Solving Coverage Problems with Embedded Graph Grammars

John-Michael McNew[1], Eric Klavins[1], and Magnus Egerstedt[2]

[1] Univ. of Washington {jmmcnew,klavins}@ee.washington.edu
[2] Georgia Institute of Technology magnus@ece.gatech.edu

Abstract. We show how Embedded Graph Grammars (EGGs) are used to specify local interaction rules between mobile robots in a natural manner. This formalism allows us to treat local network topologies, geometric transition conditions, and individual robot dynamics and control modes in a unified framework. An example EGG is demonstrated that achieves sensor coverage in a provably stable and correct manner. The algorithm results in a global network with a lattice-like triangulation.

1 Introduction

The overarching scientific question facing the area of networked robot systems is how global behaviors arise from local rules in a well-defined and predictable manner. In a dynamic task, as agents move in and out of each others sensory or communication ranges, the network changes, resulting in inherently hybrid dynamics. Hybrid dynamics also result from the fact that individual agents can assume different roles depending on local network adjacency or on the geometrical characteristics of the local environment. Networked robot systems require methods that control both the network dynamics and the individual robot behaviors. This suggests a handful of potential formalisms [1, 2]. Unfortunately, the available languages are not appealing for networked systems because the local network topology associated with each robot is not cleanly represented. Hybrid automata [3, 4], while certainly expressive enough, are cumbersome in this setting. They additionally suffer from the state explosion problem since a distinct state typically has to be enumerated for every possible combination of role assignments and network topologies.

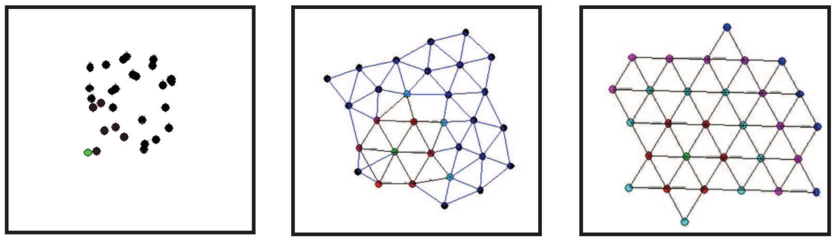


Fig. 1. Simulated formation of a δ-triangulation using an embedded graph grammar.

Thus, we take as our starting point *graph grammars* which provide a compact representation of subgraph transitions arising from local interactions. Although graph grammars have been successfully used to control robot systems [5], they best describe changing networks, not, for example, low-level robot motion. To address this, we augment the notion of a graph grammar, introducing the *Embedded Graph Grammar*(EGG) model and associated analysis methods.

To demonstrate and challenge the approach, we extend the geometric problem of *coverage* (see also [6]) to include specifications on network topology. In particular, we suppose that three communicating robots forming an equilateral triangle of side-length δ *cover* the convex hull of their locations. The goal (described more formally in Section 3) is to produce a triangulation of an entire region in which any three nearby robots form a δ-triangle and there are no holes in the resulting mesh (see Figure 1). Similar lattice-like geometries are generated by virtual potential leaders in [7] where the system shares a global coordinate system. What is novel here is (1) the EGG formalism that encodes the system dynamics, local network topology, guard conditions for switching between control modes, and inter-robot communication rules in a unified manner; (2) the lack of a common global reference frame among the robots; and (3) the superposition and successful coordination of several complementary control objectives.

2 Embedded Graph Grammar Definitions

2.1 Graphs and other Notation

If Σ is a set of labels, a labeled graph is the quadruple (V, E, l, e) where V is a set of vertices, E is a set of edges, l is a vertex labeling function that maps vertices into Σ, and e is an edge labeling function that maps edges into Σ. We denote by $\mathcal{G}$ the space of labeled graphs. Sometimes the label set Σ is a cartesian product of atomic label spaces we refer to as *fields*. For example the system introduced in this paper has a node label space where the fields are named $(mode, dist)$. We use dot notation to indicate the values of label fields for specific robots. For instance $i.dist = 10$ indicates that robot i has the value 10 in its $dist$ field.

If S is a set of vertices, $G[S]$ denotes the subgraph of G induced by S. We denote by V_G and E_G the edges of a graph G and when there is no danger of confusion we write V to indicate the vertex set of the system under consideration. If i and j are vertices in a graph, we denote their graph distance by $d(i, j)$.

If f is a function defined on a domain A, we denote the restriction of the function to $B \subset A$ by $f_{|_B}$. A function f is *well-defined* with respect to an equivalence relation $\sim$ if $x \sim y$ implies $f(x) = f(y)$.

2.2 Embedded Graphs

Consider a system of N communicating robots with identical state space X.

Definition 1. *An embedded graph γ is a pair $\gamma = (G, x)$ where G is a labeled graph and $x : V \to X$ is a* realization function. *The space of all embedded graphs is denoted by Γ.*

A vertex $i \in V$ indicates the index of the *ith* robot. The presence of an edge $ij \in E$ corresponds to a physical and/or *maintained* communication link between robots i and j. The vertex label indicates the operational mode of robot i and (along with the edge label $e(ij)$) keeps track of local information. The function x assigns to each robot a continuous state or *realization* in its state space X.

We write $G_\gamma, x_\gamma, V_\gamma$, and E_γ to denote the labeled graph, continuous state, vertices, and edges associated with an embedded graph γ. If $S \subseteq V_\gamma$, then the the embedded graph induced by S, $\gamma[S]$, is given by the pair $(G[S], x_{|S})$. We define the distance between two embedded graphs, γ and ρ, by

$$d(\gamma,\rho) \triangleq \begin{cases} \infty & \text{if } G_\gamma \neq G_\rho \\ ||x_\gamma - x_\rho|| & \textit{otherwise.} \end{cases}$$

The distance between an embedded graph γ and a set of embedded graphs T is denoted by $d(\gamma, T) = \min_{\rho \in T} d(\gamma, \rho)$.

2.3 Embedded Graph Transition Systems

A trajectory of an *embedded graph transition system* describes how the network topology and continuous states of a group of robots changes over time.

