Automatic Synthesis of Rulesets for Programmable Stochastic Self-Assembly of Rotationally Symmetric Robotic Modules

Bahar Haghighat and Alcherio Martinoli

Distributed Intelligent Systems and Algorithms Laboratory
School of Architecture, Civil and Environmental Engineering
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
bahar.haghighat@epfl.ch, alcherio.martinoli@epfl.ch

Abstract. Programmable stochastic self-assembly of modular robots provides promising means to formation of structures at different scales. One way to address the design of dedicated control rulesets for self-assembling robotic modules is to leverage formalisms based on graph grammar. While these tools are powerful and allow for formal analysis of the resulting controllers, expressing the embodiment of the robotic modules and therefore the physical structure of assemblies of such modules is not readily possible with such formalisms. This typically results in inefficient representation of ruleset controllers and poses limitations on automatizing ruleset synthesis methods, requiring manual design or tuning of the rules before deployment on the robotic modules. In this work, we consider robotic modules endowed with identical latching connectors arranged in a rotationally symmetric configuration. We extend a grammar formalism based on graphs and propose a new encoding of the modules' internal states. This allows for formulating formal methods capable of automatically deriving the rules based on the morphology of the robotic modules, in particular their number of connectors. The derived rules are directly applicable to robotic modules with no further tuning. In addition, we show that our method allows for a reduced complexity in the rulesets, a particularly welcome feature in the case of limited on-board storage, computation, and communication resources. In order to illustrate the application of our method, we extend two synthesis algorithms from the literature, namely Singleton and Linchpin, to automatically synthesize rules applicable to our resource-constrained robotic modules. In order to increase the prototyping speed and the thoroughness of the validation for the synthesis algorithms, we leverage two complementary simulation frameworks capturing the system at different levels of abstraction. Finally, employing the generated rulesets, we conduct experiments with our robotic platform to demonstrate several assemblies.

1 Introduction

Self-assembly (SA) plays a key role in many of the natural structuring phenomena at all scales (Whitesides & Grzybowski, 2002). SA is formally defined as the reversible and spontaneous phenomenon of an ordered spatial structure emerging from the aggregate behavior of simpler preexisting entities, through inherently local and random interactions in the system (Bušev, 1994). In recent years, SA has been extensively studied both as an enabling technique for micro/nano-fabrication, and as a spatial coordination mechanism for distributed robotic systems of miniaturized modules with limited capabilities, where highly stochastic sensing, actuation, and interactions are inevitable (Haghighat et al., 2016a; Yan et al., 2003). One main motivation for employing SA techniques in distributed engineered systems is to replicate the sort of scalability and robustness observed in the natural instances of SA where a global spatial order is achieved through inherently local interactions. The engineering question is thus that of finding proper assembly directives governing the local interactions, in order to achieve a global spatial order, i.e., a desired target structure. The set of assembly directives or rules guiding the assembly process is denoted as "ruleset" hereafter. Depending on the characteristics of the system, such as the SA medium and the capabilities of the modules, different approaches have been used.

Several engineered systems have demonstrated SA (Rubenstein et al., 2014), (O'Grady et al., 2009), (Tolley & Lipson, 2010), (Salemi et al., 2006), (Ayanian et al., 2008), (Klavins, 2007). The robotic modules in these systems have a symmetric design; i.e., their latching connectors are gender-less and arranged in a rotationally symmetric fashion on the module's body. Programmable SA has been demonstrated in (Rubenstein et al., 2014), and (Klavins, 2007) where active modules self-assemble into predefined desired 2D structures following a set of assembly rules. In (Rubenstein et al., 2014), the miniaturized self-locomoted Kilobot robots coordinate using a deterministic and quasiserial approach in a large swarm of 1000 robots. Module transportation may also be achieved by taking advantage of the stochastic ambient dynamics, realizing stochastic SA. This in turn can allow for simplifying the modules' internal design. In (Klavins, 2007), the programmable parts stochastically self-assemble on an air table based on their internal ruleset controller. In that work, a ruleset is generated using an automatic rule synthesis algorithm which starts from a description of a target structure in the form of an abstract graph and automatically generates proper rulesets for SA of bodiless modules. The resulting rules are then manually tuned to suit the specific morphology of the physical robotic modules. An alternative method for generating self-assembly rules is to use powerful metaheuristic methods. For instance, employing the abstract Tile Assembly Model (aTAM) (Rothemund, 2001), (Lathrop et al., 2009), evolutionary computing has been used to generate rules for self-assembly in a system of passive modules in 2D (Bhalla et al., 2010), and a system of real and simulated passive modules in 3D (Bhalla et al., 2012). The off-line evolved rules are then encoded in the physical characteristics of the passive modules in each case in the form of a magnetic bit pattern, enabling modules with matching patterns to assemble successfully. Manually designed ruleset controllers are utilized in a case study of stochastic SA of simulated underwater robotic modules in 3D in (Ganesan & Chitre, 2016), where the authors manually define ruleset controllers specifically tailored to their robotic modules and the target structures under study.

The problem of ruleset synthesis for programmable SA of graphs is first addressed in (Klavins, 2002) where the self-assembling system consists of bodiless modules represented as the graph vertices, and the connections between the modules are represented by the graph edges. The system graph evolves as the SA process progresses, following the specified assembly rules. Graph rewriting systems may be used to express algorithmically how a new graph is created given an initial one and a set of directives. In (Klavins et al., 2006b), the formalism of graph grammar is formally applied to the SA of graphs and two rule synthesis algorithms are presented. The synthesized rules represent local changes in the system graph based on locally available information. The local nature of the interactions leading to formation of edges between vertices may produce deadlock situations, blocking the system from further progression towards the global objective. The deadlock situation is discussed in (Klavins et al., 2006b), where the number of copies of the target structure being assembled in parallel is higher than the maximum feasible number, considering the total number of initially available modules. In the same work, in order to avoid deadlocks the authors propose a disassociation rule that requires implementing a consensus algorithm among the communicating modules. Alternatively, (Fox & Shamma, 2015) employs a graph grammar formalism and show that the SA of graphs can be achieved while avoiding deadlocks by introducing probabilistic dissociating rules. Two formal rule-synthesis algorithms, Singleton and Linchpin, are introduced in the same work. While in the case of assembling bodiless modules it is only the existence of edges among vertices that specifies the graph structure, in the case of physical modules their embodiment plays a crucial role. For instance, the orientation of the links formed between the modules is restricted by their morphology, in particular by the placement of their latching connectors, and therefore the space of the possible assembled structures is ultimately determined by such local embodiment. In an attempt to apply similar formalisms to the case of robotic modules, (Fox & Shamma, 2010) leverages weighted graphs in a case study to encode the geometric orientations of the edges.

Describing self-assembling robotic systems by means of an appropriate graph grammar formalism offers several advantages. First, a graphical description inherently suits the SA problem where the typically fixed number of constituting modules can be represented by the graph vertices and the bonds forming and severing throughout the SA process can be represented by the graph edges, evolving over time. This provides an efficient model for capturing self-assembling systems characteristics at a high level (Napp et al., 2006). Second, such a description enables the application of several formal rule synthesis algorithms, originally developed for SA of abstract graphs, to the case of SA of robotic modules. While several formal rule synthesis algorithms leveraging graph grammar have been proposed for programmable SA of graphs, their synthesized ruleset controllers are not directly applicable to SA of robotic modules where orientation of the forming links determines the resulting assembled structures.

In this work, we consider the specific but widely common case of rotationally symmetric robotic modules endowed with genderless latching connectors. In order to overcome the problems mentioned above, we extend the concept of abstract graphs by augmenting vertices with link slots, introducing extended vertices, where a link between two extended vertices can only form through specific slots. Additionally, we propose a new way of encoding the robotic modules' internal state by introducing extended labels; a module's internal state evolves according to the ruleset controller and corresponds to the module's local perception of the assembly it is part of. This allows for formulating general methods for synthesizing rules directly applicable to robotic modules endowed with an arbitrary number of genderless connectors arranged in a rotationally symmetric fashion. The self-assembling robotic system is therefore modeled using an extended graph, with each module being associated with one extended vertex in the graph and its internal state being encoded by a control state label and a latching orientation index. For a rotationally symmetric module with N genderless connectors, we provide a proof that our extended formalism achieves a ruleset complexity of $O(N)$ compared with $O(N^2)$ obtained by assigning one vertex and label per connector as presented in (Klavins, 2007). The reduced ruleset complexity is of particular interest for the case of miniaturized modules where very limited memory and communication resources are available. In particular, it allows for a reduction of the memory required for storing the rules as well as that of the overall volume of data shared among modules. Leveraging our formalism, we extend the Singleton and Linchpin rule synthesis algorithms from (Fox & Shamma, 2015), obtaining the SingletonR and LinchpinR algorithms. Compared with the original Singleton and Linchpin, the two new extended algorithms allow for synthesis of rulesets which are directly applicable to robotic modules. We then use SingletonR and LinchpinR to synthesize rules for our resource-constrained floating robotic modules considering case studies on two specific target structures. Finally, in order to increase the prototyping speed and the thoroughness of the validation for the synthesis algorithms, we leverage two complementary simulation frameworks capturing the system at different levels of abstraction.

The remainder of this paper is organized as follows. We begin in Section 2 by describing the SA process in our robotic system. In Section 3, the graph grammar formalism for the SA of graphs is summarized. Section 4 discusses our proposed extended formalism enabling rule synthesis for the SA of rotationally symmetric robotic modules. Section 5 describes the two synthesis algorithms generating rules for SA of our robotic modules. In Section 6, we detail the simulation frameworks. Simulated and experimental results are presented in Section 7, with the conclusions offered in Section 8.

2 Fluidic self-assembly of lily robotic modules

Figure 1 depicts our system which consists of two main components: 1) the Lily robots, originally presented in (Haghighat et al., 2015), which serve as the build-

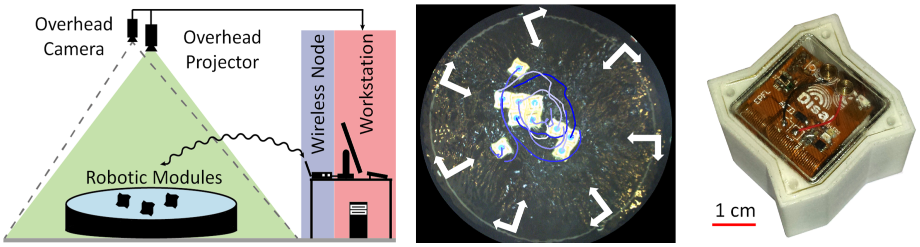

Fig. 1: An overview of the system: The experimental setup consisting of a water-filled tank with peripheral pumps agitating the fluidic environment, an overhead camera for visually tracking the robots and a projector for modulating the lighting in the environment (perceivable by the Lily robots for control purposes, not used in the experimental studies of this work), a wireless node for establishing the radio link between the workstation and the Lily robots (left). Visual tracking of ten Lily robots during an experiment; the blue lines show a short trajectory history for each robot and the white arrows depict the fluid outflow from the peripheral pumps (middle). The Lily robotic module (Haghighat et al., 2015) (right).

ing blocks of the SA process, and 2) the experimental setup built around them (Haghighat & Martinoli, 2016a). The system has several features and capabilities which accommodate a range of experimental studies. While here we briefly review the full list of system features, only a limited subset has been leveraged in the experimental work reported in this paper (features not used are mentioned explicitly in the text). Lilies are endowed with four custom-designed Electro-Permanent Magnets (EPM) to latch and also to communicate locally with their neighbors. They can also communicate over a radio link to a base station whose primary functionality is to allow for wireless programming of the robots without the need of opening the sealed plastic shell for establishing a wired connection. The base station can also issue query commands to receive specific information, for instance to learn about the robots' battery voltage level or internal state. Being power-autonomous, the robots can actively take part in the assembly process at all times. Given a target structure, an appropriate ruleset is derived as detailed in Section 5, and deployed on all robots through wireless bootloading. The robots' EPM latches are by default enabled, resulting in a default latching upon meeting another robot. Once latched, the EPM-to-EPM inductive communication channel is physically established. The robots then exchange their internal states and look for an applicable rule in their ruleset. If no applicable rule is found, they unlatch by switching off their EPM latches; otherwise they remain latched and update their internal states accordingly. Each robot then updates the base station with its new internal state over the radio. Lilies are not self-locomoted, they are instead stirred by the flow field produced within a tank by several peripheral pumps. An overhead camera is used to monitor the evolution of the system by visually tracking a passive marker on the top of each robot using the tracking software SwisTrack (Lochmatter et al., 2008). Three further centralized features of this flexible setup are available but not leveraged in the experimental work of this paper. First, the system is capable of adjusting the agitation in real time. For instance, the agitation mode in the fluidic tank may be modified according to the progress of the SA process to facilitate assembly of the desired target structure. Similarly, the luminosity of the environment, perceivable by the light sensor on the robots, can be modified via an overhead projector to induce a global change in the parameters of the robots' ruleset controller. Specific messages can also be sent to selected robots, inducing a local change in the controllers through the radio link with the base station.

