A GRAMMATICAL APPROACH TO COOPERATIVE CONTROL

JOHN-MICHAEL MCNEW ERIC KLAVINS

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

In many cooperative control methods, the geometric state of the system is abstracted to the underlying graph or *network topology*. In this paper we present a grammatical approach to modeling and controlling the network topology of cooperative systems based on graph rewriting. By restricting rewrites to small subgraphs, *graph grammars* provide a useful method for programming the concurrent behavior of large decentralized systems of robots. We illustrate the modeling process through an ongoing example and demonstrate mathematical tools for reasoning about the system's behavior. Finally, we briefly describe methods to design continuous controllers that augment the grammar so that geometric requirements may also be satisfied.

1. Introduction

Inexpensive peer-to-peer networking technologies have spurred the investigation of control methods for large-scale networks of complex concurrent systems such as automated highway systems, air-traffic control systems and cooperative systems of robots. Traditional control objectives for individual plants such as stabilization are insufficient to capture the complex behaviors desired from these systems. Furthermore, the scale and complexity of these systems requires that local control of each robot, node, or subsystem must be used exclusively to produce the desired global behavior. In the natural world, members of decentralized systems often *self-organize* in response to environmental stimuli and to each other to produce complex global behaviors. One of the central questions for *engineered self-organization* is: Given a specification of a global behavior, can we synthesize a set of local controllers that produce that global behavior and are robust to uncertainties about the environmental conditions.

In many cooperative control methods, the geometric state of the system

is abstracted to the underlying graph or *network topology*. In this paper we focus our efforts on specifying and controlling the evolution of the network topology. In particular, we are interested in controlling the network topology using only local interactions. Some of the control problems that interest us include coordinating multiple vehicles, sequencing tasks in a concurrent environment, and reconfiguring the network topology. Most research in this area assumes a connected network. However, in this paper we are concerned with tasks that often require a partially disconnected network. Our point of departure is the use of *graph grammars* to model how the network topology changes due to local interactions among agents. The graph grammar model is amenable to many standard tools from concurrency theory, which can be used to show that systems meet their specifications.

In this paper, we examine systems that combine exploration and formation forming in response to environmental stimuli. In particular, we consider an example we refer to as "Wandering Scouts." In Section 3 we model this system as a *graph grammar*. In Section 4 we introduce notation to specify behaviors of graph transition systems. In Section 5 we use equivalence classes to partition the set of reachable graphs into *macrostates*. In Section 6.1 we adapt standard concurrency methods to the current setting and prove that for a class of initial systems, eventually it is always the case that the terminal graph of the system meets a desired criterion. In Section 6.2 we show that for a larger class of initial conditions, the grammar proposed has at least one trajectory where deadlock occurs. We augment the system and prove it meets the criterion for the larger class of graphs. Finally, in Section 7 we briefly explain how to design continuous controllers that use the topologies generated by the grammar to guarantee proper formation forming.

2. Related Work

One of the earliest compelling models of self-organization in a continuous state space is the local interaction model proposed by Reynolds to simulate bird flocking behavior. Reynolds motivates motion by three steering behaviors: separation, alignment and cohesion. More recently, Leonard and Fiorelli[9] analyze flocking behavior using potential function theory and Lyapunov methods. Fax and Murray[3] use graph theoretic methods to analyze the stability of such formations, while Tabuada et al[12] show which formation graphs have feasible non-trivial trajectories. In these efforts, the connection topology is fixed and the underlying graph is connected. Jad-

babaie, Lin and Morse[4] use ergodic matrix theory to demonstrate that under certain restrictions a discrete time simplification of the Reynolds model is stable for essentially arbitrary switching sequences.

With the occasional exception of a *leader* robot, these results utilize essentially homogeneous controllers on all the robots. We are interested in the concurrent execution of multiple tasks, thus we examine programmed switching between heterogenous controllers. Olfati-Saber[11] describes controlled switching of graph topology using a hybrid automaton for the purpose of squeezing through tight spaces. However, similar to most of the previous results, a fully *connected graph* is assumed. Since we are interested in scenarios where smaller teams of robots complete tasks concurrently, we model systems wherein the overall topology is not necessarily always *connected*.

Klavins[5] describes self-organization of robot formations as a graph process where the discrete states of robots are represented by symbols. Klavins, Ghrist, and Lipsky[7] introduce graph grammars to assemble prespecified graphs 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. Klavins et al.[5,6,7] demonstrate the use of *graph grammars* to define local interaction rules for assembly, replication and other tasks. An application of graph grammars to robotic systems is demonstrated wherein free-floating robots use graph grammars to assemble into larger structures in a predictable and robust manner[1].

3. Systems and Graphs

3.1. *A Motivating Example*

We informally present an example cooperative control scenario we refer to as "Wandering Scouts." Throughout the paper we use this example to illustrate the process of converting a system to a formal graph grammar model and the process of reasoning about that model.

Suppose a group of robotic scouts with only local communication and sensing capabilities patrols an area to protect against enemy incursions. If three robotic scouts surround an enemy agent, they can capture it, and transport it to a detention center. One possible strategy is to send out the scouts in teams of three. However, we do not know a priori the location or strength of the enemies. We choose rather to send out the scouts to patrol individually, thus covering a greater area. If a scout is in patrol mode and

Robotic Scouts	Enemies
w: a patrol or *wandering scout*	e: an undisrupted *enemy*
h: a pursuer or *hunter*	d_k: a *disrupted enemy* with degree k
l: a *leader*	c: a *captured enemy*
f: a *follower*	p: a detained enemy or *prisoner*

Figure 1. Operational modes and the associated labels for the robotic scouts and enemies.

senses an enemy, the scout chases it, thereby disrupting its activities. Once a scout is pursuing an enemy, it may recruit other nearby patrolling scouts to help encircle, capture, and transport the enemy to a detention center. There are four essential subtasks in our problem.

(1) Random patrol coverage,
(2) Disruption of the enemies' activities,
(3) Capture and transport of enemies, and
(4) Detention of enemies.

Informally, since we can neither specify the controllers and objectives of the enemy nor their initial density and spatial distribution, we often consider the enemies to be an "environmental stimuli". The graph topology for this system arises from local interactions between the robotic system and the environmental stimuli. Although there is no formal connection between the network topology and the spatial distribution, certain initial spatial distributions give rise to characteristic orderings of the local interactions.

3.2. *Graph Grammars*

A *simple labeled graph* over an alphabet Σ is a triple $G = (V, E, l)$ where V is a set of *vertices*, E is a set of *edges*, and $l : V \to \Sigma$ is a labeling function. In this paper, a graph is a model of the *network topology* of an interconnected collection of robots, vehicles or particles. A vertex x corresponds to the index of a robot. The presence of an edge xy corresponds to a physical and/or communication link between robots x and y. We use the label $l(x)$ of robot x to keep track of local information and also to indicate the operational mode of the robot.

Example 3.1. The labels in Figure 1 indicate the operational modes of the robotic scouts and the enemy agents. Additionally we denote by j a detention center or *jail*.