Definition 2. *An* embedded graph transition relation *is a relation $\mathcal{A} \subseteq \Gamma \times \Gamma$ such that $(\gamma_1, \gamma_2) \in \mathcal{A}$ implies $x_{\gamma_1} = x_{\gamma_2}$.*

Definition 3. *An* embedded graph transition system *is a triple $(\gamma_0, \mathcal{A}, u)$ where γ_0 is an initial embedded graph and $u : V \times \Gamma \to TX$ is the vector field describing the continuous flow.*

Definition 4. *A* trajectory *is a map $\sigma : \mathbb{R}^{\geq 0} \to \Gamma$ such that there exists a sequence $\tau_0, \tau_1, \tau_2, ...$ where*

1. *$x_{\sigma(t)}$ is continuous.*
2. *$\tau_k \leq \tau_{k+1}$ and if the sequence has any finite length N, $\tau_N \triangleq \infty$.*
3. *For all $t, t' \in [\tau_k, \tau_{k+1})$, $G_{\sigma(t)} = G_{\sigma(t')}$.*
4. *$\tau_i \neq \infty$ and $i > 0$ if and only if there exists a transition$((G, x^*), (H, x^*)) \in \mathcal{A}$ such that $(G, x^*) = \lim_{t \to \tau_i} \sigma(t)$ and $\sigma(\tau_i) = (H, x^*)$.*
5. *For all $i \in V$ and $t \in [\tau_i, \tau_i + 1)$, $\frac{d}{dt} x_{\sigma(t)}(i) = u(i, \sigma(t))$.*

We denote the set of nondeterministic trajectories of a system by $\mathcal{T}(\gamma_0, \mathcal{A}, u)$. Clearly an embedded graph transition system is a *globally* defined non-deterministic hybrid automata where the discrete states are labeled graphs.

2.4 Locality and Embedded Graph Grammars

Our interest is in modeling and implementing embedded graph transition systems in a *local* and *distributed* fashion. In other work [8, 9], we model notions of "local" that include geometric restrictions on sensing and communications. In this paper we focus exclusively on the notion of local graph neighborhoods and we develop a model where:

1. Graph matching involves only a small subset of vertices.
2. The flow and transition relations use discrete information from the graph neighborhood.
3. The flow and transition relation are permutation-invariant.

We refer to the neighborhood of $i \in V_\gamma$ as the *friends* of i and denote this set by $F(i)$. In keeping with existing graph literature, we write $F[i]$ to mean $F(i) \cup i$ and denote the closed out-neighborhood by $F^+[i]$.

Definition 5. *Consider the pairs* (A, γ) *and* (B, ρ) *where* γ *and* ρ *are embedded graphs,* $A \subset V_\gamma$*, and* $B \subset V_\rho$*.* $(A, \gamma) \sim (B, \rho)$ *if there exists a bijective map* ν *between* V_γ *and* V_ρ *such that*

1. *For all* $i \in A$*,* $\nu(i) \in B$*,*
2. *For all* $k \in V_\gamma$*,* $x_\gamma(k) = x_\rho(\nu(k))$*, and*
3. *For all* $i \in A$*,* ν *is a label preserving isomorphism between* $G_\gamma[F[i]]$ *and* $G_\rho[F[\nu(i)]]$*.*

If $(A, \gamma) \sim (B, \rho)$ *we say* γ from the point of view of A is equivalent to ρ from the point of view of B .

Definition 6. *A control law* $u : V \times \Gamma \to TX$ *is* locally implementable *if* u *is well defined with respect to the point of view equivalence relation* $\sim$*.*

This definition captures the requirement that robots use discrete information from their local graph neighborhood in a permutation invariant manner. Robots form new links and update their labels on a *local* scale by using *guarded rules.*

Definition 7. *A* guard g *is a function* $g : \mathcal{P}(V) \times \Gamma \to \{true, false\}$*. A guard is* locally checkable *if it is well-defined with respect to the point of view equivalence relation on sets,* $\sim$*.*

Definition 8. *A* guarded rule *(or just rule),* $r = (g, L, R)$*, is a pair of labeled graphs over some* small vertex set $V_L = V_R$ *and a locally checkable guard* g*.*

Figure 4 shows an example of a guarded rule. Each breakout box contains a dummy variable on the left used to identify specific vertices in a rule. The right hand side of the box contains the labeling of that vertex. When the topology and labeling of a small group of robots matches that of the left hand graph of the rule and the robots are in a "safe" configuration (as defined by the guard), then they can update their state to match the right hand graph in the rule.

More formally suppose γ represents a possible state of a system and h is a label preserving subgraph isomorphism from V_L into G_γ such that $g(h(V_L), \gamma)$ is true. We call h a witness and the pair (r, h) an action. A rule r is *applicable* if a witness h can be found. The application of an action (r, h) on an embedded graph $\gamma = (G, x)$ produces a new embedded graph $\gamma' = ((V, E', l', e'), x)$ defined by

$$E' = (E - \{h(i)h(j) | ij \in E_L\}) \cup \{h(i)h(j) \mid ij \in E_R\}$$
$$l'(i) = \begin{cases} l(i) \text{ if } i \notin h(V_L) \\ l_R \circ h^{-1}(i) \text{ otherwise.} \end{cases} \qquad e'(ij) = \begin{cases} e(ij) \text{ if } i \notin h(V_L) \text{ or } j \notin h(V_L) \\ e_R \circ h^{-1}(i)h^{-1}(j) \text{ otherwise.} \end{cases}$$

That is, we replace $h(L)$ (which is a copy of L) with $h(R)$ in the graph G_γ. We write $\gamma \xrightarrow{r,h} \gamma'$ to denote that we obtain $G'_{\gamma'}$ from G_γ by applying action (r, h).

