Proceedings of the 45th IEEE Conference on Decision & Control
Manchester Grand Hyatt Hotel
San Diego, CA, USA, December 13-15, 2006

Locally Interacting Hybrid Systems with Embedded Graph Grammars

John-Michael McNew Eric Klavins

Department of Electrical Engineering
University of Washington
Seattle, WA 98195
{jmmcnew,klavins}@ee.washington.edu

Abstract— **In many cooperative control methods, the *network topology* in¤uences the evolution of its continuous states. In turn, the continuous state may in¤uence the network topology due to local restrictions on connectivity. In this paper we present a grammatical approach to modeling and controlling the switching of a system's network topology, continuous controllers, and discrete modes. The approach is based on *embedded graph grammars*, which restrict interactions to small subgraphs and include spatial restrictions on connectivity and progress. This allows us to direct the behavior of large decentralized systems of robots. The grammatical approach also allows us to *compose* multiple subsystems into a larger whole in a principled manner. In this paper, we illustrate the approach by proving the correctness of a cooperative control system called the *load balanced multiple rendezvous* problem.**

I. Introduction

Large scale networked systems such as automated highway systems, air-traf£c control systems, and cooperative robotic systems present unique modeling and control challenges. Sensory, communication, or computational limitations often require that a desired global behavior is produced using only local control of each robot, node, or subsystem. Furthermore, these subsystems often exhibit complex interactions between their continuous modes, network topology, and logical modes.

In this paper, we introduce the notion of an *embedded graph grammar* which extends the *graph grammar* model [1] from a purely topological construct to one that includes geometry, continuous dynamics and local conditions on the evolution of those dynamics. We present this model through an example that also highlights a hierarchical and systems-oriented approach to using graph and embedded graph grammars. Methods such as the composition of grammars, structured labeling and lexicographic ordering of Lyapunov functions on grammars allow us to design grammars in a principled manner.

After brie¤y discussing related work, we present a motivating example called the *load balanced multiple rendezvous* problem. Section IV de£nes *embedded graph grammars* and discusses our use of temporal logic on graph transition systems. Sections V through VII present grammars and controllers designed to address each of the subtasks in the example. In Section VIII, we compose these elements to create an *embedded graph grammar* and prove it meets the desired speci£cation.

The authors are supported by the AFOSR via the 2006 MURI Award *Speci£cation, Design and Veri£cation of Distributed Embedded Systems*.

II. Related Work

Researchers are currently uncovering rich connections between graphs, *information ¤ow*, and formation control [2],[3],[4]. Local restrictions on communication may induce arbitrary switching in the network. In light of this, Ji and Egerstedt [5] design graph-based controllers that maintain connectivity as robots rendezvous. Jadbabaie et al. [6] approach the issue differently, constructing controllers that are stable under arbitrary switching sequences.

Klavins [7] examines the notion of *explicitly* controlling the graph structure by describing the self-organization of robot formations as a graph process. Klavins, Ghrist, and Lipsky[1] introduce graph grammars to assemble any pre-speci£ed graph from an initially disconnected graph. By restricting rewrites to small subgraphs, graph grammars provide a useful method to program the concurrent behavior of large decentralized systems of robots [8], [9]. Olfati-Saber [10] investigates controlled switching in robot formation using a *global* hybrid automata.

However the interplay between all these elements (information ¤ow, hybrid switching, local restrictions on sensing, and continuous dynamics) in cooperative control remains largely uncharted territory. Our goal is to introduce *embedded graph grammars* as a single *graph-centric* model containing all these elements.

III. Motivating Example

Consider a group of networked robots with range limited communication executing formation control on Mars in search of ice. Simultaneously, a number of robots sense ice. We call these robots *bases*. A reasonable goal would be to route other robots (commuters) in essentially equal numbers to the base locations while maintaining connectivity in the overall system. The problem is complex, containing elements of well known formation control problems such as multiple-leader rendezvous, connectivity maintenance, and load balancing.

As shown in Figure 1, our solution to the problem is to compose three distributed algorithms that, when run in parallel, result in the desired global behavior. These are: *Distributed Cycle Removal* (DCR), which removes cycles from the graph; *Load Balancing* (LB), which distributes

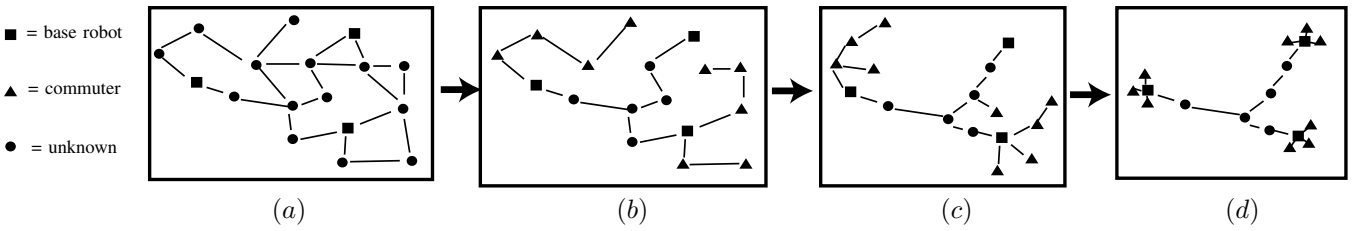


Fig. 1. Subtasks in the load balanced multiple rendezvous problem. a) An initial state. (b) Cycle removal. (c) Load Balancing. (d) A £nal state.

commuters equally among the bases; and *Connectivity Maintenance and Rendezvous* (CMR), which maintains connectivity while allowing commuter robots to rendezvous at the appropriate base. The solution combines controlled switching of a locally restricted network and continuous spatial dynamics. *Labeled graphs* and their *realizations* in continuous space are a natural model for such systems. So in Section IV we develop the formal de£nitions of *embedded graph grammars* and in Section IV-E we return to this example and formally specify the *Load Balanced Multiple Rendezvous Problem*(LBMR). We then propose a *embedded graph grammar* to solve the problem and prove its correctness through *composition*.

IV. Embedded Graph Grammar Definitions

A. Graphs and other Notation

For clarity, we introduce some 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 the vertex labeling function that maps vertices into Σ, and e is the edge labeling function that maps edges into Σ. We denote by $\mathcal{G}$ the space of labeled graphs and by $\bar{\mathcal{G}}$ the space of unlabeled graphs. Often the label set Σ is a cartesian product of atomic label spaces we refer to as *£elds*. We often name £elds. For example the system introduced in Section IV-E has a node label space given by $\Sigma = \{base, unknown\} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$ where the £elds are named $(mode, owner, dist, degree)$. We use dot notation to indicate the values of label £elds for speci£c robots. For instance $i.owner = 10$ indicates that robot i has the value 10 in its $owner$ £eld.

