planarity of the system. □

This is a very broad result, independent of the tile geometry, so long as the rules do not have cycles as right hand sides.

Obstructions to planarity arise on the geometric as well as topological levels. These obstructions can be *local* in nature (e.g., a tile assembly system which attempts to assembly more than four square tiles around a shared vertex), or, more difficult to detect, *global* (e.g., the tendency to build long non-convex protuberances which are locally planar but globally not so). Determining these global geometric obstructions is a challenging task.

4.2. Simply connected regions

For the remainder of the paper, we will restrict attention to the following classes of geometric tile assembly systems:

1. All rules are binary.
2. The alphabet is finite.
3. There is a unique asymptotically stable product.

Question 4.2 *Which connected aggregates of tiles in the plane can be constructed by a geometric tile assembly set satisfying the above assumptions?*

From Theorem 4.1, we can restrict attention to those geometric tile assembly systems which construct simply connected aggregates of tiles in the plane, since those with nontrivial cycles cannot be geometric for sets of binary (hence acyclic) rules. Note that this problem is very easy in dimension one: the rule set of Example 2.1 gives an example which has a simple geometric realization. The problem is more challenging in dimension two. It is, nevertheless, possible to write a finite set of local rules which assembles arbitrarily large simply connected regions of the plane.

Theorem 4.2 *There exists a geometric tile assembly system which assembles the entire plane as the unique asymptotically stable output. The rule set can be chosen to be constructive, with a finite set of binary rules and a finite alphabet.*

The proof is constructive and tiles the plane using square tiles which spiral about a seed. The rule set which does so is a specific example from the family of rule sets given in the next section. The analysis that the output is unique and asymptotically stable is relatively straightforward and left as an exercise.

We note that the problem is simple for finite aggregates given a large enough alphabet, since one can put a total ordering on the assembly sequence and then use a sufficiently large alphabet to force this sequential assembly. For infinite aggregates, a finite rule set is insufficient:

Theorem 4.3 *Not all simply connected aggregates of tiles in the plane can be constructed uniquely by a finite rule set with a finite alphabet.*

PROOF: This is a simple cardinality argument: there are an uncountable number of simply connected planar shapes which can be built out of (e.g., square) tiles. Only a countable number of these can be realized via finite rule sets which construct a unique limit set, since the assembly sequence gives a discrete sequence of rule applications. □

5. Building spirals grammatically

In this section, we demonstrate that it is possible to build the infinite spiral of Fig. 1 as a unique and asymptotically stable product of a finite set of binary grammatical rules for a tile assembly system. Grammatically, this rule set creates a linear chain of vertices whose corner symbols are spaced out with regularly incrementing period. The geometric realization $\mathcal{G}$ for this system attaches square tiles end-to-end except for the corner symbols, at which the tiles are glued with a turn.

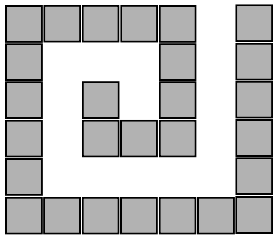


Figure 1. A graph assembly system with square tiles can assemble into an unbounded planar spiral with each tile communicating with only one neighbor.

5.1. Building spirals

There are eighteen symbols in Σ and twenty-one rules in the rule set for building the spiral of Fig. 1. The initial graph is a field of symbols with label I_0. After an initial seed-building, the rule set builds two consecutive edges of the spiral, the terminal edge labeled with A symbols and the penultimate edge labeled with B symbols. The rule set uses pairwise rules to propagate signals from the leading edge of the spiral back along the A edge and the B edge. These signals effectively count the length of the A edge and compare it to that of the B edge. This counting procedure signals when to turn a corner (T and C symbols), after which, the former B edge is converted to an F (finished) edge, which is inert to further rules. The symbols are as follows:

A_0,A_1,A_2,A_3,A_4	A (leading) edge
B_0,B_1,B_2,B_3,B_4	B (penultimate) edge
F_0,F_1	F (finished) edges
C_0,C_1,C_2,C_3	corner tiles
T_0	turn tiles
I_0	Initial tiles

In the general case of an intermediate step in the construction, there are n tiles in the terminal A edge. The number and types of B symbols used in the neighboring edge is the counter that the signaling wave uses to decide when to turn a corner. The picture is intended to suggest that there are, in total, n tiles labeled A_0 and n tiles labeled B_1.