▲

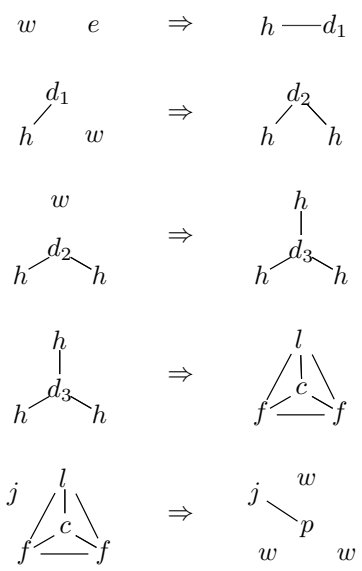

Figure 2. A grammar Φ for the wandering scouts example.

A *graph grammar* consists of a set Φ of rules. Each rule $r = (L, R)$ is a pair of labeled graphs over some small vertex set $V_L = V_R$. Let G be a larger graph representing a possible state of a system and let h be an injective, label and edge preserving map from V_L into G. We call h a witness. The pair (r, h) describes an action on G that produces a new graph $G' = (V, E', l')$ defined by

$$E' = (E - \{h(x)h(y)|xy \in E_L\}) \cup \{h(x)h(y) \mid xy \in E_R\}$$
$$l'(x) = \begin{cases} l(x) \text{ if } x \notin h(V_L) \\ l_R \circ h^{-1}(x) \text{ otherwise.} \end{cases}$$

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

Example 3.2. In Figure 2 we pose the rule set Φ as a way to model the wandering scouts system.

By convention we refer to the rules in the order they are displayed. So we refer to the rule at the top of Figure 2 as rule one or r_1, the next rule down as rule two or r_2 and so on. In our system, an edge between two vertices indicates one or both of the agents try to maintain a specified interagent distance. Thus, execution of the first rule, r_1, in Φ indicates

when a local interaction occurs between a patrolling scout w and an enemy e, the scout gives chase creating an edge and changing its mode to h. This disrupts the enemy's activities, thus its label changes to d_1. (The subscript "1" indicates that one robot is connected to the enemy). The second and third rules recruit additional robots to chase the enemy.

The fourth rule adds edges between the scouts. In the full system (i.e. including spatial aspects), the edges and labels in the right hand side of rule r_4 will be used by the scouts' continuous controllers to"encircle" the enemy and capture it. Since the system will ultimately use leader-follower formation control, rule r_4 also changes one robot's label to l and the other's to f. Note that we informally refer to the right hand side of r_4 as an *encirclement* component. Finally the fifth rule transfers the captured enemy to the detention center and returns the robotic scouts to patrol mode w. ▲

3.3. *Systems and Trajectories*

A *system* (G_0, Φ) consists of an initial graph G_0 and a set of rules Φ. A *trajectory* is a (finite or infinite) sequence

$$G_0 \xrightarrow{r_1,h_1} G_1 \xrightarrow{r_2,h_2} G_2 \xrightarrow{r_3,h_3} ...$$

where $r_i \in \Phi$. If the sequence is finite, then we require that there is no rule in Φ applicable to the terminal graph.

A system (G_0, Φ) defines a non-deterministic dynamical system whose states are the labeled graph over V_{G_0}. The system is non-deterministic since, at any step, many rules in Φ may be simultaneously applicable, each possibly via several witnesses. This results in a family of trajectories we denote by $\mathcal{T}(G_0, \Phi)$.

Example 3.3. Suppose N, M, and K are positive integers such that N is the number of vertices initially labeled by w, M is the number of vertices initially labeled by e and K is the number of vertices labeled by j. For the wandering scouts example, consider initial graphs of the form

$$G_0(N, M, K) = \{\{1, ..., N + M + K\}, \varnothing, l_0\} \tag{1}$$

where initially there are no edges (so that $E_0 = \varnothing$), and the initial labeling l_0 is defined by

$$l_0(i) = \begin{cases} w & i \leq N \\ e & N < i \leq N + M \\ j & N + M < i \leq N + M + K. \end{cases}$$

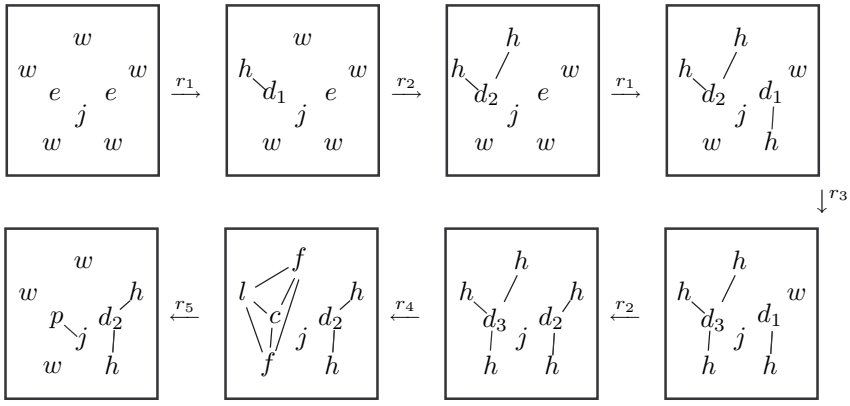

Figure 3. A trajectory of (G_0, Φ) demonstrating concurrent capturing of two enemies.

We can define the class of graphs of interest for the wandering scouts scenario by

$$\mathcal{G}_0 = \{G_0(N, M, K) \mid N, M, K \in \mathbb{N}^+ \wedge N \geq 3\}.$$

To illustrate we choose $G_0 \in \mathcal{G}_0$ with $N = 5, M = 2$, and $K = 1$. Figure 3 shows a partial trajectory of the system (G_0, Φ). Initially there are two enemies that the scouts must capture and transport. In this trajectory all of the scouts concurrently attempt to chase and capture the two enemies. This trajectory models the situation where the enemies and scouts are spatially interspersed.

Figure 4 shows a second possible trajectory of the system. In this trajectory neither of the scouts on the bottom attempts to chase or capture the enemy. This trajectory might correspond to a situation where a group of scouts captures and transports an enemy, then returns to patrolling. The next incursion of an enemy occurs in the area they are patrolling. Since we do not know beforehand the spatial distribution of the enemies it is important for our grammar to work for both of these types of trajectories. ▲

The set of all graphs reachable from G_0 via some trajectory is called the *reachable set* $\mathcal{R}(G_0, \Phi)$. The set of all connected components of graphs in $\mathcal{R}(G_0, \Phi)$ up to isomorphism is denoted $\mathcal{C}(G_0, \Phi)$. We suppose that each reachable component type has a single representative in $\mathcal{C}(G_0, \Phi)$. Let $\mathcal{G}$ be the set of all labeled graphs. The components of a grammar $\mathcal{C}(\Phi)$ are