Definition 9. *An embedded graph transition $((G, x), (H, x))$ is* consistent *with a rule r if there exists a witness h such that (r, h) is applicable to (G, x) and $G \xrightarrow{r,h} H$. We denote by $\mathcal{A}(r)$ the set of transitions consistent with rule r. If Φ is a set of rules, $\mathcal{A}(\Phi) = \cup_{r \in \Phi} \mathcal{A}(r)$.*

Definition 10. *An* embedded graph grammar system (EGG) *is a triple (γ_0, Φ, u) where γ_0 is an embedded graph representing the initial state, Φ is a set of rules, and u is a locally implementable controller.*

Proposition 1. *The set of trajectories of a local embedded graph grammar, $\mathcal{T}(\gamma_0, \Phi, u)$ are equivalent to the trajectories of an embedded graph transition system under the transition relation consistent with Φ, $\mathcal{A}(\Phi)$, and the vector field described by the locally implementable controller u, i.e.*

$$\mathcal{T}(\gamma_0, \Phi, u) = \mathcal{T}(\gamma_0, \mathcal{A}(\Phi), u).$$

Figure 2 shows an embedded graph grammar trajectory. Note that discrete transitions involve small sets of vertices and that concurrent application of rules is possible.

3 Coverage via δ-triangulation

3.1 Preliminaries

By a *triangle*, we mean any subgraph composed of three fully connected vertices (say i, j, k). A *δ-triangle* is a triangle where $||x_i - x_j|| = ||x_j - x_k|| = ||x_i - x_k|| = \delta$. We say that a region of the plane $A \subset \mathbb{R}^2$ is *covered* if A lies within the convex hull of the positions of three robots in a δ-triangle.

A graph G is *planar* if there exists an embedding $x : V \to \mathbb{R}^2$ such that when the edges are drawn as straight lines in the plane, no edges intersect except at vertices. An embedded graph $\gamma = (G, x)$ is a *plane graph* when G is planar via the realization function x. The regions of a plane graph bounded by the edges are called *faces* and every finite planar graph has exactly one unbounded face called the *outer face*.

Definition 11. *An plane graph γ is a* near-triangulation *if the outer face is a cycle and all inner faces are triangles.*

Definition 12. *A near-triangulation γ is a* δ-triangulation *if every inner face corresponds to a δ-triangle. We say a δ-triangulation γ is* maximal *if for every graph H created by adding an edge to G_γ, there does not exist a realization x such that (H, x) is a δ-triangulation.*

We denote by T the set of all δ-triangulations and by T_{max} the set of all maximal δ-triangulations.

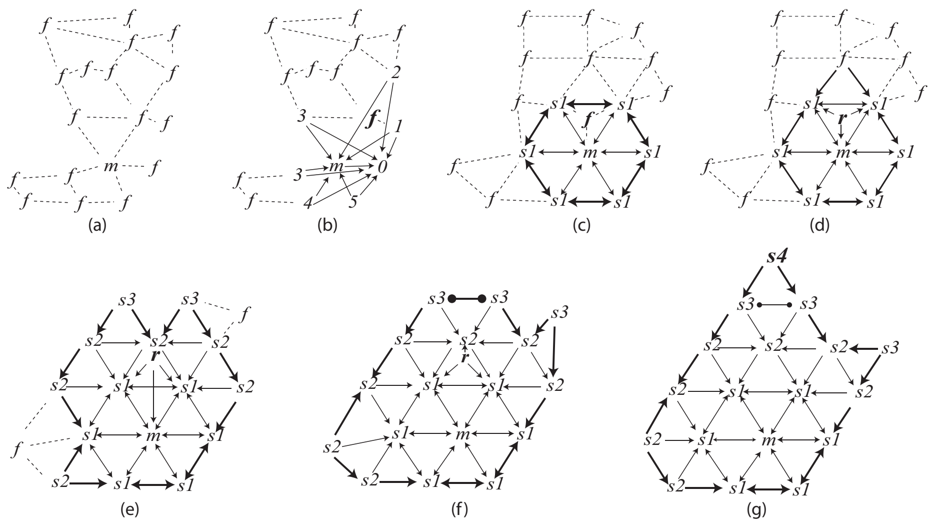


Fig. 2. A Sample Trajectory. (a) Initial state all `free` robots, f, are running the Gabriel graph pre-sorting controllers (indicated by dashed edges). (b) Hexagonal seed formation. (c) After the hexagonal seed is formed by R_2, the robot in bold labeled **f** is trapped inside the hexagon. (d) To correct this "error", rule R_6 changes the robot to `repulse` mode, **r**. (e)-(f) Utilizing the `repulsing` controller, the robot in bold moves away from the center. Also Rules R_5 and R_6 of the crystal growth process are applied. (g) By repeatedly applying the error correction rules, the robot in bold works its way out of the crystal where it can be absorbed as the final element in the structure.

3.2 General Algorithm

Consider N robots moving in the plane without a common reference frame and obeying integrator dynamics given by $\dot{x}_i = u$. Suppose Γ_0 is the class of initial embedded graphs where: (1) $E_{\gamma_0} = \varnothing$, (2) there is a unique vertex 0 labeled $l(0).mode =$ `master` and (3) all other vertices are labeled by $l(i).mode =$ `free`.
Task. *Design an embedded graph grammar,* (γ_0, Φ, u) *such that for all trajectories* $\sigma \in \mathcal{T}(\gamma_0, \Phi, u)$

$$\lim_{t \to \infty} \sigma(t) \in T_{max}.$$

Our method of constructing δ-triangulations is similar to that of growing "crystals" where placing a "seed crystal" in a "solution" causes the solution to replicate the seed and crystalize. We often view EGGs as collections of small behaviors interacting locally, a fact highlighted by our δ-triangulation grammar, Φ, which describes the four concurrent processes shown in Figure 2:

1. *Pre-sorting*–The robots not involved in the other three processes try (without communication) to organize into a mesh that is *close* to a δ-triangulation.
2. *Hexagon Formation*–The `master` chooses six robots and creates a hexagonal "seed" formation, labeling all edges on the boundary by $ij.boundary =$ `outer`.

3. *Crystal Growth*–The δ-triangulation or "crystal" is grown by waiting until robots in a boundary edge are near settling, then a robot on the exterior of the crystal "attaches" to these robots to form a new triangle or two robots already in the crystal add an edge to enclose a δ-triangle.
4. *Error Correction*–Occasionally robots executing the pre-sorting algorithm become trapped in the interior of the crystal structure. The error correction controller routes these robots to the exterior using local label information.

