Design, Modeling, and Control Methods for Fluid-Mediated Programmable Self-Assembly of Resource-Constrained Robotic Modules

THÈSE N^{O} 8599 (2018)

PRÉSENTÉE LE 22 JUIN 2018

À LA FACULTÉ DE L'ENVIRONNEMENT NATUREL, ARCHITECTURAL ET CONSTRUIT
LABORATOIRE DE SYSTÈMES ET ALGORITHMES INTELLIGENTS DISTRIBUÉS
PROGRAMME DOCTORAL EN ROBOTIQUE, CONTRÔLE ET SYSTÈMES INTELLIGENTS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Bahar HAGHIGHAT

acceptée sur proposition du jury:

Prof. J. Paik, présidente du jury
Prof. A. Martinoli, directeur de thèse
Prof. R. Nagpal, rapporteuse
Prof. M. Sitti, rapporteur
Prof. A. Ijspeert, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Acknowledgements

This work would not have been possible without the help of many people. I would like to start by thanking my advisor Prof. Alcherio Martinoli for his invaluable guidance and unconditional support all throughout the course of this work. His enthusiasm and creativity has greatly influenced the development of this thesis. I am also very grateful for his constant encouragement and support to allow me shape the course of my research.

I would like to thank Emmanuel Droz, who actively contributed to several aspects of the design and implementation of the experimental robotic platform, as well as Massimo Mastrangeli and Grégory Mermoud who introduced me to the field and shared their expertise during earlier stages of this research.

I would like to acknowledge the hardworking and talented students whom I had the pleasure to supervise: Alexandre Cherpillod, Beat Geissmann, Loic Waegeli, Luca Brusatin, Brice Platerrier, Matthias Ruegg, Maximilian Mordig, and Hala Khodr.

I would like to express my gratitude to all the members of the DISAL lab for their kind support. In particular, I would like to acknowledge the backbone of our laboratory, Corinne Farquharson and Denis Rochat.

Finally, I am thankful to all my friends at EPFL. With some I already shared a culture and with some I came to share one over the years. I shall forever remember the unparalleled friendship, kindness, and generosity through which I always found myself in a family in Lausanne.

My deepest gratitude and most sincere appreciation goes to my family: my sister Elaheh, and my parents Nasrin and Hessam whose boundless love, encouragement, and support has always empowered me. I owe you beyond what words may ever describe.

Lausanne, 25 March 2018 Bahar Haghighat

Abstract

THE newly emerged and quickly growing science of nanotechnology has been recognized as one of "the twenty-first century's great leaps forward in scientific knowledge". Self-assembly provides a powerful enabling technique for nanotechnology by providing a bottom-up solution as an alternative to the conventional top-down approach in nano-fabrication. Employing self-assembly in nanotechnology seems in fact inevitable. As we try to build ever smaller structures as big as only a few atoms, utilizing tools for putting the molecular building blocks together proves more and more inefficient and impractical. Alternatively, we may let the building blocks put themselves together, let the molecules do what they do best, self-assembling themselves into useful structures. The big question today is thus, can we learn to build things the way nature does? In order to answer this question and to be able to fully employ the power of self-assembly in nanotechnology, we need to understand the principles of what nature does as she puts structures together through self-assembly. Today, understanding and employing self-assembly is a quest pursued by researchers and engineers in almost every imaginable scientific field. In the pursuit of this goal, our research considers a programmable self-assembling distributed robotic system where the self-assembly building blocks are miniaturized robotic modules of a few centimeters in size. Our work leads us, on a high level, to ask this question: how can we influence and control the process of self-assembly in our system so that our robotic modules put themselves together into specific predefined structures? Addressing this question, we follow three distinct though highly intertwined research directions concerned with the mechatronic design, modeling, and control of our self-assembling robotic system.

A core element of our work is the experimental robotic system. With the goal of realizing a distributed robotic system in which the resource-constrained robotic modules build pre-defined target structures through programmable stochastic self-assembly, our developments are centered around the 3-cm-sized water-floating Lily robotic module. Furthermore, we implement a controllable setup around the Lily robotic modules where several environmental features such as the fluidic flow in the environment as well as the ambient luminosity perceived by the modules can be controlled in order to influence the self-assembly process towards the target structure. The experiments reported in this dissertation has been carried out with up to 15 Lily robotic modules.

Abstract

Developing models that accurately describe the assembly process dynamics is a key component in studying programmable stochastic self-assembling systems. Such models help in: (1) accurately predicting the performances (assembly rate and yield) of the distributed system, and (2) evaluating and optimizing control strategies, whether distributed (e.g., ruleset controllers programmed on the modules) or centralized (e.g., modulating environmental features such as mixing forces deriving random interactions among modules), based on model predictions. We develop models at three abstraction levels, namely submicroscopic, microscopic, and macroscopic. The submicroscopic model provides a realistic replication of the real experimental setup, faithfully capturing the physics of the self-assembly process in the system. The purpose of the microscopic model is to allow for comparing the intrinsic performance of the synthesized rulesets, i.e., the final yield and the convergence rate determined by the ruleset concurrency, in absence of any influence of physical phenomena on the application of the rules. The macroscopic model captures the dynamics of the system in terms of Markovian models. In particular, we utilize the chemical reaction network formalism and develop a Markov model of the system based on the robotic modules' ruleset controllers. The Markov model is then automatically refined into a more complex and accurate hidden Markov model.

Programmable self-assembly defines a subclass of self-assembly processes where the building blocks carry information about the final desired target structure. It is through modifying this information that the outcome of the self-assembly process can be *programmed*. The problem of distributed control for programmable self-assembly is thus one of designing a global-to-local behavioral compiler. The problem of ruleset synthesis for programmable self-assembly of bodiless modules has been studied in the literature by employing graph grammar formalism. We extend the graph grammar formalism and take into account the morphology of the robotic modules. This allows for formulating automatic rule synthesis methods for self-assembly of robotic modules, where the synthesized rules can be directly deployed on the robotic modules, with no further tuning. Moreover, we propose a new rule synthesis algorithm for synthesizing assembly rules which further promote parallelization in the self-assembly process without losing guarantees on the completeness of the achieved target.

Keywords: Programmable self-assembly, distributed robotic systems, mechatronic design, multi-level modeling, distributed and centralized control.

Résumé

La science nouvellement émergée et en croissance rapide de la nanotechnologie a été reconnue comme l'un des "grands progrès du XXIe siècle en science." L'auto-assemblage offre une puissante technique de réalisation pour la nanotechnologie, en fournissant une solution ascendante comme alternative à l'approche descendante conventionnelle en nanofabrication. L'emploi de l'auto-assemblage en nanotechnologie semble en effet inévitable. Alors que l'on essaye de construire des structures de plus en plus petites, ne consistant que de quelques atomes, l'utilisation d'outils pour assembler les briques moléculaires se montre de plus en plus inefficace et peu pratique. Alternativement, on pourrait laisser les briques se rassembler, laisser les molécules faire ce qu'elles font le mieux, s'auto-assembler en structures utiles. La grande question aujourd'hui est donc, pouvons-nous apprendre à construire les choses comme le fait la nature ? Afin de répondre à cette question et d'être en mesure d'employer entièrement le pouvoir de l'auto-assemblage dans la nanotechnologie, nous devons comprendre les principes de ce que fait la nature en assemblant les structures par l'auto-assemblage. Aujourd'hui, la compréhension et l'utilisation de l'auto-assemblage est une quête poursuivie par des chercheurs et des ingénieurs dans quasiment tous les domaines scientifiques imaginables. Dans la poursuite de cet objectif, notre recherche considère un système robotique distribué programmable auto-assemblant dont les briques d'auto-assemblage sont des modules robotiques miniaturisés de quelques centimètres. Notre travail nous amène, de façon abstraite, à poser cette question : comment pouvons-nous influencer et contrôler le processus d'auto-assemblage dans notre système, afin que nos modules robotiques se réunissent en structures spécifiques et prédéfinies ? Pour répondre à cette question, nous suivons trois directions de recherche distinctes, mais très étroitement liées, portant sur la conception mécatronique, la modélisation, et le contrôle de notre système robotique d'auto-assemblage.

Un élément essentiel de notre travail est le système robotique expérimental. Dans le but de réaliser un système robotique distribué, dans lequel les modules robotiques à ressources limitées construisent des structures prédéfinies, grâce à un auto-assemblage stochastique programmable, nos développements sont centrés autour du module robotique flottant Lily, mesurant 3 cm. En outre, nous implémentons une configuration contrôlable autour des modules robotiques Lily, dans laquelle plusieurs caractéristiques environnementales, telles que le

flux fluidique dans l'environnement ainsi que la luminosité ambiante perçue par les modules robotiques, peuvent être contrôlées afin d'influencer le processus d'auto-assemblage vers la structure désirée. Le travail expérimental rapporté dans la dissertation a été réalisé avec jusqu'à 15 modules robotiques Lily.

Le développement des modèles qui décrivent précisément la dynamique du processus d'assemblage est un élément clé dans l'étude des systèmes d'auto-assemblage stochastiques programmables. Ces modèles aident à : (1) prédire avec précision les performances (vitesse d'assemblage et rendement) du système distribué et (2) évaluer et optimiser les stratégies de contrôle, qu'elles soient distribuées (par exemple, les règles programmées sur les modules) ou centralisées (par exemple, moduler les caractéristiques environnementales telles que les forces de fluctuation menant à des interactions aléatoires entre les modules), basées sur des prédictions du modèle. Nous développons des modèles à trois niveaux d'abstraction, à savoir sous-microscopique, microscopique et macroscopique. Le modèle sous-microscopique offre une réplication réaliste de la configuration expérimentale réelle, reflétant fidèlement la physique du processus d'auto-assemblage dans le système. Le but du modèle microscopique est de permettre d'établir une comparaison de la performance intrinsèque des règles synthétisées, c'est-à-dire, le rendement final et le taux de convergence déterminés par le parallélisme dans les règles, en l'absence de toute influence de phénomène physique sur l'application des règles. Le modèle macroscopique représente la dynamique du système en termes de modèles Markoviens. En particulier, nous utilisons le formalisme du réseau de réaction chimique et développons un modèle de Markov du système, basé sur les règles des modules robotiques. Le modèle de Markov est ensuite affiné automatiquement en un modèle de Markov caché plus complexe et plus précis.