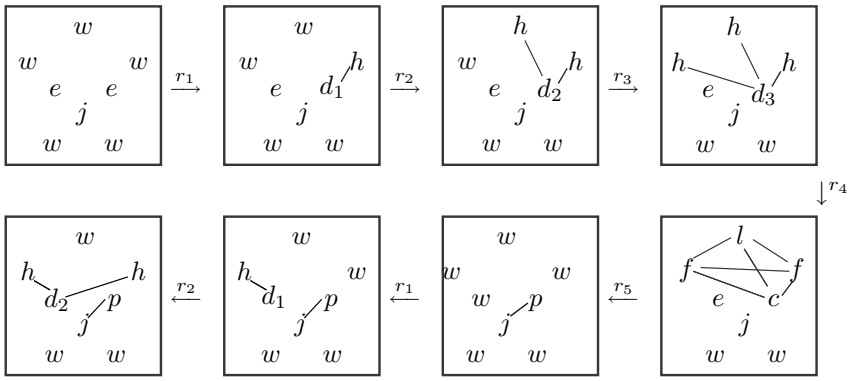

Figure 4. A partial trajectory of (G_0, Φ) illustrating the sequential capture of two enemies.

given by

$$\mathcal{C}(\Phi) = \bigcup_{G_0 \in \mathcal{G}} \mathcal{C}(G_0, \Phi).$$

If no rules in Φ can alter a reachable component, the component is said to be *stable*. The set of stable components of a grammar is denoted $\mathcal{S}(\Phi)$.

Example 3.4. Suppose $G_0 = G_0(5, 2, 1)$ is the initial graph defined in Example 3.3. The components of the system (G_0, Φ) are those pictured in Figure 5. The system only produces these components because there are only two enemies in the initial graph.

We will refer to the components of the grammar as $C_1, C_2, ...$ in the order they appear in Figure 5. The ellipsis indicates that while the set of components of the system $\mathcal{C}(G_0, \Phi)$ is finite, the set of components of the grammar, $\mathcal{C}(\Phi)$ is infinite. Specifically, it indicates a sequence of graphs C_k beginning at C_7 where C_k is a star-graph with a vertex labeled j at its center and all other vertices labeled p. Then C_{k+1} is a star graph with one more vertex labeled p.

▲

4. Propositions about Graphs

Let $\mathcal{G}$ be the set of all labeled, finite graphs. By a *proposition*, we simply mean a subset $P \subseteq \mathcal{G}$ of graphs. By defining propositions in this manner, we avoid having to define a syntax and semantics for logical statements about graphs. Informally, we will describe propositions by logical formula

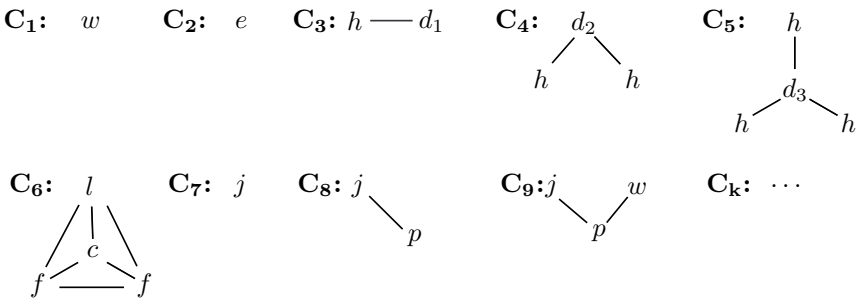

Figure 5. Components of the grammar Φ, $\mathcal{C}(\Phi)$.

and use double brackets to denote the set of graphs that satisfy the formula. For example,

$$[\![l(1) = b \wedge \exists x \exists y . xy \in E]\!]$$

denotes the set of graphs $G = (V, E, l)$ such that $1 \in V$, $l(1) = b$ and $E \neq \varnothing$. In general, any closed formula about labels and edges using finite quantification over V or E and using constant symbols for elements in V is permitted. If P is a proposition, the we define

$$(P, \Phi) = \{(G_0\Phi) \mid G_0 \models P\}$$

to be a class of systems.

Example 4.1. Define the proposition P_0 to be

$$P_0 = [\![G \in \mathcal{G}_0 \mid N > 2M]\!].$$

Then (P_0, Φ) denotes a restricted class of systems to which the system (G_0, Φ) defined in Example 3.3 belongs. ▲

Definition 4.1. Let $\sim \subseteq \mathcal{G} \times \mathcal{G}$ be an equivalence relation on $\mathcal{G}$. A proposition P is *preserved* by $\sim$ if, for all $G, G' \in \mathcal{G}$, if $G \sim G'$ then

$$G \in P \Leftrightarrow G' \in P.$$

If AP is a set of propositions, then $AP_\sim$ is the subset of propositions in AP that are preserved.

We often wish to know what propositions are preserved by a given equivalence relation. For example, suppose $\sim$ is the relation *labeled graph isomorphism*, denoted $\simeq$. Any proposition that can be represented by a formula not using constant symbols to represent vertices in V is preserved by $\simeq$.

This paper makes limited use of Linear Time Logic (LTL) to specify properties. In particular we use temporal logic formulas of the form

$$f = \mathbf{A}\ \mathbf{FG}\ P.$$

Here **A** is the path quantifier that denotes "Along all trajectories." The symbol F denotes "eventually" and G denotes "always." Thus for graph grammar systems the above formula reads "Along all trajectories it is eventually always the case that the next graph is in P." We write $(G_0, \Phi) \models f$ if the trajectories of the system are consistent with the formula f. Additionally if P_0 is a proposition we write

$$(P_0, \Phi) \models f$$

when for all $G_0 \in P_0$, $(G_0, \Phi) \models f$.

5. Macrostates

While graph grammars provide a method of programming individual robots, it is often easier to reason about grammars in the abbreviated notation of *macrostates*. Given a temporal logic formula over a set of propositions, $\{P_1, P_2 ... P_k\}$ it may be possible to find an equivalence relation in which all propositions in the set are preserved. McNew and Klavins[10] use equivalence relations to reduce the size of a graph grammar *model* to make it amenable to *model checking*. Here our goal is to find equivalence relations that preserve the underlying transition system, thus allowing us to reason about issues of progress and safety without reference to the details of rule application and the underlying graph.

The most obvious equivalence relation on graphs is graph isomorphism. This is quite natural given that the grammars we consider regard all vertices as essentially identical. In the context of self-organization, it is often useful to represent an equivalence class generated by the isomorphism relation by listing the number of each component type present in graphs in the class. Thus, suppose that $\mathcal{C}(G_0, \Phi) = \{C_1, C_2, ...\}$. Then $\mathbf{v} : \mathcal{C}(G_0, \Phi) \to \mathbb{N}$ represents all graphs $G \in \mathcal{R}(G_0, \Phi)$ with $\mathbf{v}(1)$ components isomorphic to C_1, $\mathbf{v}(2)$ components isomorphic to C_2 and so on. We write these representatives in vector notation. For the system presented in Example 3.3 where C_1 is a component of type w, $C_2 = e$ and $C_3 = d_1 - h$, and so on, the vector

$$\mathbf{v} = (\ 4,\ 0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0\ )^T$$

denotes that $\mathbf{v}(1) = 4$, $\mathbf{v}(3) = 1$, $\mathbf{v}(8) = 1$, and all other entries are zero. If $\mathbf{v}$ represents the equivalence class $[G]_\simeq$ of a graph G, we write $G \models \mathbf{v}$

to denote that G is consistent with $\mathbf{v}$ and we may write $\mathbf{v}_G$ instead of just $\mathbf{v}$. This highlights the fact that $\mathbf{v}$ is a proposition. Note that if $H \models \mathbf{v}_G$, then $H \simeq G$. In keeping with the self-assembly paradigm which is typically addressed in the context of statistical mechanics, we call $\mathbf{v}$ an *isomorphism macrostate*.