Purpose	Label	Control
Pre-Sort	f	$-\sum_{ij\in Gabriel(i)} \nabla_{x_i} U_{ij}$
Hexagon	m	0
Formation	0	$-\sum_{ij\in F^+(i)} \nabla_{x_i} U_{ij}$
	$1-5$	$-\ \nabla_{x_i}(\|\|x_i - P(i)\|\|)^2$
Crystal Growth	s	$-\sum_{ij\in F^+(i)\cap Con(i)} \nabla_{x_i} U_{ij}$
Error Correction	r	$-\ \nabla_{x_i}(\|\|x_i - p(i)\|\|)^2$

Table 1. Control Law u for the δ-triangulation solution grammar. Robots execute the control law corresponding to their label.

Suppose Φ_{HEX} is the grammar for hexagon formation (Figure 3), Φ_{CG} is the grammar for crystal growth (Figure 4),Φ_{EC} is the grammar for error correction (Figure 6), and the solution grammar is $\Phi = \Phi_{HEX} \cup \Phi_{CG} \cup \Phi_{EC}$. The following notation is useful in describing the solution system (γ_0, Φ, u). Fix a trajectory σ and define $\gamma(t) = \gamma_{\sigma(t)}, x(t) = x_{\gamma(t)}, G(t) = G_{\gamma(t)}$. By $r(t)$ and $h(t)$ we denote the rule and witness applied at time t. The grammar Φ uses two vertex label fields, *mode* and *dist*(the graph distance from the `master`) and two edge label fields, *boundary* and *control*. The mode labels are: `master`,`0`, `1`,..., `5`, `slave`, `repulse`, and `free`. Figures refer to mode labels by their first letter.

The function $Con(V)$ takes any set of vertices V and returns another set of vertices V' where V' is the set of vertices reachable from V via paths where every edge is labeled as a `control` edge. The *crystal* function $C(t)$ is defined as the restriction of the embedded graph $\gamma(t)$ to the set of vertices labeled *`slave`* or *`master`* and defines the growing crystalline structure. We define the *interior* function $I(t)$ as the closed union of the faces in the crystal $C(t)$. We denote by $d(i, l_{jk})$ the distance of x_i from the line l_{jk} through x_j and x_k. If l_{jk} defines a half plane in $\mathbb{R}^2$, we denote by $H^i_l(j,k)$ the region of state space where i lies in the half plane opposite l.

3.3 Pre-Sorting using Gabriel Graphs

We use a geometric switching algorithm by Schucker, Murphey and Bennet [10] to produce *near* δ-triangulations. The algorithm uses the edges of the Gabriel graph as a switching criterion and does not require explicit communication.

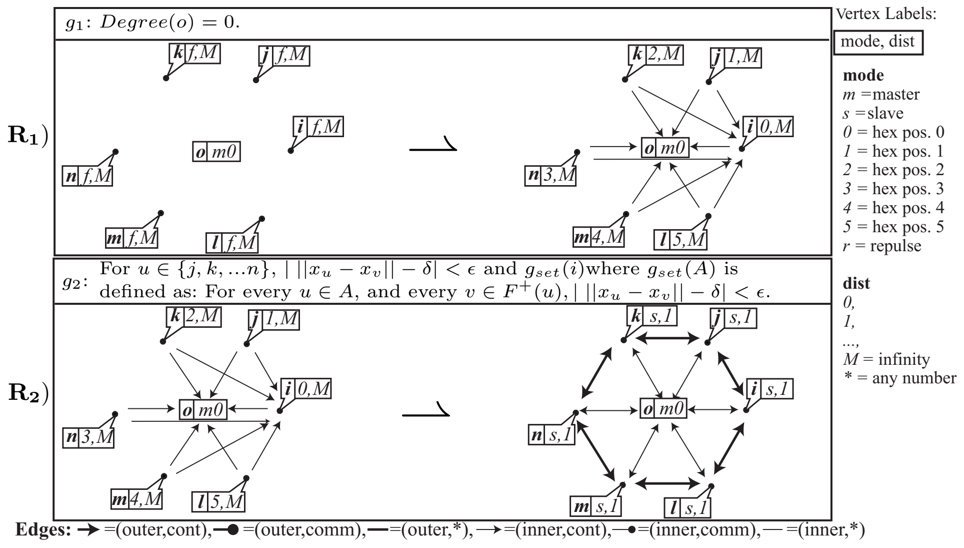

g_1: $Degree(o) = 0$.
g_2: For $u \in \{j, k, ...n\}$, $| \, ||x_u - x_v|| - \delta| < \epsilon$ and $g_{set}(i)$where $g_{set}(A)$ is defined as: For every $u \in A$, and every $v \in F^+(u)$, $| \, ||x_u - x_v|| - \delta| < \epsilon$.
R1)
R2)
k f,M
j f,M
i f,M
o m0
n f,M
m f,M
l f,M
k 2,M
j 1,M
i 0,M
n 3,M
m 4,M
l 5,M
k s,1
j s,1
i s,1
n s,1
m s,1
l s,1
Vertex Labels:
mode, dist
mode
m =master
s =slave
0 = hex pos. 0
1 = hex pos. 1
2 = hex pos. 2
3 = hex pos. 3
4 = hex pos. 4
5 = hex pos. 5
r = repulse
dist
0,
1,
...,
M = infinity
* = any number
Edges: =(outer,cont), =(outer,comm), =(outer,*), =(inner,cont), =(inner,comm), =(inner,*)

Fig. 3. Rule set Φ_{HEX} for hexagonal seed formation. Rule R_1 establishes the topology used to move the robots into a hexagonal configuration. Once the robots are in the attracting region, rule R_2 switches to the desired topology and control modes

Definition 13. *Given a realization x, we denote the Gabriel graph of x by $Gabriel(x)$, where $ij \in E_{Gabriel(x)}$ if and only if for all $k \neq i$ or j,*

$$||x_i - x_j||^2 < ||x_i - x_k||^2 + ||x_j - x_k||^2.$$

For an edge ij we define an edge based potential $U_{ij} = (||x_i - x_j|| - \delta)^2$. As seen in Table 1, `free` robots follow the negative gradient of U_{ij} for Gabriel graph edges ij, trying to make the distance between those robots δ. The first few panels of Figure 2 show the Gabriel graph controller pre-sorting the robots.

