Non-deterministic Reconfiguration of Tree Formations

John-Michael McNew   Eric Klavins

***Abstract*— We consider a network of mobile agents in an initially unknown acyclic network configuration and the problem of reconfiguring them into a desired network topology and formation geometry while maintaining connectivity in an asynchronous network. We model the system and solution as an *embedded graph grammar* and use a method of *lexicographically ordered Lyapunov functions* to show the system converges non-deterministically regardless of the initial network structure and the order of the communication events.**

I. Introduction

Systems of networked mobile agents of increasing complexity are found in automotive, aircraft, and military applications. Many of these systems are safety-critical and the design of correct, safe and fault-tolerant communication and control protocols that are robust to initialization is essential. The interplay between constraints on control, geometry, and communication and the asychrony in the network make designing solutions difficult. Verification of correct behavior is also problematic because the large state spaces.

To begin to address such issues, we examine a simple class of cooperative control algorithms that captures some of the complexity of these systems . The problem we consider exhibits hybrid dynamics over an asynchronous network, decentralized control under constrained communication, and requires robustness to unknown initial conditions and incomplete information.

In particular, we consider a network of mobile agents in an initially unknown acyclic network configuration and the problem of reconfiguring them into a desired network topology and formation geometry while maintaining connectivity in an asynchronous network. Our goal is to show that the system converges on the desired final state, regardless of the initial condition and the order of the communication events. We model the system as an *embedded graph grammar* and use a method of *lexicographically ordered Lyapunov functions* to show the system converges. We note that *embedded graph grammars* provide an appropriate level of abstraction in that the algorithm presented here involves explicit cooperation among groups of three agents, and so would be cumbersome to express as a message passing protocol between individual agents.

II. Previous Work

The formation control problem where the objective is to drive the formation error to zero is a central one in multi-agent control [1]. The relationship between graph structure and convergence is explored in [2] and [3] shows that formation control can be reformulated as a consensus problem. Solutions for formation control under communication and sensing constraints are proposed for centralized systems in [4] and decentralized system in [5].

Most of these results apply to formation control problems where the assignment of an agent to fulfill a role in the formation is known a priori. In this paper, we consider the dynamic assignment of roles in a network where the initial topology is unknown and where network connectivity limits allowable motion. The optimal assignment of agents to formation targets is examined in [6], where the problem is discretized over weighted graphs but communication constraints are not considered. The work in [7] frames the problem as an optimal control problem over target rotations, target translations and target permutations. Both papers require global information and the second suggests sub-optimal methods due to the intractability of the problem. Additionally [7], requires first the agents be driven to a stage where the communication graph is fully connected. Our paper presents a decentralized algorithm for restructuring the network which does not require the agents begin in any particular network structure.

In [8], a set of local interactions (a *graph grammar)* is synthesized to solve the *self-assembly* problem where isomorphic copies of a desired graph are assembled from an initially disconnected graph. *Embedded graph grammars*(EGGs) [9], [11] augment the graph grammar formalism by including local geometric pre-conditions on switching and continuous controllers to form an suitable for modeling cooperative control scenarios undergoing asynchronous communications. In this paper we present a control and communication protocol to drive a set of agents from an unknown *tree* formation to a desired one while maintaining connectivity in the graph using only local communication and control. Additionally, we simplify the notation for EGGs, building upon among others, the notation for I/O Automata [10].

III. Framework

A. Labeled Graphs

A *labeled graph* is a tuple $G = (V, E, l, e)$ consisting of a set V of vertices, E of edges, a function l assigning information $l(i)$ to each vertex $i \in V$, and a function e assigning information $e(ij)$ to each edge $ij \in E$. In this paper, a considerable quantity of information may be associated with each vertex or edge, and thus we use dot notation, common in data structures, to keep track of it. For example, if a vertex i has a field called *mode* having

JM McNew (jmmcnew@ee.washington) is a graduate student in Electrical Engineering at the University of Washington.

Eric Klavins (klavins@ee.washington.edu) is an Assistant Professor in Electrical Engineering at the University of Washington, Seattle, WA, 98195.

a value a, then we write $i.mode = a$. Similarly, if the edge ij has a field called *offset* taking values in $\mathbb{R}^2$, then we write, for example, $ij.\mathit{offset} = (1.1, 3.0)$. In summary, we use dot notation instead of a more cumbersome notation involving $l(i)$ or $e(ij)$. If $A \subseteq V$, we write $G[A]$ to be the subgraph of G induced by the vertices in A. We use T (for "tree") to represent a connected acyclic labeled graph. And we denote by $N_G(i)$ the neighbors of i in graph G.

The following definition allows comparison between graphs possessing possibly different vertex and edge label fields.

Definition 3.1: Suppose G and H are two graphs and Q is a set of vertex and edge label fields. We define an equivalence relation $\stackrel{Q}{\sim}$, where $G \stackrel{Q}{\sim} H$ if

1) (V_G, E_G) is isomorphic to (V_H, E_H) under some witness h and
2) For every $q \in Q$, if q is a vertex field, for every $i \in V_G$, $i.q = h(i)_H.q$. Otherwise for every $ij \in E_G$, $ij.q = h(i)h(j)_H.q$.

If this is true we say G is *structurally equivalent* to H over the fields in Q.

B. Robotic Networks and Communication

Consider a system of N kinematic agents, where each agent i has a continuous state $\mathbf{x}_i(t) \in \mathbb{R}^2$. We denote the continuous state of the entire system by $\mathbf{x}(t) \in \mathbb{R}^{2N} = X$. An *embedded graph* is the pair $(\mathbf{x}, G)$. We use embedded graphs to model the state of a robotic network as follows. The vertices of G represent the id's of the N agents. A vertex label, $l(i)$, abstracts the current software states, hardware configuration and operational mode of agent i. An edge label $e(ij)$ represents information required by pairs of cooperating robots (for instance inter-robot spacing constraints) and can only be altered by mutual agreement.

We capture the notion of communication constraints using a proximity function $\psi : X \times \mathcal{G} \to \mathcal{G}$. When its dependence on $(\mathbf{x}, G)$ is clear we write G_ψ to mean $\psi(\mathbf{x}, G)$ and call this graph the communication graph. For example, if robots can communicate when they are less than 10 meters apart and $H = \psi(\mathbf{x}, G)$, then $V_H = V_G$ and $ij \in E_H$ if and only if $||\mathbf{x}_i - \mathbf{x}_j|| < 10$. It is typically required that $E_G \subseteq E_H$, meaning that pairs of robots sending information can actually receive it.