Suppose G and G' are reachable graphs where $\mathbf{v}_G$ and $\mathbf{v}_{G'}$ denote the associated isomorphism macrostates. Let (r, h) be the action such that $f_{r,h}(G) = G'$. Let $\mathbf{a} = \mathbf{v}_{G'} - \mathbf{v}_G$. For example, $\mathbf{a}$ may have the form

$$\mathbf{a} = (\ -1,\ -1,\ 1,\ 0,\ 0,\ 0, 0)$$

indicating that components of type C_1 and type C_2 are combined into a component of type C_3. If $\mathbf{a}(i) = m < 0$, then m components of type C_i are destroyed by applying the action (r, h). If $\mathbf{a}(i) = m > 0$, then m components of type C_i are created. We call the vector $\mathbf{a}$ a *macro-action*.

Definition 5.1. Fix a rule set Φ. Let (G_0, Φ) be a system. A macro-action $\mathbf{a}$ is in the *action set* of a system, $\mathcal{A}(G_0, \Phi)$, if there exists graphs $G, G' \in \mathcal{R}(G_0, \Phi)$, and an action (r, h) such that $f_{r,h}(G) = G'$ and $\mathbf{a} = \mathbf{v}_{G'} - \mathbf{v}_G$. The action set a grammar, $\mathcal{A}(\Phi)$, is given by

$$\mathcal{A}(\Phi) = \{\mathbf{a} \mid \mathbf{a} \in \mathcal{A}(G_0, \Phi) \text{ for some initial graph } G_0 \in \mathcal{G}\}.$$

Additionally we call the matrix whose columns are the actions in $\mathcal{A}(\Phi)$ (or $\mathcal{A}(G_0, \Phi)$) the *action matrix* denoted by $\mathbf{A}(\Phi)$ (or $\mathbf{A}(G_0, \Phi)$). We write $\mathbf{A}$ to denote the action matrix when its dependence on Φ and possibly G_0 is clear.

Proposition 5.1. *Let* $\mathbf{v}$ *be an isomorphism macrostate and* $\mathbf{a}$ *be an action in* $\mathcal{A}(\Phi)$. *Then* $\mathbf{a}$ *is applicable to* $\mathbf{v}$ *if and only if for every* i *such that* $\mathbf{a}(i) < 0$, $\mathbf{v}(i) + \mathbf{a}(i) \geq 0$. *Furthermore, the new macrostate is given by* $\mathbf{v}' = \mathbf{v} + \mathbf{a}$.

Example 5.1. For the system (G_0, Φ) in Example 3.3, the action matrix

$\mathbf{A}(G_0, \Phi)$ is given by

$$\mathbf{A}(G_0, \Phi) = \begin{pmatrix} -1 & -1 & -1 & 0 & 3 & 3 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (2)$$

▲

If the number of components in $\mathcal{C}(\Phi)$ is finite, then the macro-actions in $\mathcal{A}(\Phi)$ have finite dimension. Under this condition, it is often easier to reason about the macrostates. If this is not the case we often identify another equivalence relation that does result in macro-action vectors of finite length.

In Example 3.4 the set of component types of our grammar $\mathcal{C}(\Phi)$ is shown to be infinite. In particular, there exists an infinite sequence of star-shaped graphs with a vertex labeled j at the center and all other vertices labeled by p. In the final state we want all enemies to be labeled p with an edge to some vertex labeled j. The degree of the vertices labeled j is not important. By exploiting the fact that with respect to the desired behavior the star-graphs are essentially the same, we introduce a new equivalence relation, $\sim$ that creates macrostates and macro-action vectors of finite length.

Definition 5.2. Let $\mathbf{v}$ be any isomorphism macrostate. Suppose that $\mathcal{C}(G_0, \Phi) = \{C_1, C_2, ...\}$ ordered as in Figure 5. We denote a new *truncated macrostate* by $\widetilde{\mathbf{v}}$ where $\widetilde{\mathbf{v}}(i) = \mathbf{v}$ if $i < 7$ and $\widetilde{\mathbf{v}}(7) = \sum_{j=7}^{\infty} \mathbf{v}(j)$.

Thus for any initial graph G_0, any reachable graph in the system (G_0, Φ) may be expressed as a vector of length 7. For G_0 given in Example 3.3 the reachable isomorphism macrostate

$$\mathbf{v} = (\ 4,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1,\ 0\ )^T$$

becomes the truncated macrostate

$$\widetilde{\mathbf{v}} = (\ 4,\ 0,\ 1,\ 0,\ 0,\ 1)^T$$

Note that no rule changes a vertex labeled j to any other label, and when j appears in the left hand side of a rule it is disconnected from the

rest of the vertices in the rule. This implies that Proposition 5.1 is also true for macrostates and macro-actions derived from the $\sim$ equivalence relation.

6. Reasoning About Graph Grammars

For the class of systems (P_0, Φ) in Example 4.1 we wish to prove that

(1) at any given time a scout robot is either patrolling (labeled by w), disrupting (labeled by h and connected to an enemy) or capturing and transporting an enemy (labeled by l or f and connected to a c) and
(2) eventually all enemies are detained and remain so.

We may determine whether the rule set Φ in Figure 2 satisfies the first specification by simply examining all possible transition types. Rules r_1, r_2, and r_3 are the only rules whose left hand sides have w and each rule changes the vertex label to h with a connection to a disrupted enemy. Rule r_4 changes h to l or f while rule r_5 relabels vertices with l and f to w and disconnects them from the entire graph. Thus the first specification is met.

Similarly, we may show that the only labels possible for an enemy are $\{e, d_1, d_2, d_3, c, p\}$. And we may show that a vertex labeled p always remains connected to a vertex labeled j since there is no rule that deletes an edge between them. This result implies that the second specification may be written as a temporal logic statement in truncated macrostate notation: That is for any system whose initial graph has N scouts and K detention centers, the second specification can be written as

$$f = \mathbf{A}\ \mathbf{FG}\ \widetilde{\mathbf{z}}$$

where

$$\widetilde{\mathbf{z}} = (\ N,\ 0,\ 0,\ 0,\ 0,\ 0,\ K\ )^T. \tag{3}$$

The formula f states that eventually it is always the case that the system is in a macrostate $\widetilde{\mathbf{z}}$ with N copies of w, and K graphs, some of which have edges to vertices labeled p.

6.1. *Lyapunov Functions on Trajectories*

In proving that our system meets the second specification, we adapt some standard methods of reasoning about concurrent systems[8] to graph systems represented in macrostate notation.

