Model-Checking and Control of Self-Assembly

John-Michael McNew Eric Klavins

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

***Abstract*— Graph grammars can be used to model highly distributed systems where local interaction rules control formation or self-assembly tasks. In this paper, we explore model-checking graph grammar systems, introducing the *zero-one-many collapse* as a way of reducing the usually enormous number of states and transitions produced by a graph grammar *system*. From this collapse, we also define a *canonical initial graph*, that captures some of the characteristic behavior of larger graphs with the same zero-one-many collapse. Finally, we show through examples how these results allow us to effectively reason about the behavior of a graph grammar – and also how to improve a graph grammar based on an analysis of its collapsed Kripke Structures.**

I. Introduction

The control of local interactions in systems involving large numbers of vehicles or robots in highly distributed environments can result in predictable global behaviors, such as formation forming, flocking, swarming and coverage. We have demonstrated the use of *graph grammars* [9] to define local interaction rules for formation, assembly and other tasks [7], [8] and have implemented the technique on *self-assembling robots* [1].

An important goal with these systems is to prove that the behaviors admitted by a graph grammar are correct. Also we may wish to *synthesize* new graph grammars from behavioral specifications. Therefore, in this paper, we begin to explore the use of *Model-Checking* [3] systems defined by graph grammars. The main difficultly is that the transition system or *Kripke Structure* on the set of graphs reachable via a graph grammar from an initial graph usually contains an enormous number of states and transitions. This is a well known problem with model-checking concurrent systems, and for some interleaved systems *Partial Order Reduction* has proven an often-successful approach.

In this paper, we explore systems that assemble robots into groups. In these systems there is often a good deal of structure that we can use to effectively *collapse* [2] the Kripke Structure induced by a graph grammar to a size that is amenable to model-checking. In particular, we examine what is called the *zero-one-many* collapse, which ignores the difference between having two or more of a given assembly type. From this collapse, we also define a *canonical initial graph*, that captures some of the characteristic behavior of larger initial graphs with the same zero-one-many collapse. Finally, we show through examples how these results allow us to effectively reason about the behavior of a graph grammar – and also how to improve a graph grammar based on an analysis of its collapsed Kripke Structures.

II. Related Work

Graph theory underlies much of the work in cooperative control [4] [6]. However, many methods require a specific initial topology. Klavins, Ghrist, and Lipsky [9] introduced graph grammars to assemble pre-specified graph topologies. By restricting rewrites to small subgraphs, graph grammars provide a useful method to program the concurrent behavior of large decentralized systems of robots. An application of graph grammars to robotic systems was demonstrated in [1] where free-floating robots used graph grammars to assemble into larger structures in a predictable and robust manner.

The verification of control strategies for groups of robots is largely unexplored territory. In [5], Fainekos, Gazit, and Pappas applied model-checking to path planning for single robot systems. We too use model-checking and our main point of departure is the paper by Clarke [2], where the notion of a *collapsed* Kripke Structure is introduced as a method to reduce the size of the state space. Furthermore, Baldan [10] and Rensink [11] introduced abstractions and collapsings for classes of graph grammars used in modeling software systems.

III. Systems on Graphs

A. 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. The label $l(x)$ of robot x is used to keep track of local information and may also indicate the operational mode of the robot.

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. The pair (r, h) describes an action on G that produces a new graph $G' = (V, E', l')$ defined by

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

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 G' was obtained from G by the application of (r, h).

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. 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 $\mathcal{C}(G_0, \Phi)$. In particular, we suppose that each reachable component type has a single representative in $\mathcal{C}(G_0, \Phi)$. If no rules in Φ can alter a reachable component, the component is said to be *stable*. The set of stable components is denoted $\mathcal{S}(G_0, \Phi)$.

Example 3.1: Define a rule set by

$$\Phi = \begin{cases} a \quad a \quad \Rightarrow \quad b - c, \\ a \quad b \quad \Rightarrow \quad d - e. \end{cases}$$

Suppose that $G_0 = \{\{1, 2, ...1500\}, \varnothing, \lambda x.a\}$. Here $\lambda x.a$ is the function assigning the label a to all vertices. The components of the system are $\mathcal{C}(G_0, \Phi_1) = \{a \;,\; b - c \;,\; d - e - c\}$, where $b - c$ indicates a robot labeled b is connected by an edge to a robot labeled c. ▲

B. 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 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.

Example 3.2: An example proposition P for the system (G_0, Φ_1) defined in Example 3.1 is given by

$P = [\![\text{All components are isomorphic to } d - e - c]\!]$.▲

Definition 3.1: 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 will often wish to know what propositions are preserved by a given equivalence relation. For example, suppose $\sim$ is the relation *labeled graph isomorphism*, denoted $\simeq$. Then any proposition that can be represented by a formula not using constant symbols to represent vertices in V is preserved by $\simeq$.

IV. Model-Checking Graph Grammars

A *Kripke Structure* M over propositions AP is a four-tuple $M = (S, S_0, R, L)$ where S is a set of states, $S_0 \subseteq S$ is a finite subset of S, R is a total transition relation over S, and $L : S \to 2^{AP}$ is a function that labels each state s with the propositions in AP that are true in s. To save space we do not review CTL^* logic, but refer the interested reader to [3].

Definition 4.1: The *concrete Kripke structure* of a graph grammar system (G_0, Φ) is the structure

$$M(G_0, \Phi) = (\mathcal{R}(G_0, \Phi), \{G_0\}, R, L)$$

where

i. $R(G, G')$ if and only if there exists a rule $r \in \Phi$ and a monomorphism h such that $G \xrightarrow{r,h} G'$.
ii. $L(G) = \{P \mid G \in P\}$.

Two problems arise immediately. First, the size of $\mathcal{R}(G_0, \Phi)$ can make the concrete Kripke structure prohibitively large for standard model-checking algorithms. Even on-the-fly algorithms have difficulty when the number of states reachable in one step from a given state is large. To illustrate the problem, for the system in Example 3.1, the number of unique graphs in the reachable set, $|\mathcal{R}(\mathcal{G}_0, \Phi)|$, is greater than $1500!$. To model check systems represented by graph grammars it is clear we must develop efficient methods of reducing the number of states to be explored. Second, the structure depends on the initial graph. For example, a grammar that directs the assembly of components of size 5 will have different reachable sets depending on whether the size of the initial graph is congruent to 5 or not. Nevertheless, under some *collapsings* of the concrete Kripke structure, different initial graphs will result in the same Kripke structures. Additionally the collapsed structure usually has substantially fewer states.