A formation graph F is an undirected edge labeled graph containing a field $ij.\mathit{offset} \in \mathbb{R}^2$ and a field $ij.head \in V_F$ where if the vertices in an edge are i, j, we require that $ij.head$ takes values in $\{i, j\}$. We interpret these two fields as a constraint. For instance, if $(\mathbf{x}, F)$ is an embedded graph and $ij \in E_F$ such that $ij.head = j$, the constraint is satisfied when $\mathbf{x}_i - \mathbf{x}_j + ij.\mathit{offset} = \mathbf{0}$, that is when j is offset from i by $ij.\mathit{offset}$. If the constraint is satisfied for every edge in E_F, we say that $\mathbf{x}$ is *consistent* with F. By associating directionality with a label in an undirected graph, the graphs can never have edges ij and ji with contradictory offset fields.

A continuous controller u for a robotic network is a mapping from $X \times \mathcal{G} \to TX$ (the tangent space). A decentralized controller u is considered *well defined with respect to the communication constraints* if agent $i \in V$ can calculate its control from the subgraph induced by its neighbors in the graph $G \cap G_\psi$ [9].

IV. Embedded Graph Grammars

In this paper, the state of a system is represented by an embedded graph $(\mathbf{x}, G)$. The graph G, the discrete part of the state $(\mathbf{x}, G)$, can be operated on by *rules* that update G. A rule is expressed syntactically by

Rule r	
Vertices : $i_1, ..., i_k$	
Precondition:	
P_1	1
...	
P_m	m
Effect:	
$var_1 := value_1$	m+1
...	
$var_n := value_n$	m+n

The symbols $i_1, ..., i_k$ are *free variables* that can be instantiated by vertices in G. The *preconditions* $P_1, ..., P_m$ denote propositions about vertex labels, edge labels, vertex embedding (continuous state), or graph topology and use $i_1, ..., i_k$ as free variables. The *effects* $var_1 := value_1, ..., var_n := value_n$ are updates, using $i_1, ..., i_k$ as free variables, of vertex or edge fields, or they are updates to the graph topology. Examples of rules can be found in Section VI, where the tree transformation algorithm is described.

A rule r of the above form describes a relation $\xrightarrow{r}$ defined as follows. We write

$$(\mathbf{x}, G) \xrightarrow{r} (\mathbf{x}, G')$$

if (1) there exist vertices $j_1, ..., j_k \in V_G$ such that $P_1, ..., P_l$ evaluate to true in G when the free variables $i_1, ..., i_k$ are instantiated with $j_1, ..., j_k$ respectively; (2) G' is obtained from G by applying the updates $var_1 := value_1, ..., var_m := value_m$ instantiated the same way; and (3) all information and structure in G not mentioned in the updates is preserved in G'. Note that, in this paper, $\mathbf{x}$ does not change upon the application of a rule. A function h that maps each free variable i_k to a vertex $j_k \in V_G$ is called a *witness* and describes where the rule is applied.

A set $\Phi = \{r_1, ..., r_n\}$ is referred to as a *graph grammar*. If a (local) controller $u : X \times \mathcal{G} \to TX$ is also supplied, then (Φ, u) is referred to as an *embedded graph grammar*.

A *system* is a tuple $((\mathbf{x}_0, G_0), \Phi, u, \psi)$ where

1) The pair $(\mathbf{x}_0, G_0) \in X$ is the *initial state* is a set of allowable initial graphs
2) Φ is a graph grammar
3) u is a controller
4) ψ is a proximity function.

Systems produce trajectories in the usual hybrid fashion: (1) the continuous state evolves according to $\dot{\mathbf{x}} = u(\mathbf{x}, G)$; (2) the discrete part G of $(\mathbf{x}, G)$ can (but not must) change to G' if there exists a rule $r \in \Phi$ with $G \xrightarrow{r} G'$; (3) no rule in Φ can be infinitely often applicable without eventually being applied (i.e. *fair* trajectories); (4) all vertices instantiating the application of a rule r must be able to communicate in

We denote the set of trajectories of a system as $\sigma \in \mathcal{T}(\mathbf{x}_0, G_0, \Phi, u, \psi)$ where for each trajectory σ 1)$r^1_\sigma, r^2_\sigma, ..., r^k_\sigma, ...$ denotes the sequence of rules applied, 2)$h^1_\sigma, h^2_\sigma, ..., h^k_\sigma, ...$ denotes the witness sequence, 3) $\tau^1_\sigma, \tau^2_\sigma, ..., \tau^k_\sigma, ...$ denotes the times the rules are applied, and 4) $\mathbf{x}_\sigma(t)$ and $G_\sigma(t)$ denote the continuous and graph states at time t. We omit the subscript σ when clear.

V. Problem Statement

In the framework introduced in the previous section, the main problem of the paper can be stated as follows. Suppose T_{des} is a desired formation tree where each edge field $ij.offset$ is labeled with the desired formation offsets. Suppose F is the set of fields $e.offset, e.head$. The main goal is:

Task 5.1: Design (Φ, u) so that every trajectory converges to some $(\mathbf{x}_f, T_f)$ where $T_f \overset{F}{\sim} T_{des}$, and $\mathbf{x}_f$ is consistent with T_f.

Our approach in Section VI is to temporarily abstract away the continuous state and consider the following task.

Task 5.2: Given a desired formation tree T_{des} and any initial tree T_0, design a graph grammar Φ such that for every non-deterministic trajectory: 1) every graph in every trajectory is a tree, and 2) every trajectory has a final graph T_f, where $T_f \overset{F}{\sim} T_{des}$.

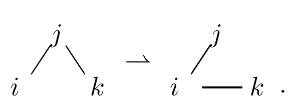
In Section VIII, we consider the continuous state and communication constraints. First we lift the solution Φ to Task 5.2 to an embedded graph grammar $\widetilde{\Phi}$ by adding the appropriate preconditions on geometry and connectivity in the communication graph. We then show what conditions $\psi, u,$ and $(\mathbf{x}_0, T_0)$ must satisfy in order for the system $((\mathbf{x}_0, T_0), \Phi', u, \psi)$ to achieve Task 5.1. This method allows us to build a single grammar that converges for a class of systems with different communication constraints.

Finally we show the effectiveness of decoupling of the asynchronous tree grammar from the continuous controller by proposing a ψ and u that satisfy these sufficient conditions. In Section IX simulation results for this specific choice of system are shown.

VI. A Graph Grammar for Universal Tree-to-Tree Conversion

The goal in this section is to solve Task 5.2. Since maintaining a connected spanning tree in formations of cooperating agents is a desirable property, the following simple lemma provides constraints on the type of rules we use.