If S is a set of vertices, $G[S]$ denotes the subgraph of G induced by S. For any graph G the neighborhood of a vertex $i \in V_G$ is denoted $N_G(i)$ and the closed neighborhood is denoted $N_G[i] = N_G(i) \cup i$. When G is a digraph, we denote the *out- neighborhood* by $N_G^+(i)$. We denote the ¤oor function by $\lfloor x \rfloor$.

De£nition 4.1: A function f is *well de£ned* with respect to an equivalence relation $\sim$ if $x \sim y$ implies $f(x) = f(y)$.

B. Embedded Graph Transition Systems

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

De£nition 4.2: 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 Γ.

Embedded graphs model the *network topology*, discrete and continuous states of an interconnected collection of robots, vehicles or particles in the following manner. 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 $l(i)$ keeps track of local information and also indicates the operational mode of robot i. The edge label, $e(ij)$, contains information *maintained* by two connected robots. The realization function x assigns to each robot a continuous state or *realization* in the 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 embedded graph induced by S, $\gamma[S]$, is given by the pair $(G[S], x_{|_S})$.

An *embedded graph transition system* is a triple $(\gamma_0, \mathcal{A}, u)$ where γ_0 is an embedded graph describing the initial state, $\mathcal{A} \subseteq \Gamma \times \Gamma$ is the *embedded graph transition relation* and $u : V \times \Gamma \to TX$ is the vector £eld describing the continuous ¤ow. In the systems we consider $(\gamma_1, \gamma_2) \in \mathcal{A}$ implies $x_{\gamma_1} = x_{\gamma_2}$, thus an *embedded graph transition system* describes a hybrid system where the discrete (or jump) states are labeled graphs and the continuous states are realization functions.

De£nition 4.3: 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 £nite 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 $\mathcal{T}(\gamma_0, \mathcal{A}, u)$. When the dependence on σ is clear we will often write $\gamma(t)$ to mean $\sigma(t)$, $x(t)$ instead of $x_{\sigma(t)}$, and G_k instead of $G_{\sigma(\tau_k)}$. Embedded graph transition systems allow the control of the evolution of the geometry, network connectivity and logic variables of a system. However, their direct implementation is undesirable because the global graph matching problem is computationally intense (especially from a distributed point of view), the model lacks *local* restrictions on sensing and communication, and the transition relation is not permutation invariant.

C. Locality and Embedded Graph Grammars

By explicitly including notions of locality in our model, we can address the issues above by ensuring

1) Graph matching involves only a small subset of vertices,

2) The ¤ow and transition relations use only the information available to robots via local sensing and communication, and
3) The ¤ow and transition relation are permutation invariant.

The £rst choice facing a system designer who will use *embedded graph grammars* is the construction of a *proximity function*, ψ, describing *local* restrictions on sensing and communication. A proximity function takes embedded graphs as inputs and returns a digraph $\psi(\gamma)$ where an edge ij is in $E_{\psi(\gamma)}$ if robot i can sense and (if it chooses) communicate with j. A proximity graph is shown in Figure 2. We call the out neighborhood of i, $N^+_{\psi(\gamma)}(i)$ the *proximity neighborhood*. We call the set of vertices $F_\gamma(i) \triangleq \{j \in V_\gamma \mid j \in N_{G_\gamma}(i) \cap N^+_{\psi(\gamma)}(i)\}$ the *friends* of i. In keeping with existing graph literature we write $F[i]$ to mean $F(i) \cup i$. We say an embedded graph γ is *edge-consistent* when every edge ij in E_γ is also in $E_{\psi(\gamma)}$. We denote by $\mathcal{D}$ the set of all edge-consistent embedded graphs. And denote by $\mathcal{D}_{G_0} = \{\gamma \in \mathcal{D} \mid G_\gamma = G\}$ the subset of edge-consistent graphs with the topology of G.

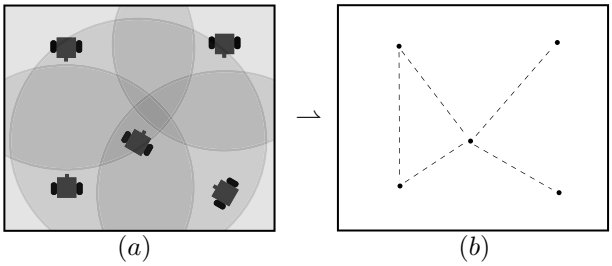


Fig. 2. Proximity Graphs. (a) Robots with range limited communication. (b) The resulting proximity graph is the disk graph.

De£nition 4.4: Consider the pairs (A, γ) and (B, ρ) where γ and ρ are embedded graphs and $A \subseteq V_\gamma$ and $B \subseteq V_\rho$. We de£ne the *point of view* equivalence relation [11] $\sim$ by $(A, \gamma) \sim (B, \rho)$ if there exists a bijective map h between the proximity neighborhoods $\bigcup_{i\in A} N_{\psi(\gamma)}[i]$ and $\bigcup_{j\in B} N_{\psi(\rho)}[j]$ such that

1) For all $i \in A$, $h(i) \in B$,
2) For all $k \in \bigcup_{i\in A} N_{\psi(\gamma)}[i]$, $x_\gamma(k) = x_\rho(h(k))$, and
3) For all $i \in A$, h is a label preserving isomorphism between $G_\gamma[F[i]]$ and $G_\rho[F[j]]$.

We say a controller $u : V \times \Gamma \to X$ is *locally implementable* if it is well de£ned with respect to the point of view equivalence relation $\sim$. As shown in Figure 3 when the vertex set contains only one element, $A = \{i\}$, the information available to compute the control $u(i, \gamma)$ is restricted to the embedded graph induced by the friends of i, $\gamma[F[i]]$ (obtained via communication) and the continuous states (obtained via sensing) of the proximity neighbors of i.