3.4 Hexagon Formation

By a *hexagonal seed graph*, we mean any embedded graph μ where $G_\mu = W_7$ is a wheel graph and x_μ is a hexagonal configuration with edge lengths of δ (i.e. $\mu \in T_{max}$). Forming the hexagonal seed is challenging because the robots lack of a common coordinate system. When rule R_1 is applied, the robot corresponding to vertex i in Figure 3 changes to $i.mode =$ `0`. The `master` robot corresponds to vertex o. The controller $\dot{x}_i = \nabla_i U_{io}$ limits the motion of i to a linear manifold defined by the configuration of i and o when the rule is applied. Since the vector $\mathbf{v}_{oi}$ is constant, the other robots (labeled by $mode \in \{1, 2, ...5\}$) form edges to robots 0 and i and use $\mathbf{v}_{oi}$ as the basis of a shared local coordinate system. Then each vertex v constructs a sink point $P(v)$ defined by their mode labels where

$$P(v) = x_o + \delta(cos(\frac{\pi}{3}v.mode), sin(\frac{\pi}{3}v.mode))^T.$$

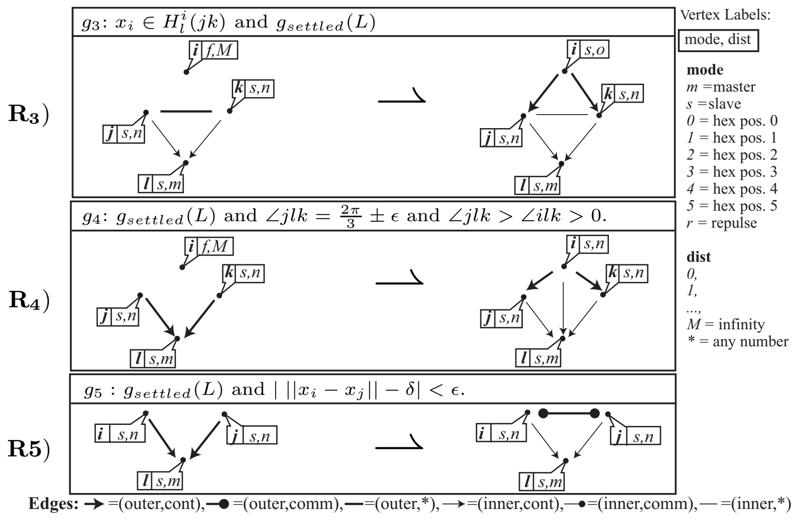

g_3: $x_i \in H^i_l(jk)$ and $g_{settled}(L)$
g_4: $g_{settled}(L)$ and $\angle jlk = \frac{2\pi}{3} \pm \epsilon$ and $\angle jlk > \angle ilk > 0$.
g_5 : $g_{settled}(L)$ and $| \, ||x_i - x_j|| - \delta| < \epsilon$.
R3)
R4)
R5)
i f,M
k s,n
j s,n
l s,m
i s,o
i s,n
Vertex Labels:
mode, dist
mode
m =master
s =slave
0 = hex pos. 0
1 = hex pos. 1
2 = hex pos. 2
3 = hex pos. 3
4 = hex pos. 4
5 = hex pos. 5
r = repulse
dist
0,
1,
...,
M = infinity
* = any number
Edges: =(outer,cont), =(outer,comm), =(outer,*), =(inner,cont), =(inner,comm), =(inner,*)

Fig. 4. Rule set Φ_{CG} for Crystal Growth. Rules R_3 and R_4 add robots to the crystal. Rule R_5 encloses δ-triangulations.

Using the simple potential controllers shown in Table 1, the robots converge towards the hexagonal configuration defined by P. Once the robots' geometry is close enough to hexagonal, the robots apply rule R_2 to switch to the δ-triangulation topology. Proposition 2 states a hexagonal seed is always formed and is proven (along with other supporting propositions) in Section 4.2. Figure 2(a)-(c)shows hexagon formation in a partial trajectory.

Proposition 2. *For any ϵ, if $\gamma(t_0) = \gamma_0$, then there exists $t_2 > t_0$ and a δ-triangulation $\mu \in T$ where $G_\mu = W_7$ such that for all $t \geq t_2$, $d(C(t)[V_\mu], \mu) < \epsilon$.*

3.5 Crystal Growth

Consider any crystal $C(t)$ and an edge labeled *jk.boundary* = `outer` (we call these edges "boundary" edges). The controllers grow the crystal by attaching a "`free`" robot from the exterior (say i) to j and k and updating the boundary labeling. The rules in Figure 4 address two different geometries. In rule R_3, if *j.dist* = *k.dist*, then i forms edges with j and k , marking the new edges *outer* and labeling *i.dist* = *j.dist* + 1 and *i.mode* = `slave`. If applied in g_3, the controller for the `slave` label shown in Table 1 moves i to a point where $||x_i - x_j|| = ||x_i - x_k|| = \delta$ corresponding to a δ-triangulation. Figure 5 shows the guards of R_3 and the other crystal growth rules and their applications. The unidirectional information flow from the hexagonal seed outward guarantees that the crystal does not deform when a new robot is added.

Rule R_4 is applied at the corners of the hexagon where there are two boundary edges jl and kl with *j.dist* = *k.dist* = *l.dist* + 1. When $\angle jlk \approx \frac{2\pi}{3}$, R_4 adds i between j and k and updates the distance by *i.dist* = *j.dist*.