$$\begin{tikzcd}
A_3 \arrow[r, dash] & A_0 \arrow[r, dash] & \cdots \arrow[r, dash] \arrow[rrdd, leftrightarrow, "n"] & A_0 \arrow[r, dash] & C_0 \arrow[d, dash] \\
& & & & B_1 \arrow[d, dash] \\
& & & & \cdots \arrow[d, dash] \\
& & & & B_1 \arrow[d, dash] \\
& & & & B_0 \arrow[d, dash] \\
& & & & \cdots \arrow[d, dash] \\
& & & & B_0 \arrow[d, dash] \\
& \cdots \arrow[r, dash] & F_0 \arrow[r, dash] & F_0 \arrow[r, dash] & F_1
\end{tikzcd}$$

The rule set is as follows:

1 2	$I_0 \quad I_0 \Rightarrow T_0 – F_1$ $I_0 \quad T_0 \Rightarrow A_3 – C_0$	Initialization Attach about a corner
3	$I_0 \quad A_3 \Rightarrow A_4 – A_1$	Attach in-line
4 5	$A_1 – A_0 \Rightarrow A_1 – A_1$ $A_1 – C_0 \Rightarrow A_1 – C_1$	Propagate *'Attach new piece'* backward along the A edge
6 7 8	$A_1 – C_2 \Rightarrow A_2 – C_0$ $A_1 – A_2 \Rightarrow A_2 – A_0$ $A_4 – A_2 \Rightarrow A_3 – A_0$	Propagate *'OK to continue'* forward along the A edge
9 10 11	$A_1 – C_3 \Rightarrow B_0 – F_1$ $A_1 – B_0 \Rightarrow B_0 – B_0$ $A_4 – B_0 \Rightarrow T_0 – B_0$	Propagate *'Turn a corner'* forward along the A edge
12	$C_1 – B_0 \Rightarrow C_2 – B_1$	Signal *'Continue'* to B edge
13	$C_1 – F_1 \Rightarrow C_3 – F_1$	Signal *'Halt'* to B edge
14 15	$C_1 – B_1 \Rightarrow C_1 – B_2$ $B_2 – B_1 \Rightarrow B_2 – B_2$	Maybe room to continue; Check backward for any B_0
16 17 18	$B_2 – B_0 \Rightarrow B_3 – B_1$ $B_2 – B_3 \Rightarrow B_3 – B_1$ $C_1 – B_3 \Rightarrow C_2 – B_1$	Yes, room to continue; Propagate *"OK to continue"* forward along the B edge
19 20 21	$B_2 – F_1 \Rightarrow B_4 – F_1$ $B_2 – B_4 \Rightarrow B_4 – F_0$ $C_1 – B_4 \Rightarrow C_3 – F_0$	No, not room to continue; Propagate *"Turn corner"* forward along the B edge

Here are two snapshots showing the addition of a tile.

$$\begin{tikzcd}
& A_3 \arrow[r, dash] & A_0 \arrow[r, dash] & A_0 \arrow[r, dash] & C_0 \arrow[d, dash] \\
& & & & B_1 \arrow[d, dash] \\
F_1 \arrow[r, dash] \arrow[d, dash] & F_0 \arrow[r, dash] & F_1 \arrow[d, dash] & & B_1 \arrow[d, dash] \\
F_0 \arrow[d, dash] & & F_1 & & B_0 \arrow[d, dash] \\
F_0 \arrow[d, dash] & & & & B_0 \arrow[d, dash] \\
F_1 \arrow[r, dash] & F_0 \arrow[r, dash] & F_0 \arrow[r, dash] & F_0 \arrow[r, dash] & F_1
\end{tikzcd}$$

$$\begin{tikzcd}
A_3 \arrow[r, dash] & A_0 \arrow[r, dash] & A_0 \arrow[r, dash] & A_0 \arrow[r, dash] & C_0 \arrow[d, dash] \\
& & & & B_1 \arrow[d, dash] \\
F_1 \arrow[r, dash] \arrow[d, dash] & F_0 \arrow[r, dash] & F_1 \arrow[d, dash] & & B_1 \arrow[d, dash] \\
F_0 \arrow[d, dash] & & F_1 & & B_1 \arrow[d, dash] \\
F_0 \arrow[d, dash] & & & & B_0 \arrow[d, dash] \\
F_1 \arrow[r, dash] & F_0 \arrow[r, dash] & F_0 \arrow[r, dash] & F_0 \arrow[r, dash] & F_1
\end{tikzcd}$$

The ease of building these spirals stems from the fact that one need merely "count" the number of tiles on the penultimate edge of the spiral and increment this by two. A slight change in the rule set increments by one instead of two, yielding a tiling of the plane and a proof of Theorem 4.2.