The formation of new links and the updating of the robots labels is executed on a *local* scale by the use of *guarded rules*. A *guard* g is a function $g : \mathcal{P}(V) \times \Gamma \to \{true, false\}$. A guard g is *locally checkable* if it is well de£ned with respect to point of view equivalence relation,

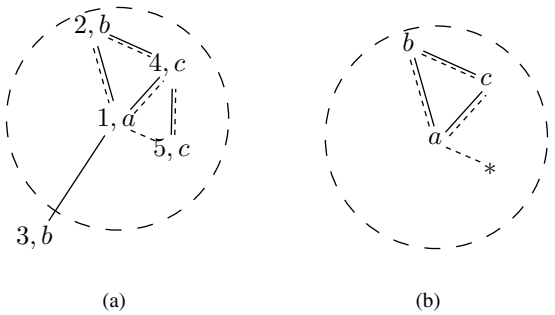


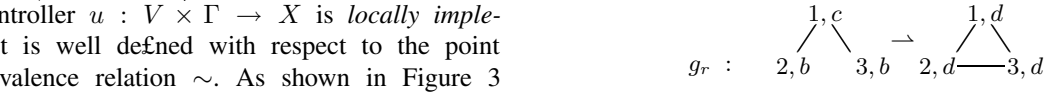
Fig. 3. Friends and Neighbors from the point of view of vertex 1. (a) An embedded graph, γ. Solid lines indicate edges in the embedded graph and dashed lines indicate proximity edges. The numbers are vertices and the letters are vertex labels. (b) Locally implementable information available to vertex 1. Vertex information is not available due to permutation invariance. Vertices with proximity edges and embedded graph edges from 1 are in the friends of 1, $F(1)$. Vertex 1 can access label, edge, and state information in $F(1)$. The vertex marked with an asterisk is in the proximity neighborhood, $N_{\psi(\gamma)}(1)$, thus only its continuous state is accessible. Note that the embedded graph is not edge-consistent because although at some time vertex 3 established an edge with 1, currently 3 lies outside the communication range of 1.

$\sim$ and $g(A, \gamma) = true$ implies $\psi(\gamma[A])$ is connected. A *guarded 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 function g. Given a rule, a *witness* is a label preserving subgraph isomorphism mapping V_L into G_γ. If a witness is found and if $g(h(V_L)) = true$, the rule is *applicable* via an action (r, h). Formally, the application of an action (r, h) on an embedded graph $\gamma = (G, x)$ produces a new embedded graph $\gamma' = ((V, E', l', e'), x)$ de£ned by

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

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).

Consider the rule (g_r, L, R), given by



(1)

where the guard function $g(A, \gamma)$ is true if the set of vertices A checking the rule only have neighbors in A, that is $\cup_{i\in A} N_{\psi(\gamma)}(i) = A$. Figure 4 demonstrates the application of this rule to an embedded graph.

We often display rules in tables and use schema to represent multiple rules with a single diagram. Figure 5 shows four rules. In rule r_1 of Figure 5, the pair of £eld names $(owner, dist)$ appears above the transition arrow, indicating that the pairs labeling the vertices in the left and right graphs have values corresponding to those £elds. For rule r_1 to be applicable we must £nd a label preserving subgraph

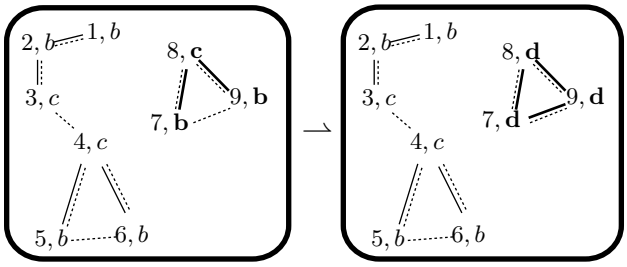


Fig. 4. Application of the rule in Equation 1. (a) An embedded graph, γ. The mapping $h_1 \triangleq \{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 3\}$ is a subgraph isomorphism but does not preserve labels, so the rule is not applicable. The mapping $h_2 \triangleq \{1 \mapsto 4, 2 \mapsto 6, 3 \mapsto 5\}$ is a label preserving subgraph isomorphism, however the guard is false because $\cup_{i\in\{4,5,6\}} N_{\psi(\gamma)}(i) = \{3,4,5,6\} \neq \{4,5,6\}$. The mapping $h_3 \triangleq \{1 \mapsto 8, 2 \mapsto 7, 3 \mapsto 9\}$ shown in bold *is* a witness that satisfies the guard so the rule is applicable here. (b) Applying the action, (r, h_3) replaces the subgraph $G[h_3(V_L)]$ with the graph $h_3(R)$ shown in bold.

isomorphism, h, mapping the left graph L into G such that if $h(1).owner = m$ and $h(2).owner = n$, then the guard $n > m$ is satisfied. Label fields not displayed in a rule are not changed if that rule is applied.

An *embedded graph grammar system (EGG)* is a quadruple $(\gamma_0, \Phi, \psi, u)$ where γ_0 is an embedded graph representing the initial state, Φ is a set of locally checkable guarded rules, ψ is a proximity function and u is a locally implementable controller. We denote the trajectories of a local embedded graph grammar by $\mathcal{T}(\gamma_0, \Phi, \psi, u)$. 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, $\mathcal{A}(\Phi) = \cup_{r\in\Phi}\mathcal{A}(r)$. The relationship between the trajectories of an embedded graph grammar and embedded graph transition system trajectories is given by

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

D. Temporal Logic and Proof Techniques

By a *proposition*, we simply mean a subset of embedded graphs, $P \subseteq \Gamma$. An embedded graph γ satisfies a proposition P, denoted $\gamma \models P$, if $\gamma \in P$. We often construct propositions that only refer to the topological elements of embedded graphs. Given a logical statement on graphs P, the indicator function $I(P)$ evaluates to 1 if the statement is true and 0 otherwise. The LTL specification eventually always P, written $\mathbf{FG}P$, means for *every sequence* in $\mathcal{T}(\gamma_0, \Phi, \psi, u)$ there exists some k (not necessarily the same) such that for all $n \geq k$, $G_n \in P$. When this is true we write $(\gamma_0, \Phi, \psi, u) \models \mathbf{FG}P$. We often prove an eventually always specification by constructing a *discrete Lyapunov function*.

Definition 4.5: Let $\preceq$ be an ordering on $\mathbb{R}^n$ with an unique zero element. A *discrete Lyapunov* function for the system $(\gamma_0, \Phi, \psi, u)$ is a function $\mathcal{V} : \mathcal{G} \to \mathbb{R}^n$ such that

a. $\mathcal{V}$ is a positive decreasing function over all trajectories,
b. $\mathcal{V} \succ \mathbf{0}$ implies that at least one action (r, h) is eventually applicable, and
c. $\mathcal{V}(G_k) = \mathbf{0}$ implies that for all future graphs G_n, $\mathcal{V}(G_n) = \mathbf{0}$.