L'auto-assemblage programmable définit une sous-classe de processus d'auto-assemblage où les briques du système contiennent des informations sur la structure finale souhaitée. C'est en modifiant cette information que le résultat du processus d'auto-assemblage peut être *programmé*. Le problème du contrôle distribué pour l'auto-assemblage programmable revient donc à celui de la conception d'un bloc interpréteur des comportements globaux aux locaux. Le problème de la synthèse des règles pour l'auto-assemblage programmable de modules abstrait a été étudié dans la littérature en utilisant le formalisme grammatical des graphes. Nous étendons le formalisme grammatical des graphes et prenons en compte la morphologie des modules robotiques. Ceci permet de formuler des méthodes de synthèse automatique des règles pour l'auto-assemblage de modules robotiques, grâce auxquelles les règles synthétisées peuvent être directement déployées sur les modules robotiques, sans réglage supplémentaire. De plus, nous proposons un nouvel algorithme de synthèse de règles pour synthétiser des règles d'assemblage qui favorisent la parallélisation dans le processus d'auto-assemblage sans perdre garanties sur l'obtention de la structure atteinte.

Mots clés : Auto-assemblage programmable, systèmes robotiques distribués, conception mécatronique, modélisation multi-niveaux, contrôle distribué et centralisé.

Zusammenfassung

DAS neu entstandene und schnell wachsende Feld der Nanotechnologie wurde als einer der "großen Fortschritte des 21. Jahrhunderts in der wissenschaftlichen Erkenntnis“ anerkannt. Die Selbstorganisation ist eine leistungsfähige Methode für die Nanotechnologie, indem sie eine „bottom-up“ Lösung als Alternativezum herkömmlichen „top-down“ Ansatz in der Nanofabrikation bietet. Der Einsatz von Selbstorganisation in der Nanotechnologie scheint unvermeidlich. Während wir versuchen immer kleinere Strukturen, die nur ein paar Atome groß sind, zu bauen, erweist sichdie Verwendung von Werkzeugen zum Zusammensetzen der molekularen Bausteine als immer ineffizienter und unpraktischer. Alternativdazu können wir die Bausteine sich zusammenfügen lassen, die Moleküle tun lassen, was sie am besten können, sich selbst zu nützlichen Strukturen zusammenfügen. Daraus ergibt sich die zentrale Frage, wie wir es schaffen, können„ Dinge so zu bauen, wie die Natur es tut? Um diese Frage zu beantworten und die Kraft der Selbstorganisation in der Nanotechnologie voll nutzen zu können, müssen wir die Prinzipien verstehen, die die Natur anwendet, wenn sie Strukturen durch Selbstorganisation zusammenbringt. Heutzutage ist das Verständnis und die Anwendung von Selbstorganisation ein Auftrag, der von Forschern und Ingenieuren in fast jedem wissenschaftlichen Bereich verfolgt wird. Um dieses Ziel zu erreichen, betrachten wir ein programmierbares selbstorganisierendes verteiltes Robotersystem, bei dem die selbstorganisierenden Bausteine miniaturisierte Robotermodule von wenigen Zentimetern Größe sind. Unsere Arbeit läuft auf die Frage hinaus: Wie können wir den Prozess der Selbstorganisation in unserem System beeinflussen und kontrollieren, so dass sich unsere Robotermodule in bestimmte vordefinierte Strukturen zusammenfügen? Um diese Frage zu beantworten, verfolgen wir drei unterschiedliche, aber stark verwobene Forschungsrichtungen, die sich mit der mechatronischen Realisierung, der Modellierung und der Kontrolle unseres selbstorganisierenden Robotersystems befassen.

Ein Kernelement unserer Arbeit ist das experimentelle Robotersystem. Unser Ziel ist es, ein verteiltes Robotersystem zu realisieren, in dem die Robotermodule mit begrenzten Ressourcen durch programmierbare, stochastische Selbstorganisation vordefinierte Zielstrukturen aufbauen. Hierbei konzentrieren sich unsere Entwicklungen auf das 3 cm große, schwimmende Robotermodul Lily. Darüberhinaus haben wir einen steuerbaren Aufbau um die Lily Robotermodule herum konzipiert und gebaut, um den Selbstorganisationsprozess in Richtung der

Zielstruktur zu beeinflussen. Verschiedene Kontrollparameter, wie die Strömungsgeschwindigkeit sowie die Umgebungshelligkeit, die von den Robotern wahrgenommen wird, können gesteuert werden. Die experimentellen Arbeiten die in dieser Dissertation beschriebenen sind, wurden mit bis zu 15 Lily Robotermodulen gleichzeitig durchgeführt.

Die Entwicklung von Modellen, die die Dynamik des Zusammensetzungprozesses genau beschreiben, ist eine Schlüsselkomponente bei der Untersuchung programmierbarer, stochastischer, selbstorganisierender Systeme. Solche Modelle helfen dabei: (1) die Leistungen (Montagerate und Ausbeute) des verteilten Systems genau vorherzusagen und (2) Regelstrategien auszuwerten und zu optimieren, ob sie verteilt (zum Beispiel durch auf den Modulen programmierten Regeln) oder zentralisiert sind (zum Beispiel durch Änderung von Umgebungsvariablen, die die zufällige Interaktionen zwischen Modulen beeinflussen), basierend auf Modellvorhersagen.Wir entwickeln dazu Modelle auf drei Abstraktionsebenen: submikroskopisch, mikroskopisch und makroskopisch. Das submikroskopische Modell liefert eine realistische Wiedergabe des realen Versuchsaufbaus, wobei die Physik des Selbstorganisationsprozesses im System wirklichkeitsgetreu erfasst wird. Der Zweck des mikroskopischen Modells ist es den Vergleich der intrinsischen Leistung der synthetisierten Regeln, also die Endausbeute und die Konvergenzrate, die durch die Gleichzeitigkeit im Regelwerk bestimmt wird, in Abwesenheit eines Einflusses physikalischer Phänomene auf die Anwendung der Regeln, zu ermöglichen. Das makroskopische Modell erfasst die Dynamik des Systems in Form von Markov Modellen. Insbesondere verwenden wir den Formalismus des chemischen Reaktionsnetzwerks und entwickeln ein Markov Modell des Systems, das auf den Regeln der Robotermodule basiert. Das Markov Modell wird dann automatisch zu einem komplexeren und genaueren versteckten Markov Modell verfeinert.

Die programmierbare Selbstorganisation definiert eine Unterklasse von Selbstorganisationsprozessen, bei der die Bausteine des Systems Informationen über die endgültige, gewünschte Zielstruktur enthalten. Durch die Änderung dieser Information, kann das Ergebnis des Selbstorganisationsprozesses *programmiert* werden. Das Problem der verteilten Kontrolle für die programmierbare Selbstorganisation besteht somit in der Entwicklung eines Global-zu-Lokal-Verhaltens-Compilers. Das Problem der Regelsatzsynthese für die programmierbare Selbstorganisation von körperlosen Modulen wurde in der Literatur untersucht, indem ein Graphgrammatikformalismus verwendet wurde. Wir erweitern den Graphgrammatikal Formalismus und betrachten dieMorphologie der Robotikmodule. Dieser ermöglicht die Formulierung automatischer Regel-Synthesemethoden für die Selbstorganisation von Robotermodulen, wobei die synthetisierten Regeln ohne weitere Anpassung direkt auf den Robotermodulen implementiert werden können. Darüber hinaus schlagen wir einen neuen Regelsynthesealgorithmus vor, um Assemblierungsregeln zu synthetisieren, die die Parallelisierung Im Selbstorganisationsprozess weiter fördern ohne die Vollständigkeit des erreichten Ziels zu verlieren.

Schlüsselwörter: Programmierbare Selbstorganisation, verteilte Robotersysteme, mechatronische Entwurf, mehrstufige Modellierung, verteilte und zentrale Regelung.

Contents

Introduction Part I

1 Self-Assembly

A recurrent theme in natural structuring phenomena is the absence of a top-down building approach. Nature does not use any assembly tools for putting things together. The elegance of this simple observation represents the essence of the science of self-assembly. From the simplest bacteria to the most intricate galaxies, nature seems to have a special way of building things: it simply lets the building blocks, some alive, and some not, to put themselves together, to *self-assemble* into structures of a wide variety of complexities and scales. Today, understanding and utilizing self-assembly is being actively investigated in several scientific disciplines [1]. In the field of biology, scientists strive to understand the origin of life and nature's principles and techniques in creating intricate structures. In the field of chemistry, researchers study characteristics and formation of ever more complex chemical systems out of molecular building blocks. In engineering fields, novel manufacturing methods are being developed that allow for realization of engineered systems at the nanoscale. In the field of computer science, scientists are learning to employ DNA building blocks to perform massively parallel computations. In the field of mathematics, scientists try to find solutions for the process design problems by developing accurate models of self-assembly. This chapter will provide a clear definition of self-assembly and outline its different categories as well as the motivations justifying why self-assembly research is of utmost interest at the present time.

1.1 Definition

Several different definitions for self-assembly have been adopted by scientists [2], [3], [4], [5], [6]. Similar to [1], we define self-assembly as the spontaneous formation of ordered structures through a reversible stochastic process that involves pre-existing building blocks, and can be controlled through proper design of the building blocks, the environment, and the driving force [1]. Each of the elements of this definition are necessary for specifically characterizing the phenomena which we consider as self-assembly. The word "ordered" distinguishes between self-assembly and aggregation processes. The phrase "pre-existing building blocks" highlights

the difference between self-assembly and pattern formation processes. Additionally, the words "stochastic," "design," "environment," and "driving force" highlight the aspects of self-assembly which play a key role in the design of engineered self-assembling systems.

Here, we highlight four particular subclasses of the class of the self-assembly phenomena [1], [7]. **Static self-assembly** processes result in structures in either local or global equilibrium. From an engineering perspective, the best understood and most highly developed engineered self-assembling systems are within this subclass. **Dynamic self-assembly** processes lead to stable structures under non-equilibrium conditions. That is, the system has to actively dissipate energy for the structures exist. The inspiration for studying such systems is the one identified by Richard P. Feynman in 1959 – biology. The principles of dynamic self-assembly, or equivalently nonequilibrium self-assembly, are yet to be studied in depth. Programmed or **programmable self-assembly** defines involves self-assembly processes where the self-assembly building blocks carry information about the form or function of the target structure. While both passive and intelligent building blocks may be employed to carry out programmable self-assembly, intelligent building blocks would allow us to consider problems such as yield optimization more flexibly. For a building block to be considered truly intelligent, it must be able to sense its state, communicate with its neighbors, and act on this information. **Templated self-assembly** employs templates for guiding the process of self-assembly. Templates are commonly used in the fabrication of macroscale objects. In the context of self-assembly, a template can be used to align or orient the building blocks so that they may bind via a secondary process or so that binding occurs more rapidly. Rather than binding to one another, the building blocks first bind to the template. The building blocks in the template then become bound together, creating the final structure.

1.2 Natural Instances

While it is not difficult to find patterns in nature, not all natural patterns result from self-assembly processes. In this section we take a brief look at several naturally occurring systems that are the result of some form of self-assembly, within both organic and inorganic natural systems. The examples discussed here are taken from [1].