Looser spirals can be constructed by incrementing some fixed large number before turning a corner. However, it follows from the proof of Theorem 4.3 that not all infinite spirals can be built in this manner, since there are an uncountable number of possible sequences of arm-lengths of the spiral. Only those sequences which are "computable" have the possibility of being constructible.

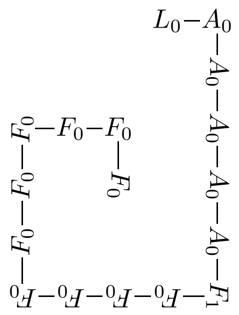
5.2. Asynchronous construction

The use of sequential propagating waves between the A and B edges in the spiral rule set presented gives a relatively slow construction: each additional tile added to the spiral requires waiting for several waves to propagate one at a time down arms of increasing length. The time (measured by the number of rules sequentially applied) needed to add a new edge to the spiral of length N is quadratic in N. By changing the rule set to make the waves of communication non-sequential we may speed up the construction.

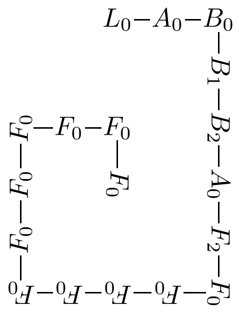
1	$I_0 \quad I_0 \Rightarrow T_0{-}F_1$	Initialization
2	$I_0 \ \ T_0 \Rightarrow L_0{-}A_0$	Initialization
3	$A_0{-}B_0 \Rightarrow B_0{-}B_1$	Propagate
4	$B_1{-}B_2 \Rightarrow B_2{-}A_0$	the B wave
5	$A_0{-}B_0 \Rightarrow B_0{-}B_1$	Propagate B wave
6	$B_1{-}B_2 \Rightarrow B_2{-}A_0$	around corner
7	$A_0{-}C_0 \Rightarrow C_0{-}C_1$	Propagate
8	$C_1{-}C_2 \Rightarrow C_2{-}A_0$	the C wave
9	$A_0{-}F_1 \Rightarrow B_0{-}F_1$	Advance F edge
10	$B_1{-}F_1 \Rightarrow B_2{-}F_2$	while starting
11	$A_0{-}F_2 \Rightarrow F_1{-}F_0$	a B wave
12	$A_0{-}F_1 \Rightarrow B_0{-}F_1$	Advance F edge
13	$B_1{-}F_1 \Rightarrow B_2{-}F_2$	around a corner
14	$A_0{-}F_2 \Rightarrow C_0{-}F_2$	while starting
15	$C_1{-}F_2 \Rightarrow C_2{-}F_3$	a B wave and then
16	$A_0{-}F_3 \Rightarrow F_1{-}F_0$	a C wave
17	$L_0{-}B_0 \Rightarrow L_1{-}B_1$	Add a new tile
18	$I_0 \ \ L_1 \Rightarrow L_0{-}B_1$	in response to
19	$L_0{-}B_2 \Rightarrow L_0{-}A_0$	a B wave
20	$L_0{-}C_0 \Rightarrow L_1{-}C_1$	Turn corner in
21	$B_1{-}C_2 \Rightarrow A_0{-}A_0$	response to C wave

This rule set is a "compressed" version of this scheme. We have reduced the number of rules and size of the alphabet by encoding the geometric realization $\mathcal{G}$ into the alphabet itself: vertices are represented with an orientation determining the bond geometry. While this is not consistent with the rules of being a strictly grammatical construction, it is possible to expand the alphabet and rule set to recover a true graph assembly system.

Rules 1 and 2 are used to create an initial graph of three vertices. The following frames show the second rule set in action. The leading edge has just finished responding to a "C" wave by turning the corner, and the trailing edge has not yet started any of the waves that will eventually cause the side on top to be built.