Theorem 4.1: Let $(\gamma_0, \Phi, \psi, u)$ be any system where the set of reachable labeled graphs is finite. Let P be a proposition on graphs and $\mathcal{V}$ be a discrete Lyapunov function for the system $(\gamma_0, \Phi, \psi, u)$ such that $\mathcal{V}(G) = \mathbf{0}$ implies $G \in P$. Then $(\gamma_0, \Phi, \psi, u) \models \mathbf{FG}\ P$.
Different literatures have different flavors of this theorem. For example, the theorem (and its proof) expressed in the temporal logic of actions can be found in [12].

E. Load Balance Multiple Rendezvous Revisited

We now have the formal language to express the LBMR problem. Let $X = \mathbb{R}^2$. Let $ij \in E_{\psi(\gamma)}$ if and only if $||x_\gamma(i) - x_\gamma(j)|| < \Delta$. Suppose B denotes the set of robots labeled *base*, $H \supseteq B$ denotes the set of vertices in a path between two bases (informally called *highway robots*) and $C = V \backslash H$ (informally called commuters). If $i \in B$ we denote the subset of commuter robots with an edge to i by C_i. Define the proposition for load balancing by

$$P_{LB} = \{\gamma \in \Gamma \mid \forall i \in B, |C_i| = \lfloor \frac{|C|}{|B|} \rfloor .\} \tag{2}$$

Define the proposition for distributed cycle removal by

$$P_{DCR} = \{\gamma \in \Gamma \mid G_\gamma \text{ is a connected tree}\}. \tag{3}$$

Finally, let P_{CMR} denote the set of embedded graphs where an edge between a base i and commuter j implies that $x_i = x_j$. We can now formally state the load balanced multiple rendezvous task.

Task 4.1: Design an embedded graph grammar $(\gamma_0, \Phi, \psi, u)$ such that for all trajectories σ, the graphs are connected, edge-consistent and

$$\lim_{t\to\infty} \sigma(t) \in P_{DCR} \cap P_{LB} \cap P_{CMR}.$$

We introduce a locally implementable controller u in section V, and two grammars Φ_{DCR} and Φ_{LB} in sections VI and VII. The embedded graph grammar $(\gamma_0, \Phi_{DCR} \cup \Phi_{LB}, \psi, u)$ defines an algorithm that removes cycles, converting highway robots to commuters. It then routes commuters to the bases along the highways. This method does not require maps or even a global coordinate system. For proof simplicity, we assume that $\frac{|F|}{|B|}$ evaluates to an integer, noting our grammar converges to an invariant set when this is not the case.

V. Connectivity Maintenance

We introduce a locally implementable controller u similar to the one defined in [5].

A. Control Law

Suppose E_1 denotes the set of edges ij where either $i.mode = base$ or $j.mode = base$ but not both and j is in $F(i)$. Let E_2 be the set where neither are bases and j is in $F(i)$. Define the function

$$\mathcal{U}_{ij}(\gamma) = \begin{cases} \frac{1}{2}(\Delta - ||x_i - x_j||)^{-1} & \text{if } ij \in E_2 \\ (\Delta - ||x_i - x_j||)^{-1} & \text{if } ij \in E_1. \end{cases}$$

and define the *total potential function* by

$$\mathcal{U}(\gamma) = \sum_{ij\in E_2\cup E_1} \mathcal{U}_{ij}(\gamma). \tag{4}$$

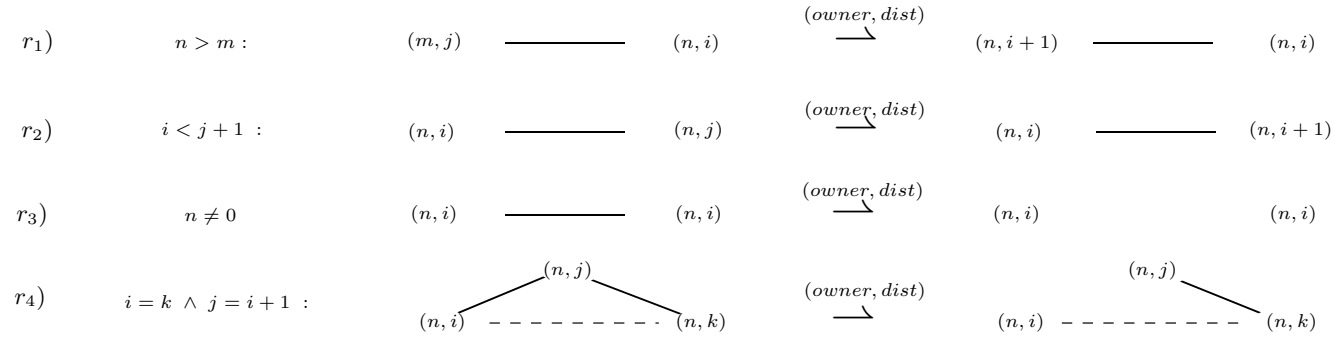


Fig. 5. Rule set for distributed cycle removal, Φ_{DCR}. The boolean statement to the left of the colon is the guard. If a match of the graph on the left side of the transition arrow is found that also satisfies the guard, it can be replaced with the graph on the right side. The label fields involved are $(owner, dist)$ and the light dashed lines in the fourth rule indicate that an edge may or may not be present.

The controller u is defined by

$$u(i,\gamma) \quad = \quad \begin{cases} 0 & i.mode = base \\ -\nabla_{x_i}\mathcal{U} & i.mode = unknown \end{cases} \qquad (5)$$

where

$$-\nabla_{x_i}\mathcal{U} = \sum_{j \mid j \in F(i)} \frac{x_j - x_i}{(||x_i - x_j|| - \Delta)^2 ||x_i - x_j||}.$$

Since the controller depends only on $F(i)$ it is locally implementable.

B. Convergence

We must show that if we fix a graph G and implement the control law u on any edge-consistent graph $\gamma(t)$, then $\gamma(t')$ is edge-consistent for all $t' \geq t$. Additionally, we must show that $x(t)$ converges to safe configurations in which if the rules introduced in section VII insert edges, the embedded graph remains edge-consistent.

Proposition 5.1: Suppose γ is any connected edge-consistent graph and that $G(t) = G$ for all $t' \geq t$. Then the continuous state $x(t)$ of the system evolving under u converges to the set of limit points $M = \{x_\gamma \mid \gamma \in \mathcal{D}_G$ and $\dot{\mathcal{U}}(x_\gamma) = 0\}$. Furthermore any point in M is a fixed point.