In the following definition $[G]_\sim$ denotes the set of all graphs equivalent to G under a fixed equivalence relation $\sim$. The following definition is standard.

Definition 4.2: Let $M(G_0, \Phi) = (S, S_0, R, L)$ be the concrete Kripke structure for (G_0, Φ). Let $\sim$ be an equivalence relation on S. The *collapsed Kripke structure* for (G_0, Φ) induced by $\sim$ is $M_\sim(G_0, \Phi) = (S', S'_0, R', L')$ where

i. $S' = \{[G] \mid G \in S\}$
ii. $S'_0 = \{[G_0]\}$.
iii. $([G], [G']) \in R'$ if and only if there exist $H \in [G]$ and $H' \in [G']$ such that $(H, H') \in R$.
iv. $L : S' \to 2^{AP_\sim}$ where $L'([G]) = L(G) \cap AP_\sim$.

The following proposition summarizes the properties of a collapsed Kripke structure. Note that "$\preceq$" is the *simulation relation* [3, Ch. 11].

Proposition 4.1: Let $M_\sim$ be the Kripke structure induced from M by the equivalence relation $\sim$. Then

i. $M \preceq M_\sim$.
ii. [3, Theorem 16] If f is an $ACTL^*$ formula and if $M_\sim \models f$, then $M \models f$.

The first property states that every behavior of M is also a behavior of $M_\sim$. The second property allows us to model check $M_\sim$ to deduce properties about M. However, we are limited to the basic propositions in $AP_\sim$ about graphs that are common to entire equivalence classes, and we are limited to checking $ACTL^*$ formulas. Furthermore, by the first property, counterexamples obtained in checking $M_\sim$ are not necessarily counterexamples in M. Nevertheless, when faced with the daunting size of the state space of M, we have little other recourse than to consider collapsings.

V. Two Useful Collapsings

A. Isomorphism

The most obvious equivalence relation on graphs is graph isomorphism. This is in fact quite natural given that the grammars we consider regard all vertices as essentially identical. In the context of self-assembly, it is 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 3.1 where C_1 is a component of type a, $C_2 = b - c$ and $C_3 = e - d - c$, the vector

$$\mathbf{v} = (\ 3,\ 2,\ 497\ )^T$$

denotes that $\mathbf{v}(1) = 3$, $\mathbf{v}(2) = 2$ and $\mathbf{v}(3) = 497$. For each component type C_j,we denote the basis vector as $\hat{C}_j$, where $\hat{C}_j(i)$ is equal to 1 if $i = j$ and equal to zero otherwise. 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 $\mathbf{v}$. Note, if $H \models \mathbf{v}_G$, then $H \simeq G$.

Also, in keeping with the self-assembly paradigm which is typically addressed in the context of statistical mechanics, we call $\mathbf{v}$ an *isomorphism macrostate*. Additionally, we denote the set of reachable isomorphism macrostates by $\mathcal{R}_\simeq(G_0,\Phi) = \{\mathbf{v}_G \mid G \in \mathcal{R}(G_0,\Phi)\}$.

Let G and G' be graphs and let $\mathbf{v}_G, \mathbf{v}_{G'}$ denote the associated isomorphism macrostates. Let (r,h) be an 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

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

indicating that components of type 2 and type 5 were combined into a component of type 3. If $\mathbf{a}(i) = m < 0$, then m components of type C_i were destroyed by applying the action (r,h). If $\mathbf{a}(i) = m > 0$, then m components of type C_i were created. We call the vector $\mathbf{a}$ a *macrostate action*. Without loss of generality, we consider only rules where the left and right graphs, L and R, each have at most two distinct components. This motivates the following definition.

Definition 5.1: Fix a rule r. We define the set of *isomorphism macrostate actions* applicable to macrostate $\mathbf{y}$ via a rule r by

$$\begin{aligned}\mathcal{A}_r(\mathbf{y}) \quad = \quad & \{-\mathbf{v}_{(C_i \amalg C_j)} + \mathbf{v}_{f_{(r,h)}(C_i \amalg C_j)} \mid \\ & \mathbf{y}(i)\text{(resp. (j))} \geq \hat{C}_i(i) + \hat{C}_j(i), and \\ & \text{r is applicable to } C_i \amalg C_j\}.\end{aligned}$$

Here $C_i \amalg C_j$ is the graph created by the disjoint union of components of type i and j.

Proposition 5.1: Let $M_\simeq(G_0,\Phi) = (S_\simeq, S_{0,\simeq}, R_\simeq, L_\simeq)$. A transition $(\mathbf{y},\mathbf{v}) \in R_\simeq$ if and only if

1) $\mathbf{y} \in \mathcal{R}_\simeq(G_0,\Phi)$.
2) There exists $r \in \Phi$ and $\mathbf{a} \in \mathcal{A}_r(\mathbf{y})$ such that

$$\mathbf{v} = \mathbf{y} + \mathbf{a}.$$

We also denote the set of macrostate actions that result in a transition from $\mathbf{y}$ to $\mathbf{v}$ as $\mathcal{A}(\mathbf{y},\mathbf{v}) = \{\mathbf{a} \mid \mathbf{a} \in \mathcal{A}_r(\mathbf{y})$ for some $r \in \Phi$ and $\mathbf{v} = \mathbf{y} + \mathbf{a}\}$. Since in self-assembly problems, we are often interested in creating many copies of small components, Proposition 5.1 implies we can more efficiently compute the isomorphism collapse by considering pairwise combinations of components rather than operating on larger graph representatives of a macrostate.

B. Zero-One-Many Equivalence

The isomorphism collapse may still contain an enormous number of states and paths. Therefore we present a refinement of the collapse that results in a more drastic reduction in the state space.