3 Graph grammars for self-assembly of bodiless modules

In this section we summarize the graph grammar formalism for formulating SA of graphs as presented in (Klavins, 2007), and (Fox & Shamma, 2015). A self-assembling system of bodiless modules can be efficiently modeled as a graph evolving over time. Each vertex in the graph represents an anonymous module in the system. While the number of vertices is finite and established at the start of the SA process, the set of edges is dynamic. A finite ruleset determines the course of the evolution of the graph, providing a distributed control scheme for the SA process. Each module maintains an internal state taking values from a discrete and finite set, represented as a labeling on the graph. A rule specifies how an edge between vertices corresponding to modules with certain internal states may be modified. In order to simulate the SA process, at each time step two modules are selected randomly. If the finite ruleset contains a rule applicable to the modules considering their current internal states, the rule gets applied and the graph is modified. In case of probabilistic rules, the rule gets actually applied only with a certain probability associated with the rule. Since the modules are considered to be bodiless, their embodiment and thus the physical orientation of the bonds they form is irrelevant. In the following, we formally define various concepts related to the SA of graphs.

Definition - Internal state of bodiless modules: Each module maintains an internal state which corresponds to its local perception of the progress of the SA process, or equivalently its local neighborhood structure. The internal state of a module evolves according to the rules specified in its ruleset, depending on its interactions with other modules and their respective internal states.

Definition - Labeled graph: A labeled graph is a triple $G = (V, E, \ell)$ where $V = \{1, ..., M\}$ is the set of vertices, $E \subset V \times V$ is the set of edges, and $\ell : V \rightarrow \Sigma$ is a labeling function, with Σ being a set of labels. A pair of vertices $\{x, y\} \in E$ is represented by xy. The vertex set, the edge set, and the labeling function of a graph G are represented by V_G, E_G, and ℓ_G respectively. The notation $n_E(x)$ represents the neighbors of vertex x relative to the edge set E.

Two graphs G_1 and G_2 are considered to be isomorphic when there exists a bijection $h : V_{G_1} \rightarrow V_{G_2}$ such that $ij \in E_{G_1} \Leftrightarrow h(i)h(j) \in E_{G_2}$. The function h

is called a witness. A label-preserving isomorphism has the additional property that $\ell_{G_1}(x) = \ell_{G_2}(h(x)), \forall x \in V_{G_1}$. Since the vertices represent identical bodiless modules, G_1 and G_2 represent the same assembly iff they are isomorphic. A graph G is said to contain a graph H if a subgraph of G is isomorphic to H.

Definition - Rule: A rule is an ordered pair of labeled graphs $r = (L, R)$ such that $V_L = V_R$. The graphs L and R are the left hand side (LHS) and right hand side (RHS) of the rule r. The rule $r = (L, R)$ essentially specifies how the LHS graph L transforms to the RHS graph R through modification of E_L to E_R and ℓ_L to ℓ_R. The size of r is defined as $|V_L| = |V_R|$. A rule specifies a local change in the system graph, meaning that $|V_G| > |V_L|$. An example of a rule of size two, i.e., a binary rule, can be visually represented as $a \quad b \rightharpoonup c - d$, with the characters denoting the labels of the two initially disconnected engaged vertices forming a bond and updating their respective internal states, i.e., the vertex with internal state a updates its internal state to c and the vertex with internal state b updates its internal state to d.

A binary rule corresponds to an interaction between two modules. Simultaneous interactions among many modules, i.e., rules of size larger than two, are generally believed to be difficult to coordinate.

Definition - Rule applicability: A rule $r = (L, R)$ is applicable to a graph G if there exists $I \subset V_G$ such that the subgraph $G \cap I$ has a label-preserving isomorphism $h : I \to V_L$.

Definition - Ruleset: A ruleset ϕ is a set of rules $r_i = (L_i, R_i)$ which specifies the evolution of the SA process towards a desired target assembly out of initially disconnected modules. The application of rules included in ϕ sequentially advances the SA progress by forming or severing bonds between modules with proper internal states.

Definition - Action: The triple (r, I, h) is called an action. The application of an action with $r = (L, R)$ to G gives a new graph $G' = (V_G, E_{G'}, l_{G'})$ defined by

$$E_{G'} = (E_G - xy : xy \in E_G \cap I \times I) \cup (xy : h(x)h(y) \in E_R)$$

$$\ell_{G\prime}(x) = \begin{cases} \ell_G(x), & \text{if } x \in V_G - I \\ \ell_R(h(x)), & \text{otherwise} \end{cases}$$

Definition - Reverse rule: The complement or reverse of a rule $r = (L, R)$, is $\bar{r} = (R, L)$, such that $G \xrightarrow{r,I,h} G' \xrightarrow{\bar{r},I,h} G'' = G$, for appropriate I and h corresponding to the rule r.

Definition - System trajectory: A trajectory of a system (G_0, ϕ), where G_0 is the initial graph of the system and ϕ is a ruleset, is a finite or infinite sequence, depending on the number of applicable rules, of $G_0 \xrightarrow{r_1,I,h} G_1 \xrightarrow{r_2,I,h} G_2 \xrightarrow{r_3,I,h} ...$

Given a ruleset ϕ, one can study the sequences of graphs obtained from successive application of the rules in ϕ. For a probabilistic ruleset, a probability is associated with each rule by the mapping $P : \phi \to (0, 1]$, indicating the tendency for the corresponding event to take place provided that the conditions

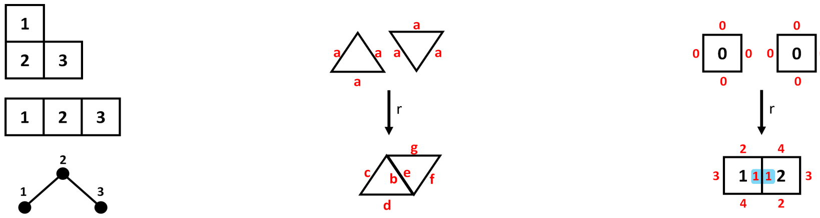

Fig. 2: Different structures represented by the same abstract graph not capturing the orientation of the formed links (left). Association of latching connectors with labels marked in red (Klavins, 2007) (middle). Relative CCW convention for hop numbering, marked in red, starting at the most recently engaged latching connector, marked in blue (right).

under which the rule is applicable are met. All formal rule synthesis methods proposed for programmable SA of graphs automatically generate a ruleset ϕ for assembling a desired target by iteratively browsing and parsing the target graph (Klavins et al., 2006b), (Klavins, 2007), (Fox & Shamma, 2015). Section 5 provides details on the functionality of such methods and how they can be extended to generate rules for SA of rotationally symmetric robotic modules.

4 Graph grammars for self-assembly of rotationally symmetric robotic modules

In this section, we explain how we extend the graph grammar formalism to formulate the problem of ruleset synthesis for programmable SA of rotationally symmetric robotic modules (Haghighat et al., 2016b). As explained in Section 3, the SA process in a system of bodiless modules can be directly modeled by an abstract graph evolving over time and the standard graph grammar formalism can be applied. For the case of robotic modules, their embodiment needs to be incorporated in the model as the modules' morphology, in particular the orientation of the links they may form, strictly determines the shape of the resulting structure. This information cannot be directly encoded in the structure of abstract graphs. Figure 2 (left) gives a simple illustration of this issue considering square-shaped modules. While in both the L shaped structure, on the top, and the chain shaped structure, in the middle, the assembly comprises three modules, with two modules having one common neighboring module, it can be seen that depending on the latching connectors which get engaged, two distinct assembly structures may exist. If the orientation of the formed links are ignored, both assemblies can be described by the same abstract graph, depicted on the bottom.

In order to employ the graph grammar formalism for the problem of SA of robotic modules two main issues should be addressed: first, how the morphology of the robotic module, in particular its latching connectors, can be incorporated into the system graph structure, and second, how the internal states of the robotic modules, which includes information on the orientation of the links formed, can be encoded and represented in the graph structure.

One approach to address the aforementioned issues is considered in (Klavins, 2007). Instead of representing a single module, each vertex in the system graph can be associated with a latching connector on a robotic module. The vertices corresponding to latching connectors of a certain robotic module are then connected using permanent links which indicate the physical coupling, as depicted in Figure 2 (middle). For the case of bodiless modules the state of the module can be encoded by a single label associated with its corresponding vertex in the system graph. For the case of robotic modules, the method in (Klavins, 2007) represents the internal state of a module as the set of labels associated with the vertices corresponding to its latching connectors. Several drawbacks may be listed for this method of representing a robotic module and its internal state within a graph grammar formalism. First, as a result of dedicating several vertices to represent a single module, i.e., one vertex per latching connector, the system graph, i.e., the graph representing the system, will be crowded with vertices and edges which encode redundant information, giving rise to an increased complexity in analyzing and simulating the model. Second, automatic synthesis of rules for robotic modules is not straightforward using this method, mainly due to the complex structure of the graph. Indeed, (Klavins, 2007) first runs a synthesis algorithm on an abstract description of the desired target, and the resulting rules are then manually tuned to account for the correct orientation of the forming links. Third, for a robotic module with N connectors each acquiring a dedicated state label, it can be shown that the ruleset complexity grows in $O(N^2)$.

In what follows, we propose an alternative approach for applying a graph grammar formalism to the SA problem of rotationally symmetric robotic modules. Our goal is not only to be able to employ such formalisms but also to formulate algorithms for the automatic synthesis of rules. To this end, we extend the notion of labeled graphs by introducing the definition of extended vertices and labels. While we are particularly interested in scenarios involving our Lily robots in 2D, the assumptions we make are general enough to be directly applied to similar platforms. The method is also easily applicable to 3D SA with similar assumptions. In essence, we augment the vertices with link slots and introduce the extended vertices, where a link between two extended vertices is formed through specific link slots. The link slots are then indexed according to an enumeration convention on the latching connectors of the robotic modules. Assuming that the latching connectors on the robotic modules are genderless and arranged in a rotationally symmetric configuration, the relative hop distance between the engaged link slots determines the relative orientation of the links formed between the modules and thus determines the shape of the structure. Following this extension, we introduce the extended labels, encoding the internal state of a robotic module as a pair of the control state and the latest engaged connector index. Formal definitions of these concepts are provided below.

Definition - Extended vertex: An extended vertex has ordered link slots which correspond to the latching connectors of a robotic module. An extended vertex v representing a rotationally symmetric robotic module endowed with N latching connectors is a N-tuple $v = (s_1, s_2, ..., s_N)$ where $s_i \in \{0, 1\}$ is a binary value representing the latching state of the corresponding latching connector on the i^{th} latching slot. The numbering of the slots is assumed to match the one on the robotic module, following a counter-clockwise (CCW) rotation convention on the module. Since the modules are assumed to be rotationally symmetric, the connectors are anonymous for an isolated module.