Definition 6.1. A *discrete Lyapunov* function is a function on isomorphism macrostates $\mathcal{V} : \mathbb{N}^n \to \mathbb{N}$ such that

(1) $\mathcal{V}$ is a positive decreasing function over all trajectories,
(2) $\mathcal{V}(\mathbf{x}) = 0$ implies for all future states $\mathbf{v}$, $\mathcal{V}(\mathbf{v}) = 0$, and
(3) $\mathcal{V} > 0$ implies that at least one action (r, h) is applicable.

Note that our definition of a discrete Lyapunov function is related to, but not exactly equivalent to the standard definition found in discrete systems literature.

Proposition 6.1. *Let P be a proposition and $\mathcal{V}$ be a discrete Lyapunov function for a system (G_0, Φ) such that $V(G) = 0$ for some $G \in P$. Then $(G_0, \Phi) \models$ **A FG** P. In other words, along all trajectories it is eventually always the case that the current and next graphs are in P.*

Finding a function $\mathcal{V}$ that meets the requirements of Definition 6.1 is highly dependent on the system and the proposition P. We have the following results for the case when the desired proposition is an isomorphism macrostate or a combination of isomorphism macrostates. The results also apply to any type of macrostate for which Proposition 5.1 holds. Although we develop the following results in terms of isomorphism macrostates, they also apply to the truncated macrostates in Definition 5.2.

Proposition 6.2. *Let $\mathcal{A}(\Phi)$ be the set of possible macrostate actions for a system (G_0, Φ) . If there exists a vector $\mathbf{w} \in \mathbb{N}^n$ such that for all $\mathbf{a} \in \mathcal{A}(\Phi)$,*

$$\mathbf{w}^T \mathbf{a} < 0$$

then $\mathbf{x} \mapsto \mathbf{w}^T \mathbf{x}$ is a positive decreasing function on all trajectories in (G_0, Φ).

Example 6.1. For the class of systems (P_0, Φ), we propose the discrete Lyapunov function $\mathcal{V}(\widetilde{\mathbf{x}}) = \mathbf{w}^T \widetilde{\mathbf{x}}$ where $\mathbf{w}$ is given by

$$\mathbf{w} = (\ 0\ 5\ 4\ 3\ 2\ 1\ 0\ )^T. \tag{4}$$

The action matrix of Φ in truncated macrostate notation is given by

$$\mathbf{A}(\Phi) = \begin{pmatrix} -1 & -1 & -1 & 0 & 3 \\ -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{5}$$

We will often refer to the actions individually, so we note that $\mathbf{A}(\Phi) = (\mathbf{a}_1\ \mathbf{a}_2\ \mathbf{a}_3\ \mathbf{a}_4\ \mathbf{a}_5)$ Then

$$\mathbf{w}^T\mathbf{A} = -(\ 1,\ 1,\ 1,\ 1,\ 1\ ).$$

Thus Proposition 6.2 holds for our grammar, which implies that $\widetilde{\mathbf{x}} \mapsto \mathbf{w}^T\widetilde{\mathbf{x}}$ satisfies the first condition of the Lyapunov function definition. ▲

Proposition 6.3. *Let* $\mathbf{w}$ *be a vector as described in Proposition 6.2. If for all* $\mathbf{a} \in \mathcal{A}(\Phi)$ *there exists an element* i *such that* $\mathbf{a}(i) < 0$ *and* $\mathbf{w}(i) > 0$*, then*

$$\mathbf{w}^T\mathbf{x} = 0 \implies \mathbf{G}\ \mathbf{w}^T\mathbf{x} = 0.$$

Example 6.2. Let $\widetilde{\mathbf{z}}$ in Equation 3 be the desired final macrostate. Then for $\mathbf{w} = (\ 0\ 5\ 4\ 3\ 2\ 1\ 0\ )^T$, $\mathbf{w}^T\widetilde{\mathbf{z}} = 0$. A review of the action set for our system (i.e. the columns of the action matrix in Equation 5) demonstrates that for every action $\mathbf{a}$ there exists an element i such that $\mathbf{a}(i) < 0$ and $\mathbf{w}(i) > 0$. Thus the function $\widetilde{\mathbf{x}} \mapsto \mathbf{w}^T\widetilde{\mathbf{x}}$ satisfies the second condition of Definition 6.1. ▲

Example 6.3. We wish to show that whenever $\mathbf{w}^T\widetilde{\mathbf{x}} > 0$, then at least one action is applicable. Suppose there exists a vector $\widetilde{\mathbf{x}}$ where $\mathbf{w}^T\widetilde{\mathbf{x}} > 0$ but no action is applicable. Action $\widetilde{\mathbf{a}}_4$ is applicable for any macrostate $\widetilde{\mathbf{x}}$ such that $\widetilde{\mathbf{x}}(5) > 0$. Action $\mathbf{a}_5$ is applicable if $\widetilde{\mathbf{x}}(6) > 0$. Thus, we must show that for all $\widetilde{\mathbf{x}}$ with

(1) $\widetilde{\mathbf{x}}(5) = \widetilde{\mathbf{x}}(6) = 0$ and
(2) $\mathbf{w}^T\widetilde{\mathbf{x}} > 0$,

actions $\mathbf{a}_1, \mathbf{a}_2$, or $\mathbf{a}_3$ are not applicable. Under these conditions, either $\widetilde{\mathbf{x}}(2), \widetilde{\mathbf{x}}(3)$, or $\widetilde{\mathbf{x}}(4)$ must be non-zero. Action $\mathbf{a}_1$ is applicable if $\widetilde{\mathbf{x}}(2) > 0$ and $\widetilde{\mathbf{x}}(1) > 0$. Action $\mathbf{a}_2$ is applicable if $\widetilde{\mathbf{x}}(3) > 0$ and $\widetilde{\mathbf{x}}(1) > 0$. Action $\mathbf{a}_3$ is applicable if $\widetilde{\mathbf{x}}(4) > 0$ and $\widetilde{\mathbf{x}}(1) > 0$. Thus it must be the case that

$\widetilde{\mathbf{x}}(1) = 0$. Since there are M vertices initially marked e and since there is exactly one of these vertices in each component of type C_2, C_3, and C_4, we have the additional constraint that $\widetilde{\mathbf{x}}(2) + \widetilde{\mathbf{x}}(3) + \widetilde{\mathbf{x}}(4) \leq M$. Note that there is one robotic scout in component C_1, zero scouts in C_2, one scout in C_3, etc. Thus the number of robotic scouts in each component type is given by the vector

$$\mathbf{b} = (1\ 0\ 1\ 2\ 3\ 3\ 0).$$

Because the number of robotic scouts remains constant, for our class of initial graphs P_0 and for any macrostate $\widetilde{\mathbf{x}}$, we require that $\mathbf{b}^T\widetilde{\mathbf{x}} = N > 2M$. However,

$$\mathbf{b}^T\widetilde{\mathbf{x}} > 2M$$

subject to the constraints

$$\begin{aligned} \widetilde{\mathbf{x}}(2) + \widetilde{\mathbf{x}}(3) + \widetilde{\mathbf{x}}(4) &\leq M \\ \widetilde{\mathbf{x}}(5) &= 0 \\ \widetilde{\mathbf{x}}(6) &= 0 \end{aligned}$$

can only be satisfied if $\widetilde{\mathbf{x}}(1) > 0$, which is a contradiction of our supposition that $\widetilde{\mathbf{x}}(1) = 0$. Thus for all $\widetilde{\mathbf{x}}$, whenever $\mathbf{w}^T\widetilde{\mathbf{x}} > 0$, then at least one action is applicable.