Definition 5.2: Two graphs G and H are *k-equivalent*, denoted $G \overset{k}{\sim} H$, if and only if for each i,

$$\mathbf{v}_G(i) \leq k \ \text{ or } \ \mathbf{v}_H(i) \leq k \Longleftrightarrow \mathbf{v}_G(i) = \mathbf{v}_H(i).$$

Taking $k = \infty$ results in isomorphism. Taking $k = 1$ results in what we call the *zero-one-many* relation: $\overset{1}{\sim}$. Two graphs are *zero-one-many* equivalent if, for each component type C_i in $\mathcal{C}(G_0,\Phi)$, the graphs both contain zero copies of C_i, they both contain one copy of C_i, or they both contain more than one copy (i.e. *many* copies) of C_i. The Kripke structure $M_{\underset{\sim}{1}}(G_0,\Phi)$ is called the *zero-one-many collapse* of $M(G_0,\Phi)$.

We also write $[G]_{\underset{\sim}{1}}$ in vector notation as in, for example,

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

which denotes that $\widetilde{\mathbf{v}}(1) > 1$, $\widetilde{\mathbf{v}}(2) = 0$, $\widetilde{\mathbf{v}}(3) = 1$ and so on. As with isomorphism macrostates, we write $G \models \widetilde{\mathbf{v}}$ when $\widetilde{\mathbf{v}}$ corresponds to $[G]_{\underset{\sim}{1}}$ and we may write $\widetilde{\mathbf{v}}_G$ instead of just $\widetilde{\mathbf{v}}$. We call $\widetilde{\mathbf{v}}$ a zero-one-many macrostate, or just macrostate. Finally, if $\mathbf{v}$ is an isomorphism macrostate, we denote by $\widetilde{\mathbf{v}}$ the zero-one-many macrostate obtained by replacing elements of $\mathbf{v}$ with m when they are greater than 1. When an isomorphism macrostate is written in terms of a macrostate $\mathbf{v}$ and a macro-action $\mathbf{a}$ as in $(\mathbf{v}+\mathbf{a})$, we denote the corresponding zero-one-many macrostate by $\widetilde{(\mathbf{v}+\mathbf{a})}$.

Proposition 5.2: Any proposition that does not use constant symbols to represent vertices in V and which does not imply the existence of more than two components of any given type is preserved by the equivalence relation $\overset{1}{\sim}$. We denote this set of propositions as $AP_{\overset{1}{\sim}}$.

Example 5.1: The proposition

$$P = [\![\exists x, y, z \in V,\ x \neq y \neq z \mid l(x) = b, l(y) = b, l(z) = b]\!]$$

is not preserved. To see this let G be the graph with three unconnected vertices, each labeled b and H be the graph with two unconnected vertices each labeled b. Then $\widetilde{\mathbf{v}}_G = \widetilde{\mathbf{v}}_H$ but G is in P, while H is not. However, the following proposition is preserved.

$$\begin{aligned}P' = [\![&\exists \text{ a component } C \in G \text{ and vertices } x, y, z \in C \\ &\text{such that } l(x) = b, l(y) = b, l(z) = b]\!].\end{aligned} \quad \blacktriangle$$

We usually consider finite initial graphs. Each initial graph, unfortunately, results in a potentially different zero-one-many collapse of $M(G_0, \Phi)$. However, we can show that the number of different possible forms for $M_{\overset{1}{\sim}}(G_0, \Phi)$ is finite if we require that the number of components of a grammar, independent of the initial graph, is finite. Let $\mathcal{G}_0$ be the set of initial graphs under consideration and define

$$\mathcal{C}(\Phi) = \bigcup_{G_0 \in \mathcal{G}_0} \mathcal{C}(G_0, \Phi).$$

Proposition 5.3: If $\mathcal{C}(\Phi)$ is finite, then there are a finite number of Kripke structures $M_{\overset{1}{\sim}}(G_0, \Phi)$ with $G_0 \in \mathcal{G}_0$.

Proof: There are at most $3^{|\mathcal{C}(\Phi)|}$ possible zero-one-many macrostates $\widetilde{\mathbf{v}}$, because $\widetilde{\mathbf{v}}(i) \in \{0, 1, m\}$ for all i. The states of each Kripke structure $M_{\overset{1}{\sim}}(G_0, \Phi)$ are taken from these macrostates, thus there are a finite number of such structures. ■

Note that this result does not hold for the isomorphism collapse, since, then, $\mathbf{v}(i) \in \mathbb{N}$.

For a fixed rule set Φ we define an equivalence relation $\equiv_1$ on initial graphs in $\mathcal{G}_0$ by

$$G_0 \equiv_1 H_0 \iff M_{\overset{1}{\sim}}(G_0, \Phi) = M_{\overset{1}{\sim}}(H_0, \Phi).$$

By Proposition 5.3, the relation $\equiv_1$ partitions $\mathcal{G}_0$ into a finite number of equivalence classes. The zero-one-many equivalence relation allows us to characterize the behavior of Φ with respect to the possibly infinite set of initial graphs in a given class.

C. Determining the Zero-One-Many Kripke Structure

Obtaining the collapsed structure $M_{\overset{1}{\sim}}(G_0, \Phi)$ by enumerating the states in concrete structure $\widetilde{M}(G_0, \Phi)$ is computationally infeasible. We instead seek a direct method of deriving $M_{\overset{1}{\sim}}(G_0, \Phi)$, especially one amenable to on-the-fly model-checking.

Let $\mathbf{c} = (|C_1|, |C_2|, ... |C_n|)^T$ be the vector of component sizes where $\mathcal{C}(\Phi) = (C_1, C_2, ..., C_n)$.

Proposition 5.4: If $\widetilde{\mathbf{v}}$ is a state of $M_{\overset{1}{\sim}}(G_0, \Phi)$, then the equation

$$\mathbf{c}^T \mathbf{x} = |G_0|$$

subject to the constraint $\widetilde{\mathbf{x}} = \widetilde{\mathbf{v}}$ has an integer solution.

Proof: If $\widetilde{\mathbf{v}}$ is a state of $M_{\overset{1}{\sim}}(G_0, \Phi)$, there exists a graph $G \in \mathcal{R}(G_0, \Phi)$ such that $\widetilde{\mathbf{v}} = \widetilde{\mathbf{v}}_G$. Define $\mathbf{x} = \mathbf{x}_G$. Since $|G| = \mathbf{c}^T \mathbf{x}$ and $|G| = |G_0|$, the equation is satisfied. Also, for all i, if $\widetilde{\mathbf{v}}(i) \in \{0, 1\}$, then $\widetilde{\mathbf{v}}(i) = \mathbf{x}(i)$ and if $\widetilde{\mathbf{v}}(i) = m$, then $\mathbf{x}(i) > 1$. ■

Note that $\mathbf{x}$ is a representative of a possible isomorphism macrostate that is consistent with $\widetilde{\mathbf{v}}$. A similar result applies to transitions in the zero-one-many collapsed structure.