Definition - Internal state of robotic modules: Similar to the case of bodiless modules, each robotic module maintains an internal state which corresponds to its local perception of the progress of the SA process, or equivalently its local neighborhood structure. The internal state of a module evolves according to the rules specified in the ruleset, depending on its interactions with other modules and their respective internal states. The difference is in the notion of the neighborhood structure.

In the case of bodiless modules, the local neighborhood structure of a module and its corresponding internal state does not contain information about the orientation of the links. However, for the case of robotic modules with specific embodiment, the orientation of the links strongly determines the structure of the assembly formed around a module, and must be encoded in the modules internal state.

For the case of rotationally symmetric robotic modules, we consider the internal state to consist of two components: a non-spatial component, called the *control state* hereafter, which encodes the same information as the internal state in the case of bodiless modules, and a spatial component, called the *latching state* hereafter, which encodes the index of the latest engaged latching connector.

Definition - Extended label: An extended label is a pair $l = (l_a, l_n)$ encoding the internal state of a rotationally symmetric robotic module. l_a represents the control state of the robotic module and l_n represents the latching state of the robotic module, i.e., the index of its most recently engaged connector. Notice that l_n may be extended to be an ordered list of recently engaged connectors.

Definition - Extended labeled graph: An extended labeled graph is a quadruple $G = (V, E, S, \ell)$ where $V = \{1, ..., M\}$ is the set of extended vertices, $E \subset V \times V$ is the set of edges, $K = \{1, ..., N\}$ is the set of link slots available on each of the extended vertices, $S : E \rightarrow K \times K$ defines which slots are involved in a link between two vertices, and $\ell : V \rightarrow \Sigma$ is a labeling function, with Σ being a set of extended labels.

Following the extension of the graphs, the rules are also extended to be described using elements which are a combination of a control state variable and a relative latching state variable as explained below. The idea is that a robotic module can only take part in an interaction governed by a certain rule if it has the appropriate control state and is participating in the interaction with the appropriate orientation.

We assume that the robotic modules exchange information of their respective internal states once their latching connectors are engaged. More specifically, once one of the connectors is engaged, the robot may communicate its internal state in the form of a relative extended label of $l = (l_a, l_h)$ with l_a being the robot's control state and l_h being a relative hop number which represents the relative orientation of the currently engaged connector with respect to its predecessor, assuming a CCW hop convention (see Figure 2, right). For a vertex with an extended label of (l_a, l_n) on a robot with N connectors $l_h = [(l_n - l_c) \bmod N] + 1$, where l_c is the index of the currently engaged connector and l_n is the index of the previously engaged connector.

Definition - Extended rule: An extended rule is an ordered pair of extended graphs $r = (L, R)$. An extended binary rule can be depicted as $l_1 \quad l_2 \rightharpoonup l_3 - l_4$, with the l_1, l_2, l_3, l_4 being pairs of the form $l_i = (l_{ia}, l_{ih})$ denoting the relative extended label of the engaged vertices.

Proposition: For a rotationally symmetric robotic module with N number of connectors, employing extended relative labels allows for a ruleset complexity of $O(N)$ compared to the one-label-per-connector approach which results in a ruleset complexity of $O(N^2)$. More formally, if χ is the set of alphabets utilized in the ruleset for encoding the modules' internal states then the extended relative labeling approach results in $|\chi| = O(N)$ while the one-label-per-connector approach results in $|\chi| = O(N^2)$.

Proof: Consider a rotationally symmetric robotic module having N number of connectors and an internal state S. Having N connectors, the module can interact with a similar one through N different orientations, i.e., one orientation per each of its connector. Assume that each one of these N configurations can potentially result in a distinct assembly. For each distinct assembly, the module's internal state S needs to be updated to a distinct value of S'.

Consider the one-label-per-connector approach; S' is encoded by assigning a new label to each connector. To encode a new distinct internal state S', this approach requires N new labels to be added to the alphabet included in χ. To encode all possible interaction outcomes, i.e., all possible updated S' values, N new labels should be added to the ruleset for each one of the N possible configurations, thus a total of N^2 new labels.

Consider the relative extended labeling approach; S' is encoded by assigning a new l_a label per interaction configuration and updating the l_n label to the currently engaged connector index. To encode all possible interaction outcomes, one new label should be added to the ruleset for each of the N possible interaction configurations, thus a total of N new labels. □

5 Synthesizing rules for rotationally symmetric robotic modules

In Section 4, we presented the extension of the graph grammar formalism for the case of SA of symmetric robotic modules and in Section 1 argued that the extended formalism may be utilized to 1) formulate automatic rule synthesis algorithms directly applicable to programmable SA of rotationally symmetric robotic modules, and 2) model and simulate the evolution of the SA process in a system of rotationally symmetric robotic modules. In this section, we showcase the application of the extended formalism to the former of the two points mentioned above. Our focus here is on demonstrating the capability of the extended formalism, and as such, we do not propose inherently novel rule synthesis algorithms. Rather, we pick two rule synthesis algorithms for SA of bodiless modules previously proposed in the literature (Fox & Shamma, 2015), Singleton and Linchpin, and utilize our extended formalism to formulate their counterparts, SingletonR and LinchpinR, for rotationally symmetric robotic modules. Therefore, the specific contribution of this section is demonstrating how our proposed extended formalism explained in Section 4 can be employed to extend automatic rule synthesis algorithms that have been previously proposed for SA of abstract graphs into algorithms capable of automatically synthesizing rules for SA of symmetric robotic modules with arbitrary number of connectors, for a given target represented as an extended graph.

Consider the discussion in the Introduction of this paper, particularly the work in (Ganesan & Chitre, 2016) where the ruleset applied on the simulated robotic modules are essentially hand-designed, or the work in (Klavins, 2007) where an automatic rule synthesis algorithm for bodiless modules is employed to generate a ruleset which is then tuned to fit the specific morphology of the programmable robotic modules. The contribution of this section eliminates the necessity for manually designing or tuning rulesets for robotic modules by providing a formal and efficient (see the proof in Section 4) tool for employing existing rule synthesis algorithms in the literature, possibly originally designed for SA of bodiless modules and transform them so that they can be directly applied to any given symmetric robotic module. Additionally, we provide a comparison between the rules synthesized for a robotic target structure and its abstract graph counterpart by the respective algorithms and show how the two rulesets relate. We then use the SingletonR and LinchpinR to synthesize rulesets for the SA of our Lily robotic modules for two specific target structures, a chain structure and a cross structure, each consisting of 6 robotic modules.

5.1 Singleton and Linchpin for self-assembly of bodiless modules

Here, we briefly introduce the Singleton and Linchpin algorithms originally presented in (Fox & Shamma, 2015). For a given acyclic target graph, these algorithms synthesize rulesets for SA of a system of bodiless modules represented as an abstract graph. The rulesets generated by both algorithms include reverse rules executed probabilisitically in order to avoid deadlock situations. While the rules are the output of the synthesis algorithms, proper execution probabilities need to be assigned separately. This is due to the fact that the synthesis algorithms are agnostic to the dynamics of the underlying self-assembling system and solely consider the necessary steps for the formation of the target.

Thanks to the use of probabilistically executed reverse rules, the rulesets synthesized by the two algorithms provide probabilistic guarantees on achieving the maximum feasible assembly yield. However, they natively differ in their temporal assembly profile. While Singleton induces a serial assembly strategy, Linchpin gives rise to a more parallel scheme. More specifically, for a given target graph G, Singleton generates a serial ruleset where each rule progresses the SA of the target graph by appending an isolated vertex to the structure. In contrast, Linchpin synthesizes a parallel ruleset, where the target graph is assembled from each leaf towards a final vertex, with the process culminating in two concurrently built subgraphs joining together (Fox & Shamma, 2015). Note that both algorithms were specifically designed for handling acyclic target graphs, i.e., trees, and thus do not synthesize valid rulesets for cyclic target graphs.

As an example consider a chain shape target graph constituting six modules $G = (V = \{1, 2, 3, 4, 5, 6\}, E = \{12, 23, 34, 45, 56\})$, assuming vertex 2 as the root vertex fed to the algorithms in (Fox & Shamma, 2015), the resulting rulesets are as below, where the forward rules and their corresponding reverse rules are denoted as r_i and $\bar{r_i}$, respectively:

$$\phi_{Singleton} = \begin{cases} 0 \quad 0 \leftrightharpoons 1-2 \qquad (r1, \bar{r1}) \\ 1 \quad 0 \leftrightharpoons 3-4 \qquad (r2, \bar{r2}) \\ 4 \quad 0 \leftrightharpoons 5-6 \qquad (r3, \bar{r3}) \\ 6 \quad 0 \leftrightharpoons 7-8 \qquad (r4, \bar{r4}) \\ 8 \quad 0 \leftrightharpoons 9-10 \qquad (r5, \bar{r5}) \end{cases} \qquad \phi_{Linchpin} = \begin{cases} 0 \quad 0 \leftrightharpoons 1-2 \qquad (r1, \bar{r1}) \\ 0 \quad 0 \leftrightharpoons 7-8 \qquad (r2, \bar{r2}) \\ 2 \quad 0 \leftrightharpoons 3-4 \qquad (r3, \bar{r3}) \\ 4 \quad 0 \leftrightharpoons 5-6 \qquad (r4, \bar{r4}) \\ 8 \quad 6 \leftrightharpoons 9-10 \qquad (r5, \bar{r5}) \end{cases}$$

Considering a system of six randomly interacting bodiless modules, all initially isolated and labeled 0, the assembly progress guided by the forward rules of the two synthesized rulesets of $\phi_{Singleton}$ and $\phi_{Linchpin}$ are depicted in Figure 3. Note the difference in the natural course of the process induced by the two rulesets. This is regarded as the assembly strategy of the ruleset. While $\phi_{Singleton}$ assembles the target graph in five sequentially executed steps, the rules r_1 and r_2 in the $\phi_{Linchpin}$ ruleset can be executed concurrently and may thus constitute one step. It is through this concurrency that the assembly strategy of $\phi_{Linchpin}$ can reduce the total required assembly time and achieve a higher assembly rate than that of $\phi_{Singleton}$. Note that the ultimate assembly rate in a system depends strongly on the system dynamics, in particular the mixing in the system which allows for the probabilistic interactions between the bodiless modules. However, assuming that all the interactions in the system take place equiprobably, the assembly process guided by $\phi_{Linchpin}$ will on average result in the target graph faster than that guided by $\phi_{Singleton}$. Below we formally define a measure for ruleset concurrency.

Definition - Concurrent steps: The number of concurrent steps required by a ruleset ϕ for building a certain target structure is the minimum number of steps that it takes to assemble the target structure out of initially isolated modules, considering that several concurrently executable rules can be executed simultaneously, and assuming that execution of one rule takes one step. Note that the measure of concurrent steps as defined above is general and applies to rulesets for the SA of both bodiless (see examples given in Section 5.1) and rotationally symmetric robotic modules (see examples given in Section 5.2).