Proof: Without loss of generality, suppose one base i has coordinates $x_i = (0,0)$. Let $\Omega_0 = \{x_\gamma \quad | x_\gamma \in \mathcal{D}_{G_0} \ \mathcal{U}(x) \leq \mathcal{U}(x_0)\}$. Ω_0 is closed and bounded by $r = |G|\Delta$. Since $\dot{\mathcal{U}} = -\frac{\partial U}{\partial x}^T \frac{\partial U}{\partial x} \leq 0$, then by LaSalle's invariance principle [13] every solution converges to M. Furthermore since whenever $\dot{\mathcal{U}} = 0$, then $\dot{x} = 0$, it follows that any $x \in M$ is a fixed point. ■

Additionally, $\gamma(t')$ must be edge-consistent for all $t' \geq t$ since $U_{ij} \to \infty$ as $\Delta - ||x_i - x_j|| \to 0$.

We call any vertex i where $degree(i) = 1$ a *leaf* vertex.

Corollary 5.1: Suppose $i, j \in V$ are any vertices such that $degree(i) = 1$ and $ij \in E$. If x_j^* denotes a final value of $x_j(t)$, then x_i converges asymptotically to x_j^*.

Proof: Suppose $x_i \neq x_j$. Because i is only connected to j,$\nabla_{x_i}\mathcal{U} = \frac{x_j - x_i}{(||x_i - x_j|| - \Delta)^2 ||x_i - x_j||} \neq 0$. Since $\dot{\mathcal{U}} = -\sum_{i \in V} (\nabla_x \mathcal{U})^T (\nabla_x \mathcal{U})$, it must be the case that $\dot{\mathcal{U}} < 0$, implying that a configuration with $x_i \neq x_j$ is not in M. ■

We have shown that each leaf converges to its neighbor's position. We next construct a grammar in Section VI that forms trees (and thus leaves). Then we construct a grammar in Section VII that changes the connectivity of leaf robots so the controller u can direct their progress toward the bases.

VI. Distributed Cycle Removal (DCR)

We wish to construct a rule set, Φ_{DCR}, such that eventually it is always the case that G_γ is a connected tree (proposition P_{DCR}). We denote by $|C_G|$ the number of cycles in a graph G.

A. Grammar

Figure 5 shows the rule set designed for distributed cycle removal. In the initial graph G_{γ_0}, $(owner, dist) = (i, 0)$ if node i is a base. Otherwise, $(owner, dist) = (0, |G_0|)$. The algorithm propagates a nondeterministic ordering based on path length from a base node and uses the ordering to determine where to cut a cycle. The guard for r_1 requires the owner field of the first node to be less than the owner field of the second node. As seen in Figure 6, when this is the case, the first node assumes the owner value of the second and sets its distance to be one unit larger. This rule guarantees that eventually all vertices will have the same owner. When two connected vertices i and j share the same owner, rule r_2 updates their *perceived* distance to the owner based on lowest value in their $dist$ fields. This rule guarantees that the perceived distance, $i.dist$, converges to the true distance, $d(i, i.owner)$. As shown in Figure 6, rules r_3 and r_4 determine when two paths from the same base node meet (and thus a cycle is present) and delete a connecting edge.

B. Proof of Correctness

In this section we show that $(\gamma_0, \Phi_{DCR}, \psi, u) \models \mathbf{FG} P_{DCR}$. Let $\mathcal{V} : \mathcal{G} \to \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ where

$$\begin{array}{rcl} \mathcal{V}_1(G) & = & \sum_{j \in V} |\max_{i \in B} i - j.owner| \\ \mathcal{V}_2(G) & = & \sum_{j \in V} |j.dist - d(j.owner, j)| \\ \mathcal{V}_3(G) & = & |C_G|. \end{array} \qquad (6)$$

Let $(\mathbb{R}^3, \preceq)$ be the lexicographic ordering defined by

$$(a_1, a_2, a_3) \prec (b_1, b_2, b_3)$$

if $a_1 < b_1$ or there exists an k such that $a_i = b_i$ for all $i \leq k$ and $a_{k+1} < b_{k+1}$. The lexicographic ordering is extremely

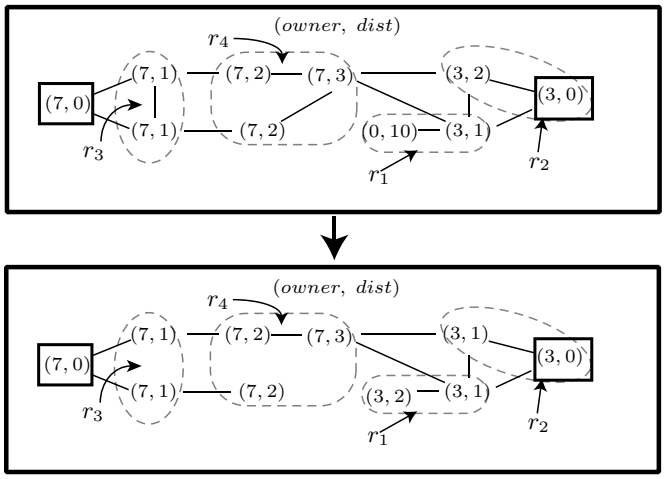


Fig. 6. Application of DCR rules. This £gure displays a graph before and after the concurrent application of rules r_1, r_2, r_3, and r_4. The dashed lines indicate the subgraphs where the rules are applied and small rectangles indicate bases. The label £elds involved are $(owner, dist)$.

useful in composing Lyapunov functions associated with concurrently executing rules and grammars.

Proposition 6.1: If for some vertex i, $i.owner \neq 0$, then there exists some path $p(i.owner, i)$ such that the length of $p(i.owner, i)$ is $i.dist$. (Proof omitted.)

Proposition 6.2: $\mathcal{V}$ in Equation 6 is a Lyapunov function for the system $(\gamma_0, \Phi_{DCR}, \psi, u)$.

Proof: We show that $\mathcal{V}$ meets the three requirements in De£nition 4.5.

$\mathcal{V}$ is positive decreasing: We have constructed $\mathcal{V}_1, \mathcal{V}_2$, and $\mathcal{V}_3$ such that applying rule r_1 to any graph decreases $\mathcal{V}_1$, applying rule r_2 decreases $\mathcal{V}_2$, and applying r_3 or r_4 decreases $\mathcal{V}_3$. Using the lexicographic ordering we must show that applying rules r_3 or r_4 does not increase $\mathcal{V}_2$ or $\mathcal{V}_1$ which is true because it does not change the $owner$ or $dist$ labels. An application of r_2 does not increase $\mathcal{V}_1$ because it does not alter the $owner$ label. Thus $\mathcal{V}$ is positive decreasing with respect to $\prec$.