Proposition 5.5: If $M_{\overset{1}{\sim}}(G_0, \Phi) = (S_{\overset{1}{\sim}}, S_{\overset{1}{\sim},0}, R_{\overset{1}{\sim}}, L_{\overset{1}{\sim}})$, a transition $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}) \in R_{\overset{1}{\sim}}$ if and only if there exists a $\mathbf{y}$ such that

1) $\mathbf{c}^T \mathbf{y} = |G_0|$ has an integer solution.
2) $\widetilde{\mathbf{u}} = \widetilde{\mathbf{y}}$.
3) $\exists r \in \Phi$, and $\mathbf{a} \in \mathcal{A}_r(y)$ such that
$$\widetilde{\mathbf{v}} = \widetilde{(\mathbf{y}+\mathbf{a})}.$$
4) $\mathbf{y}$ is reachable.

The reachability of $\mathbf{y}$ requires a path of macrostate actions from $\mathbf{v}_{G_0}$ to $\mathbf{y}$ in $M_{\simeq}(G_0, \Phi)$. Each of these actions may further constrain the values of $\mathbf{y}$. For example, suppose that $\mathbf{a}$ is an action that deletes an edge in C_i, producing two components of type C_j. If $\mathbf{a}$ is the only action capable of generating components of type C_j and no actions destroy component C_j, then $\mathbf{y}(j)$ must be even. Thus the reachability of a state $\mathbf{y}$ is determined by propagating all constraints implied by actions and initial conditions. However, by limiting the rules we consider to *non-destructive* rules we can present a simpler method. A rule r whose right hand side R is a single connected component is called *non-destructive*.

Proposition 5.6: Let G_0 be an initial graph such that for all x, $l_{G_0}(x) = a$ and $E_{G_0} = \varnothing$. Let Φ be any non-destructive rule set. If there exists a $\mathbf{y}$ such that

$$\mathbf{c}^T \mathbf{y} = |G_0|,$$

then $\widetilde{\mathbf{y}}$ is reachable in $M_{\overset{1}{\sim}}(G_0, \Phi)$.

Proof: Suppose that H is a graph containing only components involved in an action (r, h). If r is non-destructive, then $f_{(r,h)}(H) = H'$ implies that H' is a single component, say C_i. This implies that any component C_i can be constructed via a trajectory where every action involves only the vertices in V_{C_i}. Thus, for any set of non-destructive rules, we can explicitly construct a trajectory ending in a graph G with $G \models \mathbf{y}$ by constructing $\mathbf{y}(i)$ copies of C_i independently. A valid trajectory for constructing G is then any interleaving of these *component* trajectories. ■

While Proposition 5.5 reveals relationships among the concrete, isomorphism, and zero-one-many Kripke structures, it does not give an explicit method to compute $M_{\overset{1}{\sim}}$ besides brute force enumeration of the states of $M_{\simeq}$. The zero-one-many Kripke structure for non-destructive grammars may be determined however, by posing Proposition 5.5 as a linear constraint satisfaction problem.

First, denote by $\mathcal{A}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}})$ the set of actions $\mathbf{a} \in \mathcal{A}_r(\widetilde{\mathbf{u}})$ that result in $\widetilde{\mathbf{v}}$ when applied to some $\mathbf{u}$ that models $\widetilde{\mathbf{u}}$. Let $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$ be a triple with $\mathbf{a} \in \mathcal{A}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}})$. For each $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$

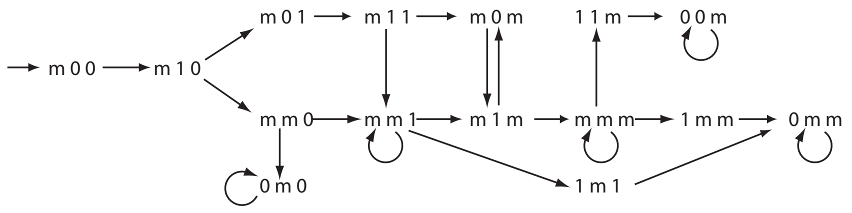

Fig. 1. The zero-one-many collapsed Kripke structure for example 5.3.

we seek an isomorphism macrostate $\mathbf{y}$ consistent with the constraints in Proposition 5.5.

We generate two types of constraints. First, define

$$\mathbf{e}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})(i)) = \begin{cases} \widetilde{\mathbf{v}}(i) - \mathbf{a}(i) & \text{if } \mathbf{a}(i) \neq 0 \text{ and } \widetilde{\mathbf{v}}(i) \leq 1 \\ \widetilde{\mathbf{u}}(i) & \text{if } \widetilde{\mathbf{u}}(i) \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

and require that whenever $\mathbf{e}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})(i) \neq 0$, then $\mathbf{y}(i) = \mathbf{e}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})(i)$. Second, define

$$\mathbf{g}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})(i) = \begin{cases} \max(2 - \mathbf{a}(i), 2) & \text{if } \mathbf{e}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a}(i) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

and require that whenever $\mathbf{g}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})(i) \neq 0$ then $\mathbf{y}(i) \geq \mathbf{g}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})(i)$. For convenience, denote $\mathbf{e}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$ by $\mathbf{e}$, and similarly for $\mathbf{g}$. Next, define the matrix $\mathbf{A}_e$, that determines for which i to apply the constraint $\mathbf{e}(i) = \mathbf{y}(i)$, by

$$\mathbf{A}_e(i,j) = \begin{cases} 1 & \text{if } i = j \wedge e(i) \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Define a similar matrix $\mathbf{A}_g$.

Finally the proposed macrostate $\mathbf{y}$ must be consistent in size with the initial graph: $\mathbf{c}^T\mathbf{y} = |G_0|$. This leads to the following proposition.