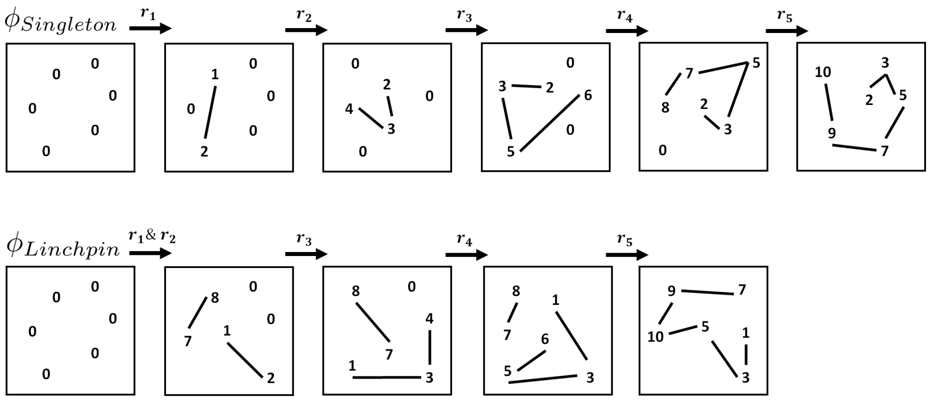

Fig. 3: Progress of the SA process for a chain shape target graph as guided by rulesets $\phi_{Singleton}$ and $\phi_{Linchpin}$.

5.2 SingletonR and LinchpinR for self-assembly of rotatinally symmetric robotic modules

Algorithm 1 depicts the pseudo codes of both Singleton and SingletonR algorithms, and Algorithm 2 depicts the pseudo codes of both Linchpin and LinchpinR algorithms. SingletonR and LinchpinR essentially have the same structure as their abstract graph counterparts. Their main difference is that at each step of the rule synthesis they determine two labels, i.e., l_a and l_h, instead of a single one. As a result of following the same rule synthesis strategy, the class of targets which are achievable by SingletonR and LinchpinR are the same as the ones of the original algorithms, i.e., solely acyclic target graphs can be handled. l, k, and $N_E(k)$ denote the largest label, the root vertex, and the neighbors of node k with respect to edge set E, respectively. For a given target graph $\hat{G}$, running $\textsc{Singleton}((V_{\hat{G}}, E_{\hat{G}}, k, 0))$ for any $k \in V_{\hat{G}}$ generates a ruleset. The ruleset allows the SA process to grow the target graph outwards from the starting vertex k. Similarly, SingletonR generates a ruleset for robotic modules based on a given target structure, represented by an extended graph $G = (V_G, E_G, S_G)$, where $S(v_i, v_j)$ returns the ordered pair of (s_i, s_j), the involved link slots on the two linked vertices. $L(v)$ returns the current extended label of a vertex, (l_a, l_n). The GVL() (short for Get Vertex Label) procedure returns the ordered pair of (l_a, l_h) by updating the value of l_h such that it indicates the relative position of the currently engaged slot, s, with respect to the previously engaged one. The SVL() (short for Set Vertex Label) procedure updates the extended label (l_a, l_n) by updating the value of l_n considering the value of the applied label. Compared to the Singleton algorithm where only the state labels are synthesized, SingletonR and LinchpinR produce the relative hop number l_h indicating the proper linking orientation as well. The combination of these two values provides a general description of the full internal state of a robotic module.

```
$Target\ graph : G = (V, E)$
$Root\ vertex : v_k$
$Initial\ label : l = 0$
procedure SINGLETON($G, v_k, l$)
    $\phi \leftarrow \emptyset$
    if $|n_E(k)| = 0$ then
        return $(l, \phi)$
    else
        $\{v_j : j = 1, 2, ..., |n_E(v_k)|\} \leftarrow n_E(v_k)$
        $\bar{l} \leftarrow l$
        for $j = 1$ to $|n_E(k)|$ do
            $\phi \leftarrow \phi \cup \{\bar{l}\ 0 \rightleftharpoons (l+1) - (l+2)\}$
            $\bar{l} \leftarrow l + 1$
            $l \leftarrow l + 2$
            Let $(V^j, E^j)$ be the component
            of $(V, E - \{v_k v_j\})$ containing $v_j$
            $G_j \leftarrow (V^j, E^j)$
            $(l_j, \phi_j) \leftarrow$ SINGLETON($G_j, v_j, l$)
            $\phi \leftarrow \phi \cup \phi_j$
            $l \leftarrow l_j$
        end for
    end if
    return $(l, \phi)$
end procedure
```

```
$Target\ graph : G = (V, E, S, L)$
$Root\ vertex : v_k$
procedure SINGLETONR($G, v_k$)
    $\phi \leftarrow \emptyset$
    if $|n_E(k)| = 0$ then
        return $(l, \phi)$
    else
        $\{v_j : j = 1, 2, ..., |n_E(v_k)|\} \leftarrow n_E(k)$
        for $j = 1$ to $|n_E(v_k)|$ do
            $(s_k, s_j) \leftarrow S(v_k, v_j)$
            $l_k \leftarrow$ GVL($L, s_k, v_k$)
            $l_j \leftarrow$ GVL($L, s_j, v_j$)
            $\bar{l} \leftarrow$ INCREMENTSTATE($l, 1$)
            $l \leftarrow$ INCREMENTSTATE($l, 2$)
            SVL($L, v_k, s_k, \bar{l}$)
            SVL($L, v_j, s_j, l$)
            $\phi \leftarrow \phi \cup \{l_k\ l_j \rightleftharpoons \bar{l} - l\}$
            Let $(V^j, E^j)$ be the component
            of $(V, E - \{v_k v_j\})$ containing $v_j$
            $G_j \leftarrow (V^j, E^j, S, L)$
            $(l, \phi_j) \leftarrow$ SINGLETONR($G_j, v_j$)
            $\phi \leftarrow \phi \cup \phi_j$
        end for
    end if
    return $(l, \phi)$
end procedure

procedure GVL($L, s, v$)
    $(l_a, l_n) \leftarrow L(v)$
    $l_h \leftarrow (l_n - s + 1) \pmod N$
    return $(l_a, l_h)$
end procedure

procedure SVL($L, v, s, l$)
    $(l_a, l_h) \leftarrow l$
    $l_n \leftarrow s$
    $L(v) \leftarrow (l_a, l_n)$
end procedure

procedure INCREMENTSTATE($l, i$)
    return $(l_a + i, l_n)$
end procedure
```

Alg. 1: Original Singleton algorithm for the SA of bodiless modules as presented in (Fox & Shamma, 2015), and the SingletonR algorithm for the SA of rotationally symmetric robotic modules obtained by applying our extended graph grammar formalism.

```
$Target\ graph : G = (V, E)$
$Root\ vertex : v_k$
$Initial\ label : l = 0$
procedure Linchpin$(G, v_k, l)$
    $\phi \leftarrow \emptyset$
    for $j = 1, 2, ..., |n_E(v_k)|$ do
        if $|n_E(v_j)| \geq 2$ then
            Let $(V^j, E^j)$ be the component
            of $(V, E - \{v_k v_j\})$ containing $v_j$
            $(l_j, \phi_j) \leftarrow$ Linchpin$(G_j, v_k, l)$
            $l \leftarrow l_j$
        else
            $l_j \leftarrow 0$
            $\phi_j \leftarrow \emptyset$
        end if
    end for
    $\phi \leftarrow \phi_1 \cup \{l_1\ 0 \rightleftharpoons (l+1) - (l+2)\}$
    $l \leftarrow l + 2$
    for $j = 2, ..., |n_E(v_k)|$ do
        $\phi \leftarrow \phi \cup \phi_j \cup \{l_j\ l \rightleftharpoons (l+1) - (l+2)\}$
        $l \leftarrow l + 2$
    end for
    return $(l, \phi)$
end procedure
```

```
$Target\ graph : G = (V, E, S, L)$
$Root\ vertex : v_k$
procedure LinchpinR$(G, v_k)$
    $\phi \leftarrow \emptyset$
    for $j = 1, 2, ..., |n_E(v_k)|$ do
        if $|n_E(v_j)| \geq 2$ then
            Let $(V^j, E^j)$ be the component
            of $(V, E - \{v_k v_j\})$ containing $v_j$
            $(l_j, \phi_j) \leftarrow$ Linchpin$(G_j, v_k)$
            $l \leftarrow l_j$
        else
            $l_j \leftarrow 0$
            $\phi_j \leftarrow \emptyset$
        end if
    end for
    $(s_k, s_1) \leftarrow S(v_k, v_1)$
    $l_k \leftarrow \mathrm{GVL}(L, s_k, v_k)$
    $l_1 \leftarrow \mathrm{GVL}(L, s_1, v_1)$
    $\bar{l} \leftarrow$ IncrementState$(l, 1)$
    $l \leftarrow$ IncrementState$(l, 2)$
    $\mathrm{SVL}(L, v_k, s_k, \bar{l})$
    $\mathrm{SVL}(L, v_1, s_1, l)$
    $\phi \leftarrow \phi_1 \cup \{l_1\ 0 \rightleftharpoons (l+1) - (l+2)\}$
    $l \leftarrow l + 2$
    for $j = 2, ..., |n_E(v_k)|$ do
        $(s_k, s_j) \leftarrow S(v_k, v_j)$
        $l_k \leftarrow \mathrm{GVL}(L, s_k, v_k)$
        $l_j \leftarrow \mathrm{GVL}(L, s_j, v_j)$
        $\bar{l} \leftarrow$ IncrementState$(l, 1)$
        $l \leftarrow$ IncrementState$(l, 2)$
        $\mathrm{SVL}(L, v_k, s_k, \bar{l})$
        $\mathrm{SVL}(L, v_j, s_j, l)$
        $\phi \leftarrow \phi \cup \phi_j \cup \{l_j\ l \rightleftharpoons (l+1) - (l+2)\}$
        $l \leftarrow l + 2$
    end for
    return $(l, \phi)$
end procedure
```

Alg. 2: Original Linchpin algorithm for the SA of bodiless modules as presented in (Fox & Shamma, 2015), and the LinchpinR algorithm for the SA of rotationally symmetric robotic modules obtained by applying our extended graph grammar formalism.

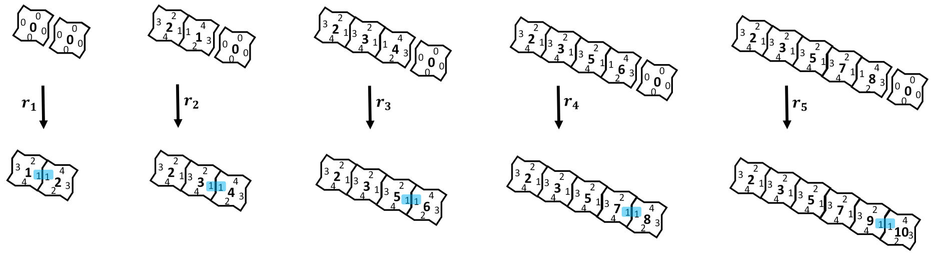

Fig. 4: Progress of the SA process for the chain shape target structure employing ϕ_{-}^{S}. The latest engaged latching connectors on the modules are highlighted with a blue mark, while the relative hop numbering starting at the most recently engaged latching connector are shown on the sides of each module.