For all systems (G_0, Φ) where $G_0 \in P_0$, the function $\widetilde{\mathbf{x}} \mapsto \mathbf{w}^T\widetilde{\mathbf{x}}$ meets all three conditions in Definition 6.1. We conclude that

$$(P_0, \Phi) \models \mathbf{A}\ \mathbf{FG}(N,\ 0,\ 0,\ ,0,\ 0,\ 0,\ K\ )^T.$$

▲

6.2. *Designing Grammars to Avoid Deadlock*

For many initial graphs, simple grammars like the one we describe in the previous section generate deadlock conditions on some of the system's trajectories. Often more complicated grammars are required to guarantee no deadlock occurs. Lyapunov functions are difficult to find for these grammars and we often use *weak Lyapunov* functions to prove such systems satisfy a specification on terminal behavior.

Example 6.4. In Section 6.1, we prove that

$$(P_0, \Phi) \models f.$$

where $f = \mathbf{A}\ \mathbf{FG}\ \widetilde{\mathbf{z}}$ and $\widetilde{\mathbf{z}} = (N,\ 0,\ 0,\ ,0,\ 0,\ 0,\ K\ )^T$. Since the size of the enemy force is unknown we would like to expand the class of initial graphs for which the grammar models f. Specifically, we would like to show that

$$(\mathcal{G}_0, \Phi) \models f.$$

However, we may show by counter example that this is not the case. Consider the initial graph $G_0(N, M, K)$ where $N = 3, M = 2$, and $K = 1$. Then the trajectory

$$\widetilde{\mathbf{v}}_0 \xrightarrow{\mathbf{a}_1} \widetilde{\mathbf{v}}_1 \xrightarrow{\mathbf{a}_2} \widetilde{\mathbf{v}}_2 \xrightarrow{\mathbf{a}_3} \widetilde{\mathbf{v}}_3 \xrightarrow{\mathbf{a}_4} \widetilde{\mathbf{v}}_4 \xrightarrow{\mathbf{a}_5} \widetilde{\mathbf{v}}_5 \xrightarrow{\mathbf{a}_1} \widetilde{\mathbf{v}}_6 \xrightarrow{\mathbf{a}_2} \widetilde{\mathbf{v}}_7 \xrightarrow{\mathbf{a}_3} \widetilde{\mathbf{v}}_8 \xrightarrow{\mathbf{a}_4} \widetilde{\mathbf{v}}_9 \xrightarrow{\mathbf{a}_5} \widetilde{\mathbf{v}}_{10}$$

where $\widetilde{\mathbf{v}}_{10} = (3,\ 0,\ 0,\ ,0,\ 0,\ 0,\ 1\ )^T$ satisfies f. Consider however the trajectory

$$\widetilde{\mathbf{v}}_0 \xrightarrow{\mathbf{a}_1} \widetilde{\mathbf{v}}_1 \xrightarrow{\mathbf{a}_2} \widetilde{\mathbf{v}}_2 \xrightarrow{\mathbf{a}_1} \widetilde{\mathbf{u}}_3$$

where $\widetilde{\mathbf{u}}_3 = (0,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1\ )^T$. No progress can be made from $\widetilde{\mathbf{u}}_3$ since no macrostate action applies to $\widetilde{\mathbf{u}}_3$. Thus the trajectory does not satisfy f. ▲

Example 6.5. We wish to define a new grammar Υ such that

$$(\mathcal{G}_0, \Upsilon) \models f.$$

We create Υ by adding the following rules to Φ.

$$\Phi' = \left\{ \begin{array}{lcl} \begin{array}{cc} & h \\ d_1 & \end{array} \quad d_1 & \Rightarrow & \begin{array}{cc} & h \\ e & \end{array} \quad d_2 \\ \begin{array}{cc} & h \\ d_1 & \end{array} \quad d_2 & \Rightarrow & \begin{array}{cc} & h \\ e & \end{array} \quad d_3 \\ \begin{array}{cc} & h \\ d_2 & \end{array} \quad d_2 & \Rightarrow & \begin{array}{cc} & h \\ d_1\,. & \end{array} \quad d_3 \end{array} \right.$$

The new grammar is $\Upsilon = \Phi \cup \Phi'$. The components of our new grammar are the same as the components of Φ so that $\mathcal{C}(\Upsilon) = \mathcal{C}(\Phi)$. Because the new rules do not involve labels p or j, we may still use the truncated macrostate notation developed in Definition 5.2. The action matrix of the new grammar is

$$\mathbf{A}(\Upsilon) = \left(\begin{array}{c|ccc} & 0 & 0 & 0 \\ & 1 & 1 & 0 \\ & -2 & -1 & 1 \\ \mathbf{A}(\Phi) & 1 & -1 & -2 \\ & 0 & 1 & 1 \\ & 0 & 0 & 0 \\ & 0 & 0 & 0 \end{array}\right). \tag{6}$$

▲

Systems with the grammar Υ require slightly different proof machinery leading to the following definitions and results.

Definition 6.2. Let $\mathbf{v}$ be a vector in $\mathbb{N}^m$. We call $\mathbf{v}$ an *application vector* since $\mathbf{v}(i)$ indicates that action $\mathbf{a}_i \in \mathcal{A}(\Phi)$ is applied $\mathbf{v}(i)$ times. If we denote $\mathbf{A}(\Phi)$ as $\mathbf{A}$, then the vector $\mathbf{Av} \in \mathbb{N}^n$ is the net change in components after applying the actions indicated by the application vector $\mathbf{v}$.

Proposition 6.4. *There exists a G_0 such that there is a cycle in the transition system of (G_0, Φ) if and only if the nullspace of the action matrix $\mathbf{A}$, Null($\mathbf{A}$), contains a vector n where the entries in the vector are all non-negative.*

Definition 6.3. Let $\mathbf{z}$ be a desired final isomorphism macrostate. Let $\mathbf{w}$ be a vector in $\mathbb{N}^n$ such that $\mathbf{w}^T\mathbf{z} = 0$ and for all actions $\mathbf{a}_i \in \mathcal{A}(\Phi)$, $\mathbf{w}^T\mathbf{a}_i \leq 0$. We call the function $\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x}$ a *weak Lyapunov function.*

Proposition 6.5. *Fix a grammar Φ, and let $\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x}$ be a weak Lyapunov function for desired macrostate $\mathbf{z}$. If*