Proposition 5.7: Let Φ be a non-destructive rule set. If $\widetilde{\mathbf{u}} \in S_{\underset{\sim}{1}}(G_0, \Phi)$, then $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}) \in R_{\underset{\sim}{1}}(G_0, \Phi)$ if there is some $\mathbf{y}$ and some $\mathbf{a} \in \mathcal{A}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}})$ such that the constraints

$$\begin{aligned} \mathbf{c}^T\mathbf{y} &= |G_0| \\ \mathbf{A}_e\mathbf{y} &= \mathbf{e} \\ \mathbf{A}_g\mathbf{y} &\geq \mathbf{g} \end{aligned}$$

have an integer solution in $\mathbf{y}$.

Proof: The proof is straightforward by Propositions 5.4, 5.5 and 5.6 and the definition of the zero-one-many collapse. ■

Example 5.2: Let $\widetilde{\mathbf{u}} = (m,\ 1,\ m)^T$ and consider the system described in Example 3.1. We want to determine if the system can transition to state $\widetilde{\mathbf{v}} = (m,\ 0,\ m)^T$. For this pair, there is one action $\mathbf{a} = (-1,\ -1,\ 1)^T$ in $\mathcal{A}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}})$. This action uses the second rule to join an "a" (component type C_1) and a "$b\ -\ c$", (type C_2) to build a component of type C_3. The constraint vectors are $\mathbf{e} = (0,\ 1,\ 0)^T$ and $\mathbf{g} = (3,\ 0,\ 2)^T$. It must be that $\mathbf{y}(1)$ is greater than or equal to 3 since there are at least 2 components of type C_1 in $\widetilde{\mathbf{v}}$, but one component of type C_1 was destroyed when the action $\mathbf{a}$ was applied to $\mathbf{y}$. The matrices are

$$\mathbf{A}_e \;=\; \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A}_g \;=\; \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The vector of component sizes is $\mathbf{c} = (1,\ 2,\ 3)^T$. Many solutions to the constraint problem in Proposition 5.7 exist, for example $\mathbf{y} = (4,\ 1,\ 498)^T$. In contrast, consider the constraints for a transition from $\widetilde{\mathbf{u}} = (m,\ 1,\ m)$ to $\widetilde{\mathbf{z}} = (1,\ 0,\ m)^T$. Although this transition is allowed in systems generated from *some* initial graphs, there are no solutions $\mathbf{y}$ that satisfy the constraint imposed by $|G_0| = 1500$. ▲

Proposition 5.7 suggests the zero-one-many structure $M_{\underset{\sim}{1}}(G_0, \Phi)$ can be found via breadth first search. The approach identifies all transitions possible from a given state by examining actions applicable to pairs of components and requiring a solution to the constraint satisfaction problem. We have implemented this in Mathematica.

Example 5.3: Figure 1 shows the zero-one-many Kripke structure for the system (G_0, Φ_1) defined in Example 3.1. Although G_0 has 1500 vertices, the collapsed Kripke structure only has 15 states. ▲

D. Properties of the Zero-One-Many Collapse

By Proposition 4.1 we may examine temporal logic properties of the zero-one-many collapsed system that are $ACTL^*$ formula.

Example 5.4: Our intent was to create a grammar that produces length 3 chains. Consider the $ACTL^*$ formula $f = \mathbf{AG}\ \mathbf{AF}\ (0,\ 0,\ m)^T$, which states that "for all paths it is generally true that eventually state $(0,\ 0,\ m)^T$ is reached." By Proposition 4.1 if $M_{\underset{\sim}{1}} \models f$, then $M \models f$. Model-checking $M_{\underset{\sim}{1}} \models f$ yields a false result. *However, $M_{\underset{\sim}{1}} \not\models f$ does not imply $M \not\models f$, because a collapsing may have cycles not present in the concrete structure.* We will show later the cycle between states $(m\ 0\ m)$ and $(m\ 1\ m)$ in Figure 1 is one such cycle.

One counterexample that invalidates f is described by the path: $(m,\ 0,\ 0)^T \to (m,\ 1,\ 0)^T \to (m,\ m,\ 0)^T \to (0,\ m,\ 0)^T \to (0,\ m,\ 0)^T$.... This suggests that $(0,\ m,\ 0)^T$ may be a final state in the concrete structure. One way to verify this is to use the formula useful for identifying final states $f' = \mathbf{AG}\ (\ (0,\ m,\ 0)^T \Rightarrow \mathbf{X}(0,\ m,\ 0)^T\ )$. It can be shown that $M_{\underset{\sim}{1}} \models f'$, which implies that once reached, the system can never progress away from state $(0,\ m,\ 0)^T$. ▲

VI. Canonical Initial Graphs

Since the zero-one-many collapsed structure may have cyclic behaviors that do not exist in the concrete structure, it is useful to identify a canonical graph whose zero-one-many behavior mimics those of larger initial graphs within the same $\equiv_1$ equivalence class. By considering the concrete

Kripke structure only for the canonical graph, we may make inferences about the smaller structure that apply to the structures of larger initial graphs.

A. *Definition of the Canonical Initial Graph*

In the following we denote the restriction of the concrete Kripke structure M to a smaller set of propositions $AP' \subset AP$ by $M_{|_{AP'}}$.

Definition 6.1: Let G and H be graphs in $[G_0]_{\equiv_1}$. Let $M_1 = M_{|AP^{\underset{\sim}{1}}}(G, \Phi)$ and $M_2 = M_{|AP^{\underset{\sim}{1}}}(H, \Phi)$. Define the relation $\sqsubseteq$ by $M_1 \sqsubseteq M_2$ if and only if for all $\mathbf{u}_1 \in M_1$ there exists a $\mathbf{u}_2 \in M_2$ such that $\widetilde{\mathbf{u}}_1 = \widetilde{\mathbf{u}}_2$ and whenever there exists a $\mathbf{v}_1 \in M_1$ with $(\mathbf{u}_1, \mathbf{v}_1) \in R_1$ via an action $\mathbf{a}$, then there exists a $\mathbf{v}_2 \in M_2$ with $(\mathbf{u}_2, \mathbf{v}_2) \in R_2$ via action $\mathbf{a}$ and $\widetilde{\mathbf{v}}_1 = \widetilde{\mathbf{v}}_2$.

Note that the relation $\sqsubseteq$ is a preorder.

Proposition 6.1: All but a finite subset of $[G_0]_{\simeq_1}$ are maximal elements with respect to $\sqsubseteq$.