$\mathcal{V} \succ \mathbf{0}$ implies that at least one action (r, h) is applicable: Assume $\mathcal{V} \succ \mathbf{0}$ but that no action is applicable. Note that $\mathcal{V}_1 > 0$ means r_1 is applicable, which contradicts our assumption. Thus, $\mathcal{V}_1 = 0$ and all vertices are labeled with the same owner. However if $\mathcal{V}_2 > 0$, then $|i.dist - d(i.owner)| > 0$. This implies that there exists a vertex j with an edge ij where either 1) $j.dist < i.dist + 1$ and r_2 is applicable or 2) $j.dist = i.dist$ and r_3 is applicable. If $\mathcal{V}_2 = \mathbf{0}$, then Proposition 6.1 implies that every node is labeled with its true distance from the $owner$ node. Finally, if $\mathcal{V}_3 = |C_G| > 0$, then there must exist two nodes i and j such that $i.dist = j.dist$ and either r_3 or r_4 is applicable.

If $\mathcal{V}(G_k) = \mathbf{0}$, then $\mathcal{V}(G_n) = \mathbf{0}$ for all $n > k$: When $\mathcal{V}_1 = \mathcal{V}_2 = 0$ the guards for rules r_1 and r_2 are false. Since no rule forms an edge, $|C_G| = 0$ for all future graphs. ■

Theorem 6.1: Given the system $(\gamma_0, \Phi_{DCR}, \psi, u)$ it is eventually always the case that G_γ is a connected tree.

Proof: The reachable set of graphs is bounded because the vertex set and label sets of the system are £nite. $\mathcal{V}_3 = 0$ implies there are no cycles in the graph and Proposition 6.1 implies connectivity. Thus if $\mathcal{V} = \mathbf{0}$, then $\gamma \in P_{DCR}$. We then invoke the Lyapunov Theorem 4.1, to show it is eventually always the case that G is a connected tree. ■

VII. Load Balancing

Our goal is to route (in balanced proportions) all the commuters to the bases. By $A \subset C$, we refer to the strict subset of $commuters$ without an edge to a highway robot and informally call these *alley* robots. For instance in the £rst panel of Figure 8, robot k is an alley robot. Suppose G_ω is a tree where every edge ij is labeled with four £elds: $(b_{ij}, s_{ij}, b_{ji}, s_{ji})$. Suppose we cut the graph at any edge ij as pictured in Figure 8. We denote the resulting tree containing vertex i (respectively j) by T_i, (respectively T_j). As seen in Figure 8, the *strength* of T_i, $s_{ij} = \mid V_{T_i} \bigcap C|$, is the number of commuter robots in the tree. The *base mass*, b_{ij}, of tree T_i is the number of bases in the tree. Since initially the structure of the graph and number of bases is unknown, we consider a set of edge-consistent embedded trees $\omega_0 \in P_{DCR}$ such that $(b_{ij}, s_{ij}, b_{ji}, s_{ji}) = (0, |G_{\omega_0}|, 0, |G_{\omega_0}|)$. We de£ne the function err to be the number of edge £elds labeled 0. We propose a grammar Φ_{LB} such that $(\omega_0, \Phi_{LB}, \psi, u) \models \mathbf{FG} P_{LB}$. Our approach is to route commuters in the alleys to the highway where they can make locally optimal choices that distribute them evenly to the bases.

A. Grammar

Figure 7 displays the rule set, Φ_{LB}, designed for this purpose. Rule r_5 collects and propagates the strength and base mass information through the graph. Rule r_6 moves alley robots toward the highways. We de£ne the *pressure differential* of an edge in the ij direction by

$$\rho_{ij} = \frac{s_{ij}}{b_{ij}} - \frac{s_{ji}}{b_{ji}}.$$

As seen in Figure 8, if the pressure differential is greater than zero, rule r_7 moves the commuter into the subtree with the lowest pressure (where a subtree is chosen non-deterministically if all pressures differentials are equal and greater than zero.)

B. Proof of Correctness

Let $\langle C_i \rangle = 1$ if the number of commuters attached to i is greater than zero, and zero otherwise. And $[x] = x$ if $x > 0$ and $[x] = 0$ otherwise. As in the previous section, we associate a function with each rule, construct a lexicographic ordering among them, and propose Υ as a Lyapunov function for the grammar where

$$\begin{array}{rcl} \Upsilon_1 & = & err \\ \Upsilon_2 & = & \sum_{i \in A} \min_{j \in H} d(i,j) - 1 \\ \Upsilon_3 & = & \sum_{i \in H} \langle C_i \rangle \sum_{j \in \mathcal{N}_i} [\frac{s_{ij}}{b_{ij}} - \frac{s_{ji}}{b_{ji}}]. \end{array} \quad (7)$$

Rules r_6 and r_7 have guards of the form $||x_n - x_j|| < \Delta$ that guarantee the system remains edge-consistent when the rules insert edges. In order to guarantee progress we must show that these guards are always satis£ed in £nite time.

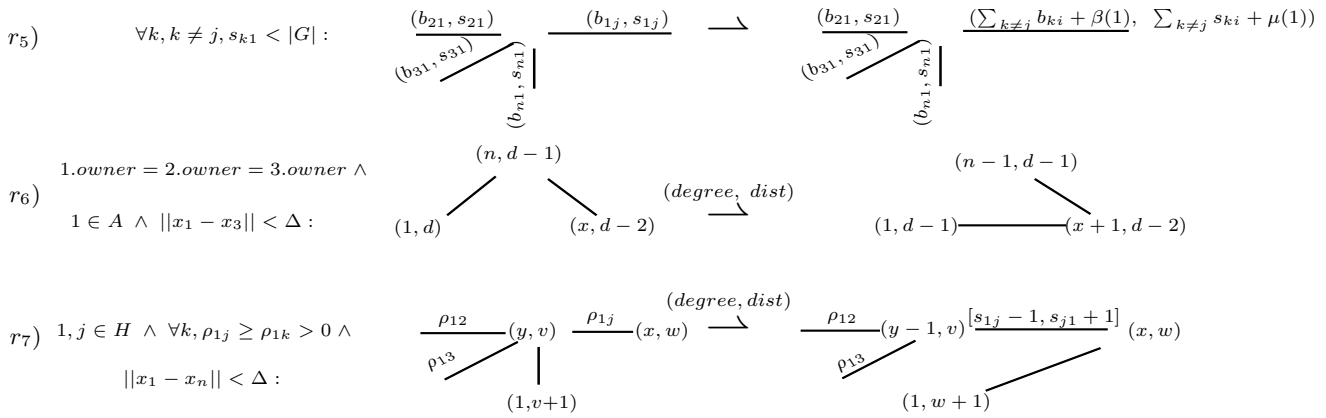


Fig. 7. Rule set for load balancing, Φ_{LB}. Note the inclusion of geometric conditions in the guards on r_6 and r_7. $\beta(i) \triangleq I(i.mode = base)$ and $\mu(i) \triangleq I(i.mode \neq base \wedge degree(i) = 1)$.