(1) The set of actions of the grammar $\mathcal{A}(\Phi)$ cannot generate a cycle,
(2) For all $\mathbf{a} \in \mathcal{A}(\Phi)$ there exists an element i such that $\mathbf{a}(i) < 0$ and $\mathbf{w}(i) > 0$, and
(3) Whenever $\mathbf{w}^T\mathbf{x} > 0$, then at least one macrostate action is applicable,

then

$$(G_0, \Phi) \models \mathbf{A}\ \mathbf{FG}\ \mathbf{z}.$$

Example 6.6. Let $\mathbf{w} = (\ 0\ 5\ 4\ 3\ 2\ 1\ 0\ )^T$ as before. Let $\mathbf{A} = \mathbf{A}(\Upsilon)$ given in Example 6. Then $\mathbf{w}^T\mathbf{A} = -(\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0\ )$. Thus $\widetilde{\mathbf{x}} \mapsto \mathbf{w}^T\widetilde{\mathbf{x}}$ is

a weak Lyapunov function. A basis for the nullspace of $\mathbf{A}$ is given by the vectors

$$\left\{ \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right\}.$$

Clearly there are no elements of the nullspace with all non-negative entries. Thus no cycle can be generated from the actions in $\mathcal{A}(\Upsilon)$. In Example 6.2 the second condition of Proposition 6.5 is shown to be true for the actions $\{\mathbf{a}_1, ..., \mathbf{a}_5\}$. For every new action in Equation 6,there exists an element i such that $\mathbf{a}(i) < 0$ and $\mathbf{w}(i) > 0$. Thus the second condition is met for the function $\widetilde{\mathbf{x}} \mapsto \mathbf{w}^T\widetilde{\mathbf{x}}$.

Finally, we must show that whenever $\mathbf{w}^T\widetilde{\mathbf{x}} > 0$, then at least one macro-action is applicable. Assume no macro-action is applicable to $\widetilde{\mathbf{x}}$. From the previous analysis in Example 6.3 we know if no macro-action applies to $\widetilde{\mathbf{x}}$ then $\widetilde{\mathbf{x}}(1) = 0, \widetilde{\mathbf{x}}(5) = 0$ and $\widetilde{\mathbf{x}}(6) = 0$. The three new actions imply that if no action is applicable either $\widetilde{\mathbf{x}}(3) = 1$ and $\widetilde{\mathbf{x}}(4) = 0$ or $\widetilde{\mathbf{x}}(4) = 1$ and $\widetilde{\mathbf{x}}(3) = 0$. Since component C_2 contains zero robotic scouts, C_3 contains one robotic scout, and C_4 contains two robotic scouts, if the number of robotic scouts N in the initial graph is greater than two, then clearly there is a contradiction. Thus $\mathbf{w}^T\widetilde{\mathbf{x}} > 0$ must imply at least one macrostate action is applicable.

The grammar Υ, the class of graphs $\mathcal{G}_0$ and the proposed weak Lyapunov function $\widetilde{\mathbf{x}} \mapsto \mathbf{w}^T\widetilde{\mathbf{x}}$, satisfy all three conditions in Proposition 6.5. Thus we may conclude that

$$(\mathcal{G}_0, \Upsilon) \models f.$$

That is, for any initial condition with at least three robotic scouts, eventually all the enemy will be detained. ▲

7. Simulation Results

In the wandering scouts example, the graph grammar describes the possible evolution of the network topology of a group of robots. A graph grammar does not, however, describe geometry. To incorporate geometry, we must

also design continuous controllers, a process we refer to as *embedding* the graph grammar. In the embedding a continuous state $x \in \mathbb{R}^2$ is associated with each vertex. In this section we use "robots" to discuss spatial information and simply vertices if discussing purely topological concepts. For the system in the wandering scouts example, we suppose the presence of an edge between two robots i and j indicates i and j have a communication link and can detect one another's labels. Without an edge, a robot cannot know the operational modes of its neighbors except during the isolated moments in which rules are being locally checked.

We are interested in scenarios where the robots have limited communication and sensing ranges. The primary issues that must be addressed when designing the continuous controllers are:

(1) Designing controllers that use the network topology to enforce geometric conditions such as the *encirclement* condition necessary to capture an enemy.
(2) Designing controllers that guarantee that if progress is possible from a graph G in the graph grammar, it is eventually possible from any spatial state with the same underlying graph G.

We briefly describe a simulation of the wandering scouts example and the continuous controllers used to embed the grammar Υ. In future papers we will present a more formal description of the embedding process in terms of hybrid systems. But these issues are beyond the scope of the current paper.

Suppose r_{ij} denotes the Euclidean distance between two robots i and j. Denote by r_c the communication and sensing radius of the robots. Let $\mathcal{U}$ be an attractive-repulsive potential function.

$$\mathcal{U} : V \times \Sigma \times \Sigma \times E \times \mathbb{R}^{2n} \to \mathbb{R}.$$

Here n is the total number of vertices. Suppose i is the vertex of a robot and j is any other vertex. If the pair of vertices ij is not in the edge set E or if $r_{ij} > r_c$, then $\mathcal{U}(i, l(i), l(j), ij, r_{ij}) = 0$. Otherwise, $\mathcal{U}$ has the form pictured in Figure 6(a).

In the figure, r^* denotes a desired interagent distance between i and j. The maximum separation distance is denoted by r^+. The discontinuity at r^+ is intended to enforce the condition that once an edge is formed it is never broken by moving outside the communication range. The discontinuity at r^- is intended to enforce collision avoidance. For any edge, the parameters r^*, r^+, and r^- may be different for different label pairs.

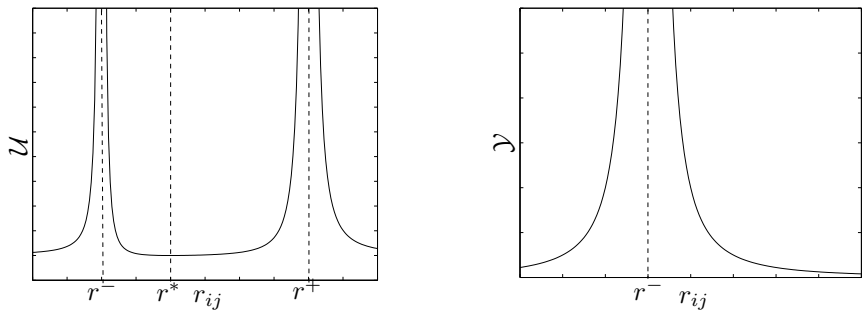

Figure 6. (a) Attractive-repulsive potential function $\mathcal{U}$ for the Wandering Scouts Scenario, (b) Repulsive potential function $\mathcal{Y}$.

We also define a purely repulsive potential function

$$\mathcal{Y} : V \times \Sigma \times E \times \mathbb{R}^{2n} \to \mathbb{R}.$$

If $ij \in E$, then $\mathcal{Y} = 0$, otherwise it has the form shown in Figure 6(b). Here r^- is only a function of a robot's own label.

The dynamics of the ith robot are given by

$$\dot{x}_i = - \sum_{\{j|ij \in E\}} \nabla \mathcal{U}(i, l(i), l(j), ij, r_{ij}) - \sum_{\{j|ij \notin E\}} \nabla \mathcal{Y}(i, l(i), ij, r_{ij}) + W.$$

W is continuous random vector process of bounded size that helps guarantee the motion of any two components is only correlated for short periods of time. If this is true and assuming a bounded spatial domain, then with high probability, every macrostate action that is possible in the grammar is also possible in the embedded system.