Inorganic Instances of Self-Assembly

The first example we consider is crystallization. Crystals exhibit atomic scale order which may be explained using packing theory. Packing problems are a class of optimization problems in mathematics that involve attempting to pack objects together into containers. Crystals also exhibit specific macro scale structures. From the point of view of self-assembly, what is of interest is not only the various structures forming through the process of crystallization, but also the process of crystallization itself which defines how such structures form.

The second example are the bubble rafts which self-assemble with the capillary forces in

action. The short and long range order in the assembled system can be explained using the optimal packing theory. The bubbles in the assembled cluster are usually of very different sizes. If one of the bubbles bursts, the cluster would quickly rearrange itself in a new configuration filling the vacancy.

The third example is polymerization. Polymers are long-chain macro-molecules formed by monomer building blocks. The polymerization process leads to a distribution of polymer chains of different lengths through a self-assembly process in which the monomer building blocks arrange themselves into ordered complex structures.

Organic Instances of Self-Assembly

The hope for employing self-assembly for building complex functional structures similar to the biological instances discussed in this section is perhaps one of the main reasons for the amount of present interest in studying self-assembly today. We consider two main examples of organic self-assembly.

The first example concerns protein structures and the phenomenon of protein folding. Fundamental in biology, proteins serve both as the structural material and the machinery of the cell. Proteins are essentially polymers, comprising long chain molecules of amino acid building blocks. The folded structure of the protein is determined by the sequence of these amino acid building blocks along the polymer chain. The mechanism through which this sequence determines the final structure defines the protein folding phenomenon.

The second example is the ribosome which provides an instance of a self-assembled nanomachine. The ribosome is comprised of protein building blocks that self-assemble to build a functional system which forms the manufacturing center of the cell producing specific protein structures according to instructions delivered by RNA. The ribosome has been cited as a perfect example of a nanoscale assembler machine, capable of building various nanomachines [8].

1.3 Principles

In order to identify the principle components playing role in self-assembly phenomena, we need to rethink the nature of each phenomenon from an abstract viewpoint. The systems we have mentioned so far, crystals, ribosomes, and bubble rafts, may seem to have little in common. However, when viewed abstractly, these systems are very similar. In fact, we argue that nature repeatedly uses the same principles in designing systems that self-assemble. In the following, we identify and highlight four principle components present in all self-assembling systems: structured building blocks, a binding force, a driving force, and an environment.

The building blocks, the units performing the assembly, represent the first principle component in self-assembly phenomena. The complexity of the final self-assembled system is

directly determined by the features of the building blocks. Consequently, tuning the features of the building blocks allows for controlling the resulting structure in the system, the first means for controlling the self-assembly process.

The binding force between the building blocks holding them together is the second principle. The reversibility of the binding between the building blocks allows the system to move from local equilibria to a global equilibrium. Tuning the binding force provides a second means for controlling the self-assembly process. This could be done by the building blocks as a result of their interactions or by external stimuli. Several types of binding forces may be observed such as capillary, electromagnetic, and chemical bonding forces.

The environment, where the self-assembly process takes place, comprises the third principle component. A proper environment is necessary in order for the binding force to act. For example, capillary forces are only useful when the building blocks sit at the surface of a liquid. Adjusting or dynamically changing the environment provides the third means of controlling the self-assembly process.

The driving force is the last principle component. The building blocks have to stochastically interact with one another in order for the self-assembly process to occur. Essentially, the driving force in the system provides the randomness and moves the system through different possible system configurations on its way to the final ordered configuration.

1.4 Outlook

To celebrate the 125th anniversary of Science, 25 big questions as well as 100 little questions were selected [1], highlighting the most likely research questions to shape the course of scientific research for the next 125 years [9]. Quite remarkably, right among ever sought after big questions such as “What is the universe made of?” and “Are we alone in the universe?” one will find “How far can we push chemical self-assembly?” How did the yet barely defined field of self-assembly rise to prominence so quickly?

This sudden ascent of self-assembly may be ascribed to simultaneous developments in several scientific fields, interactively reinforcing understanding of one another [1]. The advent of nanotechnology quickly highlighted the fact that we need to be able to understand and efficiently utilize self-assembly to operate at such small length scales. It was first at the 1959 annual meeting of the American Physical Society where Richard P. Feynman introduced the idea that humans may create functional machines at the nanoscale. At the same time, several core developments took place in other fields of science and engineering including mathematics, computer science, chemistry, and biology that lead us to where we stand today. We are now on the verge of understanding and utilizing the power of self-assembly. Table 1.1 summarizes these landmarks in the history of self-assembly. Self-assembly is being pursued by researchers in almost every imaginable scientific field today, bringing us closer to recreating the elegance of nature every day.

Time	Event
1930s	Alan Turing develops the theory of universal computation.
1950s	John von Neumann develops theory of automata replication.
1953	James D. Watson and Francis Crick discover the structure of DNA.
1955	H. Fraenkel-Conrat and R.C. Williams self-assemble the tobacco mosaic virus in a test tube.
1957	Penrose and Penrose construct a simple self-replicating system.
1961	HaoWang develops "Wang Tiles" demonstrating the equivalence of tiling problems and computation.
1991	Nadrian C. Seeman and Junghuei Chen self-assemble a cube from DNA.
1994	Leonard Adleman launches the field of DNA computation by using DNA to solve a Hamiltonian path problem.
1996	Kazuo Hosokawa's group demonstrates microscale self-assembly using surface tension.
2000	George M. Whitesides's group self-assembles electrical networks from millimeter scale polyhedra.
2004	William Shih adapts the methods of Seeman to self-assemble a DNA octahedron.
2004	Eric Winfree and Paul Rothemund self-assemble a Sierpinski triangle from DNA demonstrating that self-assembly may be used for computation.
2000s	Self-assembly research explodes drawing the interest of researchers from every imaginable field.

Table 1.1 – Landmarks in the history of self-assembly [1].

Summary

Self-assembly is defined as the spontaneous formation of ordered structures through a reversible stochastic process that involves pre-existing building blocks and can be controlled through proper design of the building blocks, the environment, and the driving force. In the quest of fully understanding self-assembly, a multi-disciplinary effort is undertaken by biologists, chemists, computer scientists, engineers, and mathematicians. In addition to precisely defining self-assembly and the means through which it may be controlled, this chapter briefly reviewed the history of self-assembly and highlighted the motivations for the present interest in studying the science of self-assembly.

2 Engineered Self-Assembling Systems

In this chapter we dedicate our attention to the engineered self-assembling systems at the centimeter scale, some comprising intelligent robotic modules and some passive modules, and review the recent developments made in the design and experimentation of systems most relevant to the focus of this thesis. Throughout our review, we outline that the development of engineered self-assembling systems involves addressing three major problems: the fabrication problem, the analysis problem, and the synthesis problem. The fabrication problem concerns all the practical aspects of fabricating and operating a typically large swarm of self-assembly building blocks. This is particularly important when the building blocks are intelligent robotic modules where programming, charging, deploying, and repairing the faulty ones requires specific careful operations. Therefore, the design and fabrication of the building blocks has to allow for these operations to be doable in large swarms. The analysis problem considers a predictive aspect: given a set of building blocks, a binding force, an environment, and a driving force, what structures will the system produce? The synthesis problem addresses the opposite aspect: given the desired target structure, how do we choose a set of building blocks, a binding force, an environment, and a driving force so that the system builds the target structure?

2.1 Overview

Being increasingly viewed as reliable models for the study of self-assembly processes at a variety of scales, engineered self-assembling systems at the centimeter scale are widely studied throughout different fields in science and technology including biology, chemistry, manufacturing, material science, microelectronics, physics, robotics, and sociology [10], [11]. During recent years, self-assembly has been extensively studied both as an enabling technique for micro-/nano-fabrication and as a coordination mechanism for distributed robotic systems. Several functionalities have been demonstrated within engineered self-assembling systems, including formation [12], [13], [14], [15], [16], [17], growth [18], [19], [20], self-reconfiguration [21], [22], [23], self-repair [24], and template replication [25]. Achieving the level of scalability

and robustness, which inherently exists in natural self-assembling system instances, both in terms of swarm sizes and the size of individual building blocks, has been a key motivation for investigating self-assembly in engineered collective systems, where typically multiple rather simple and small building blocks are deployed.

Simplicity in the building blocks' internal design allows for reducing costs, increasing robustness and also further reduction of the blocks' size, ultimately resulting in finer resolutions in the assembled target structures. While intelligent building blocks can actively take part in the self-assembly process and thus allow for distributed control approaches [15], [26]–[28], the self-assembly process of passive building blocks can only be controlled through a centralized approach modulating the environmental features [17], [29], [30]. Centralized control approaches become quickly computationally intractable and unfeasible as the size of the swarm grows. Distributed control approaches on the other hand, offer the advantage of high scalability as they remain feasible for large swarms. A less explored area of study is where a combination of distributed and centralized control approaches can be applied: the guided programmable self-assembly of miniaturized robots, where the minimal design allows for realizing large swarms of intelligent modules at small scales while a controllable environment can additionally guide the self-assembly process.

In order to understand the system dynamics and to guide the design of the self-assembling systems, both regarding the hardware aspects and the control aspects, models at different abstraction levels are investigated. Various modeling approaches have been employed for understanding the process of self-assembly. Three main approaches can be identified, physical or low-level models where the physical details of the self-assembling system are carefully captured in the model [31]–[33], abstract or high-level models where the model mostly describes the functionality of the underlying assembly process in the system [34]–[37], and multi-level modeling approach where starting from a low-level model a suite of incrementally abstract models are created [38], [39]. In the following sections, we take a closer look at engineered self-assembling systems along their three major aspects of fabrication or mechatronic characteristics, analysis and modeling approach, and control and synthesis approach.

2.2 Mechatronic Characteristics

A wide variety of engineered self-assembling systems have been reported in the literature, comprising both passive and intelligent modules sizing from a few centimeters to half a meter, and weighing from 4 to 11,000 g. While the mechatronic design of the system involves both the design of the environment as well as the constituting modules, typically, most of the sophistication lies in the module design. Tables 2.1 and 2.2 detail the physical and electrical characteristics of the modules in a number of prominent engineered self-assembling systems, broadly categorized according to the mobility of their constituting modules.