Since the last frame, the leading edge has completely responded to one "B" wave, and another "B" wave is propagating up the side.



In the best case, the time to build an edge of the spiral of length N is linear in N; in the worst case, in which all rules are applied with the sequential protocol of the prior subsection, it is quadratic in N. Otherwise said, in the optimal case, it is constant time to add the next tile, independent of location in the spiral. Since this is not a deterministic system, we cannot force the best-case scenario; however, we suspect that randomized asynchronous execution would yield the best-case run times.

6. Concluding remarks

We have presented a formalism for ascribing geometric content to a grammatical assembly scheme which formerly contained only topological (network) content. We then demonstrated that the problem of writing rule sets which are consistent with the geometry of the plane is possible (even in the case of building infinite spiral structures) but by no means simple. However, there are simple obstructions computable from the topology and geometry of the outputs: unless there are nontrivial cycles of communication, any rule set which builds a cycle in the plane will also want to build non-planar objects as well.

Acknowledgments

Ghrist is supported in part by NSF PECASE DMS-0337713. Lipsky is supported in part by NSF VIGRE DMS-9983160.

References

[1] B. Berger, P. Shor, L. Tucker-Kellogg, and J. King. Local rule-based theory of virus shell assembly. *Proceedings of the National Academy of Science, USA*, 91(6):7732–7736, August 1994.

[2] N. Bowden, A. Terfort, J. Carbeck, and G. M. Whitesides. Self-assembly of mesoscale objects into ordered two-dimensional arrays. *Science*, 276(11):233–235, April 1997.

[3] B. Courcelle. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, chapter on Graph Rewriting: An Algebraic and Logic Approach, pages 193–242. MIT Press, 1990.

[4] E. Klavins, R. Ghrist and D. Lipsky. Graph grammars for self assembling robotic systems. In *Proc. Intl. Conf. Robotics & Automation*, 2004.

[5] H. Ehrig. Introduction to the algebraic theory of graph grammars. In V. Claus, H. Ehrig, and G. Rozenberg, editors, *Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69, 1979.

[6] H. O. Jacobs, A. R. Tao, A. Scwartz, D. H. Gracias, and G. M. Whitesides. Fabrication of a cylindrical display by patterned assembly. *Science*, 296:4763–4768, April 2002.

[7] C. Jones and M. J. Matarić. From local to global behavior in intelligent self-assembly. In *International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[8] E. Klavins. Automatic synthesis of controllers for distributed assembly and formation forming. In *Proceedings of the IEEE Conference on Robotics and Automation*, Washington DC, May 2002.

[9] E. Klavins. Automatically synthesized controllers for distributed assembly: Partial correctness. In S. Butenko, R. Murphey, and P. M. Pardalos, editors, *Cooperative Control: Models, Applications and Algorithms*, pages 111–127. Kluwer, 2002.

[10] D. Lammers. Motorola speeds the move to nanocrystal flash. *EE Times*, December 8, 2003.

[11] N. Mokhoff. Crystals line up in IBM flash chip. *EE Times*, December 8, 2003.

[12] K. Saitou. Conformational switching in self-assembling mechanical systems. *IEEE Transactions on Robotics and Automation*, 15(3):510–520, 1999.

[13] Y. S. Smentanich, Y. B. Kazanovich, and V. V. Kornilov. A combinitorial approach to the problem of self assembly. *Discrete Applied Mathematics*, 57:45–65, 1995.

[14] R. L. Thompson and N. S. Goel. Movable finite automata (MFA) models for biological systems I: Bacteriophage assembly and operation. *Journal of Theoretical Biology*, 131:152–385, 1988.

[15] T. Toth-Fejel. Modeling Kinematic Cellular Automata. NASA Institute for Advanced Concepts Phase I: CP-02-02. General Dynamics Advanced Information Systems Contract # P03-0984. April 30, 2004.

[16] H. Wang. Notes on a class of tiling problems. *Fundamenta Mathematicae*, pages 295–305, 1975.

[17] G. M. Whitesides and B. Grzybowski. Self assembly at all scales. *Science*, 295:2418–2421, March.

[18] E. Winfree. Algorithmic self-assembly of DNA: Theoretical motivations and 2D assembly experiments. *Journal of Biomolecular Structure and Dynamics*, 11(2):263–270, May 2000.