Lemma 6.1: The set of all connected acyclic labeled graphs of size N is invariant to the application of rules of the form



Field	Description
$ij.offset$	The field $ij.offset \in \mathbb{R}^2$ holds the desired offset of j from i.
$ij.head$	The field $ij.head \in \{i, j\}$ establishes an order on the edge that is useful for formation control.
$ij.order$	The function $ij.order : \{i, j\} \to \{1, 2, ..., N\}$. In particular, $ij.order(i) = \lvert T^j_i \rvert$ and $ij.order(j) = \lvert T^i_j \rvert$
$i.tree$	The field $i.tree$ is either undefined (denoted $\perp$) or it contains some tree T_d where $T_d \overset{F}{\sim} T_{des}$. Additionally these trees contain fields $v.mode \in \{t, a\}$ and vw.order as above.
$i.role$	The field $i.role$ takes values in $V_{i.tree}$ or it is undefined, (denoted $\perp$). If $i.role = v$, the interpretation is that i assumes the role of v in $i.tree$.
$i.mode$	The labels $i.mode \in \{t, m, a\}$ are used by the solution grammar as follows. The label $i.mode = t$ indicates i should reconfigure its neighborhood to one structurally equivalent to the vertex identity in $i.role$. The label m implies that vertex i should merge two of its subtrees. The label a is the initialization label.

TABLE I

Label fields for the solution grammar Φ.

Proof: Suppose T is any tree. The graph $T - \{ij, jk, ik\}$ is a forest containing 3 trees T_i, T_j and T_k. The transformation above preserves the property that there is exactly one path between $i, j,$ and k and hence between T_i,T_j, and T_k. ■

We begin by reviewing results and terminology from previous work [9]. Suppose T is any tree and ij is any edge in E_T. If we remove the edge ij, two trees remain, one containing vertex j but not vertex i (denoted T^i_j) and the other containing i but not j (denoted T^j_i). In [9], we developed a graph grammar that given any connected graph G, marks G with a spanning tree T and labels each edge ij with $\lvert T^i_j \rvert$ and $\lvert T^j_i \rvert$. Here, we build on that approach and assume an edge field $ij.order$ contains these values.

The non-deterministic tree reconfiguration system is given by the following initialization condition T_0 and graph grammar Φ. Table I shows the edge and vertex label fields used by Φ. Note that in addition to the tree composed of the agents, the field $i.tree$ can be populated with a labeled tree corresponding to the desired topology. To avoid confusion, if T is a tree, we denote vertices corresponding to agents by $\{i, j, k...\} \in V_T$ and for any vertex $i \in V_T$, we denote vertices in $i.tree$ by $\{v, w, ...\} \in V_{i.tree}$.

Initialization: We define the set of initial trees $\mathcal{T}_0$, where $T_0 \in \mathcal{T}_0$ is defined over the fields in Table I,T_0 has only one vertex i labeled $i.mode = t$, $i.role = 1$, and $i.tree = T^0_d$ where T^0_d indicates vertex 1 is labeled $1.mode = t$ and all other vertices $w \in i.tree$ are labeled $w.mode = a$. All other vertices $j \in T_0$ are labeled $j.mode = a, j.role = \perp, j.tree = \perp$. Figure 1 panel (a) shows an initial graph.

Grammar: For any vertex i, if $i.role = v \neq \perp$, we denote by b_i

$$b_i = \max_{vw \in E_{i.tree} \mid w.mode = a} vw.order(w).$$

That is b_i is the largest branch in $N(v)$ remaining to be matched. Then the non-deterministic tree reconfiguration grammar Φ is shown in Table II.

Rule r_1

Vertices : i, j

Precondition:

$i.mode = t \wedge j.mode = a$ 1

Denote $i.role$ by v. $\exists w \in N_{i.tree}(v)$ such that $(w.mode = a \wedge vw.order(w) = ij.order(j))$ 2

$G[\{i,j\}] = i - j$ 3

Effect:

$w.mode := t, \quad j.tree := i.tree$ 4

$j.mode := t, \quad j.role := w$ 5

$ij.offset := vw.offset, \quad ij.head := j$ 6

Rule r_2

Vertices : i, j, k

Precondition:

$i.mode = t \wedge j.mode = a \wedge k.mode = a$ 1

$ij.order(j) > b_i$ 2

$jk.order(k) \geq b_i \vee ij.order(j) - jk.order(k) \geq b_i$ 3

$G[\{i,j,k\}] =$ $\begin{tikzcd}[column sep=small] & j \arrow[dl, dash] \arrow[dr, dash] & \\ i & & k \end{tikzcd}$ 4

Effect:

$G[\{i,j,k\}] :=$ $\begin{tikzcd}[column sep=small] & j \arrow[dl, dash] & \\ i \arrow[rr, dash] & & k \end{tikzcd}$ 5

Rule r_3

Vertices : i, j, k

Precondition:

$i.mode = t \wedge j.mode = a \wedge k.mode = a$ 1

$ij.order(j) < b_i \wedge ik.order(k) < b_i$ 2

$G[\{i,j,k\}] =$ $\begin{tikzcd}[column sep=small] & j \arrow[dl, dash] & \\ i \arrow[rr, dash] & & k \end{tikzcd}$ 3

Effect:

$G[\{i,j,k\}] :=$ $\begin{tikzcd}[column sep=small] & j \arrow[dl, dash] \arrow[dr, dash] & \\ i & & k \end{tikzcd}$ 4

Rule r_4

Vertices : i, j, k

Precondition:

$i.mode = t \wedge j.mode = a \wedge k.mode = a$ 1

$ij.order(j) > b_i$ 2

$jk.order(k) < b_i \wedge ij.order(j) - jk.order(k) < b_i$ 3

$G[\{i,j,k\}] =$ $\begin{tikzcd}[column sep=small] & j \arrow[dl, dash] \arrow[dr, dash] & \\ i & & k \end{tikzcd}$ 4

Effect:

$i.mode := m$ 5

Rule r_5

Vertices : i, j, k

Precondition:

$i.mode = m \wedge j.mode = a \wedge k.mode = a$ 1

$G[\{i,j,k\}] =$ $\begin{tikzcd}[column sep=small] & j \arrow[dl, dash] & \\ i \arrow[rr, dash] & & k \end{tikzcd}$ 2

Effect:

$G[\{i,j,k\}] :=$ $\begin{tikzcd}[column sep=small] & j \arrow[dl, dash] \arrow[dr, dash] & \\ i & & k \end{tikzcd}$ 3

$i.mode := a$ 4

TABLE II

RULES IN THE TREE RECONFIGURATION GRAMMAR Φ.

Description: The basic notion is that a vertex i labeled $i.mode = t$ uses operations allowed by Lemma 6.1 to rebalance the tree so that the neighborhood of i is locally structurally equivalent to T_{des}. In particular we require that $ij.order(j) = vw.order(w)$. When this is the case, the branch beginning with edge $ij \in T$ contains enough vertices to be transformed into the branch beginning with $vw \in T_{des}$ (although the topology of the subtree rooted at j may be quite different than topology of the subtree rooted at w). Groups of two or three agents may employ asynchronous communications to apply the rules of Φ and execute the following basic operations.