Proof: Let M_i be the concrete Kripke structure of graph G_i. Choose any graph G_1 in $[G_0]_{\equiv_1}$. Either M_1 is maximal or there is another graph $G_2 \in [G_0]_{\equiv_1}$ with $|G_2| > |G_1|$ in whose structure M_2 there exists a transition $(\mathbf{u}_2, \mathbf{v}_2) \in R_2$ via an action $\mathbf{a}$ such that for all $(\mathbf{u}_i, \mathbf{v}_j) \in R_1$ where $\widetilde{\mathbf{u}}_i = \widetilde{\mathbf{u}}_2$ and $\widetilde{\mathbf{v}}_j = \widetilde{\mathbf{v}}_2$, the transition $(\mathbf{u}_i, \mathbf{v}_j)$ is not via $\mathbf{a}$. Since the number of actions available at any state is finite and the number of states mapping to any macrostate $\widetilde{\mathbf{u}}$ is finite, there must exist a maximal element. ■

Definition 6.2: For a fixed rule set Φ, the *canonical initial graph*, G^* is the maximal element of the preorder $\sqsubseteq$ with the smallest initial graph.

Intuitively, a state with a self loop in the zero-one-many structure must have an action that applies to a component with ”Many” copies in the graph and produces another component of which there are ”Many” copies in the graph. Several such actions may be possible at a transition. The canonical initial graph is the smallest initial graph such that at every self-loop $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{u}})$, every action in $\mathcal{A}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{u}})$ is possible.

B. *Determining the Canonical Initial Graph*

Given an initial graph G for which we have constructed $M_{\underset{\sim}{1}}(G, \Phi)$, we want to determine G^* without identifying every graph $H \in [G_0]_{\equiv_1}$. For every transition and action $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$ possible for a member of the equivalence class, a representative macrostate $\mathbf{u}$ must exist in $M_{|_{AP^{\underset{\sim}{1}}}}$ where the action applies. Then for each $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$ possible in the zero-one-many structure, we pose the constraint satisfaction problem introduced in Proposition 5.7: Find $\mathbf{x}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$, subject to the constraints $\mathbf{x} \geq \mathbf{g}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$ and $\mathbf{x} = \mathbf{e}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a})$. As we generate the zero-one-many Kripke structure, we concatenate each constraint problem into new vectors $\bar{\mathbf{x}}, \bar{\mathbf{g}}, \bar{\mathbf{e}}$ and the matrices $\bar{\mathbf{A}}_e, \bar{\mathbf{A}}_g$. Each solution must represent a graph with the same number of vertices. This generates the constraint that for all $\mathbf{x}$, $\mathbf{c}^T\mathbf{x}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a}) = \mathbf{c}^T\mathbf{x}(\widetilde{\mathbf{y}}, \widetilde{\mathbf{z}}, \mathbf{a}')$. We enforce this constraint by constructing the matrix

$$\bar{\mathbf{A}}_c = \begin{bmatrix} c^T & -c^T & 0 & \dots & & 0 \\ c^T & 0 & -c^T & 0 & \dots & 0 \\ & & \ddots & & & \\ c^T & 0 & \dots & & & -c^T \end{bmatrix}.$$

The above constraints may imply the presence of more transitions than are allowed by systems in the equivalence class $[G]_{\equiv_1}$. Thus, we define one more set of constraints as follows. If $\widetilde{\mathbf{u}} \in S_{\underset{\sim}{1}}(G^*, \Phi)$ but $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{z}}) \notin R_{\underset{\sim}{1}}(G^*, \Phi)$, then Proposition 5.7 states that for every possible $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{z}}, \mathbf{a})$ there does not exist a solution to $\mathbf{c}^T\mathbf{x}(\widetilde{\mathbf{u}}, \widetilde{\mathbf{z}}, \mathbf{a}) = |G^*|$. As we determine the zero-one-many Kripke structure, we collect the disallowed triples $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{z}}, \mathbf{a})$ into the set $\mathcal{N}$. Since we wish to minimize the size of G^*, we arrive at the following linear integer programming problem.

Theorem 6.1: The number of vertices in the canonical initial graph, $|G^*|$, can be found by finding $\bar{\mathbf{x}}$ such that

$$(\ \mathbf{c}\ \ \mathbf{0}\ )^T\bar{\mathbf{x}} \tag{1}$$

is minimized subject to the constraints

$$\begin{array}{rcl} \bar{\mathbf{A}}_\mathbf{c}\bar{\mathbf{x}} & = & \mathbf{0} \\ \bar{\mathbf{A}}_\mathbf{e}\bar{\mathbf{x}} & = & \bar{\mathbf{e}} \\ \bar{\mathbf{A}}_\mathbf{g}\bar{\mathbf{x}} & \geq & \bar{\mathbf{g}} \\ \bar{\mathbf{x}}(i) & \in & \mathbb{N} \end{array} \tag{2}$$

and for all $(\widetilde{\mathbf{u}}, \widetilde{\mathbf{v}}, \mathbf{a}) \in \mathcal{N}$ there does not exist a vector $\mathbf{y}$ of positive integers such that

$$\begin{array}{rcl} \mathbf{c}^T\mathbf{y} & = & \bar{\mathbf{x}}(1) \\ \mathbf{A}_e\mathbf{y} & = & \mathbf{e} \\ \mathbf{A}_g\mathbf{y} & \geq & \mathbf{g}. \end{array} \tag{3}$$

By convention $\bar{\mathbf{x}}(1)$ is the number of vertices in the desired initial graph. We find a candidate solution $\bar{\mathbf{x}}$ to Equation 1 subject to the constraints in Equation 2 using standard integer linear programming (ILP). Suppose the size of the candidate canonical initial graph is $\bar{\mathbf{x}}(1) = N$. If no solution exists to the constrained equations in Equation 3, then $|G^*| = N$. Otherwise we add the constraint $(\mathbf{c}\ \mathbf{0})^T\bar{\mathbf{x}} > N$ to the above ILP problem. We iterate this process until a solution is found.

Example 6.1: The canonical initial graph for the system (G_0, Φ) in Example 3.1 has 18 vertices compared with 1500 in G_0. Two iterations of the ILP are required to find G^*. The first ILP solution has 15 vertices. However, the corresponding Kripke structure contains more transitions than in $M_{\underset{\sim}{1}}(G_0, \Phi)$. Thus the additional constraint $(\mathbf{c}\ \mathbf{0})^T\bar{\mathbf{x}} > 15$ was added to the ILP for the second iteration. ▲

C. *Shared Propositions and Properties*

Our goal is to understand which behaviors of large systems we can capture by examining the behaviors of the corresponding canonical initial graphs.