In systems with self-propelled modules, self-propulsion may be realized with a differential drive or tracks, providing good locomotion on flat and rough terrains respectively. Modules of

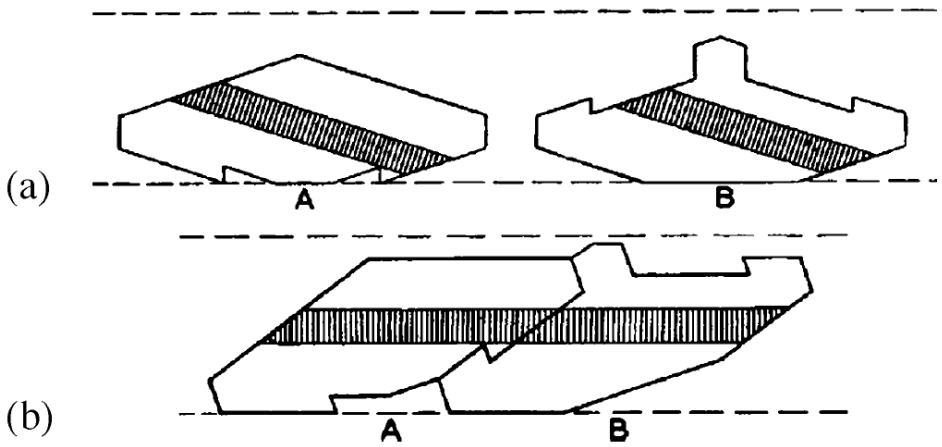
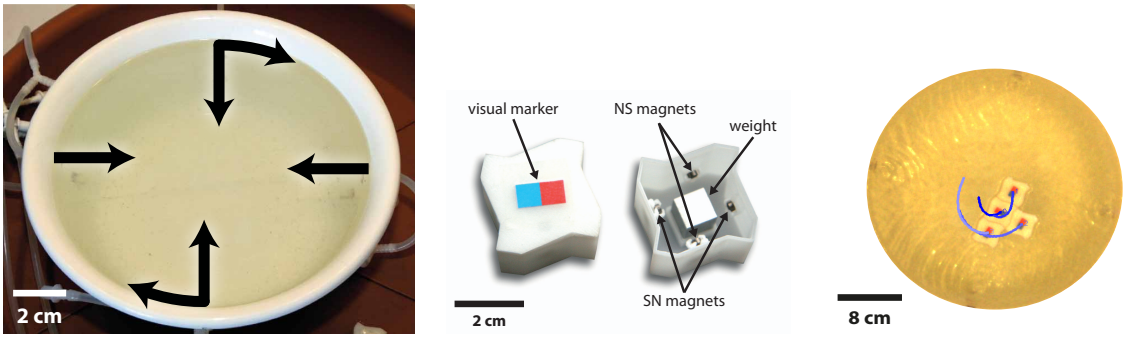


Figure 2.1 – Schematic of Penrose's simple model of self-replication as illustrated in [11].



(a) **(b)** **(c)**

Figure 2.2 – Passive self-assembling Lily modules setup [17]: (a) Water-filled tank; (b) Passive Lily module, including the latching mechanism composed of four permanent magnets with different pole orientation north-south (NS) and south-north (SN), respectively; (c) Real-time visual tracking of modules (the blue lines show a short history of the trajectory of each block)

.

swarm-bot combine both mechanisms [40]. Modules of CEBOT Mark III have nozzles providing propulsion on flat terrain [41]. In some systems with self-propelled modules, by having modules move within the constructed structure, achieving self-reconfiguration. Examples of such systems include CONRO [42], PolyBot [43], and M-TRAN [44]. Self-propelled modules typically have high power consumption, limiting their lifetime without external power supply. Perceiving their surroundings and actively moving towards selected modules is usually part of the capabilities of such modules, putting a great demand on their design.

In systems with externally propelled modules, the modules encounter each other on a random basis. In these systems, the modules are designed to operate in a limited range of environments which may impose constraints on the design. In the system of Griffith et al. [45] and in PPT [15], modules are endowed with on-board batteries. Therefore, in principle, any two modules can bind and communicate with each other upon collision. In White et al.'s systems

[46] and [47], modules are not powered initially. A special seed module is directly linked to an external power supply. Following binding to the seed structure, modules receive power through the connection link. This power sharing scheme and the existence of a seed module is also the case with the Pebble robots system [48]. In general, computing requirements for externally propelled modules are relatively low. In these systems, modules can bind passively upon collision, and use computation to make the decision whether to stay assembled or not. A number of different environmental media have been utilized, providing for the random collisions and interactions among the modules including employing a shaking table or container [34], [48]–[50], an air table [15], [45], [46], agitating the modules floating within a fluid [47], or agitating the modules floating on a fluid surface [17], [51]. All the mentioned systems operate at the centimeter scale. At the sub-millimeter scale, however, utilizing a liquid provides the most efficient way for moving the modules [52].

System	Fig.	Mass	Battery	Processor	Environment	Communication	Latching
Penrose & Penrose [49]	2.1	N	N	N	1D shaking	N	mechanical interlocking upon collision
Hosokawa et al. [34]	2.3(a)	3.6 g	N	N	2D rotating flat box	N	permanent magnets
Breivik [51]	2.3(b)	30 g	N	N	2D fluid	N	permanent magnets
White et al. [46]	2.3(c)	165 g	N	8-bit Basic Stamp II-SX, 50 MHz	2D air table	serial link between connected modules	electromagnets
White et al. [46]	2.3(d)	165 g	N	8-bit Basic Stamp II, 50 MHz	2D air table	serial link between connected modules	swiveling permanent magnets
Grifith et al. [45]	2.3(e)	26 g	Y	8-bit ATmega8, 8 MHz	2D air table	4 wireless electromagnetic local transmitters	mechanical latch, regulated electromagnetically
White et al. [47]	2.3(f)	895 g	N	8-bit Basic Stamp II-SX, 50 MHz	3D fluid	serial link between connected modules	electromagnets and permanent magnets
White et al. [47]	2.3(g)	1480 g	N	8-bit Basic Stamp II, 50 MHz	3D fluid	serial link between connected modules	pressure of fluid flow, regulated by valves
PPT [15]	2.3(h)	110 g	Y	8-bit PIC18F242, 3.6 MHz	2D air table	3 infrared emitters/receivers	swiveling permanent magnets
Bhalla & Bentley [50]	2.3(i)	N	N	N	2D shaking	N	permanent magnets
Pebbles [48]	2.3(j)	N	N	8-bit ATmega328, 8 MHz	2D shaking	4 wireless electromagnetic local transmitters	electro-permanent magnets
Mermoud et al. [17]	2.2	17.3 g	N	N	2D fluid	N	permanent magnets

Table 2.1 – Externally propelled modules' characteristics in various engineered self-assembling systems. Table data is taken from [11] and extended.

System	Fig.	Mass	Battery	Processor	Sensor	Communication	Latching
RSD I [53]	2.4(a)	N	N	relay (1 head, 2 tail)	bump switch (0 head, 3 tail)	parallel link between connected modules	impulse and friction
CEBOT, Mark II [54]	2.4(b)	2700 g	N	sub CPU (+ main CPU off-board)	4 infrared detectors (3 rigid, 1 rotational), 3 ultrasonic distance (1Tx and 2Rx)	9 infrared emitters (8 rigid, 1 rotational), parallel link between connected modules	actuated mechanical hook
CEBOT, Mark III [41]	2.4(c)	N	N	sub CPU (+ main CPU off-board)	9 infrared detectors, 6 ultrasonic distance (3Tx and 3Rx)	9 infrared emitters, parallel link between connected modules	mechanical pin/hole & SMA
CEBOT, Mark IV [55]	2.4(d)	4100 g	N	16-bit 8086, 5-10 MHz	2 infrared detectors	2 infrared emitters, wireless (RS-232C)	actuated mechanical hook
Bererton & Khosla [56]	2.4(e)	250 g	Y	8-bit PIC16C73A, 20 MHz + off-board	B&W camera (320x240), bump switch	wireless (RF)	mechanical pin hole
PolyBot, G2 [43]	2.4(f)	416 g	N	32-bit PowerPC 555 (MPC555), 40 MHz	4 infrared detectors	8 infrared emitters, 2 CANbus	mechanical pin/hole & SMA
PolyBot, G3 [43]	2.4(g)	200 g	N	32-bit PowerPC 555 (MPC555), 40 MHz	8 infrared detectors, 2 2-axis inclinometers, 8 1-axis force	8 infrared emitters, 2 CANbus	mechanical pin/hole & SMA
CONRO [42]	2.4(h)	114 g	Y	8-bit Basic Stamp II-SX, 50 MHz	4 infrared detectors	4 infrared emitters	mechanical pin/hole & SMA
Swarm-bot [40], [57], [58]	2.4(i)	700 g	Y	32-bit XScale, 400 MHz + 13 8-bit PIC16F876/7, 20 MHz	19 infrared proximity, color camera (640x480, omnidirectional), 2-axis force, torque, 4 microphones, 8 light, 3-axis inclinometer, 2 humidity, 2 light barriers	8 RGB LEDs changing body color, 2 speakers, Wi-Fi	actuated mechanical hook
SMC [59]	2.4(j)	11000 g	Y	32-bit Pentium MMX, 233 MHz	color camera (2 per parent: 640x416, 2-3 per child: 320x240), 1-axis force	Wi-Fi	actuated mechanical hook
M-TRAN III [44]	2.4(k)	420 g	Y	32-bit HD64F7047, 48 MHz, 3 16-bit HD64F3687/94, 16MHz + off-board	13 infrared detectors, 3-axis inclinometer	13 infrared emitters, CANbus, wireless (BlueTooth)	actuated mechanical hook
3D M-block [28]	2.4(l)	150 g	Y	32-bit Nordic nRF51422, 16MHz	N	infrared link between neighboring modules	permanent magnets
Kilobot [60], [26]	2.4(m)	N	Y	8-bit ATmega328, 8 MHz	N	infrared link between neighboring modules	N

Table 2.2 – Self-propelled modules' characteristics in various engineered self-assembling systems. Table data is taken from [11] and extended.