Pass a role– Rule r_1 passes a role and is applied to the tree in Figure 1, Panel (a) via the witness $h \triangleq \{i \mapsto i, j \mapsto k\}$. That is we apply rule 1 by replacing j with k and i with i. The tree representing the agents is seen above the dashed line. Clearly the tree satisfies the pre-condition $i.mode = t$ and $k.mode = a$. The value of $i.tree$ is pictured below the dashed line. Since $ik.order(k) = 1$, we satisfy the precondition in line 3 of rule r_1 by associating vertex w in the rule with vertex 4 in $i.tree$.

Figure 1 Panel (b) shows the new tree formed by applying the Effect of rule r_1. That is $5.mode$ is given the value t to create a new value of $i.tree$. Then $j.mode$ becomes t, $j.role$ becomes 4, $ik.offset$ is given the value of $14.offset$ and $ik.head$ becomes k. Now the edge ik is equivalent to the edge vw in the desired formation and agent k knows its role in reconfiguring the topology is that of vertex 4.

Split a branch– Suppose a branch beginning with edge $i - j$ with mode labels $t - a$ contains too many vertices to match any of the unmatched branches of $i.role \in i.tree$ (line 3 of the pre-condition of rule r_2). Suppose additionally, splitting the branch beginning with $i - j - k$ via r_2 results in one of the two branches containing at least as many vertices as b_i (line 4 of the pre-condition). This implies rule r_2 is applicable. Rule r_2 is applied to the tree in Figure 1 panel (b) via $h \triangleq \{i \mapsto i, j \mapsto l, k \mapsto j\}$ to yield the tree in panel(c).

Merge two branches– If i is labeled $i.mode = t$ and there are two branches beginning with edges ik, ij where $ij.order_j < b_i$ and $ik.order_k < b_i$. Then applying rule r_3 merges these branches into a larger branch (as shown in Figure 2(f)-2(g)). Alternatively when a branch $i - j - k$ with mode labels $t - a - a$ and $ij.order(j) > b_i$ cannot be split because the resulting branches would both be smaller than b_i, rules r_4 and r_5 are applied to merge subtrees of j (as shown in Figure 2(a) - 2(c)).

VII. PROOF OF CORRECTNESS

Theorem 7.1: Let T_{des} be any desired formation tree and $T_0 \in \mathcal{T}_0$ an initial tree such that $|T_{des}| = |T_0|$. Then every reachable graph of the system of the system (T_0, Φ) is a tree and every trajectory has a final graph T_f such that $T_f \overset{F}{\sim} T_{des}$.

The main thrust of this section is to show that even though the grammar contains both merge and split operations, no trajectories exhibit livelock or deadlock behaviors.

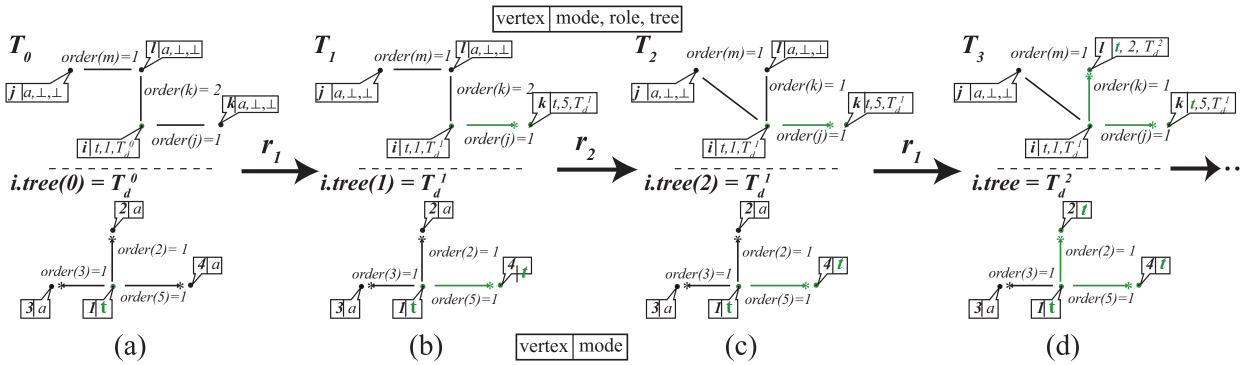


Fig. 1. A partial trajectory the sequence of graphs T_k representing the agents is shown above the dashed line. The value of the field $i.tree$ at time k is shown below the dashed line. The green edges and vertices indicate the growing isomorphism between T_k and $i.tree(k)$. For every edge ij, the relevant values of $ij.order(j)$ are shown. Panel (a) shows the initial graph where agent i is labeled with $i.role = 1$. Since $ij.order(j) = 1$ and $14.order(4) = 1$, rule r_1 is applied via the witness $i \mapsto i, j \mapsto k$ and vertex w in rule r_1 is associated with vertex 4. The resulting tree T_1 appears in panel (b). In panel (b), the largest unmatched branch order $b_i = 3$. Since $il.order(l) > b_i$, rule r_2 is applied yielding T_2 in panel (c). No update of $i.tree$ occurs. Finally rule r_1 is applied to il. Note in panel (d), $l.tree \neq k.tree$ but the structural equivalence witnessed by $i.role, l.role, k.role$ is consistent.

A. *Method of Proof*

In graph grammars the order in which actions are applied is non-deterministic. The resulting state spaces are large and directly exploring them using a method such as model checking is daunting [12]. In [9] we introduced the notion of a *lexicographically ordered discrete Lyapunov* function as a method for proving that systems converge to a desired set of graphs. We briefly describe the method here.

Definition 7.1: Suppose Φ is a set of rules and $A \subset \mathcal{G}$ is invariant to applications of rules in Φ. Let $\preceq$ be an ordering on $\mathbb{R}^k$ with an unique zero element. A function $\mathbf{U} : A \to \mathbb{R}^k$ is a *discrete Lyapunov function* for the graph grammar Φ if for all $G \in A$,

- *i* $\mathbf{U}(G) \prec \mathbf{0}$ implies at least one rule is applicable.
- *ii* $\mathbf{U}(G) = \mathbf{0}$ implies no rule is applicable.
- *iii* When $\mathbf{U}(G) \succ \mathbf{0}$, every applicable rule r decreases $\mathbf{U}$.

Theorem 7.2 (from [9]): Suppose (G_0, Φ) is a system, P is a set of desired final graphs, A is set of graphs invariant to the application of rules in Φ and U is a discrete Lyapunov function such that $\mathbf{U}^-1(\mathbf{0}) \subseteq P$. If $G_0 \in A$, then every trajectory converges to a final graph in P.