5.3 Rulesets for self-assembly of Lily robotic modules

The rulesets returned by SingletonR for a chain and a cross structure, ϕ_{-}^{S} and ϕ_{+}^{S} respectively, as well as the rulesets returned by LinchpinR for a chain and a cross structure, ϕ_{-}^{L} and ϕ_{+}^{L} respectively, using six rotationally symmetric robotic modules, are reported below. The (l_a, l_h) notation is used for the relative extended labels expressed in the rules as explained in Section 4, and the reverse rules are separated.

$$\phi_{-}^{S} = \begin{cases} (0,0) \quad (0,0) & \xrightarrow{r1} \quad (1,1)-(2,1) \\ (1,3) \quad (0,0) & \xrightarrow{r2} \quad (3,1)-(4,1) \\ (4,3) \quad (0,0) & \xrightarrow{r3} \quad (5,1)-(6,1) \\ (6,3) \quad (0,0) & \xrightarrow{r4} \quad (7,1)-(8,1) \\ (8,3) \quad (0,0) & \xrightarrow{r5} \quad (9,1)-(10,1) \\ (1,1)-(2,1) & \xrightarrow{\bar{r1}} \quad (0,0) \quad (0,0) \\ (3,1)-(4,1) & \xrightarrow{\bar{r2}} \quad (1,3) \quad (0,0) \\ (5,1)-(6,1) & \xrightarrow{\bar{r3}} \quad (4,3) \quad (0,0) \\ (7,1)-(8,1) & \xrightarrow{\bar{r4}} \quad (6,3) \quad (0,0) \\ (9,1)-(10,1) & \xrightarrow{\bar{r5}} \quad (8,3) \quad (0,0) \end{cases} \qquad \phi_{+}^{S} = \begin{cases} (0,0) \quad (0,0) & \xrightarrow{r1} \quad (1,1)-(2,1) \\ (1,2) \quad (0,0) & \xrightarrow{r2} \quad (3,1)-(4,1) \\ (3,3) \quad (0,0) & \xrightarrow{r3} \quad (5,1)-(6,1) \\ (5,4) \quad (0,0) & \xrightarrow{r4} \quad (7,1)-(8,1) \\ (8,3) \quad (0,0) & \xrightarrow{r5} \quad (9,1)-(10,1) \\ (1,1)-(2,1) & \xrightarrow{\bar{r1}} \quad (0,0) \quad (0,0) \\ (3,1)-(4,1) & \xrightarrow{\bar{r2}} \quad (1,4) \quad (0,0) \\ (5,1)-(6,1) & \xrightarrow{\bar{r3}} \quad (3,3) \quad (0,0) \\ (7,1)-(8,1) & \xrightarrow{\bar{r4}} \quad (5,1) \quad (0,0) \\ (9,1)-(10,1) & \xrightarrow{\bar{r5}} \quad (8,3) \quad (0,0) \end{cases}$$

$$\phi_{-}^{L} = \begin{cases} (0,0) \quad (0,0) & \xrightarrow{r1} \quad (1,1)-(2,1) \\ (0,0) \quad (0,0) & \xrightarrow{r2} \quad (3,1)-(4,1) \\ (2,3) \quad (0,0) & \xrightarrow{r3} \quad (5,1)-(6,1) \\ (4,3) \quad (0,0) & \xrightarrow{r4} \quad (7,1)-(8,1) \\ (8,3) \quad (6,3) & \xrightarrow{r5} \quad (9,1)-(10,1) \\ (1,1)-(2,1) & \xrightarrow{\bar{r1}} \quad (0,0) \quad (0,0) \\ (3,1)-(4,1) & \xrightarrow{\bar{r2}} \quad (0,0) \quad (0,0) \\ (5,1)-(6,1) & \xrightarrow{\bar{r3}} \quad (2,3) \quad (0,0) \\ (7,1)-(8,1) & \xrightarrow{\bar{r4}} \quad (4,3) \quad (0,0) \\ (9,1)-(10,1) & \xrightarrow{\bar{r5}} \quad (8,3) \quad (6,3) \end{cases} \qquad \phi_{+}^{L} = \begin{cases} (0,0) \quad (0,0) & \xrightarrow{r1} \quad (1,1)-(2,1) \\ (0,0) \quad (0,0) & \xrightarrow{r2} \quad (3,1)-(4,1) \\ (0,0) \quad (4,4) & \xrightarrow{r3} \quad (5,1)-(6,1) \\ (0,0) \quad (6,3) & \xrightarrow{r4} \quad (7,1)-(8,1) \\ (2,3) \quad (8,2) & \xrightarrow{r5} \quad (9,1)-(10,1) \\ (1,1)-(2,1) & \xrightarrow{\bar{r1}} \quad (0,0) \quad (0,0) \\ (3,1)-(4,1) & \xrightarrow{\bar{r2}} \quad (0,0) \quad (0,0) \\ (5,1)-(6,1) & \xrightarrow{\bar{r3}} \quad (0,0) \quad (4,2) \\ (7,1)-(8,1) & \xrightarrow{\bar{r4}} \quad (0,0) \quad (6,3) \\ (9,1)-(10,1) & \xrightarrow{\bar{r5}} \quad (2,3) \quad (8,4) \end{cases}$$

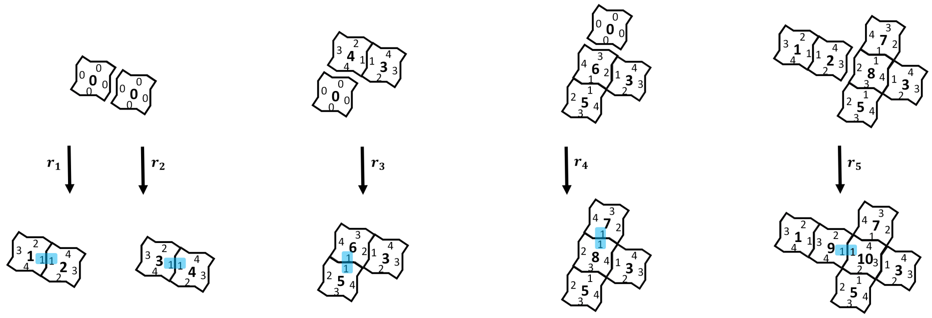

Fig. 5: Progress of the SA process for the cross shape target structure employing ϕ_{+}^{L}. The latest engaged latching connectors on the modules are highlighted with a blue mark, while the relative hop numbering starting at the most recently engaged latching connector are shown on the sides of each module.

The resulting rulesets are not easy to understand at first glance. Here, we provide additional explanations and visualizations in order to bring additional intuition on their operation. Consider the ϕ_{-}^{S} whose SA progress course using six Lily robotic modules is visualized in Figure 4. While the state labels returned by SingletonR for Lilies are similar to the ones for a chain shape in the case of bodiless modules presented in 5.1, it can be seen that the values of $l_h = 3$ on the left-hand-side (LHS) of the rules dictate two hops on the link slots between the successive latching events, resulting in a linear structure considering the square-shaped modules. The reverse rules all have $l_h = 1$ at the LHS, indicating that the rule's corresponding interaction happens at the link slot engaged the latest.

Consider the ϕ_{+}^{S} whose SA progress course using six Lily robotic modules is visualized in Figure 5. Each square represents a Lily, labeled with its internal state l_a value in the middle. The most recent engaged link slot is indicated with a blue mark, while the relative hop numbers of l_h are shown on the modules' sides. For each Lily, numbering the slots always starts with $l_h = 1$ at the most recently engaged slot and follows a CCW convention. Note that the synthesis algorithms only generate the rules; appropriate probabilities should be associated with forward and reverse rules in order to allow the system to recover from deadlocks, while reliably forming the target.

6 Simulation frameworks

In order to compare the performance of different rulesets synthesized by SingletonR and LinchpinR algorithms for SA of our Lily robotic modules and to study the transient system behavior corresponding to each of these ruleset controllers, we utilize two simulation frameworks: a non-spatial microscopic and a spatial submicroscopic framework, each shedding complementary light on specific aspects of the process.

The purpose of the microscopic framework is to allow for the comparison of the intrinsic performance of the derived rulesets, i.e., the final yield and the convergence rate determined by the concurrency in the ruleset, in absence of any influence of physical phenomena on the application of the rules. This is particularly interesting considering that the two rule synthesis algorithms, i.e., SingletonR and LinchpinR, are agnostic about the spatial aspects of the system, analogous to their abstract graph counterparts, the Singleton and Linchpin algorithms. More specifically, given a target structure, the relevant metric when comparing rulesets synthesized by SingletonR and LinchpinR is the number of concurrent steps as defined in Section 5.1 for formal definition. In reality, the realization of the conditions under which each step can be executed depends directly on the spatial characteristics of the system influenced by, for instance, the density of the modules and their mobility due to agitation in the environment, effects that are not taken into account in this microscopic simulation framework.

The submicroscopic framework, on the other hand, provides a realistic replication of the real experimental setup, faithfully capturing the physics of the SA process in the system. This framework allows for the comparison of the performance of the ruleset controllers in simulation under realistic conditions, revealing the outcome of the interplay of the physical characteristics of the system and the assembly strategy of the ruleset controllers. This is particularly interesting considering that the functionality of the ruleset controllers depends on the robotic modules' randomly arranged encounters. The nature of these random encounters is strongly determined by the physical characteristics of the system. More specifically, since the Lily robotic modules are not self-locomoted and are assumed to be driven around by the environmental agitation, we are essentially relying on diffusion for module transportation and thus the performance of the assembly process can be hindered by the diffusion limitations in the system. In other words, if the robotic modules do not have the chance for proper interactions, the target structure will never form, regardless of any well-designed features of the employed ruleset controllers.

6.1 Microscopic simulation framework

The microscopic simulation framework is based upon the abstract model for randomized interactions among bodiless modules introduced in (Fox & Shamma, 2015). We build on this method in two ways. First, in order to model interactions between robotic modules the notion of 'extended graphs‘ along with appropriate geometrical constraints is utilized. Second, we introduce a new shape recognition method which is an extension over a graph isomorphism check to track the progress of the SA process in the system. This method is directly generalizable to rotationally symmetric modules with an arbitrary number of latching connectors.

Random pairwise interactions In our extended formalism, a random pairwise interaction dynamics is defined as a quadruple (G, F, ϕ, P). Rule probabilities are assigned by $P : \phi \rightarrow (0, 1]$. The set of pairs of disjoint vertices is

```
procedure GroupToShape($V_G, E_G, S_G$)
    $\forall\ v_i : i = 1, 2, ..., |V_G|, pos(v_i) = (0, 0)$
    $\forall\ v_i : i = 1, 2, ..., |V_G|, prev(v_i) = 0$
    $\forall\ v_i : i = 1, 2, ..., |V_G|, dir(v_i) = 1$
    $unvisited \leftarrow \{v_1\}$
    $visited \leftarrow \{\varnothing\}$
    while ($unvisited \neq \{\varnothing\}$) do
        $v_i \leftarrow unvisited(1)$
        $\bar{s_i} \leftarrow (dir(v_i) - 1 + 2) \pmod 4 + 1$
        $s_i \leftarrow S_G(v_i, prev(v_i))$
        $d \leftarrow \bar{s_i} - s_i$
        if $d < 0$ then
            $d \leftarrow d + 4$
        end if
        $\{v_j : j = 1 : |n_{E_G}(k)|\} \leftarrow n_{E_G}(k)$
        for $j = 1$ to $|n_{E_G}(k)|$ do
            if $v_j \notin visited$ then
                $s_j \leftarrow S_G(v_j, v_i)$
                $\bar{s_j} = (s_j - 1 + d \pmod 4) + 1$
                for $k = 1 : 4$ do
                    if $\bar{s_j} == k$ then
                        Let $R(\theta)$ be the 2D
                        rotation matrix
                        $pos(v_j) \leftarrow pos(v_i) +$
                        $[-1, 0]R(k \cdot \pi/2)$
                    end if
                end for
                $dir(v_j) \leftarrow \bar{s_j}$
                $prev(v_j) \leftarrow v_i$
                $unvisited \leftarrow unvisited \cup \{v_j\}$
                $visited \leftarrow visited\ \cup\ \{v_i\}$
            end if
        end for
        $unvisited(1) \leftarrow \{\varnothing\}$
    end while
    $(x_{min}, y_{min}) = \{(x, y)|$
    $(x, y) = pos(v_i), \forall j, pos(v_j) > pos(v_i)\}$
    $\forall i \quad pos(v_i) \leftarrow pos(v_i) - (x_{min}, y_{min})$
    return ($pos$)
end procedure
```

Alg. 3: Pseudo code of the shape recognition algorithm for the case of square-shaped Lily robotic modules.

defined as $PW(G) = \{(x, y) : \nexists I \subset G | (x, y) \in E_I, x \neq y\}$, where I is a connected subgraph of G. The set $PW(G)$ specifies the modules among which an interaction is feasible as they are not connected to the same subassembly. $F(G)$ maps an extended graph G to probabilities of pairwise vertex selections from V_G. A random trajectory of the system, is generated by sampling $F(G_t)$ at each time instant to obtain a pair (x, y) and then executing an appropriate action on the selected pair. For the two selected vertices to interact, engaged link slots are chosen randomly from the available slots. Sampling from $F(G_t)$ introduces an inherent stochasticity to the trajectories of the system even if the ruleset contains only deterministic rules. The interaction probabilities, defined by $F(G_t)$, depend on the current graph G_t and can be calibrated based on experimental data to reflect the spatial aspects of the underlying SA process. More specifically, it is the dynamics of mixing in the physical system which affects the interaction chances of different assemblies. For instance, larger assemblies may move around the arena more slowly, or orient themselves in the fluidic field in such a way that certain encounters are less probable. In the current work, our goal is to employ the microscopic simulation framework to study the intrinsic performance of the synthesized rulesets, similar to the studies conducted in (Fox & Shamma, 2015). Therefore, the interaction probabilities are kept uniform similar to (Fox & Shamma, 2015).