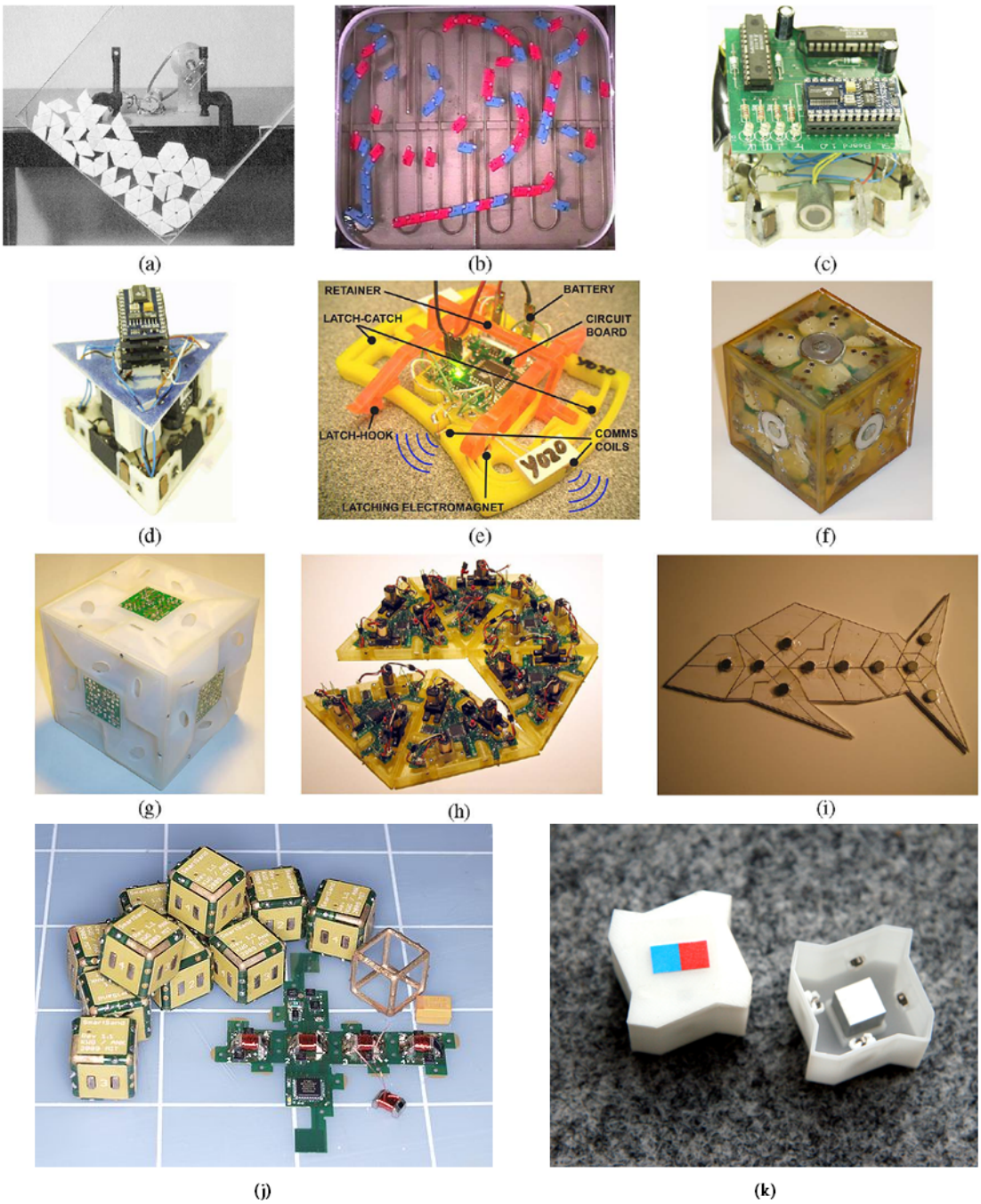


Figure 2.3 – Systems with externally propelled components, figure data is adapted from [11]: (a) Hosokawa et al.'s self-assembling hexagons [34]. (b) Breivik's template-replicating polymers [51]. (c), (d) White et al.'s self-assembling programmable modules. (e) Griffith et al.'s electromechanical assemblers [61]. (f) White et al.'s first system for self-assembly in 3D. (g) White et al.'s second system for self-assembly in 3D. (h) Programmable parts testbed [15]. (i) Bhalla and Bentley's self-assembling special purpose modules [50]. (j) Pebble robots [48]. (k) Mermoud et. al.'s passive Lily modules [17].

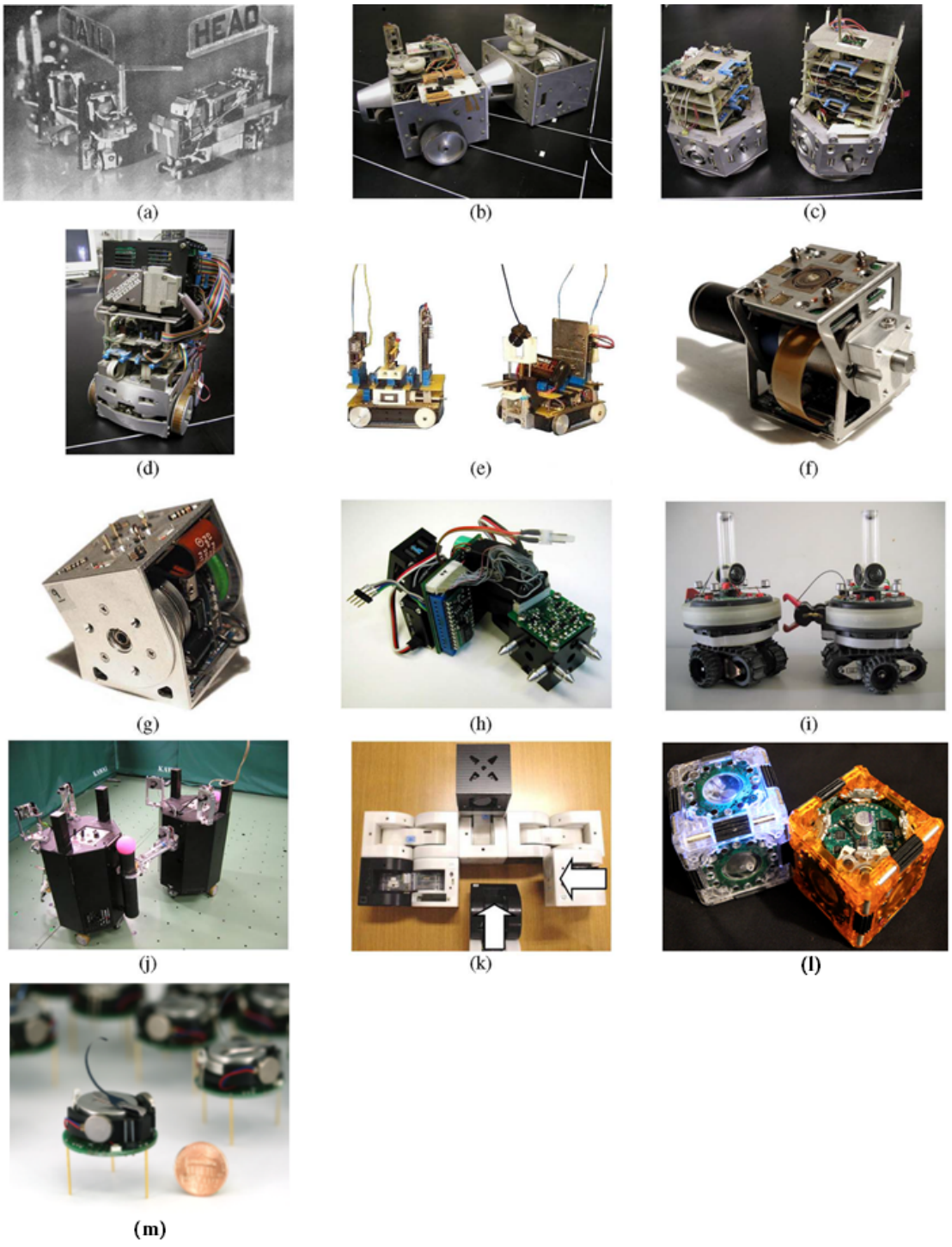


Figure 2.4 – Systems with self-propelled components, figure data is adapted from [11]: (a) RSD I [53]. (b) CEBOT Mark II [54]. (c) CEBOT Mark III [41]. (d) CEBOT Mark IV [55]. (e) Bererton and Khosla's system [56]. (f), (g) PolyBot G2 and PolyBot G3 [43]. (h) CONRO [42]. (i) Swarm-bot [40]. (j) Super Mechano Colony [59]. (k) M-TRAN III [44]. (l) 3D-Mblock [28]. (m) Kilobot robot [60].

2.3 Analysis Methods and Modeling

Probabilistic models were developed for various aggregation and self-assembly experiments [62]. Miyashita et al. proposed an interesting modeling and simulation framework that includes both microscopic simulations and macroscopic models for capturing the dynamics of Tribolon modules that perform stochastic self-assembly at the surface of water [63]. A comprehensive theoretical study of microscopic robot coordination in viscous fluids has been carried out by Hogg [64]. Kumar and colleagues have extensively investigated stochastic modeling and distributed control of swarms of robots [65]. Their approach originates from the study of chemical systems [66], in which randomness plays a key role. The chemical formalism is particularly suitable to the study of aggregation and self-assembly, as pointed out by Hosokawa et al. [34]. Hosokawa et al. analyzed the yield of desired products as well as the process dynamics (with six discrete components per entity) [34]. In the swarm-bot system [40], the analysis addressed the reliability and speed by which individual modules connect into single entities, as well as the additional capabilities and functions such processes may provide (with up to 16 discrete components per entity).

2.4 Synthesis Methods and Control

The process of self-assembly is governed by the modules' interactions with each other and by their spatially anisotropic binding preferences. In simpler systems with passive building blocks, modules have static binding preferences and are externally propelled. Examples include the systems of Hosokawa et al. [34], Bhalla and Bentley [50], and Mermoud et al. [17]. In these systems, the course of the assembly process is typically guided through actively controlled environment, while the space of achievable structures is limited by the binding preferences of the passive modules. In more complex systems with intelligent building blocks, a module's motion and/or binding preferences can depend on its state which may change following interactions with other modules and/or the environment. For instance, a module's state changes by mechanical interactions with other modules in the system of Penrose [49]. In the system of Breivik, a module's state can be affected by the temperature of the environment [51]. In the swarm-bot system, a module's state depends on the state of connectivity of its neighboring modules. In several systems, self-assembly starts from a seed which may be a single module or a modular entity, static or mobile, usually designated by the experimenter. The seed may also be autonomously chosen by the system as in the case of [58].

In several systems, modules execute a deterministic finite-state machine which may be embedded in the hardware or software. In the systems of Penrose [49], RSD I [53], and the system of Breivik [51] this logic is embedded in the hardware characteristics of the modules. For the case of intelligent building blocks the logic is embedded in the module's software. For instance for the case of PPT [15], each of the programmable modules executes a program which defines the module's binding preferences, specified by a graph grammar. In the case of the swarm-bot system, designing the behavioral controller is automated using evolutionary algorithms [40].

Summary

Over the course of the past 60 years, a variety of self-assembling systems have been designed with building blocks at the centimeter scale. These systems demonstrate several basic functionalities such as formation, growth, self-reconfiguration, self-repair, and template replication. In this chapter we highlighted the three major problems one has to address when developing engineered self-assembling systems, the fabrication problem concerned with mechatronic realization and operation of the system, the analysis problem concerned with modeling, predicting and evaluating the system performance, and the synthesis problem concerned with influencing the self-assembly process towards building specific desired target structures.

3 Scope of this Thesis

THIS chapter highlights where this thesis stands with respect to the state of the art and defines our specific contributions. It provides an outline of the thesis along with the main objectives of the research. Finally, we present our contributions in each of the main parts of the thesis and describe their link to our published record.

3.1 Objectives and Outline

The research efforts of this thesis can be basically divided into two main thrusts: (i) a **technological** research thrust concerned with the design, the fabrication, and the packaging of the experimental platform, and (ii) a **methodological** research thrust concerned with the development of distributed, stochastic control schemes supported by a flexible modeling framework.