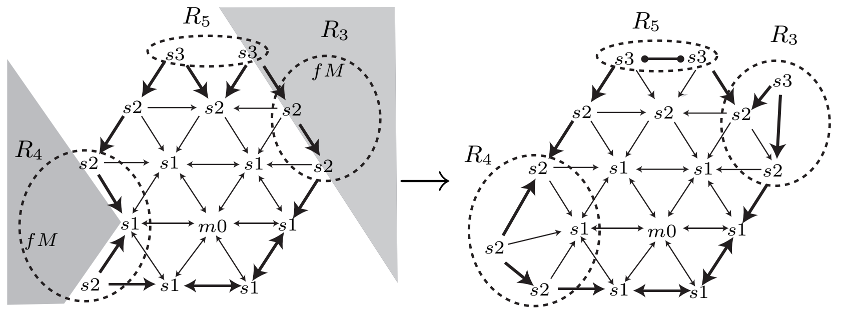


Fig. 5. Applications of Crystal Growth Rules. The gray areas indicate the regions in which the guards g_3 and g_4 are true. R_3 and R_4 add vertices to the crystal. R_5 closes a δ-triangulation. Note that after the application of the rules, the new labeling of the outer boundary remains consistent with the geometry.

Proposition 3. *For all $\nu > \epsilon$ and for $\rho \in T$, if $d(C(t_1), \rho) = \nu$, then there exists $t_2 > t_1$ such that either $r(t_2) \in \{R_3, R_4\}$ or $d(C(t_2), \rho) = \epsilon$.*

Finally, if there are two robots, j and k where $j.dist = k.dist$, jl and $kl \in E$ and $\angle jlk \approx \frac{\pi}{3}$, then the grammar encloses a triangle by adding an edge jk via rule R_5. This guarantees that the resulting δ-triangulation is maximal. The edge is labeled $e(jk) = (outer, comm)$, indicating it is not used as a control input.

Proposition 4. *If $d(C(t_1), \rho) < \epsilon$, then there exists $t_2 > t_1$ such that either*

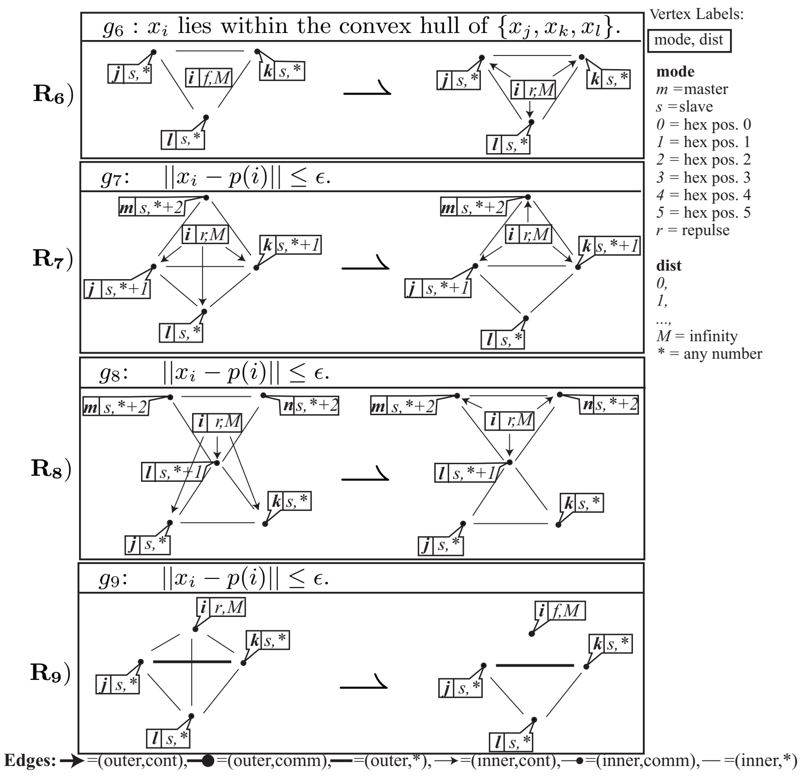
i. *$r(t_2) = R_3$ or $r(t_2) = R_4$ or*
ii. *There exists $\eta \in T_{max}$ such that (1) $d(C(t_2), \eta) < \epsilon$ and (2) if $i.mode = free$ and $x_i \notin I(t)$, R_3 or R_4 are applicable to i.*

3.6 Error Correction

Rules R_6-R_9 (Figure 6) and the controller (Table 1) associated with the label `repulse` remove robots trapped in the crystal structure. Suppose three robots $\{j, k, l\}$ define a face f_1 and a robot i (with $i.mode =$ `free`) lies in the closure of f_1. Then applying rule R_6 changes $i.mode$ to `repulse`. The `repulsing` robot calculates a sink point $p(i)$ such that when $l.dist = k.dist < j.dist$ then $p(i) = 2\delta(x_j - \frac{1}{2}(x_k + x_l))/(||x_j - \frac{1}{2}(x_k + x_l)||)$ and when $j.dist < k.dist = j.dist$, then $p(i) = 2\delta(\frac{1}{2}(x_k + x_l) - x_j)(||x_j - \frac{1}{2}(x_k + x_l)||)$. The `repulsing` robot then moves to p via the simple potential controller defined in Table 1. If x_i is near p and p is inside another face farther away from the center, either rule R_7 or R_8 are applied to continue moving away from the `master`. Or if $p \notin I(t)$, rule R_9 returns robot i to `free` mode. Applications of these rules can be seen in Figure 2(c)-(f).

Proposition 5. *For any $\rho \in T$, if $i \in I(t_1)$ and $d(C(t_1), \rho) < \epsilon$, then there exists $t_2 > t_1$ such that $i \notin I(t)$ or $r(t_2) = R_3$ or $r(t_3) = R_4$.*

Corollary 1. *There exists ϵ such that if $d(C(t_1), \rho) < \epsilon$ and R_9 is applicable, then there exists $t_2 \geq t_1$ such that $r(t_2) \in \{R_3, R_4\}$.*



g_6 : x_i lies within the convex hull of $\{x_j, x_k, x_l\}$.
g_7: $||x_i - p(i)|| \leq \epsilon$.
g_8: $||x_i - p(i)|| \leq \epsilon$.
g_9: $||x_i - p(i)|| \leq \epsilon$.
$\mathbf{R_6}$)
$\mathbf{R_7}$)
$\mathbf{R_8}$)
$\mathbf{R_9}$)
Vertex Labels:
mode, dist
mode
m =master
s =slave
0 = hex pos. 0
1 = hex pos. 1
2 = hex pos. 2
3 = hex pos. 3
4 = hex pos. 4
5 = hex pos. 5
r = repulse
dist
0,
1,
...,
M = infinity
* = any number
Edges: =(outer,cont), =(outer,comm), =(outer,*), =(inner,cont), =(inner,comm), =(inner,*)

Fig. 6. Error Correction rules, Φ_{EC}. Rule R_6 initiates the error correction process, R_7 and R_8 move vertices towards the outside of the crystal and R_9 terminates the error correction process when the robot is sufficiently far from the crystal.