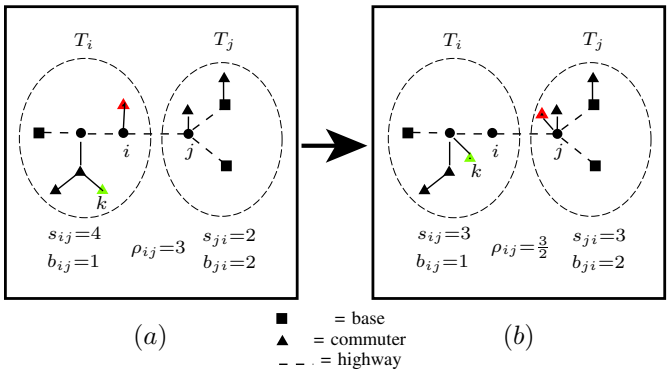


Fig. 8. Graph transition due to the concurrent application of rules r_6 and r_7. The £gure shows the value of the pressure differential, ρ_{ij}, between subtrees T_i and T_j (marked by dashed ovals). Rule r_7 moves a commuter from i to j, decreasing the pressure differential.

Proposition 7.1: Let γ be edge-consistent with a £xed topology G where $ij, jk \in E_G$ and i is a leaf. Then x_i converges to a set of points where $||x_i - x_k|| < \Delta$ in £nite time.

The proposition follows directly from Corollary 5.1 and the fact that $\dot{\mathcal{U}} < 0$.

Proposition 7.2: Υ as de£ned in Equation 7 is a Lyapunov function for the system $(\omega_0, \Phi_{LB}, \psi, u)$.

Proof: We must again show the following.

Υ is positive decreasing: Showing Υ_1 and Υ_2 decrease with respect to the lexicographical ordering $\prec$ is straightforward. Rule $r_7$is applicable when $[\frac{s_{ij}}{b_{ij}} - \frac{s_{ji}}{b_{ji}}] > 0$. Applying rule r_7 across edge ij decreases the value of the ρ_{ij} to $0 \leq [\frac{s_{ij}-1}{b_{ij}} - \frac{s_{ji}+1}{b_{ji}}]$. Since no other edges are affected, Υ_3 decreases.

If $\Upsilon \succ \mathbf{0}$ at least one action is applicable: Assume $\Upsilon \succ \mathbf{0}$ but no rule applies. The tree structure guarantees if $err > 0$, then rule r_5 is applicable, so it must be the case that $\Upsilon_1 = 0$. By Proposition 7.1, eventually the guard $||x_i - x_j|| < \delta$ for rules r_6 and r_7 is satis£ed. Thus if $A \neq \varnothing$, then r_6 applies. It follows that $\Upsilon_2 = 0$ and all the commuters are attached to the highway. However, if $\Upsilon_3 = \sum_{i\in H} \langle C_i \rangle \sum_{j\in\mathcal{N}_i} [\frac{s_{ij}}{b_{ij}} - \frac{s_{ji}}{b_{ji}}] > 0$, then r_7 applies because there is a commuter connected to a node with a pressure differential. This contradicts our original assumption, thus $\Upsilon \succ 0$ implies an action is applicable.

If $\Upsilon(G_k) = \mathbf{0}$, then $\Upsilon(G_n) = \mathbf{0}$ for all $n > k$: The guards on the rules r_6 and r_7 are false when $\Upsilon = \mathbf{0}$. Applying rule r_5 does not change the graph since all £elds are correct. ■

Theorem 7.1: $(\omega_0, \Phi_{LB}, \psi, u) \models \mathbf{FG}P_{LB}$.

Proof: We must show that if $\Upsilon = \mathbf{0}$ then for all $k \in B$, $|C_k| = \frac{|C|}{|B|}$. Assume $|C_k| < \frac{|C|}{|B|}$. Since $\Upsilon = \mathbf{0}$ implies all commuters are attached to bases, there is a base i such that $|C_i| > \frac{|C|}{|B|}$. Since $\Upsilon_3 = \sum_{i\in H} \langle C_i \rangle \sum_{j\in F(i)} [\frac{s_{ij}}{b_{ij}} - \frac{s_{ji}}{b_{ji}}] = 0$, this implies that $\frac{s_{ji}}{b_{ji}} = \frac{|C|}{|B|}$. The total strength of the graph is $|C| = |C_i| + |C_{G\setminus C_i}| = |C_i| + \sum_{ij\in E} b_{ji} \frac{|C|}{|B|}$. This gives rise to the contradiction $|C| - \frac{|C|}{|B|} < |C| - \frac{|C|}{|B|}$. Thus if $\Upsilon = \mathbf{0}$, then for all $i \in B$, $|C_i| = \frac{|C|}{|B|}$. Since Υ is a Lyapunov function, we may invoke Theorem 4.1. ■

VIII. Composition of EGG Systems

The preceding sections introduce two grammars, Φ_{DCR} and Φ_{LB} and a continuous control law u such that the system $(\gamma_0, \Phi_{DCR}, \psi, u) \models \mathbf{AF}P_{LB}$and $(\omega_0, \Phi_{LB}, \psi, u) \models \mathbf{AF}P_{LB}$. The correctness of each of the grammars is proved using discrete Lyapunov functions. In this section, we compose the grammars and show that for γ_0 with the edge labeling $(0, |G_{\gamma_0}|, 0, |G_{\gamma_0}|)$

$$(\gamma_0, (\Phi_{DCR} \cup \Phi_{LB}), \psi, u) \models \mathbf{FG}(P_{DCR} \cap P_{LB}). \quad (8)$$

A. Total System – Proof of Correctness

Building on previous results, we form a new function

$$\mathcal{X} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \Upsilon_1, \Upsilon_2, \Upsilon_3)$$

where $\mathbb{R}^6$ is lexicographically ordered. We must show that applying any rule from Φ_{LB} does not change the ordering property of Proposition 6.1 since this property is necessary to prove that $\mathcal{V}$ is a Lyapunov function. This is true since Φ_{LB} repairs the *dist* £eld when it moves a commuter. And we note that the convergence property in Proposition 7.1 still holds for the composition.