Shape recognition Tracking the progress of the SA process of the simulated system requires a mapping between the connected components of the graph of the system and the shape of the corresponding assemblies. For the case of SA of graphs, where the system is represented by an abstract graph at each time instant, this describes a problem of graph isomorphism (Fox & Shamma,

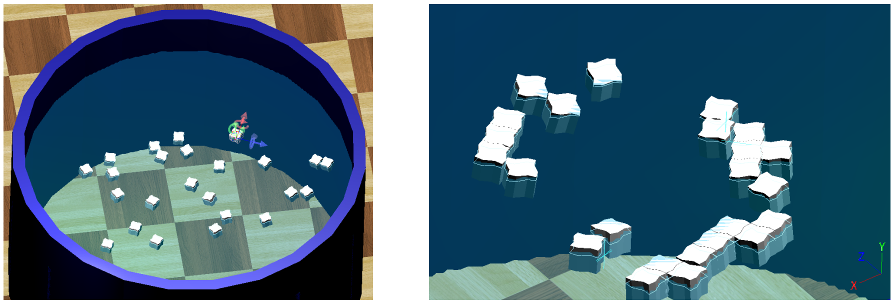

Fig. 6: Simulated world of Lily robots in Webots (Michel, 2004) (left), along with a close-up of the floating simulated Lily robotic modules (right).

2015). However, for the case of our extended graphs, the relative position of the engaged slots needs to be taken into account to recognize the shape of the resulting assembly. We propose a simple method for recognizing the shapes based on traversing the connected components of the extended graph and constructing a series of locations of the Center Of Mass (COM) of the robotic modules. The relative ordering of the link slots on the neighboring modules determines the orientation of each traverse. The series of locations are then rotated and translated such that all coordinates are positive. The resulting ordered set is used as the identifier of the structure. This method can be applied to modules with a variety of shapes. Our method is sufficient for the case of structures confined in 2D and is substantially less computationally expensive than general approaches such as the ones presented in (Asadpour et al., 2009), (Golestan et al., 2013). The pseudo code of our proposed shape recognition algorithm for the case of Lily robotic modules is shown in Algorithm 2.

6.2 Submicroscopic simulation framework

Figure 6 depicts the submicroscopic simulation of the system. In this context, submicroscopic reflects the fact that the model provides a higher level of detail than a canonical microscopic model, faithfully reproducing intra-robot features (e.g., body shape, individual sensors and actuators). With this level of details, a submicroscopic simulator can keep track of a number of state variables such as the exact pose of the robotic node, the specific forces exerted by one of its actuators, or the signal perceived by one of its sensors. In order to faithfully recreate our self-assembling system in simulation, we use Webots (Michel, 2004), a physics-based robotics simulator. Webots uses the Open Dynamics Engine (ODE) for simulating rigid body dynamics. Additionally, in order to simulate specific non-natively supported physics, it is possible to employ custom-designed physics plugins. The Lily robotic modules' CAD design as well as the robots' controller software are imported into the Webots simulated world. The rulesets programmed on the simulated robots are also identical to the case of the microscopic simulation in the previous section. The latest version of Webots supports

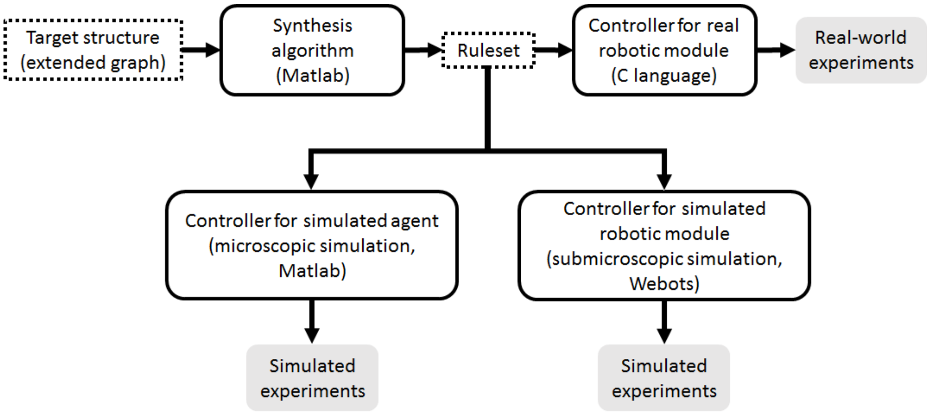
Target structure (extended graph)
Synthesis algorithm (Matlab)
Ruleset
Controller for real robotic module (C language)
Real-world experiments
Controller for simulated agent (microscopic simulation, Matlab)
Controller for simulated robotic module (submicroscopic simulation, Webots)
Simulated experiments
Simulated experiments

Fig. 7: Diagram of the overall software framework. The "synthesis algorithm" block may utilize different rule synthesis algorithms. In the current work SingletonR and LinchpinR synthesis algorithms have been used.

a basic fluid node which allows for a simple uniform stream velocity, but is not capable of simulating a complex fluidic field. We used a similar approach as (Di Mario et al., 2011) to reproduce the complex flow field and the corresponding hydrodynamic forces. In particular, we developed a dedicated physics plugin for the simulated world in Webots that applies the drag force to the simulated Lily robotic module based on the velocity of the module and the flow velocity at its location at each time instant. The details of this development can be found in (Haghighat & Martinoli, 2016b).

7 Experiments and results

We have conducted simulated experiments using the microscopic and submicroscopic simulation frameworks described in Section 6, as well as real-world experiments using the platform described in Section 2. The performance of the rulesets synthesized by Singleton and Linchpin algorithms for SA of bodiless modules have been comparatively studied in (Fox & Shamma, 2015). The purpose of this section is primarily to validate the rulesets generated by the SingletonR and LinchpinR algorithms using the different simulated and real platforms and secondarily, to provide a comparison between the performance of the rulesets synthesized by the two algorithms for the SA of Lily robotic modules. We expected to observe similar trends in the performance of the rulesets of SingletonR and LinchpinR compared to the ones of Singleton and Linchpin. More specifically, the concurrency in the rulesets synthesized by LinchpinR should allow for an intrinsically (i.e., disregarding spatial effects) higher assembly rate than that of SingletonR rulesets. We investigated this aspect in our first set of experiments conducted within the microscopic simulation framework. In our second set of experiments, we employed the submicroscopic simulation framework

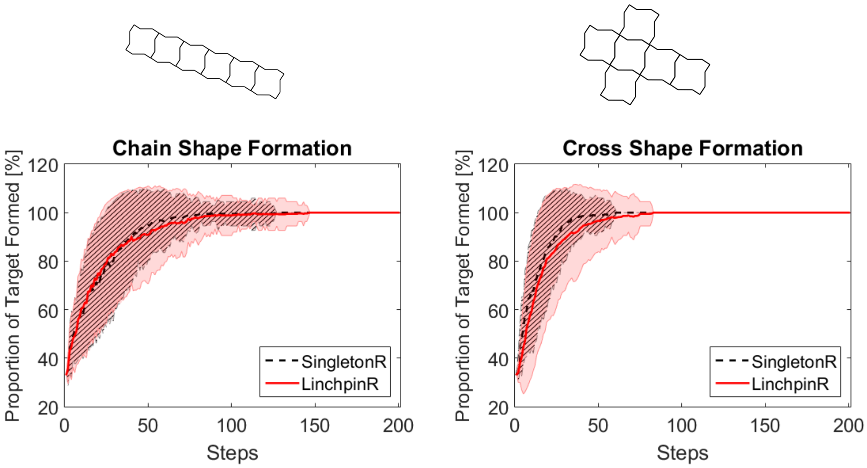

Fig. 8: Comparison of the rulesets synthesized by SingletonR and LinchpinR algorithms in the microscopic simulation using six Lily robots for the two target shapes. The lines (dashed for SingletonR and continuous for LinchipR) and shaded regions (striped for SingletonR and uniform for LinchpinR) indicate the mean and standard deviation of 100 runs, respectively.

to investigate the performance of the rulesets in more realistic conditions where the spatial aspects of the underlying SA process are carefully modeled. Finally, we used physical Lily robotic modules to evaluate the rulesets performances in real world experiments.

Figure 7 depicts the structure of the overall software framework developed and employed in this work. Rulesets for SA of Lily robotic modules are synthesized utilizing the SingletonR and LinchpinR algorithms in the synthesis algorithm block depicted in Figure 7 to derive rules for 1) the chain shape target assembly, and 2) the cross shape target assembly, both of size six. The synthesized rulesets, explained in detail in Section 5.3, are deployed at all the three implementation levels (microscopic and submicroscopic simulations as well as real world) using six modules and only in simulation (microscopic and submicroscopic) using 24 modules. For forward rules $P(.) = 1$ and for reverse rules $P(.) = 0.1$ are chosen. In addition, within a ruleset, all the rules with identical LHS are set to be equi-probable and share the $P(.) = 1$ which is set for forward rules. This concerns only rules in the LinchpinR ruleset where the two rules forming dimers label them probabilistically. As a result each rule is executed with $P(.) = 0.5$. The finishing rule is chosen to be irreversible in all the rulesets, i.e., $P(\bar{r}_5) = 0$, giving rise to stable target assemblies once they are formed. There exists a breadth of research on optimization of ruleset controllers for programmable self-assembling systems (Klavins et al., 2006a), (Matthey et al., 2009). However, with the focus of our current work being on automatizing rule

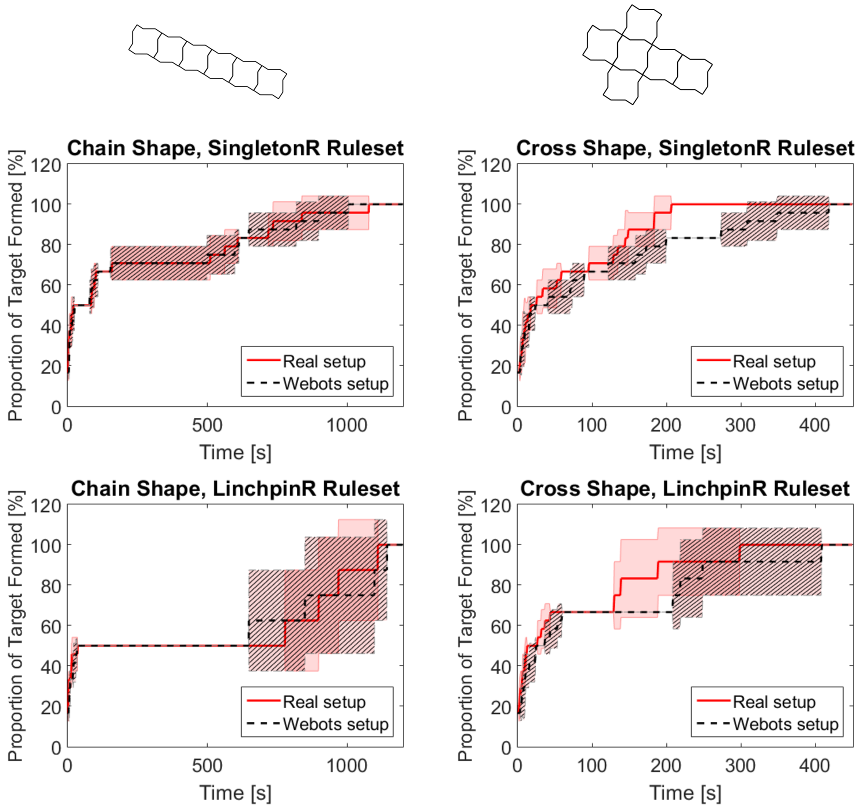

Fig. 9: Comparison of experimental results obtained with the real set-up and the submicroscopic simulation framework (Webots) using six Lily robotic modules for the two target shapes. The lines (dashed for the real setup and continuous for the Webots setup) and shaded regions (striped for the real setup and uniform for the Webots setup) indicate the mean and standard deviation of five runs, respectively.

synthesis for robotic modules, the rule probabilities are chosen empirically and are not necessarily optimal. The reverse rules probability is chosen such that the dissociation of advanced assemblies is roughly less probable than the formation of a more advance assembly. In other words, the assemblies are stable enough not to disassemble before a further rule can be applied in order to progress the assembly process, for the chosen agitation regime in the fluidic arena. Two other values were tested as well, $P(.) = 0.5$ which typically resulted in having all the modules isolated and $P(.) = 0.01$ which typically resulted in having the modules all stuck in a dimer formation.