We use the *lexicographic ordering* $(\mathbb{R}^n, \preceq)$ defined by

$$(a_1, a_2, ..., a_n) \prec (b_1, b_2, ..., b_n)$$

if $a_1 < b_1$ or there exists an k such that $a_i = b_i$ for all $i \leq k$ and $a_{k+1} < b_{k+1}$. Additionally if any rule r is applied to any tree T, we denote by T' and $field'$ the new tree and the value of $field$ in the new tree.

B. *Invariant Set*

The following notation is useful in constructing the invariant set. For any tree, we denote by $\boldsymbol{V}_t(T)$ the set $\{i \in V_T \mid i.mode = t\}$. We define the function $\eta : \{i \in V_T) \mid i.mode = t\} \to V_{T_d^*}$ as $\eta(i) \triangleq i \mapsto i.role$. We next define the set of values that can appear in the field $i.tree$.

Definition 7.2: Define $T_d \in \mathcal{T}_{\text{des}}$ by: (1) $T_d \overset{F}{\sim} T_{des}$ under the identity mapping; (2) the label field $v.mode \in \{t, a\}$; and (3) $T_d[\boldsymbol{V}_t(T_d)]$ is a directed tree rooted at vertex 1.

We denote by $T_d^0 \in \mathcal{T}_{\text{des}}$ the unique tree where only vertex 1 is labeled by $l(1).mode = t$. We denote by $T_d^* \in \mathcal{T}_{\text{des}}$ the unique tree where every vertex v is labeled $v.mode = t$.

Definition 7.3: Define a set of trees $\mathcal{T}_{\text{agent}}$ having the fields in Table I, such that if $T \in \mathcal{T}_{\text{agent}}$ it satisfies the following labeling constraints.

- *i.* There exists $B \subset T_d^*$ such that $T[\boldsymbol{V}_t] \overset{F}{\sim} T_d^*[B]$ via the witness η.
- *ii.* If $i \in \boldsymbol{V}_t, T[\boldsymbol{V}_t \cap N(i)] \overset{F}{\sim} i.tree[N(i.role) \cap \boldsymbol{V}_t(i.tree)]$ via the witness η.
- *iii.* If $i \in \boldsymbol{V}_t$, then $i.role \in V_{T_d^*}$ and $i.tree \in \mathcal{T}_{\text{des}}$.
- *iv.* If $j.mode = m$, $N(j)$ contains exactly one vertex i such that $i \in \boldsymbol{V}_t$.
- *v.* If $i.mode = m$, then r_5 is applicable to i.

We next show that $\mathcal{T}_{\text{agent}}$ is an invariant set.

Lemma 7.1: If T is a connected acyclic graph, the application of any action in Φ results in a connected acyclic graph.

Proof: An inspection of the preconditions and effects of the reconfiguration rules in Table II shows they satisfy Lemma 6.1. ■

Lemma 7.2: If $T \in \mathcal{T}_{\text{agent}}$, then after the application of any rule, conditions (*i*), (*ii*) and (*iii*) of Definition 7.3 are true.

Proof: We need only consider applications of rule r_1, because it is the only rule to alter $\boldsymbol{V}_t$. Suppose r_1 is applicable to some $T \in \mathcal{T}_{\text{agent}}$ where the vertices y and z instantiate the vertices i and j in r_1. Condition (*i*) requires $T[\boldsymbol{V}_t(T)] \simeq T_d^*[B]$ via witness η for some B and condition (*ii*) guarantees that for all $i \in \boldsymbol{V}_t$, $T[\boldsymbol{V}_t \cap N(i)] \overset{F}{\sim} i.tree[N(i.role) \cap \boldsymbol{V}_t(i.tree)]$. Since line 3 requires $w.mode = a$, B clearly does not contain w. Since the effect in line 4 is $w.mode := t$, it follows that $T'[\boldsymbol{V}_t(T) \cup z] \simeq T_d^*[B \cup w]$. Additionally since this rewrite is recorded in the new value of $z.tree$ it follows that $T[\boldsymbol{V}_t \cap N(y)] \simeq y.tree[N(y.role) \cap \boldsymbol{V}_t]$. Furthermore, since $w \in N(v)$, condition (*iii*) holds. ■

Lemma 7.3: If $T \in \mathcal{T}_{\text{agent}}$, then after the application of any rule in Φ, condition (*iv*) and condition (*v*) of Definition 7.3 hold.

Proof: Suppose $T \in \mathcal{T}_{\text{agent}}$, $ij \in E$, $i \in \boldsymbol{V}_t(T)$ and $j.mode = m$. Since no rule change a $mode$ label of t, $\boldsymbol{V}_t(T) \subseteq \boldsymbol{V}(T')$. This implies vertex j is connected to at least one vertex labeled $mode = t$. Now suppose vertices y and z instantiate vertices i and j in rule r_4. The precondition of rule r_4 requires z be connected to a vertex y with $y.mode = t$ and the effect is to change $y.mode$ to m, thus y will be connected to at least one vertex labeled $mode = t$. Furthermore any vertex j with $j.mode = m$ can be connected to at most one vertex with $mode = t$ since condition (*ii*) and Definition 7.2 imply that $T[\boldsymbol{V}_t]$ is connected. Thus condition (*iv*) of Definition 7.3 is true. Since the precondition of rule r_4 line 3 implies the existence of two branches jk, jl, condition (*v*) is met. ■

Proposition 7.1: The set $\mathcal{T}_{\text{agent}}$ is invariant.

Proof: The proposition follows from Lemmas 7.1, 7.2, and 7.3 ■

C. Discrete Lyapunov Function

The following sets are useful in constructing a discrete Lyapunov function satisfying Definition 7.1.

Symbol	Description
$E_{ta} =$	$\{ij \in E \mid i.mode = t \wedge j.mode = a\}$. If rule r_1 can be applied, members of this set must be involved.
$E_{>b_i} =$	$\{ij \in E_{ta} \mid j.mode = a \Rightarrow ij.order(j) > b_i\}$. If rule r_2 or r_4 can be applied, a member of this set must be involved.
$E_{t\neg ta} =$	$\{i-j-k \in T \mid i.mode = t \wedge j.mode \neq t \wedge k.mode = a \wedge i-j \in gbi\}$. This is the number of size three branches beginning with a vertex labeled $mode = t$. If rules r_2, r_4, and r_5 can be applied, members of this set must be involved.
$E_{\geq b_i} =$	$\{ij \in E_{ta} \mid j.mode = a \Rightarrow ij.order(j) \geq b_i\}$.
$E_{<b_i}$	$\{ij \in E_{ta} \mid j.mode = a \Rightarrow ij.order(j) < b_i\}$. If rule r_3 can be applied, it must be applied to two members of this set.
$E_{tma} =$	$\{i-j-k \in E_{\text{No Split}} \mid j.mode = m\}$.