The thesis is laid out in five parts:

Part I - Introduction In this part, we briefly introduce the concept of self-assembly in both natural and engineered systems. We explain why studying the science of self-assembly is of importance today and how investigating self-assembly in robotic systems may contribute to future technological developments in nano-technology.

Part II - Robotic Platform In this part, we present the mechatronic system design process of our robotic platform. The design challenges and system requirements as well as devised solutions are described. We explain the functionalities of the realized robotic platform and characterize the system features through several experiments.

Part III - Modeling Self-Assembly This part lays out our multi-level modeling approach in order to capture the dynamics of the self-assembling system at different model abstraction levels. We explain how the models are built based on a description of the embedded controllers defining the robotic modules' behaviors and as well as observations of the self-assembly process in the system.

Part IV - Controlling Self-Assembly This part focuses on how the self-assembly process may be guided in order for certain global patterns to form. We explain how appropriate embedded controllers can be synthesized and deployed on real robotic modules. In particular, we show how the self-assembly process in our robotic platform can be controlled mainly through a distributed fashion while global centralized feedback may also affect the course of the process.

Part V - Conclusion In this final part, we summarize the research conducted in this thesis and highlight the core contributions of the work. Considering the achievements and the challenges yet to overcome, we provide an outlook on possible research directions following up our work.

3.2 Contributions

The contributions of this thesis are along three main axes: design, modeling, and control of a programmable self-assembling robotic platform. Each of the following parts contributes along one of these axes to the state of the art.

Part II - Robotic Platform Our first contributions concern design and development of an experimental fluidic self-assembly robotic platform. Our system consists of two main components: 1) the floating Lily robotic modules, and 2) the controllable experimental setup built around them.

- B. Haghighat, E. Droz, and A. Martinoli, "Lily: a miniature floating robotic platform for programmable stochastic self-assembly", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1941–1948
- B. Haghighat and A. Martinoli, "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)*, 2016, pp. 2778–2783
- B. Haghighat, M. Mastrangeli, G. Mermoud, F. Schill, and A. Martinoli, "Fluid-mediated stochastic self-assembly at centimetric and sub-millimetric scales: design, modeling, and control", *Micromachines*, vol. 7, no. 8, p. 138, 2016

Part III - Modeling Self-Assembly The third contribution explores the development of models which capture the dynamics of the self-assembling system at different abstraction levels. In particular we develop models at three abstractions levels: (i) submicroscopic level, where the physics of the system is accurately captured in a realistic robotic simulator; (ii) microscopic level, where the physics of the problem is abstracted out into probabilistic interactions of the modules; (iii) macroscopic level, where we employ Markov modeling techniques to capture the evolution of the state of the system as a whole.

- B. Haghighat, R. Thandiackal, M. Mordig, and A. Martinoli, "Probabilistic modeling of programmable stochastic self-assembly of robotic modules", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4656–4663

Part IV - Controlling Self-Assembly The second main contribution is the synthesis of embedded ruleset controllers which allow the self-assembly process of the system to be guided towards a specific desired target structure in a distributed fashion. Additionally, we show how rules giving rise to a parallel assembly scheme achieving higher assembly rates can be synthesized for robotic modules.

- B. Haghighat and A. Martinoli, "Automatic synthesis of rulesets for programmable stochastic self-assembly of rotationally symmetric robotic modules", *Swarm Intelligence*, vol. 11, no. 3-4, pp. 243–270, 2017
- B. Haghighat, B. Platerrier, L. Waegeli, and A. Martinoli, "Synthesizing rulesets for programmable robotic self-assembly: a case study using floating miniaturized robots", in *International Conference on Swarm Intelligence (ANTS)*, vol. 9882 of LNCS, 2016, pp. 197–209
- B. Haghighat and A. Martinoli, "A rule synthesis algorithm for programmable stochastic self-assembly of robotic modules", in *Proceedings of the 13th Int. Symp. on Distributed Autonomous Robotic Systems (DARS)*, vol. 6, 2018, pp. 329–343

Summary

This thesis focuses on developing design, modeling, and control methods for programmable stochastic self-assembly in a swarm of robotic modules. It is driven by two main thrusts: (i) a technological research thrust concerned with the design, the fabrication, and the packaging of the experimental platform, and (ii) a methodological research thrust concerned with the development of distributed, stochastic control schemes supported by a flexible modeling framework. Part II contributes to the technological research thrust, while parts III and IV contribute to the methodological research thrust with each part building on top of the preceding part.

System Design Part II

4 Introduction

CHAPTER 3 revealed that this thesis comprises two main research thrusts: (i) a technological research thrust, and (ii) a methodological research thrust. This chapter describes the details of the technological research thrust in our work, in particular, the development of our experimental fluid-mediated self-assembly robotic platform built around a specifically designed water-floating robotic module, called Lily. We discuss the design choices and specific manufacturing details of the Lily robotic module as well as functionality validation results. Moreover, we explain the design and characterization of the full experimental setup through several basic self-assembly experiments.

4.1 Related Work

Inspired by the nature, self-assembling robotic systems typically consist of multiple simple and small robotic modules as building blocks. Simplicity in the building blocks' internal design is in favor of reducing manufacturing costs and increasing robustness as well as allowing for reduction of the blocks' size, ultimately resulting in finer resolution in the assembled structures. While obtaining low-power reliable latching and communication is generally challenging at small physical scales, taking advantage of the stochastic forces in the environment for locomotion allows for further simplification of the blocks' internal design, in particular in terms of power, computation, and actuation on-board resources.

The self-assembly robotic modules has been studied in numerous works, where a wide range of hardware implementations, including fully-autonomous robots and controllable environments, along with corresponding control approaches, have been developed. These works differ mainly in the capabilities of the robots, i.e., the self-assembly building blocks, the type of the environment and its level of controllability, and the approach employed to guide the self-assembly process towards the target. The cubic modules presented in [28] are capable of forming structures in three dimensions deploying magnetic latching and a lattice-based locomotion approach on a test table. Self-assembly of a swarm of autonomous floating robots in 2D has been studied in [73]. In [26], programmable self-assembly has been demonstrated

to be a powerful means for the formation of structured patterns in two dimensions in a large swarm of miniaturized robots, the Kilobots. While the motion of the Kilobots is inherently noisy, the primitive collective behavior programmed implements a deterministic and quasi-serial approach to shape formation. Taking advantage of the stochastic ambient dynamics for module transportation can, however, allow for the simplification of the internal design of the modules, as well as increased parallelization of the self-assembly process. 3D stochastic self-assembly of passive modules on an active substrate is investigated in [29]. Tribolon modules stochastically assemble into floating structures [63]. Tribolons are actuated using vibrating motors, and the environment is not capable of providing any control to guide the assembling process; the robots have a pantograph for both energy supply and control, including a latching mechanism based on the Peltier effect. The intelligent programmable parts in [15] are capable of local communication via infra-red and have controllable permanent magnet-based latches. The modules stochastically self-assemble on an air table, based on their internal behavior. The system of Pebble robots in [27] starts the process of shape formation with an ordered lattice; the stochastic forces in the environment are then used to detach unwanted blocks. Pebbles are only powered once they connect to the structure formed around the seed module; once connected and powered, they are capable of local communication among themselves. Abundant research has been dedicated to theoretical and experimental aspects of self-replication [74]. Self-replication of robotic units, for which self-assembled structure formation is crucial, is demonstrated using five 2D coding strings as templates [75] and also using self-swiveling microcontroller-based gripping blocks in [76]. A wide variety of platforms were developed in the context of modular robotics, some of which have the capabilities of autonomous locomotion and docking [23], [47], [73], [77]–[79].

To distinguish our contribution from the literature, we highlight that in almost all the above mentioned research activities on robotic self-assembly, no priority is given neither to real-time tracking of the modules' trajectories nor to related quantitative investigations of the influence of driving strategies and stirring conditions on the evolution, efficiency, and yield of the processes. Moreover, while in the case of Miyashita's triangular Tribolons the modules are set at the water-air interface similar to our Lily robotic modules, they are connected to external pantographs for energy supply, and they share the underlying "substrate" (i.e., electrolytes-loaded water) as second, shared electrode to perform propulsion or Peltier latching, and they are not endowed neither with on-board sensing nor computation; as an additional, relevant difference, their set-up does not allow to precisely control the underlying fluid flow apart from that deriving from the mechanical vibrations of the tank. Our modules are instead energy-autonomous, endowed with sensing, computation (microcontroller), and actuation (for latching/unlatching purposes), stirred by programmable fluid flows, and eligible to construct more articulated structures as compared to Tribolon's closed polygonal (hexagonal or triangular) ones. That is, our modules will allow investigations within the same platform both structural and informational aspects concerning aggregation and self-assembly, thereby combining both attention to graph-theoretic constructions (as studied mainly by Klavins' group) and focus on the effects of blocks morphology on assembly yield (as represented by Miyashita's work on Tribolons).

Our approach is also substantially different from the work in [79], where the fluidically stirred blocks explore a physical assembly space of higher dimensionality than that possible with our Lily robotic module. The growth of the assembled structures in [79] starts from a single fluidic sink located on the underlying active substrate; and only blocks physically attached to the structure rooted on the energy-providing substrate can be actuated, because electricity is transmitted upon contact among blocks. Furthermore, apart from the electrical actuation of internal valves, the modules feature neither on-board sensing nor computational capabilities.

4.2 Problem Statement

Our work considers the Lily robotic modules that, while compensating for their very limited autonomy of movement by scavenging momentum from their local fluidic environment, are still endowed with software controlled, intentional though primitive capabilities of interaction. The basis for our developments is the work in [17] on self-assembling passive Lily modules mentioned in Chapter 2. The Lily robotic modules' capabilities are based on active sensing, actuation, latching, and internal memory states able to encode rules for the building of predefined target structures. The Lily platform consequently embodies the minimalist tendency toward reduced fabrication costs, reduced energy budgets, and improved operating robustness of swarming robotic modules—as represented by the simplification of the blocks' functions concerning locomotion and interactions—along with the complementary tendency toward the purposeful, smart exploitation of environmental fields for leveraging interaction and aggregation activities. Such set of properties is coherent and very flexible as far as its defining parameters can be tuned and adapted to different scenarios. Particularly, it allows for a software design of behavioral rules of the robots, thereby extending the possibilities provided by physically hardwired interactions only. Additionally, the pose of the Lily robotic modules can be easily tracked using a standard overhead camera, and their internal states can be retrieved in real time using standard low-power wireless communication. Therefore, the Lily platform and its related experimental tools represent a setup of increased ease of handling and troubleshooting, ideally suited to investigate the general open issues posed by fluid-mediated self-assembly, especially given the substantial amount of features shared with similar systems at smaller length scales (e.g., external agitation, geometrical shape-matching and alignment, flotation at an interface).