The function $\mathcal{U}$ has a local minimum at r^*. However the region of attraction is limited to $r^- < r_{ij} < r^+$. In fact outside of this region, $-\nabla \mathcal{U}$ may drive r_{ij} away from r^*. Since $\mathcal{U}$ is only non-zero when there is an edge, and edges are created or destroyed via the application of rules, we require that a rule can only be applied when $r^- < r_{ij} < r^+$. We accomplish this by placing guards on the rules that are boolean functions of the geometry.

Additionally, note that we assume for progress to be guaranteed the motion of the components may only be correlated for small periods of time. Because the potential fields are both attractive and repulsive, if certain geometric conditions are not met, it is possible for components to become permanently interlinked. In particular for any edges ij and kl, we must prevent the embedding of those edges from crossing.

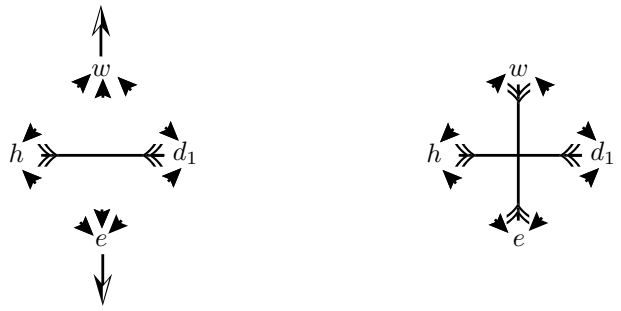

Figure 7. (a)Virtual forces on robots before the application of rule r_1 (b) Undesirable stable equilibrium after the application of rule r_1.

Figure 7 demonstrates the simplest case involving an edge existing between two vertices marked h and d_1. The other vertices marked w and e can apply rule r_1 of the rule set Υ. In panel (a) of Figure 7 the double arrowheads along the edges indicate the direction of the virtual forces generated by $-\nabla\mathcal{U}$. The solid arrowheads show the repulsive virtual forces generated by $-\nabla\mathcal{Y}$. The half-filled arrows indicate the net virtual forces applied to each robot. As these arrows show, the net repulsive force on the robots labeled w and e will eventually drive those robots out of communication and sensing range, thus this configuration is not stable. Note that there are not half-filled arrows associated with the vertices marked h and d_1 because in this configuration the sum of the attractive and repulsive forces is zero. Panel (b) shows how the forces change if rule r_1 is applied. For each of the four robots, the virtual forces sum to zero. Thus this configuration represents an undesirable stable equilibrium in which the motion of the two components will remain correlated. Suppose h is the label preserving injective mapping of L_{r_1} into G. To ensure an application of the first rule in Υ creates the proper geometry and that progress is guaranteed, we place the following guard, gd_1, on rule r_1.

$$gd_1 \Longleftrightarrow r^- < r_{h(1)h(2)} < r^+ \bigwedge_{k,m \in sense(h(L))} \neg cross(h(1), h(2), k, m).$$

A robot with vertex k is in the set $sense(h(L))$ if k is not in $h(L)$ and the distance between i and k, r_{ik} is less than the sensing distance, r_c. The boolean function *cross* determines if possible embedded edges between two

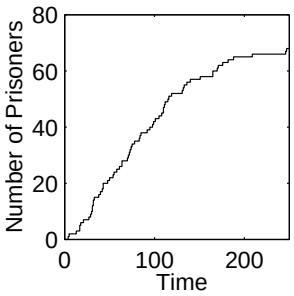

Figure 8. The number of captured enemy over time.

pairs of robots cross.

By carefully defining the guards on the rule set Υ and by carefully designing the potential functions $\mathcal{U}$ and $\mathcal{Y}$ we can guarantee that: The only way of changing the network topology is through the application of rules, the network topology and controllers result in the desired geometries, and with high probability progress always occurs.

We created a MATLAB simulation of the wandering scouts scenario utilizing the potential function controllers $\mathcal{U}$ and $\mathcal{Y}$ and the grammar Υ enhanced by guards. We ran simulations of systems with initial graphs in $\mathcal{G}_0$ ranging in size from 20 to 500 vertices and various distributions of scouts, enemies and jails. Figure 8 shows the number of captured enemies over time for a representative run of a system with 200 robotic scouts, 80 enemies and 5 detention centers. We chose purely random motion in the wandering scouts mode w as the patrol strategy. The decreasing rate of prisoner detention occurs because the likelihood of randomly encountering an enemy decreases as the number of enemy not detained decreases. A more structured patrol strategy[2] might result in faster convergence to the point where all enemy are captured. However, all scenarios regardless of distribution demonstrated converging behavior. Therefore we conclude that the simulation of the embedding of the grammar is consistent with the behavior of the grammar.

8. Discussion

In Section 6.1 we created a one-of-a-kind proof for the simple wandering scouts scenario using Lyapunov methods. These methods are often useful for the design of small subsystems, but as the specifications and systems become more complex, one-of-a-kind methods are more difficult to apply

and we expect to eventually use formal verification methods such as *Model Checking*. One of the challenges in model checking graph grammars is the enormous state space generated by graph isomorphism. We demonstrated methods to drastically reduce the size of the model for a limited class of grammars[10]. Model size is a major hindrance in model checking large-scale concurrent networked systems and in future work, we plan to broaden the class of grammars for which we may efficiently compute reduced models.

In section 7 we created continuous controllers and guards on the rule set so that these controllers worked in tandem with the graph grammar to achieve the desired geometries and topologies. This essentially created a locally defined hybrid system we refer to as an *embedded graph grammar*. In future work, we plan to present a formal model of the embedded graph grammar.

We believe the wandering scouts example belongs to a class of problems for which we may be able to automatically synthesize the grammar, controllers, and guards of a solution embedded graph grammar. As the size and complexity of networked systems grows, we expect automatic controller synthesis to become necessary. A key requirement to defining and programming solutions to this class of problems appears to be a method of specification that directly relates the continuous and discrete aspects of the problem and includes a formal notion of "locality."

References

1. J. Bishop, S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen. Self-organizing programmable parts. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ Robotics and Automation Society, 2005.
2. J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.
3. J. Alexander Fax and Richard Murray. Graph laplacians and stabilization of vehicle formations. In *15th IFAC Congress*, 2002.
4. 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.
5. 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.
6. Eric Klavins. Universal self-replication using graph grammars. In *The 2004 International Conference on MEMs, NANO and Smart Systems*, Banff, Canada, 2004.

7. Eric Klavins, Robert Ghrist, and David Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 2005. To Appear.
8. L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.
9. N.E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol.3:2968 – 73, 2001.
10. John-Michael McNew and Eric Klavins. Model-checking and control of self-assembly. In *American Control Conference*, 2006. Submitted.
11. 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.
12. Pedro Lima Paulo Tabuada, George J. Pappas. Motion feasibility of multi-agent formations. *IEEE Transactions on Robotics*, Vol. 21 (3):387–392, 2005.