Mean and standard deviation of several sample runs are used for performance study and comparison in (Fox & Shamma, 2015). We use the same statistical in-

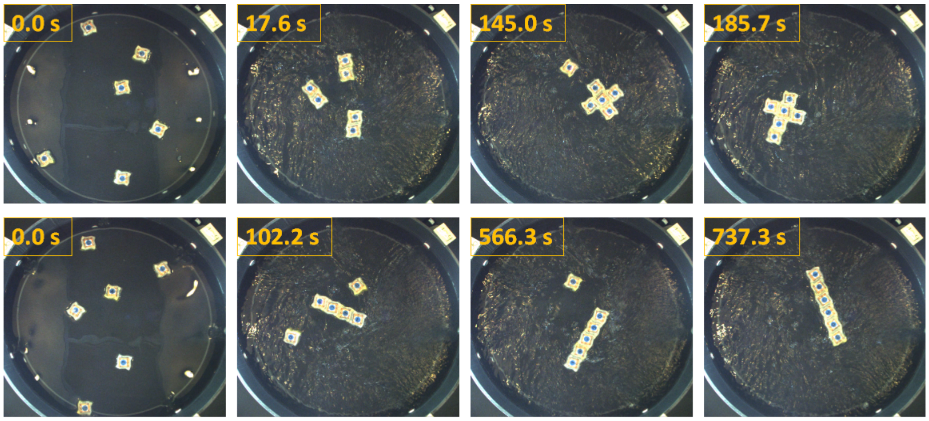

Fig. 10: Snapshots of the SA process employing SingletonR rulesets for the cross shape target (arranged in the top row) and chain shape target (arranged in the bottom row) and using six Lily robots.

dicators due to roughly symmetric quasi-Gaussian distribution of our data for all the plots in this paper. Figure 8 shows the performance of the rulesets derived by SingletonR and LinchpinR for the two target structures in microscopic simulation with a total of six available modules, all initially isolated. With a maximum feasible yield of one (i.e., six available robotic modules and targets of size six), the vertical axis shows the proportion of modules in the correct placement. The horizontal axis shows the number of steps, with each step representing a formation event in the system as a result of application of a forward deterministic rule. For both target structures, the rulesets derived by SingletonR and LinchpinR exhibit similar assembly rates, in other words, the curves have similar slopes. However, it is interesting to note that the LinchpinR rulesets generally exhibit slightly higher variability around the mean performance value. Both of these observations can be explained considering the assembly strategies of the corresponding rulesets. The LinchpinR ruleset by design builds the target structure in fewer concurrent steps than the SingletonR ruleset (three versus five concurrent steps for the case of the chain structure and four versus five concurrent steps for the case of the cross structure). However, such concurrency is unexploited if not enough modules are supplied to the ruleset. Indeed, as we will see later (Figure 11 depicting results achieved with 24 modules), it is sufficient to increase the absolute number of available modules (and therefore increase the number copies of the objective target) to see a clear exploitation of the Linchpin superior concurrency. More specifically, considering the rules in ϕ^L_- and ϕ^L_+, LinchpinR builds dimers with two possible labelings assigned probabilistically with equal probability (as introduced in the ruleset), while SingletonR adds modules one by one, labeled deterministically. With exactly six modules available, if the probabilistically assigned labeling happens not to be of the type needed, the progress of

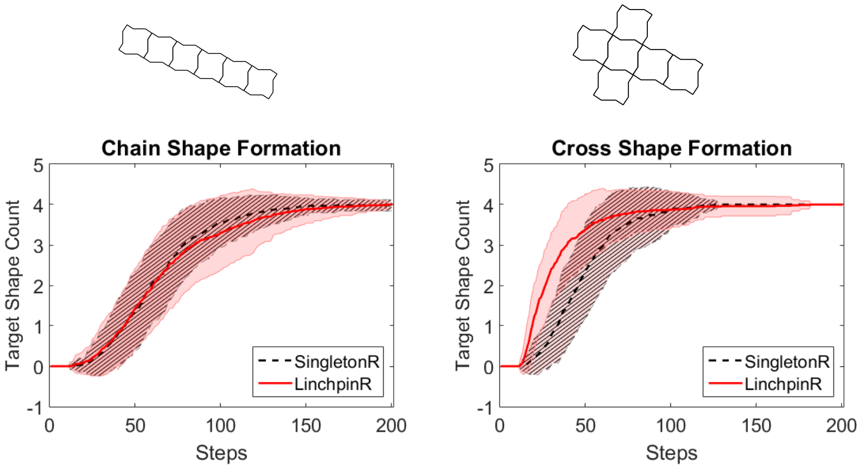

Fig. 11: Microscopic simulation results of rulesets derived by the two extended synthesis algorithms for the two target structures of chain and cross shape. The lines (dashed for SingletonR and continuous for LinchipR) and shaded regions (striped for SingletonR and uniform for LinchpinR) summarize the mean and standard deviation of 100 runs, respectively.

the SA process is delayed until a dimer disassembles and reassembles with the required labeling.

Real-world experiments were conducted by programming six Lily robots with the four derived rulesets to build the two target structures. Each experiment was repeated five times. The same experiments were also conducted in the submicroscopic simulation framework, each repeated five times, to provide a direct comparison between the simulated and the real-world setups. Figure 9 shows the evolution of the target structure in the simulated and real-world setups. With a maximum feasible yield of one (i.e., six available robotic modules and targets of size six), the vertical axis shows the proportion of modules in the correct placement. For all the simulated and real experiments, the maximum yield of one was achieved as depicted in Figure 9. These results roughly indicate a good matching between the two setups. Table 1 details the formation time statistics from the real-world experiments. The low number of runs (five runs per experiment) limits the significance of the gathered statistics. However, as we will show below through leveraging the submicroscopic simulation framework, the observations from these experiments are confirmed in simulated experiments repeated for a larger number of runs and when higher number of modules are available. Considering the real-world results for the chain shape, while the median formation time for SingletonR is less than that of LinchpinR, the minimum and maximum formation times achieved by the two rulesets are close (see Table 1 results and horizontal width of the blue region in Figure 9, left column). Linch-

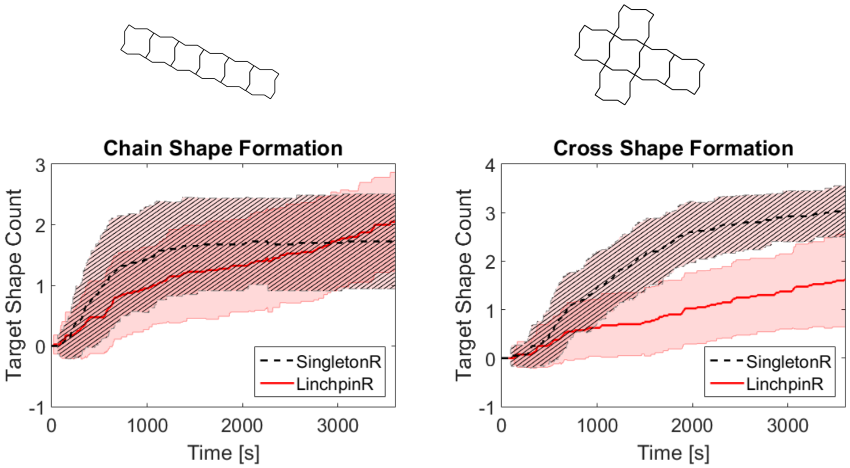

Fig. 12: Submicroscopic simulation results of the performance of rulesets derived by the two extended synthesis algorithms for the two target structures of chain and cross shape. The lines (dashed for SingletonR and continuous for LinchipR) and shaded regions (striped for SingletonR and uniform for LinchpinR) summarize the mean and standard deviation of 50 runs, respectively.

pinR builds the target out of dimers and requires two dimers labeled differently. Since the labeling is done at random, when the available modules are scarce this can easily result in longer formation times. In other words, LinchpinR does not necessarily make the best use of the available resources. This explains how SingletonR manages to achieve lower median. Additionally, the specific interaction configuration that LinchpinR requires for chain formation, i.e., two chains of size three joining to form the target chain of size six, is particularly difficult to arrange, while for the case of the SingletonR ruleset, the isolated Lily module seems to manage more easily to reach the interaction site. For the cross shape, both the smallest and the largest formation times were obtained by LinchpinR. This can be explained by considering the interaction between the intermediate subassemblies. While LinchpinR builds the target through four concurrent steps as opposed to SingletonR's five, the relative orientation of the connecting subassemblies is more easily achieved for SingletonR where one component, i.e., the isolated Lily, is always symmetric. Figure 10 depicts the progress of the SA process for the two target structures deploying the SingletonR ruleset on the Lily robots. These observations highlight the importance of having simulation tools at different modeling levels as each tool manages to shed light on aspects which remain out of reach of the other tools.

Figure 11 shows the performance of the rulesets derived by the two extended synthesis algorithms for the two target assemblies in microscopic simulation using 24 available modules, all initially isolated. The vertical axis shows the number of copies of the target assembly in the system at each step. The four rulesets exhibit interestingly different performance in comparison with the previous scenario employing only six modules. For the cross shape target, the naturally serial ruleset of SingletonR is outperformed by the more concurrent one of LinchpinR, achieving the target with fewer rule executions. For the chain shape target, the rulesets of the two algorithms perform similarly. With a large number of available modules compared to the desired target size, the strategy of LinchpinR to build dimers with two possible labelings assigned probabilistically proves efficient in comparison to the one of SingletonR, adding modules one by one with deterministic labelings.

Figure 12 shows the submicroscopic simulation results of the derived rulesets using 24 available modules all initially isolated. These findings further verify the results of the real-world experiments. For the chain shape, both rulesets exhibit a high variability in the formation time, and perform similarly in effect. However, for the case of the cross shape, the performance of the two rulesets is significantly different. SingletonR outperforms LinchpinR in this case achieving lower average formation time as well as lower standard deviation. This is in agreement with the observations of the real experiments and highlights the strong spatial effects. Even though the LinchpinR ruleset is capable of forming the target with fewer rule executions, the final rule forming the target requires a specific configuration which is not easily achieved in the system.