One important argument in favor of our proposed approach to leverage centimeter-scale robots with the aim of allowing for investigating future sub-millimeter devices is the experimental ease of use and flexibility offered by our proposed robotic platform at this scale. The communication capabilities of the Lily allows one to probe the internal states of each robotic module in real time during the experiment. In the context of experiments with a large number of modules, this type of information is invaluable for both planning, analysis, and debugging purposes. Moreover, the larger size of Lily makes visual tracking, handling, and troubleshooting much easier than in the case of sub-millimeter-sized modules. Here below, we concretely list the design objectives considering the development of our experimental robotic platform.

- *Design Objective I: minimal design complexity, miniature size, centimeter scale.* At the centimeter scale, the Lily robotic modules endowed with minimal design complexity will be eminently representative of the self-assembly processes of passive MEMS building blocks. However, unlike the passive building blocks, we intend for the intelligent Lily robotic modules to be able to host and perform a considerable variety of tailored experiments without any hardware re-design, rather through exploiting their software programmability.

- *Design Objective II: software controlled primitive capabilities of interaction.* The robotic modules should be easily re-programmable, allowing for running different self-assembly experiments. The program on the robotic modules determines the outcome of the local interactions among them. These local interactions eventually result in a specific global structure to emerge. By programming the robotic modules with different software we may change the target structure emerging through self-assembly thus realizing programmable self-assembly.

- *Design Objective III: power autonomy, running on battery.* The robotic modules should be power autonomous by employing an on-board battery, allowing for the modules to take part in the assembly process at all times. This will also eliminate the need for a seed module. Employing on-board batteries is fairly affordable at centimeter scale and reduces the overall system design complexity compared to the case where the modules need to harvest energy from the environment.

- *Design Objective IV: low-power controllable latching.* The robotic modules need to employ a controllable latching mechanism in order to actively take part in forming or severing bindings according to the programmed software behavior. The controllable latching mechanism needs to be low power in order for the limited on-board battery capacity to allow for long enough assembly experiments.

- *Design Objective V: local communication.* The robotic modules need to have a means for communicating with their neighboring modules. This local communication enables the modules to make a mutual decision, based on their programmed software behavior, for forming or severing a binding with another module. Through these local decisions the self-assembly process will progress towards the desired target structure emerging at the global level.

- *Design Objective VI: global communication.* Existence of a global radio communication link between the robotic modules and a base station allows for collecting information about the experiments as the self-assembly process is in progress. Such link may also be exploited for sending commands or feedback to the robotic modules during the experiments. Additionally, one may perform functionality health check on the robots through this radio link as well as programming them.

- *Design Objective VII: minimalist sensing.* The robotic modules need to be equipped with minimal sensing means to (i) perceive if they have neighboring modules, and (ii)

perceive possible environmental clues guiding the self-assembly process.

- *Design Objective VIII: scalable operation, charging, and programming.* Our self-assembly experiments are intended to include tens of modules. With large numbers it can be tedious to turn on or off, charge, or program the robots individually. For these reasons, the robotic modules need to be designed such that they do not require to be handled individually for these operations.

- *Design Objective IX: non-self-locomoted water-floating modules.* The Lily robotic modules are intended to float on water and be driven around by the flow in the environment. This places certain constraints on the design. For the robots to be floating two key conditions need to be met: the payload should be below the buoyancy threshold and evenly distributed for a balanced flotation. Another crucial aspect of the design is proper sealing of the electronics inside while maintaining the ability to quickly dis-assemble, repair, and re-assemble the faulty modules.

- *Design Objective X: controllable environment.* The environment should be able to actively take part in the self-assembly process by adapting the global conditions such as the flow field moving the robotic modules around or communicating specific feedback to the robotic modules.

Considering the design objectives outlined above, we will explain the final design choices made in the following chapter. Here below, we briefly list the design choices made for the realization of the final experimental platform.

- *Design Choice I: minimal design complexity, miniature size, centimeter scale* The Lily robotic module has a cubic shell of 3 cm size. In order to simplify the design, no moving parts are included. The Lilies are not self-locomoted. Instead, they float on water and are driven around by the flow created in the environment. This ambient fluidic flow field drives the mixing in the system and provides for the various interactions among the robotic modules ultimately guiding the self-assembly.

- *Design Choice II: software controlled primitive capabilities of interaction.* The self-assembly of the Lily robotic modules is mainly controlled by the software embedded on the modules on-board 8-bit microcontroller in a distributed fashion. The software defines a rule-based behavioral controller where each rule determines if a binding between two Lily robots in a certain configuration should be formed or severed in order for the assembly process to progress towards the desired target structure. The robotic modules employ a wireless bootloader which allows for programming the robots with a new behavioral controller wirelessly.

- *Design Choice III: power autonomy, running on battery.* The robotic modules employ a small LiPo battery which sits at the bottom of the plastic shell.

- *Design Choice IV: low power controllable latching.* The robotic modules are endowed with four electro-permanent magnetic latches which can be placed in two controllable states: the on state where magnetic attraction is applied, or the off state where no magnetic attraction is applied. Unlike electromagnets which hold the assumed state only while consuming power, electro-permanent magnets require power only during the transient time when changing from one state to the other and consume zero power while holding a state. As a result, for low switching frequencies, electro-permanent magnets offer a very low-power controllable latching mechanism.

- *Design Choice V: local communication.* Once latched, the electro-permanent magnetic latches form an inductive communication channel. The robotic modules employ this channel to communicate to their neighbors.

- *Design Choice VI: global communication.* The robotic modules employ an integrated transceiver within the on-board microcontroller and a chip antenna for radio communication with a base station. This radio link is used for (i) wireless programming of the robots, and (ii) communicating with the robots during assembly experiments for data collection or providing specific commands.

- *Design Choice VII: minimalist sensing.* The robotic modules are equipped with a light sensor as a minimal means of sensing the environment. Additionally, via the local inductive communication channel formed through their electro-permanent magnetic latches they can find out if another robot exists in their neighborhood.

- *Design Choice VIII: scalable operation, charging, and programming.* The Lily robotic modules have been designed not to prevent individual handling for operations such as turning on or off, charging, or programming as well as starting or stopping an experiment. Multiple Lilies can be programmed with a new software through a radio link at the same time using the multicast mode, thus the overall programming time is independent of the swarm size. For charging a group of Lilies, they are all placed in a charging rack with powered rails connected to a power supply. The robotic modules have on-board charging chips which take care of a proper scheme for safely and efficiently charging the LiPo batteries. Lilies have a small push-button controller chip on board that eliminates the need for a physical switch for turning the modules on or off. In order to switch on the robotic modules, one may simply switch the power supply off and on once, providing a pulse of voltage which will switch on all the modules in the power rack.

- *Design Choice IX: non-self-locomoted water-floating modules.* The Lily's shell is a 3D-printed water-resistant structure of 1 mm thickness. It encloses the module and defines the available volume, determining the payload limit to be 35 g. Four small cavities are devised in the shell to allow for trimming and fine tuning weight distribution for a balanced flotation. The design includes a narrow gap of 1mm depth and width between the shell and the cap which is sealed using silicone paste. To open the module one only needs to cut the paste using a sharp blade and lift the cap.

- *Design Choice X: controllable environment.* While the self-assembly process towards a desired target structure is mainly guided by the programmed behavior of the Lily robotic modules in a fully distributed and thus scalable fashion using a rule-set description, the environment has the capability to actively take part in the process by adapting the flow field, changing the ambient luminosity pattern perceivable by the modules, or sending global feedback information to the modules over radio.

Summary

In this chapter we reviewed the related work in self-assembling robotic systems at the centimeter scale. In particular we focused on the mechatronic design of the robotic modules and the environment. Moreover, we explained our system design objectives and subsequent design choices. In the following chapters, we will carefully describe the development of the Lily robotic modules as well as the experimental fluidic self-assembly platform built around them.

5 Lily Robotic Module

As the first step towards developing our experimental platform, we focus on the building blocks which actually perform the self-assembly. In particular, we consider the mechatronic design and fabrication of the Lily robotic modules. Having inherited their external shell shape from the passive Lily modules firstly employed in a preceding research effort [17], Lily robotic modules employ low-power controllable Electro-Permanent Magnet (EPM) latches which they can disable to refuse certain interactions according to their embedded behavioral rule-set controller as well as the information about their neighbors acquired by communication through inductive EPM channels. Lilies can also communicate over a radio link to a base station to receive commands, new firmware, or to report specific information. Being powered by a small LiPo battery, Lilies can actively take part in the assembly process at all times allowing for a parallel assembly scheme where sub-structures of the target can be built separately, eventually joining to complete the overall target. Lily robotic modules are unique in the field in that they are the only power-autonomous fluid-mediated self-assembling building blocks capable of both wireless communication to a base station and local communication to neighbors through their custom-designed latches. Fig. 5.1 shows a Lily robotic module, a 35 mm roughly cubic-shaped power-autonomous unit. The choice of a latching mechanism was severely restricted by the small size of the robotic module, in particular a low-power mechanism was crucial. For the robots to be floating, two key conditions were necessary: the payload needed to be below a threshold and the weight needed to be evenly distributed. The following sections explains the design and development of the Lily robotic modules in detail.

As depicted in Fig. 5.2, each robotic module is composed of several components: a LiPo battery, four EPMs serving as physical latching and local communication channel, and a flexible circuit board. The flexible board is a two-layer design of total 260 μm thickness, on which a microcontroller unit with an integrated radio transceiver, an analog radio front-end, and a switching and power circuitry are placed. The battery is placed at the bottom of the shell. The flexible board with the four EPMs soldered on it is then folded and placed on top of the battery with the EPMs snapping in the sockets on the walls. A small frame is then placed in the

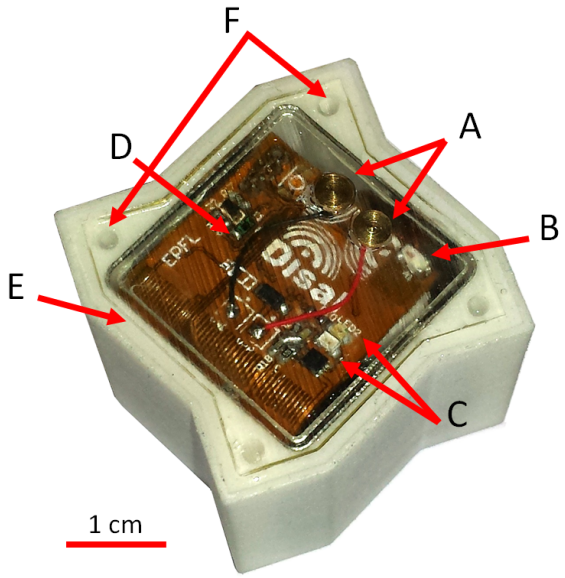


Figure 5.1 – Picture of a Lily robotic module. Some key features visible in the picture are: charging contacts (A), chip antenna (B), two LEDs signaling board status (C), ambient light sensor (D), sealing gap filled with silicone paste (E), and two of the four trimming holes (F).

middle to hold the EPMs in place. The gap between the transparent cap and the shell is sealed to protect the electronics inside from moisture. Table 5.1 summarizes the mass breakdown for a Lily robotic module.