Definition 6.3: Let $\mathcal{I} \subseteq [G^*]_{\equiv_1}$. A proposition P is *shared* by graphs reachable from graphs in $\mathcal{I}$ via Φ if

$$P \cap \mathcal{R}(G, \Phi) = \varnothing \Leftrightarrow P \cap \mathcal{R}(H, \Phi) = \varnothing.$$

Proposition 6.2: Any proposition that is shared over $[G^*]_{\equiv_1}$ may use constant symbols to represent vertices in V provided

1) The largest constant symbol is less than or equal to $|G^*|$ and
2) the proposition P does not imply the existence of more than two components of any given type.

We denote the set of shared propositions by $AP^+_{\underset{\sim}{1}}$.

Example 6.2: The proposition

$$P = [\![l(1) = b, l(2) = b, l(3) = b]\!]$$

is not shared. As a counter example consider an initial graph H with three vertices each labeled b and another initial graph G with only two vertices, each labeled b. $H \in P$. But clearly $P \cap \mathcal{R}(G^*, \Phi) = \varnothing$ regardless of Φ. However, the following proposition is shared.

$$P' = [\![\exists C \mid \{1, 2, 3\} \in V_C \wedge l(1) = b, l(2) = b, l(3) = b]\!].\blacktriangle$$

A temporal logic formula f is shared among the canonical system and the systems for larger initial graphs H in $[G^*]_{\equiv_1}$ if

$$M_{|_{AP^+_{\underset{\sim}{1}}}}(G^*, \Phi) \models f \Leftrightarrow M_{|_{AP^+_{\underset{\sim}{1}}}}(H, \Phi) \models f. \qquad (4)$$

Although, for CTL^* and LTL, some properties will not be shared, many important properties related to final behavior or infinite behavior in the system are shared. One such formula is

$$f = \mathbf{AFG}\ P,$$

which can be used to identify the stable set. Also, we can detect infinite cycles with

$$f = \mathbf{EG}(\ P \implies \mathbf{F}\ \neg P\ \wedge \neg P \implies \mathbf{F}P).$$

Finally, under some restrictions, formulas describing sequences of labelings and possible connections for a vertex are preserved. For example the formula

$$f = \ \mathbf{AG}\ (\ l(1) = a\ \mathbf{U}\ l(1) = b\ ),$$

which states that vertex 1 is labeled a until it is labeled b, is shared. This type of property cannot be investigated using either the isomorphism collapse or the zero-one-many collapse, since propositions that use constants to represent vertices are not preserved under either collapse. We have yet to characterize the full set of formulas preserved by the canonical system.

Example 6.3: In Example 5.2 we stated that the cycle between $(m,\ 0,\ m)^T$ and $(m,\ 1,\ m)^T$ in $M_{\underset{\sim}{1}}(G_0, \Phi)$ was a spurious by-product of the collapsing process. We can verify this using the concrete Kripke structure $M_{|_{\underset{\sim}{1}}}$ of the canonical initial graph from Example 6.1. In particular, we check whether

$$M_{|_{\underset{\sim}{1}}}(G^*, \Phi) \models \mathbf{A}\ \neg\mathbf{GF}(\ m\ 1\ m\ )),$$

which turns out to be false. Thus, the cycle identified in the zero-one-many collapse did not correspond to a cycle in the concrete system.

Example 6.4: Unlike in the zero-one-many collapse, we may refer to distinct vertices in a limited fashion. For example, the following formula expresses the different sequences of labelings a vertex may assume.

$$\begin{aligned} f \quad = \quad & \mathbf{AG}\ \ (\mathbf{F}\ l(1) \neq a\ \wedge \\ & l(1) = a\ \mathbf{U} \\ & (l(1) = c\ \vee\ l(1) = e\ \vee\ (l(1) = b\ \mathbf{U}\ l(1) = d)\ )). \end{aligned}$$

The formula states that the label of vertex 1 *must* change from a to either $c, b,$ or e. And if it is labeled b it *may* change to d. We find that $M_{|_{\underset{\sim}{1}}}(G^*, \Phi) \models f$. Note, most classes of initial graphs $[H]_{\equiv_1}$ do not model this property. $\blacktriangle$

VII. Implementation and Examples

We have constructed a program in Mathematica to demonstrate the ideas presented in this paper. The program takes a rule set and an initial graph and computes the zero-one-many Kripke structure and the size of the canonical initial graph. Suppose f is a temporal logic formula and Φ is a rule set designed to guarantee that $M(G_0, \Phi)$ models f. Suppose also that Γ is a set of rules generated by the environment or some uncontrollable subsystem (e.g. See [1]). We think of Γ as a disturbance and define the following control problem: if

$$M(G_0, \Phi \cup \Gamma) \not\models f,$$

construct a "controller" rule set Ψ such that

$$M(G_0, \Phi \cup \Gamma \cup \Psi) \models f.$$

As an example, consider an initial graph G_0 with 1600 vertices labeled a. Define a rule set Φ by

$$\Phi = \left\{ \begin{array}{ccccc} a & a & \Rightarrow & b - c, \\ b & b & \Rightarrow & d - c, \\ d & d & \Rightarrow & c - c. \end{array} \right.$$

The desired stable component is the eight vertex acyclic graph denoted by C_4 in Figure 2. Consider the $ACTL^*$ formula

$$f = \forall x(\mathbf{AG}\ \widetilde{\mathbf{v}}(x) = m \implies \mathbf{AFG}\ \widetilde{\mathbf{v}}(x) \geq m) \implies x = 4,$$

which describes the desired behavior of our system. The formula is equivalent to the statement that the component C_4 is the only member of the stable set, $\mathcal{S}(G_0, \Phi_1)$. We may verify in the collapsed structure that $M(G_0, \Phi) \models f$.