8 Conclusion

In this paper, we addressed the problem of rule synthesis for programmable SA of rotationally symmetric robotic modules endowed with genderless latching connectors. More specifically, we focused on a case study involving robotic modules relying on the surrounding environment for their mobility. The SA process in the system was guided towards achieving a global target structure in a distributed fashion by means of appropriate ruleset controllers programmed on the robotic modules. The robotic modules maintained an internal state corresponding to their local perception of the progress of the SA process. The ruleset controller programmed on the modules regulated the outcome of the random interactions between two robotic modules based on their internal states.

Table 1: Real experiment results of the four rulesets derived by the two extended synthesis algorithms for the two target structures of chain and cross shape. Formation time statistics are reported for five runs of each experiment.

Algorithm	Target	Median (s)	Mean (s)	Min. (s)	Max. (s)	Std. (s)
SingletonR	Chain shape	788	844	720	1080	166
LinchpinR	Chain shape	935	941	780	1112	139
SingletonR	Cross shape	185	181	146	208	26
LinchpinR	Cross shape	165	190	131	300	78

In a broad sense, the engineering question that motivates this effort can be formulated as follows: given a desired target structure composed of several robotic modules, how can we design the proper ruleset controllers to be deployed on the individual robotic modules? Our focus in this work has been thus on formulating rule synthesis algorithms for the SA of robotic modules. In particular, we considered the specific but widely common case of rotationally symmetric robotic modules endowed with genderless latching connectors.

The main contribution of this work is introducing the extended graph grammar formalism. This was motivated by the fact that previous research employing the standard graph grammar formalism for formulating SA of bodiless modules had demonstrated the relevance and application of the formalism and successfully presented several automatic rule synthesis algorithms. However, the rulesets synthesized by these algorithms were not directly applicable to robotic modules. As a result, one had to further tune the rulesets synthesized by these algorithms to fit the specific morphology of the robotic modules used in their study or to manually design them. The extended graph grammar formalism allows for taking into account the morphology of the robotic modules by introducing the notion of extended graphs comprising extended vertices with ordered link slots representing the robotic modules' connectors. Additionally, we introduced the notion of extended labels. The internal state of each module is encoded by an extended label. We provided the proof that this extended formalism allows for the generation of rulesets of $O(N)$ complexity, with N being the number of genderless connectors available on a robotic module. Using our extended formalism, we extended two synthesis algorithms originally introduced for the SA of bodiless modules, namely Singleton and Linchpin. Doing so, we obtained their counterparts for the SA of rotationally symmetric robotic modules, namely SingletonR and LinchpinR. Studies on the synthesized rulesets in simulation and reality considering two specific target structures were conducted to validate the functionality and evaluate the relative performance of the synthesized rulesets.

The strength and limitations of our proposed formalism can be summarized as follows. For a rotationally symmetric robotic module with N genderless connectors, our extended formalism allows for a ruleset complexity of $O(N)$ compared with $O(N^2)$ obtained through the convention of assigning one vertex and label per connector. The reduced ruleset complexity is of particular interest for the typical case of miniaturized modules in engineered self-assembling systems as it allows for a reduction of the memory required for storing the rules as well as the amount of data shared among the modules. Employing the extended graph grammar formalism for the SA of rotationally symmetric robotic modules offers the possibility of formulating automatic rule synthesis algorithms capable of deriving rules directly applicable to embodied robotic modules. Such algorithms systematically process a description of the target structure in the form of an extended graph; they go through the graph structure recursively severing edges and generating a new rule for each severed edge, in a computationally efficient way. This systematic scheme allows for designing rule synthesis algorithms with different strategies and characteristics, depending on how the algorithm goes through the graph structure. As opposed to metaheuristic rule synthesis methods, the rule synthesis methods based on graph grammars can be formally studied, the characteristics of their synthesized rulesets can be deduced, and their rule synthesis process is computationally very efficient. However, the limitation of the formal algorithms based on graph grammars, including also our proposed extended formalism, is that the rulesets will not necessarily be optimal under realistic or real-world conditions. Consider the case of LinchpinR and SingletonR studied in this paper. While the SingletonR generates concurrent rulesets by design, under realistic conditions, the advantage of concurrency is not always observable. In other words, while the theoretical and inherent characteristics of the rulesets designed by graph grammar algorithms can be deduced, they may fail to perform accordingly well under realistic conditions. This highlights a limitation of this approach as opposed to metaheuristic rule synthesis algorithms: the effect of the environment and the conditions under which the SA process takes place is not considered when deriving the rulesets. Along the same lines, a second limitation is the fact that rule synthesis algorithms based on graph grammars only generate the structure of the rulesets. The ruleset controller's parameters, i.e., the probabilities associated with the rules, are however undetermined and need to be assigned through a different process.

To summarize, we believe that, in the long run, engineered programmable self-assembling systems comprising miniature active modules have the potential of several applications, most notably in space and medical domains, where robust structure formation out of miniature building blocks in a reversible and re-programmable fashion is of significance. This work provides a graph grammar formalism for formulating models and control methods for the SA of rotationally symmetric robotic modules. Using this formalism, previously proposed rule synthesis algorithms for SA of bodiless modules can be readily transformed into algorithms able to generate rules directly applicable to symmetric robotic modules. Additionally, novel algorithms can be formulated for synthesizing ruleset controllers for robotic SA (Haghighat & Martinoli, 2016b). Future work will be followed on several fronts. We will investigate novel rule-synthesis algorithms allowing for higher concurrency in the process by considering geometrical features of the target. Additionally, optimization techniques will be considered towards assigning optimal rule probabilities, which in this work were essentially empirically chosen, as well as towards generating rulesets which are optimal with regards to environmental characteristics. To this end, we will leverage the high-fidelity calibrated submicroscopic simulation framework for providing a systematic way of evaluating the performance of rulesets with different parametrizations of their associated rule probabilities. Finally, we plan to fully exploit our setup to evaluate the performance of the synthesized rulesets in real experiments involving up to 50 Lily robotic modules.

Acknowledgments.

The authors gratefully acknowledge the highly constructive and insightful assistance of the reviewers of this manuscript. This work has been sponsored by the Swiss National Science Foundation under the grant numbers 200021_137838/1 and 200020_157191/1.

Bibliography

Asadpour, Masoud, Mohammad Hassan Zokaei Ashtiani, Alexander Sproewitz, & Auke Ijspeert 2009. Graph signature for self-reconfiguration planning of modules with symmetry. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5295–5300. IEEE.

Ayanian, Nora, Paul J White, Adám Hálász, Mark Yim, & Vijay Kumar 2008. Stochastic control for self-assembly of xbots. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pages 1169–1176. IEEE.

Bhalla, Navneet, Peter J Bentley, & Christian Jacob 2010. Evolving physical self-assembling systems in two-dimensions. In International Conference on Evolvable Systems, pages 381–392. Springer.

Bhalla, Navneet, Peter J Bentley, Peter D Vize, & Christian Jacob 2012. Programming and evolving physical self-assembling systems in three dimensions. Natural Computing, 11(3):475–498.

Bušev, M. 1994. Synergetics: Chaos, order, self organization. World scientific.

Di Mario, E., G. Mermoud, M. Mastrangeli, & A. Martinoli 2011. A trajectory-based calibration method for stochastic motion models. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4341–4347.

Fox, Michael, & Jeff Shamma 2015. Probabilistic performance guarantees for distributed self-assembly. IEEE Transactions on Automatic Control, 60(12):3180–3194.

Fox, Michael J, & Jeff S Shamma 2010. Communication, convergence, and stochastic stability in self-assembly. In IEEE International Conference on Decision and Control, pages 7245–7250.

Ganesan, Varadarajan, & Mandar Chitre 2016. On Stochastic Self-Assembly of Underwater Robots. IEEE Robotics and Automation Letters, 1(1):251–258.

Golestan, Keyvan, Masoud Asadpour, & Hadi Moradi 2013. A new graph signature calculation method based on power centrality for modular robots. In International Symposium Distributed Autonomous Robotic Systems (DARS), pages 505–516.

Haghighat, Bahar, Emmanuel Droz, & Alcherio Martinoli 2015. Lily: A Miniature Floating Robotic Platform for Programmable Stochastic Self-Assembly. In IEEE International Conference on Robotics and Automation (ICRA), pages 1941–1948.

Haghighat, Bahar, & Alcherio Martinoli 2016a. Characterization and validation of a novel robotic system for fluid-mediated programmable stochastic self-assembly. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2778–2783. IEEE.

Haghighat, Bahar, & Alcherio Martinoli 2016b. A Rule Synthesis Algorithm for Programmable Stochastic Self-Assembly of Robotic Modules. In To appear in proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS).

Haghighat, Bahar, Massimo Mastrangeli, Grégory Mermoud, Felix Schill, & Alcherio Martinoli 2016a. Fluid-Mediated Stochastic Self-Assembly at Centimetric and Sub-Millimetric Scales: Design, Modeling, and Control. Micromachines, 7(8):138.

Haghighat, Bahar, Brice Platerrier, Loic Waegeli, & Alcherio Martinoli 2016b. Synthesizing rulesets for programmable robotic self-assembly: A case study using floating miniaturized robots. In International Conference on Swarm Intelligence (ANTS), volume 9882 of LNCS, pages 197–209. Springer.

Klavins, Eric 2002. Automatic synthesis of controllers for distributed assembly and formation forming. In IEEE International Conference on Robotics and Automation (ICRA), pages 3296–3302.

Klavins, E. 2007. Programmable self-assembly. IEEE Control Systems, 27(4):43–56.

Klavins, E., S. Burden, & N. Napp 2006a. Optimal Rules for Programmed Stochastic Self-Assembly. In Proceedings of Robotics: Science and Systems, Philadelphia, USA.

Klavins, Eric, Robert Ghrist, & David Lipsky 2006b. A grammatical approach to self-organizing robotic systems. IEEE Transactions on Automatic Control, 51(6):949–962.

Lathrop, James I, Jack H Lutz, & Scott M Summers 2009. Strict self-assembly of discrete Sierpinski triangles. Theoretical Computer Science, 410(4-5):384–405.

Lochmatter, Thomas, Pierre Roduit, Chris Cianci, Nikolaus Correll, Jacques Jacot, & Alcherio Martinoli 2008. Swistrack-a flexible open source tracking software for multi-agent systems. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4004–4010.

Matthey, Loïc, Spring Berman, & Vijay Kumar 2009. Stochastic strategies for a swarm robotic assembly system. In IEEE International Conference on Robotics and Automation (ICRA), pages 1953–1958. IEEE.

Michel, O 2004. WebotsTM: Professional Mobile Robot Simulation. Advanced Robotic Systems, 1(1):39–42.

Napp, Nils, Samuel Burden, & Eric Klavins 2006. The statistical dynamics of programmed self-assembly. In IEEE International Conference on Robotics and Automation (ICRA), pages 1469–1476. IEEE.

O'Grady, Rehan, Anders Lyhne Christensen, & Marco Dorigo 2009. SWARMORPH: multirobot morphogenesis using directional self-assembly. IEEE Transactions on Robotics, 25(3):738–743.

Rothemund, Paul Wilhelm Karl 2001. Theory and experiments in algorithmic self-assembly. University of Southern California.

Rubenstein, M., A. Cornejo, & R. Nagpal 2014. Programmable self-assembly in a thousand-robot swarm. Science, 345(6198):795–799.

Salemi, Behnam, Mark Moll, & Wei-Min Shen 2006. SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3636–3641.

Tolley, M., & H. Lipson 2010. Fluidic manipulation for scalable stochastic 3D assembly of modular robots. In IEEE International Conference on Robotics and Automation (ICRA), pages 2473–2478.

Whitesides, George M, & Bartosz Grzybowski 2002. Self-assembly at all scales. Science, 295(5564):2418–2421.

Yan, Hao, Sung Ha Park, Gleb Finkelstein, John H Reif, & Thomas H LaBean 2003. DNA-templated self-assembly of protein arrays and highly conductive nanowires. Science, 301(5641):1882–1884.