3.7 Process Interaction

Theorem 1. *For all* $M \in \{6, 7, ..., |V| - 1\}$*, if* $|C(t_1)| = M$*, then there exists* $t_2 \geq t_1$ *such that* $r(t_2) \in \{R_3, R_4\}$ *and* $|C(t_2^+)| = M + 1$.

Proof. Assume to the contrary that $|C(t_1)| = M < V$ but neither rule R_3 nor R_4 are applied at any time $t' \geq t_1$. Since at time t_1, there is some $\rho \in T$ such that $d(C(t_1), \rho) < \infty$, then by Proposition 3 if R_3 or R_4 are not applied the system will flow to a state where $d(C(t_1), \rho) < \epsilon$. By Proposition 4, we know that if $i \notin I(t)$ then eventually R_3 or R_4 are applicable. Thus eventually it must be the case that $i \in I(t)$. However, by proposition 5 and Corollary 1, $i \in I(t)$ leads to R_3 or R_4 being applied. Thus it must be the case one is applied and for some i, $i.mode$ changes to `slave`, therefore $|C(t_2^+)| = M + 1$. □

Proposition 6. *For all* $t_1 < t_2$, $|C(t_1)| \leq |C(t_2)|$.

Proof. No rule in Φ changes the vertex labels `master` or `slave`. □

Theorem 2. *For all trajectories, $\gamma(t_0) = \gamma_0$ implies there exists a time t such that for all $t' \geq t$, $|C(t')| = |V|$.*

Proof. Proposition 2 implies eventually $|C| = 6$. Theorem 2 follows by induction on Propositions 1 and 6 and the finiteness of the initial graph.

Theorem 3. *For all trajectories $\gamma(t) \in \mathcal{T}(\gamma_0, \Phi, u)$,*

$$\lim_{t\to\infty} \gamma(t) \in T_{max}.$$

Proof. By Theorem 2 and Proposition 4 we have that eventually $G_\gamma(t) = G_\rho$ for some $\rho \in T_{max}$. By Proposition 3 we know the positions converge.

The embedded graph grammar system (γ_0, Φ, u) was simulated for a variety of initial conditions with graph sizes ranging from 12 to 100. All simulations resulted in δ-triangulations. Figure 1 shows snapshots from one such simulation. Although the error correction rules are central to the proofs above, in simulation error correction rules are rarely applied. Note that the final shape is non-deterministic and highly dependent on the initial conditions. Nonetheless, once an initial δ-triangulation is established, simple rules can be constructed to form almost any global shape from local control.

4 Proof of Supporting Propositions

4.1 Auxiliary Properties

Lemma 1. *Let $A \subset V$ be any set such that $Con(A) = A$, then the equilibrium and dynamics of A are independent of $V - A$.*

Proposition 7. *If $\gamma(t_0) = \gamma_0$, then there exists a time t_2 such that $r(t_2) = R_2$.*

Proof. $r(\tau_1) = R_1$ since only R_1 is applicable to G_{γ_0}. $Im(h_1)$ is the set of vertices to which R_1 is applied. R_2 is the only rule applicable to vertices in h_1. Since Lemma 1 holds for $Im(h_1)$ we construct a Lyapunov function for $x_{|_{Im(h_1)}}$ by

$$V(t) = \sum_{i \in Im(h_1) - \{0,j\}} ||x_i - P(i)||^2 + (||x_j - x_0|| - \delta)^2$$

If we treat x_0 as a constant, then $V \geq 0$, $\dot{V} = -\nabla V^T \nabla V \leq 0$. Furthermore, if $\dot{V} = 0$, $||x_i - P_i|| < \epsilon$, so we can satisfy the guard g_2. $\square$

Proposition 8. *Suppose $C(t_k^-) = B$, $r(t_k) = R_3$, and $C(t_k^+) = D$. If for some ϵ, $d(B, T) < \epsilon$, then $\lim_{t\to\infty} C(t)[V_D] \in T$*

Proof. Suppose robot i is added to $C(t_k^-)$ by forming `control` edges ij and ik to robots j and k to form D. Since for any u, v the rules never allow an edge uv where $u.dist < v.dist$, Lemma 1 holds and we may consider the dynamics of $C(t)[V_D]$ without reference to the vertices in $V - V_D$. We choose ϵ in $g_{settled}$ small enough so that if for R_3, $g_{settled}(L)$ is satisfied for j and k, their motion relative to the coordinate frames of the robots to which they have directed control edges is zero. Since every time a robot is added to the structure, $g_{settled}(L)$ must be satisfied, by induction we may treat x_j and x_k as constants.

Let $V = (||x_i - x_j|| - \delta)^2 + (||x_i - x_k|| - \delta)^2$. $\dot{V} = \nabla V^T \frac{dx}{dt} = -\nabla_{x_i} V^T \nabla_{x_i} V \leq 0$. V is a Lyapunov function with stable fixed points where x_i^* satisfies $||x_i^* - x_j|| - \delta = ||x_i^* - x_k|| - \delta = 0$. We must show that the region of attraction of one of these fixed points is the open half plane defined by line l_{jk} containing x_i^*.