TABLE III

SETS USED IN THE DISCRETE LYAPUNOV FUNCTION, $\mathbf{U}$

Define a function $\mathbf{U} : \mathcal{G} \to \mathbb{R}^6$ as follows

- $U_1(T) = |T| - |\boldsymbol{V}_t|$. This is the number of vertices that have yet to be matched to the target graph.
- $U_2(T) = (U_1)(N - 1 - |E_{\geq b_i}|)$. The number of edges that can be used in an application of r_3.
- $U_3(T) = \frac{1}{|E_{>b_i}|} \sum_{E_{>b_i}} ij.order(j) - b_i$. The average distance from b_i by the branches that are too large.
- $U_4(T) = \sum_{E_{<b_i}} b_i - ij.order(j)$, the summed distance of all sites to which r_3 applies.
- $U_5(T) = |E_{t\neg ta}|$.
- $U_6(T) = |E_{t\neg ta}| - |E_{tma}|$

In general, the dependence of $\mathbf{U}$ on T is understood and so we will write $\mathbf{U}$ instead of $\mathbf{U}(T)$. We now show that this function meets the requirements of a discrete Lyapunov function for our system.

Lemma 7.4: For every $T \in \mathcal{T}_{\text{agent}}$, if $\mathbf{U} \succ \mathbf{0}$, then some action is applicable to T.

Proof: Assume to the contrary that $\mathbf{U} \succ \mathbf{0}$ but no rule is applicable. By condition (*v*) of Definition 7.3, $E_{tma} \neq \varnothing$ implies rule r_5 is applicable. Assume E_{tma} is empty, but $U_5 > 0$. Since any element of $E_{t\neg ta} - E_{tma}$ satisfies either the precondition of rule r_5 or of rule r_2, it must be that $U_5 = 0$ and $U_6 = 0$. But this implies that U_3 is zero. Since line 4 of rule r_4 is not satisfied, line 4 of rule r_2 is satisfied.

Now assume U_3, U_5, and U_6 are zero, but $U_4 > 0$ and $i.mode = t$. If r_3 is not applicable, then there can be at most one vertex j with $ij \in E_{ta}$ and $ij.order(j) < b_i$. Furthermore, if i is the head of k then $|T_k^i| - 1 = \sum_{ij \in E, j \neq k} ij.order(j)$. It follows that $U_4 > 0$ implies $E_{>b_i}$ is non-empty which contradicts our assumption that $U_3 = 0$. Thus $U_k = 0$ for $k \geq 3$ if no rule is applicable. However, we have shown that the sets $E_{<b_i}$ and $E_{>b_i}$ are empty. Therefore if $U_1 > 0$, then rule r_1 is applicable. Since $U_1 = 0$ implies $U_i = 0$ for all other i, $\mathbf{U} \succ \mathbf{0}$ implies an action is applicable. ■

Lemma 7.5: For every $T \in \mathcal{T}_{\text{agent}}$, $\mathbf{U} = \mathbf{0}$ implies no action is applicable.

Proof: When $U_1 = 0$, every vertex is labeled $i.mode = t$. Since the precondition for every rule contains at least one vertex not labeled $mode = t$, no rule is applicable. ■

Lemma 7.6: For every $T \in \mathcal{T}_{\text{agent}}$, if $\mathbf{U} \succ \mathbf{0}$, then the application of any action decreases $\mathbf{U}$.

Proof: We must show that if $T\ in \mathcal{T}_{\text{agent}}$ and $r \in \Phi$, then $\mathbf{U}(T) \prec \mathbf{U}(T')$. Table IV summarizes the information proved below, indicating the relative change for each U_i when each rule r_j is applied. Here we prove this by explicitly looking at an application of each rule.

r_1) Rule r_1 labels a vertex by $mode = t$, decreasing U_1.

r_2) Suppose $ij.order(j) > b_i$ is the order of the branch before r_2 splits the branch. There are two cases. In the first case, when the branch $i-j-k$ is split, $ij.order(j)' \geq b_i$ and $ik.order(k)' \geq b_i$, which implies $U_2' < U_2$. In the second case, $U_2' = U_2$ because only one branch (say $ij.order(j)'$) is greater than or equal to b_i. Since $ij.order(j)' < ij.order(j)$, U_3 decreases and U_1 is unchanged.

r_3) Suppose ij and jk are merged as in rule r_3 and $ij.order(j)' = ij.order(j) + ik.order(k)$. Suppose $ij.order(j)' < b_i$. Then $b_i - ij.order(j)' = b_i - (ij.order(j) + ik.order(k)) < b_i - ij.order(j) + b_i - |T_k^i|$, implying $U_4' < U_4$. Now suppose $ij.order(j)' > b_i$, then $|E_{\geq b_i}|' = |E_{\geq b_i}| + 1$, and $U_2' < U_2$.

r_4) Applying r_4 implies $U_6' = |E'_{\text{No Split}}| - |E'_{tma}| = |E_{\text{No Split}}| - (|E_{tma}| + 1) < U_6$. This relabeling does not alter U_i for $i < 6$.

r_5) Applying r_5 implies $U_5' = |E'_{\text{No Split}}| = U_5 - 1$ since the merge eliminates one branch. The operation does not alter U_i for $i < 5$.

Proposition 7.2: $\mathbf{U}$ under the *lexicographic ordering* $\preceq$ is a Lyapunov function for the grammar $\Phi$with respect to the invariant set $\mathcal{T}_{\text{agent}}$.

	$\mathbf{U_1}$	$\mathbf{U_2}$	$\mathbf{U_3}$	$\mathbf{U_4}$	$\mathbf{U_5}$	$\mathbf{U_6}$
$\mathbf{r_1}$	$\downarrow$	$\downarrow$	0	0	0	0
$\mathbf{r_2}$	0	$(0,\downarrow)$	$\downarrow$	?	?	?
$\mathbf{r_3}$	0	$(\downarrow,0)$	$(?,0)$	$\downarrow$	?	?
$\mathbf{r_4}$	0	0	0	0	$\downarrow$	$\uparrow$
$\mathbf{r_5}$	0	0	0	0	0	$\downarrow$

TABLE IV

DIRECTION OF CHANGE IN $U_1, ..., U_6$ WHEN RULES IN Φ ARE APPLIED.