Let $\widetilde{\mathbf{z}}$ be any zero-one-many macrostate. Consider another formula that detects cyclic behaviors

$$g = \forall\widetilde{\mathbf{z}}\ \mathbf{A}\ \neg\mathbf{G}(\widetilde{\mathbf{z}} \implies \mathbf{F}\neg\widetilde{\mathbf{z}}\ \wedge\ \neg\widetilde{\mathbf{z}} \implies \mathbf{F}\widetilde{\mathbf{z}}).$$

We can consider this question in the concrete structure of the canonical initial graph. Using the method described in Theorem 6.1 we determine that G^* has only 40 vertices. We can then show that

$$M_{|_{AP_{\underset{\sim}{1}}}}(G^*, \Phi_1) \models g \implies M(G_0, \Phi_1) \models g.$$

Now suppose we add the following disturbance rule, representing a previously unmodeled interaction:

$$\Phi_U = \left\{\ b \quad b \quad \Rightarrow \quad e - f, \right.$$

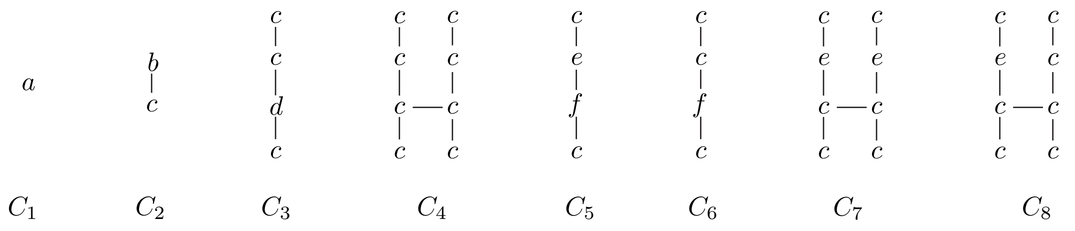

Fig. 2. Components of the grammar $(G_0, \Phi \cup \Gamma \cup \Psi)$. $C_1 - C_4$ are the components of Φ and C_4 is the desired stable component. C_5 is the unwanted component created by the disturbance rule Γ. $C_5 - C_7$ are the components created via the controller Ψ.

Call the new system $(G_0, \Phi_1 \cup \Gamma)$. Using the Mathematica program, we compute the zero-one-many collapsed Kripke structure for the new system $M_{\underset{\sim}{1}}(G_0, \Phi_1 \cup \Gamma)$, which has 125 states. The zero-one-many structure reveals that the system produces an additional component C_5, which is pictured in Figure 2 . The zero-one-many collapse does not model f. The program returns a list of final states $(0,0,0,0,m)^T$, $(0,0,1,0,m)^T$, $(0,0,1,m,m)^T$, $(0,0,0,m,m)^T$, $(0,0,0,m,0)^T$, $(0,0,1,1,m)^T$, $(0,0,0,1,m)^T$ and $(0,0,1,m,1)^T$. It is clear from this list that the controller rule set Ψ needs to convert two C_5 components to a C_4 component and convert a C_3 and a C_5 component to a C_4 component. The control rules are given by

$$\Psi = \begin{cases} f \quad f \;\Rightarrow\; c - c, \\ f \quad d \;\Rightarrow\; c - c, \\ c - e \;\Rightarrow\; c - c. \end{cases}$$

The zero-one-many collapse $M_{\underset{\sim}{1}}(G_0, \Phi \cup \Gamma \cup \Psi)$ has approximately 6500 states and generates three more components shown in Figure 2 . Some action is applicable to every reachable state congruent with 1600 vertices except $(0,0,0,m,0,0,0,0)^T$ thus

$$M(G_0, \Phi \cup \Gamma \cup \Psi) \models f.$$

VIII. Discussion

We have presented the zero-one-many collapse which reduces the potentially enormous state space generated by a graph grammar. For systems with only non-destructive rules, we have shown that calculating members of the reachable set corresponding to zero-one-many macrostates is equivalent to solving a linear integer constraint satisfaction problem. We may also allow destructive rules if we include non-linear integer constraints. We are currently investigating computationally efficient ways of including such constraints. Also, we restricted our arguments to rules involving at most two components. In general, if the rules have at most n components, the collapse to consider is the *zero one two ...* $n-1$ *many* collapse.

The restriction of the verifiable behaviors to those expressed by $ACTL^*$ formulas over properties preserved by the collapse is somewhat limiting. By identifying the canonical initial graph, we were able to examine a much larger class of behaviors using a graph smaller than our initial graph.

Finally, we demonstrated the use of these ideas by examining a simple control problem where unwanted components are generated when two grammars are composed. This type of control must begin with the identification of unwanted components. Since this is the domain of model-checking, we believe that model-checking and control of graph grammar systems are intrinsically linked. We were able to apply the Mathematica program to grammars that produced small numbers of components and we are currently implementing our methods in faster languages. In the future we plan to fully analyze the time and space complexity of the zero-one-many collapse algorithm.

In the future, we would like to integrate these methods with on-the-fly model-checking. Additionally there are a number of open questions: When is $\mathcal{C}(\Phi)$ finite? Are there collapsings more appropriate to classes of systems where $\mathcal{C}(\Phi)$ is not finite? How can model-checking techniques inform the synthesis of rule sets to meet logical specification over graphs?

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] E. M. Clarke and O. Grumberg. Avoiding the state explosion problem in temporal logic model checking algorithms. In *ACM Symposium on Principles of Distributed Computing*, Vancouver, Canada, 1987.

[3] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.

[4] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 5(1), 2004.

[5] Hadas Kress Gazit George Fainekos and George J. Pappas. Temporal logic planning for mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, Barcelona, Spain, 2005. To Appear.

[6] A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6), 2003.

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

[8] Eric Klavins. Universal self-replication using graph grammars. In *The 2004 International Conference on MEMs, NANO and Smart Systems*, Banff, Canada, 2004.

[9] Eric Klavins, Robert Ghrist, and David Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 2005. To Appear.

[10] B. Knig P. Baldan, A. Corradini. Verifying finite-state graph grammars: an unfolding-based approach. In P. Gardner Conference Proceedings and N. Yoshida, editors, *CONCUR'04*, volume 3170 of *Lecture Notes in Computer Science*, pages 83–98. Springer-Verlag, 2004.

[11] Arend Rensink. Canonical graph shapes. In D. A. Schmidt, editor, *Programming Languages and Systems — European Symposium on Programming (ESOP)*, volume 2986 of *Lecture Notes in Computer Science*, pages 401–415. Springer-Verlag, 2004.