We have that $\dot{x}_i \propto (||v_{ij}|| - \delta)\hat{v}_{ij} + (||v_{ik}|| - \delta)\hat{v}_{ik}) = \alpha \hat{v}_{ij} + \beta \hat{v}_{ik}$. We show that there is no path from $x_i \notin l_{jk}$ to $x_i \in l_{jk}$. Suppose we pick x_i near l_{jk} such that $||v_{ij}|| < ||v_{ik}||$ and $\beta > \alpha > 0$. Then $\angle \hat{v}_{ij} < \angle \dot{x}_i < \angle \hat{v}_{ik}$. This remains true (near l_{jk}) until $||v_{ij}|| = \delta$. Now $\alpha \leq 0$and $\angle \hat{v}_{ik} \leq \angle \dot{x}_i < \pi + \angle \hat{v}_{ik}$. This suggests if the trajectory intersects l_{jk} it must do so between x_j and x_k. However, in this region, $\alpha < 0$ and $\beta < 0$ which means there is a component of $\dot{x}_i$ away from the line l_{jk}. Thus there is no path leading from $x_i \notin l_{jk}$ to $x_i \in l_{jk}$.

Proposition 9. *Suppose* $C(t_k^-) = B$, $r(t_k) = R_4$, *and* $C(t_k^+) = D$. *If for some* ϵ, $d(B,T) < \epsilon$, *then* $\lim_{t\to\infty} C(t)[V_D] \in T$. (See proof above).

4.2 Supporting Proposition Proofs

Proof of Proposition 2. By Proposition 7, rule R_2 is executed. We propose the Lyapunov function $V = \sum_{i \in Im(h_2)} \sum_{j \in F^+(i)} 1/2 U_{ij}$. $\dot{V} = \nabla V_x \dot{x} = -\nabla V_x^T \nabla V_x \leq 0$. In addition to fixed points corresponding to δ-triangulations, the `slave` controllers operating under the topology created by R_2 have local minima at some positions where $x_i = x_j$. However, by choosing ϵ in g_2 small enough ($18\epsilon^2 < \delta^2$), then for all V in the guard region, $V \leq 18\epsilon^2 < \delta^2$ where δ^2 is the contribution to the Lyapunov function for a single edge where $x_i = x_j$. Thus $\lim_{t\to\infty} \gamma[Im(h_2)](t) \in T$. □

Proof of Proposition 3. By propositions 8 and 9 and the fact that edges added by R_5 are not used by the controllers, every vertex that is not settled has its own decreasing Lyapunov function parameterized by the edge lengths. □

Proof of Proposition 4. Since Proposition 3 holds and since a small neighborhood near $\rho \in T$ satisfies guard g_5, every application of R_5 must be applied. Furthermore, since the intersection of the half-planes and cones defined by the boundary edges is a superset of the exterior of the crystal, it must be the case that $x_i \notin I(t)$ implying rule R_3 or R_4 are applicable. □

Proof of Proposition 5. When $d(C(t), \rho) < \epsilon$ for very small ϵ, it is clear that either (1) $p(i)$ lies in a face where the sum of the distances is greater or (2) $p(i) \notin I(t)$. In case (1), as $x_i \to p(i)$ either R_7 or R_8 becomes applicable. Since the graph is finite, by induction on rules R_7 and R_8 eventually $p(i) \notin I(t)$ (i.e case (2)). By the convergence property of the controllers, eventually $x_i \notin I(t)$. □

Proof of Corollary 1. If $d(C(t), \rho) < \epsilon$ and if $\delta + \epsilon \ll \sqrt{3}/2\delta - \varepsilon$, there are no Gabriel graph edges between i and vertices on the interior of the crystal. This implies that if i executes R_9 and is labeled `free`, while $||x_i - x_j|| < \delta$ for $j \in C(t)$, the motion is away from the crystal. Thus i remains in g_3 or g_4. □

5 Conclusions and Future Work

Embedded graph grammars are a unique combination of concurrency and hybrid systems in which we can model networked robotic systems. Our solution to the δ-triangulation coverage problem is meant to demonstrate how embedded graph grammars can be used to specify and reason about the correctness of complex, multi-mode coordination problems. We note that issues such as robustness or performance under sensing and communication limitations can be addressed by constructing more complicated grammars. However, these questions lie outside the scope of this paper.

We believe that as systems incorporate more complex combinations of reactive tasks, the need for the unified modeling of communication protocols and the ensuing hybrid dynamics becomes pronounced. Unfortunately, proving that such systems are correct is unwieldy as the proofs in this paper suggest. We plan to explore real time temporal logics, compositional methods, and automated verification techniques, thereby completing the formalism introduced here.

6 Acknowledgement

Eric Klavins and John-Michael McNew are partially supported by the AFOSR via the 2006 MURI award *Design, Specification and Verification of Distributed Embedded Systems*. The work by Magnus Egerstedt is supported under a grant from the NASA, Earth Science Technology Office, AIST Program.

References

1. Egerstedt, M., Brockett, R.: Feedback can reduce the specification complexity of motor programs. IEEE Transactions on Automatic Control (2003)
2. V. Manikonda, P.S.K., Hendler., J.: Languages, Behaviors, Hybrid Architectures and Motion Control. In: Mathematical Control Theory. (1998)
3. Lygeros, J., Johansson, K.H., Simic, S., Zhang, J., Sastry, S.: Dynamical properties of hybrid automata. IEEE Transactions on Automatic Control (2003)
4. Henzinger, T.: The theory of hybrid automata, 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press (1996)
5. Klavins, E., Burden, S., Napp, N.: Optimal rules for programmed stochastic self-assembly. In: Robotics: Science and Systems, Philadelphia, PA (2006)
6. Cortes, J., Martinez, S., Bullo, F.: Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. IEEE Transactions on Automatic Control (2006)
7. Leonard, N., Fiorelli, E.: Virtual leaders, artificial potentials and coordinated control of groups. Proc. of the IEEE Conference on Decision and Control (2001)
8. McNew, J.M., Klavins, E.: Locally interacting hybrid sytems using embedded graph grammars. In: Proc. of the Conference on Decision and Control. (2006)
9. Smith, B., McNew, J., Egerstedt, M., Klavins, E., Howard, A.: Embedded graph grammars for multi-robot coordination. In: Proc. of the International Conference on Robotics and Automation. (2007) submitted.
10. B. Shucker, T. Murphey, J.: A method of cooperative control using occasional non-local interactions. In: Proc. of the American Control Conference. (2006)