Proof: The proof follows directly from Definition 7.1 and Lemmas 7.4, 7.5, and 7.6. ■

D. Proof of Theorem 7.1

For $T_f \in \mathcal{T}_{\text{agent}}$, $\mathbf{U} = \mathbf{0}$ implies every node is labeled by $i.mode = t$. By condition (i) of Definition 7.3, $T_f \overset{F}{\sim} T_{des}$. Proposition 7.1 establishes $\mathcal{T}_{\text{agent}}$ as the invariant set and clearly T_0 is in the invariant set. By Proposition **??** $\mathbf{U}$ is a Lyapunov function, the proof of Theorem 7.1 then follows from Theorem 7.2.

VIII. CONTINUOUS CONTROLLER SPECIFICATIONS

In the previous section, we developed an abstract graph grammar Φ for reconfiguring formation trees. The goal of the paper is formation reconfiguration and thus we want to construct an embedded graph grammar system $((\mathbf{x}_0, T_0), \widetilde{\Phi}, u, \psi)$ so that the system converges to the desired formation. In this section, we lift the graph grammar rules in Φ to a set of embedded graph grammar rules $\widetilde{\Phi}$.

Definition 8.1: For every rule in $r \in \Phi$ and for every edge ij that appears in r, we construct $\widetilde{r} \in \widetilde{\Phi}$ by adding the precondition $ij \in E_{G_\psi}$.

Proposition 8.1: Suppose a controller u and a proximity function ψ satisfy the following specifications

- *i* Local Safety–For any $(\mathbf{x}_0, T)$ such that $T \in A \subset \mathcal{T}_{\text{agent}}$ and $E_T \subseteq E_{T_\psi}$, if $\dot{\mathbf{x}} = u(\mathbf{x}, T)$ then for all $t > 0$, $ij \in E_T \subseteq E_{T_\psi}$.
- *ii* Local Progress–Suppose $T \in A \subset \mathcal{T}_{\text{agent}}$, $r \in \Phi$ and h maps L into T. If T is static and $\dot{\mathbf{x}} = u(\mathbf{x}, T)$, then there exists a t_f such that for all $t \geq t_f$ and all edges ij in rule r, $ij \in E_{G_\psi}$.

Then $G_\sigma \in \mathcal{T}(\mathbf{x}_0, T_0, \widetilde{\Phi}, u, \psi) \Leftrightarrow G_\sigma \in \mathcal{T}(T_0, \Phi)$.

Proof: Suppose there is a trajectory $G_\sigma \in \mathcal{T}((\mathbf{x}_0, T_0), \widetilde{\Phi}, u, \psi)$ that is not in $\mathcal{T}(T_0, \Phi)$. This implies that G_σ has a final graph T_f, but in the system (T_0, Φ) there exists a rule r whose precondition is satisfied by T_f. However, the local progress condition guarantees the pre condition of the embedded graph grammar rule r' in Definition 8.1 is satisfied. Furthermore any sequence can be generated simply by waiting until the desired action satisfies the geometric pre-condition of Definition 8.1. ■

Essentially, $\widetilde{\Phi}$ guarantees that when a rule is applied, if the rule adds an edge ij the controller u is still well-defined since $ij \in G_\psi$. This suggests that we may abstract the behavior of this hybrid system as a graph grammar on two types of edges. By adding rules that implement the changes in the communication graph implied by the conditions of Proposition 8.1 by adding or deleting edges in the communication graph, we can create a grammar that simulates the topological behaviors of the system $((\mathbf{x}_0, T_0), \widetilde{\Phi}, u, \psi)$.

Proposition 8.2: Suppose u satisfies Proposition 8.1. Additionally suppose that for any $T \in \mathcal{T}_{\text{agent}}$, $T \overset{F}{\sim} T_d^*$ and any $\mathbf{x}_0$ such that $E_T \subseteq E_{\psi(\mathbf{x}_0, T)}$. If the limit of $\mathbf{x}(t)$ as $t \to \infty$ under the dynamics $\dot{\mathbf{x}} = u(\mathbf{x}, T)$ is some $\mathbf{x}^*$ where $\mathbf{x}^*$ is consistent with T, then, the limit as $t \to \infty$ of every trajectory σ of a system $((\mathbf{x}_0, T_0), \widetilde{\Phi}, u, \psi)$, is some $(\mathbf{x}^*, T^*)$ where $T^* \overset{T}{\sim}_{des}$ and $\mathbf{x}^*$ is consistent with T^*.

Essentially the proposition says that local progress and local safety are sufficient until the tree is structurally equivalent to T_{des}.

Any number of proximity functions and controllers satisfy the requirements of Theorem 8.2. Here we consider the disk graph proximity function $\{ij \in E_{T_\psi} \Leftrightarrow |\mathbf{x}_i - \mathbf{x}_j| < \Delta\}$ In [5], graph based controllers for connectedness preserving formation control on the disk graph communication topology were introduced. The controllers have the form

$$\dot{\mathbf{x}}_i = u_i(\mathbf{x}, T) = -\sum_{j \in N(i)} \frac{2(\Delta - ||\mathit{off}||) - ||\mathbf{x}_i - \mathbf{x}_j - \mathit{off}||}{(\Delta - ||\mathit{off}|| - ||\mathbf{x}_i - \mathbf{x}_j - \mathit{off}||)^2} \tag{1}$$

where $\mathit{off} \in \mathbb{R}^2$ such that $||\mathit{off}|| < \Delta$. The global hybrid scheme introduced first sets the value of off on every edge to drive the agents towards consensus. Once the robots sense the global condition $\psi(\mathbf{x}, G) = K^N$, the variable off is given the desired formation offset values. The scheme we propose here uses the labeling changes propagated via the graph grammar to switch the value of off locally.

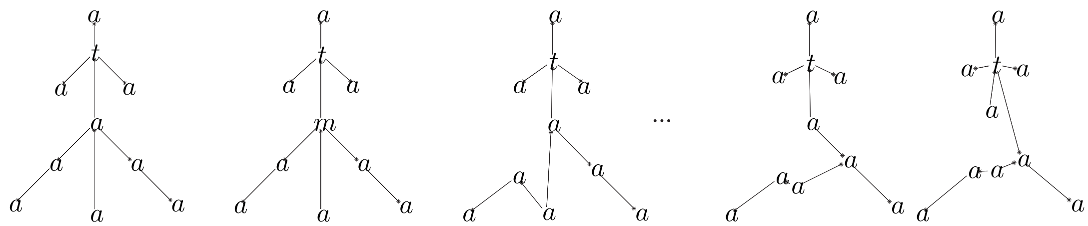
$$\mathit{off} = \begin{cases} (0,0)^T & \text{if } i.mode = t \text{ and } j.mode = a \\ ij.\mathit{offset} & \text{otherwise.} \end{cases} \tag{2}$$

Proposition 8.3: Suppose $T_0 \in \mathcal{T}_{\text{agent}}$, ψ is the disk graph, u is defined by Equations 1 and 2, and $\mathbf{x}_0$ satisfies $ij \in E_{T_0} \implies ij \in E_{G_\psi}$. Then every trajectory σ of the system $((\mathbf{x}_0, T_0), \widetilde{\Phi}, u, \psi)$ converges to some $(\mathbf{x}^*, T^*)$ where $T^* \overset{F}{\sim} T_{des}$ and $\mathbf{x}^*$ is consistent with T^*.