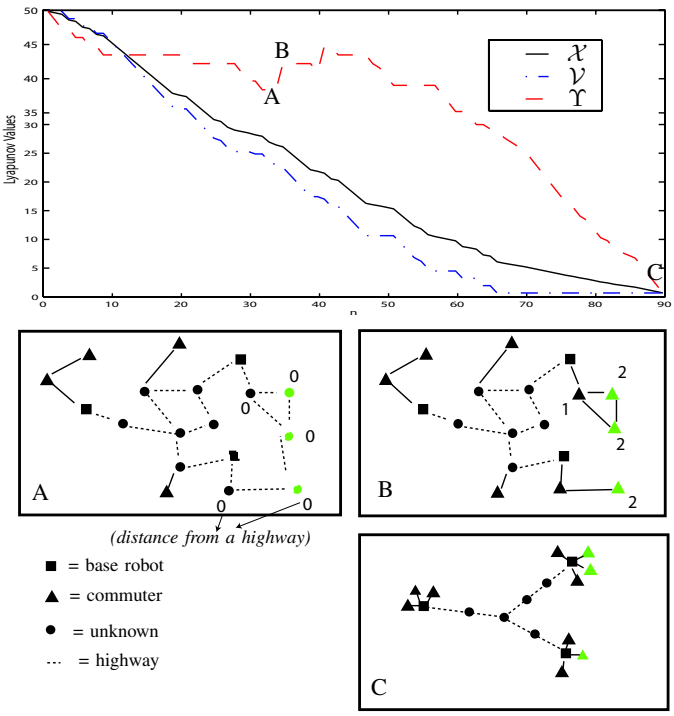


Fig. 9. Lyapunov functions for a representative run. A) Prior to an application of r_3. B) After an application of r_3. C) The £nal state. Note the lexicographic ordering allows $\mathcal{X}$ to be positive decreasing even though Υ increases at B.

To show that $\mathcal{X}$ is positive decreasing, under the ordering, we only need to show that applying a rule from Φ_{LB} does not increase $\mathcal{V}$. This is the case since Φ_{LB} does not alter the *owner* £eld, does not create cycles, and repairs the *dist* £eld. To show that $\mathcal{X} \succ \mathbf{0}$ implies an action is applicable we note that since the rules from Φ_{LB} have not changed the information used in Φ_{DCR}, $\mathcal{V} \succ \mathbf{0}$ implies that a rule is applicable. If $\mathcal{V}(G) = \mathbf{0}$, then the graph G is a tree and thus the Φ_{LB} grammar has a valid initialization point. Since Φ_{DCR} does not alter edge labels, $\mathcal{V} = \mathbf{0} \wedge \Upsilon \succ \mathbf{0}$ implies an action is applicable. Invoking the Lyapunov theorem proves the temporal logic statement in Equation 8. By the convergence properties proved in sectionV we have the desired result that for all trajectories $\sigma \in \mathcal{T}(\gamma_0, \Phi_{DCR} \cup \Phi_{LB}, \psi, u)$

$$\lim_{t \to \infty} \sigma(t) \in P_{DCR} \cap P_{LB} \cap P_{CMR}.$$

Note that Φ_{DCR} and Φ_{LB} are really compositions of single rules. Given a collection of smaller grammars, our design methodology is to: 1) Identify initialization conditions and invariant properties of those grammars such as the ordering property of Proposition 6.1. 2) Compose the grammars and show the composition satis£es the initialization and invariant properties. 3) Form a Lyapunov function for the new system under a lexicographic ordering that allows us to build upon previous results.

B. Simulation Results

We simulated the system in Matlab using cyclic initial graphs ranging in size from 30 to 70 vertices. Every simulation terminated with a correct topology and trajectories where the commuters were converging on the bases. Figure 9 shows a representative run and the three Lyapunov functions $\mathcal{V}, \Upsilon$ and $\mathcal{X}$. Note the £gure presents qualitative information since we use a normalizing scheme to represent the lexicographic ordering. One of the more persuasive arguments for our design is that the system can execute in a concurrent fashion as shown in Figure 9. The £gure displays the states just before and after an execution of rule r_3 where Υ increases because commuters are added to the alleys. However, by ordering the $\mathcal{V}$ elements of $\mathcal{X}$ before the Υ elements, we can guarantee that $\mathcal{X}$ decreases.

IX. Discussion

The embedded graph grammar formalism models networked hybrid systems with local restrictions on their interactions. We have shown a mapping from an embedded graph grammar to a non-deterministic hybrid automata. The automata may be large, unwieldy and may obscure information about the local processes controlling the evolution of the embedded graph grammar. Nonetheless using this mapping, we hope to £nd conditions on the structure of the embedded graph grammar that will allow us to build upon existing results in hybrid systems literature.

Additionally, we developed a notion of the composition of grammars in our proof. In future work, we will de£ne formally the semantics of graph grammar composition, examine which properties and behaviors are preserved in grammar composition, and develop formal design methods for composing grammars.

References

[1] Eric Klavins, Robert Ghrist, and David Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 2005. To Appear.
[2] A. Muhammad and M. Egerstedt. Connectivity graphs as models of local interactions. *Journal of Applied Mathematics and Computation*.
[3] J. Alexander Fax and Richard Murray. Graph laplacians and stabilization of vehicle formations. In *15th IFAC Congress*, 2002.
[4] Paulo Tabuada, George J. Pappas, and Pedro Lima. Motion feasibility of multi-agent formations. *IEEE Transactions on Robotics*, 2005.
[5] Meng Ji and Magnus Egerstedt. Connectedness preserving distributed coordination control over dynamic graphs. *Proceedings of the American Control Conference*, 2005.
[6] A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6), 2003.
[7] 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.
[8] John-Michael McNew and Eric Klavins. A grammatical approach to cooperative control. In R. Murphey O.Prokopyev and P. Pardalos, editors, *Cooperative Control and Optimization*. 2005. In press.
[9] Eric Klavins, Samuel Burden, and Nils Napp. Optimal rules for programmed stochastic self-assembly. In *Robotics: Science and Systems*, Philadelphia, PA, 2006.
[10] Reza Olfati-Saber and Richard M. Murray. Distributed structural stabilization and tracking formations of dynamic multi-agents. In *IEEE Conference on Decision and Control*, 2002.
[11] Eric Klavins. Self-assembly from the point of view of its pieces. In *American Control Conference*, 2006.
[12] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.
[13] J. LaSalle. Some extensions of Liapunov's second method. *IRE Transactions on Circuit Theory*, 1960.