Proof: In [5], they introduce an edge tension $\mathcal{V}_{ij} = \frac{||\mathbf{x}_i - \mathbf{x}_j - off||}{\Delta - ||off|| - ||\mathbf{x}_i - \mathbf{x}_j||}$ and show that $\mathcal{V} = \sum_{ij \in E} \mathcal{V}_{ij}$ is a Lyapunov function for the system. In particular they show that

- *i* Since $V_{ij} \to \infty$ as $||\mathbf{x}_i - \mathbf{x}_j|| \to \Delta$, if T is static if $ij \in \psi(\mathbf{x}_0, T)$ then $\forall t > 0, ij \in \psi(\mathbf{x}(t), T)$ otherwise V must increase.
- *ii* If the local safety condition is met at time $t = 0$, then $\lim_{t \to \infty} \mathbf{x}(t)$ satisfies the formation constraint off on every edge. The proof is by LaSalle's invariance principle.

Since $\mathit{off} = (0,0)^T$ on edges ij where $i.mode = t$ and $j.mode = a$ and since $||off||$ is strictly less than Δ, there exists a finite time t_f satisfying the local progress condition of Proposition 8.1. ■



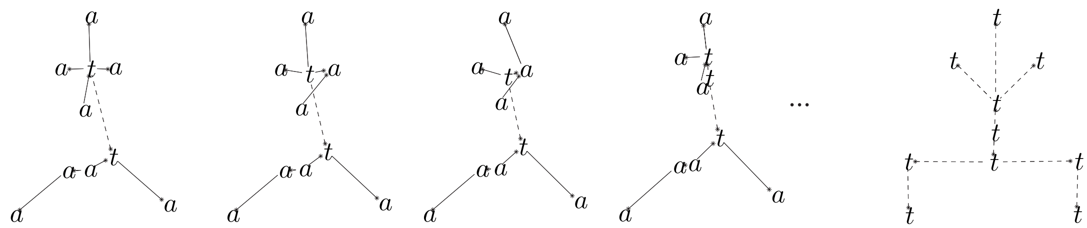
(a) Initial State. The letters indicate the value of the $mode$ label.

(b) Rule r_4 changes the mode label of a vertex to m.

(c) Rule r_5 merges two branches of the vertex labeled m in the previous snapshot.

(d) Another merge operation has occurred via rule r_4 and r_5.

(e) Rule r_2 splits a branch.



(f) Rule r_1 labels a vertex $mode = t$. Dashed edges occur between vertices in $\boldsymbol{V}_t$.

(g) Rule r_3 merges two branches.

(h) Another merge via r_3

(i) After one more merge, rule r_1 labels another vertex $mode = t$.

(j) The system converges to the desired tree and formation shown here.

Fig. 2. A sample trajectory for a 10 vertex system.

IX. Simulation

Matlab simulations of the system $(x_0, T_0, \Phi', u, \psi)$ were run on randomly generated initial trees and target trees, T_d. All simulations converged on the desired formation tree under isomorphism.

Figure 2 shows a sample trajectory for a system containing 10 vertices. Each rule is applied at least once. Figure 3 shows the value of U and the values of U_3 and U_4.

X. Future Work

There area number of obvious extensions to the simple grammar presented here. For instance, the reconfiguration algorithm we present requires a single initialization vertex, where in general one might prefer any number of vertices to begin the reconfiguration process. Another natural extension is to add rules to the grammar to make it more robust to the addition of deletion of vertices or edges. Furthermore, there are reasonable proximity functions where the elementary moves of the reconfiguration algorithm must result in disconnection in the communication graph. Whether or not one may make a general grammar for these systems similar to what was done here is not currently known.

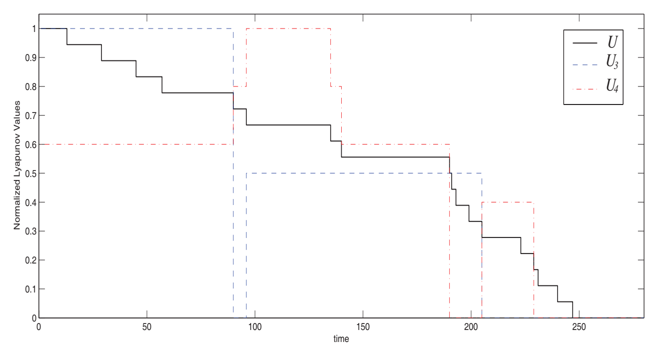


Fig. 3. Values of the discrete Lyapunov function U and components U_3 and U_4 for the trajectory in Figure 2. Note that U decreases with the application of every rule. Also note that whenever U_4 increases, U_3 decreases.

References

[1] B. Young J. Lawton, R. Beard. A decentralized approach to formation maneuvers. *IEEE Trans. on Robotics and Automation*, 2003.

[2] R. Olfati-Saber and R. Murray. Consensus problems in networks of agents with switching topologies and time delays. *IEEE Transactions on Automatic Control*, 2004.

[3] J. Alexander Fax and Richard Murray. Graph laplacians and stabilization of vehicle formations. In *15th IFAC Congress*, 2002.

[4] G. Pappas M. Zavlanos. Potential fields for maintaining connectivity of mobile networks. *IEEE Trans. on Robotics and Automation*, 2007.

[5] M. Ji and M. Egestedt. Distributed formation control whil preserving connectedness. In *Conference on Decision and Control*, 2006.

[6] M.E. Broucke. Disjoint path algorithms for planar reconfiguration of identical vehicles. In *American Control Conference*, 2003.

[7] S. Azuma M. Ji and M. Egerstedt. Role-assignment in multi-agent coordination. *International Journal of Human-friendly Welfare Robotic Systems*, 2006.

[8] Eric Klavins, Robert Ghrist, and David Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 2006.

[9] J. M. McNew and E. Klavins. Locally interacting hybrid sytems using embedded graph grammars. In *Proceedings of the Conference on Decision and Control*, 2006.

[10] N. Lynch. *Distributed Algorithms*. 1996.

[11] J. McNew, E. Klavins, and M. Egerstedt. Solving coverage problems using embedded graph grammars. *Hybrid System:Computation and Control*, 2007.

[12] John-Michael McNew and Eric Klavins. Model-checking and control of self-assembly. In *American Control Conference*, 2006.