5.1 External Shell

The Lily shell is a 3D-printed watertight structure of 1 mm thickness. It encloses the module and defines the available volume, determining the payload limit to be 35 g. The shells were ordered to the US-based 3D printing company Shapeways. The shells are made of a UV-cured Polymer which is also heat resistant up to 80 °C. This material is printed using the Multi-Jet

Table 5.1 – Mass break-down for a Lily robotic module.

Item	Mass
Shell	9.5 g
Cap	1.3 g
Battery	7.8 g
Populated board	3 g
4 x EPM	6.8 g
EPM frame	0.5 g
2 x Charging pin	0.2 g
Sealing paste	0.5 g
Total	29.6 g

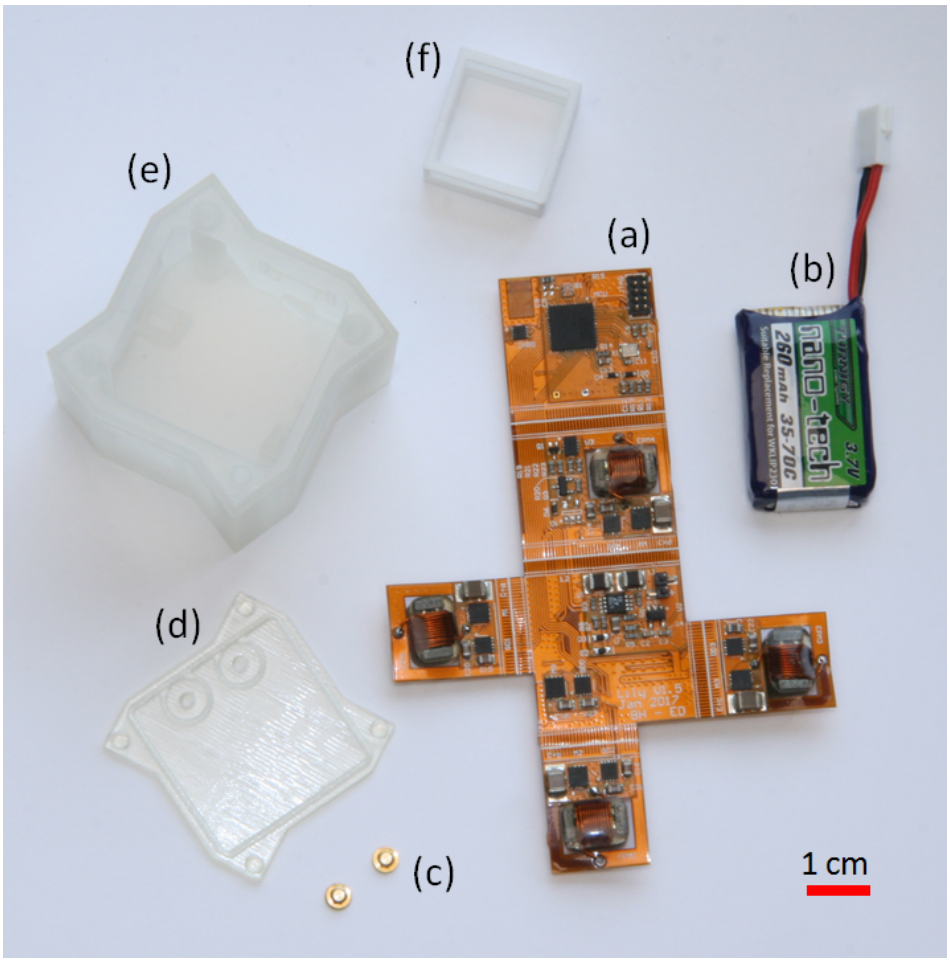
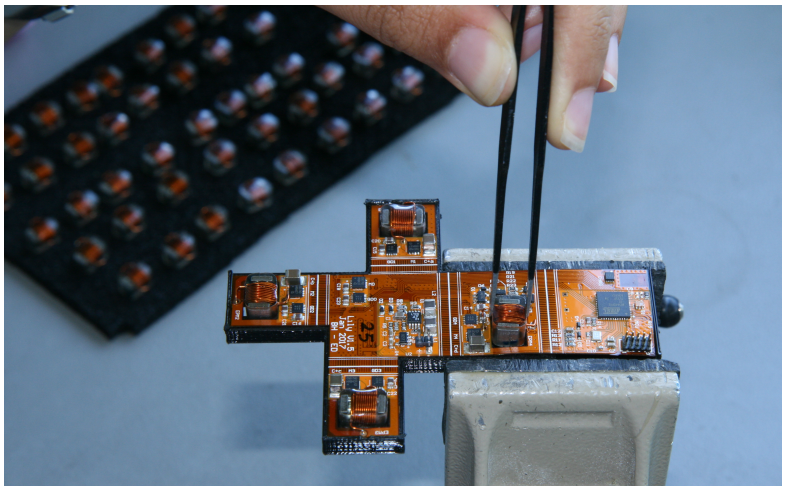


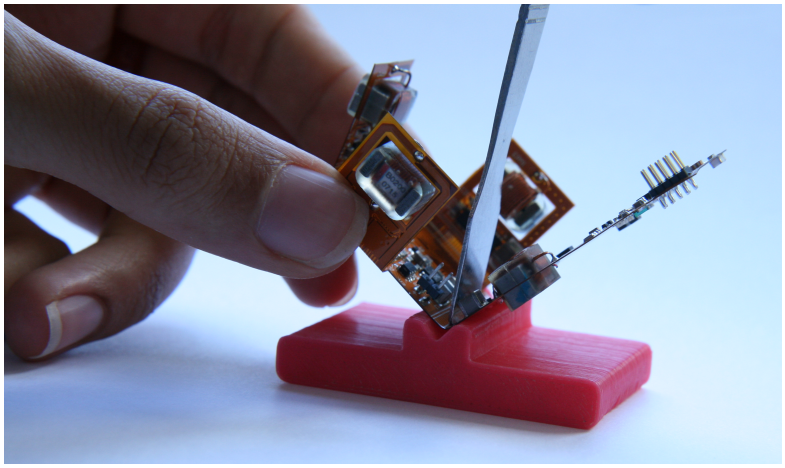
Figure 5.2 – Each Lily robotic module is composed of (a) a flexible circuit board with four EPM latches soldered on it, (b) a 240 mAh LiPo battery, (c) two brass charging pins protruding the transparent cap, (d) a 3D printed transparent cap, (e) a 3D printed shell, and (f) a 3D printed frame for holding the EPMs in place.

Modeling (MJM) process. Molten plastic is deposited onto an aluminum build platform in layers using several nozzles, essentially like a large print that sweeps across the build layer. As the heated material jets onto the build plate, it solidifies instantly. After each layer is deposited, it is cured, or polymerized, by a wide area UV lamp. The next layer is then applied, and through this repeated process, layers of thermoplastic build up into a model. Each layer has a thickness of 0.029 mm. When printing is finished, the models are removed from the tray and placed into an oven that melts away the wax support material. Next, they are placed into an a ultrasonic oil bath to remove any remaining wax residues, and then a ultrasonic water bath to remove any oil on the model.

The thickness of the shell wall is 1 mm everywhere except for the windows devised on each wall for the placement of the EPMs where the wall thickness is 0.3 mm. As a result, when two Lilies are in close contact, their neighboring EPMs are 0.6 mm apart. As depicted in Fig. 5.2,



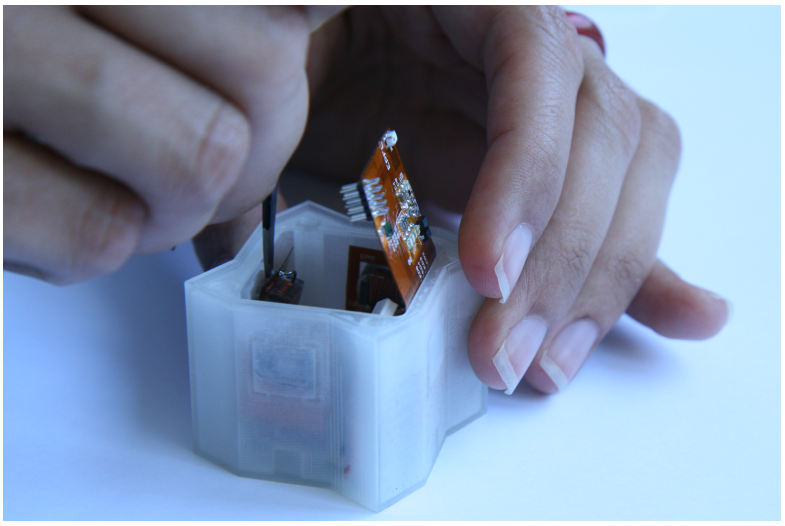
(a)



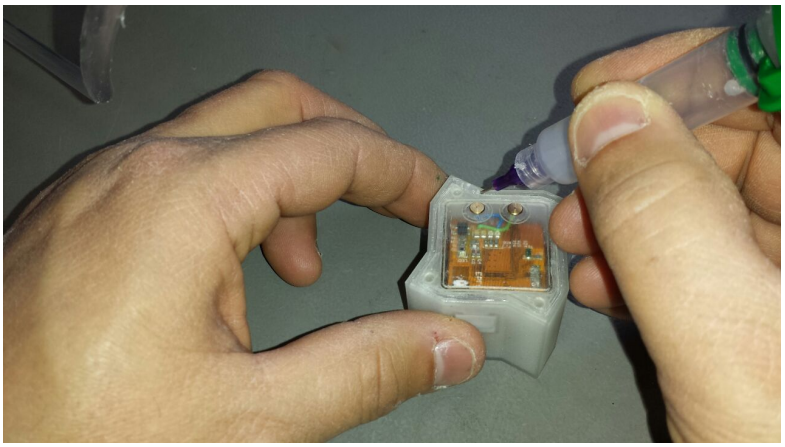
(b)

Figure 5.3 – (a) Placing the EPM latches on the flexible PCB lying on a 3D-printed support. The 3D-printed support allows for ensuring precise placement. The EPMs are then manually soldered on the PCB using a soldering iron. (b) After soldering the EPMs on the flexible PCB, the board is bent along 5 bending axes to fit inside the 3D-printed cubic shell. In order to bend the flexible PCB a small 3D-printed support piece is used to ensure uniform and repeatable bending and across several flexible boards and along all bending axes.

the shell has a specific rugged shape designed to prevent the latched units from easily slipping away due to high energy agitations in the fluid. Four small cavities are devised in the shell to allow for trimming and fine tuning weight distribution for a balanced flotation. For this, cylindrical brass pieces cut into a variety of predefined sizes were used as trimming weights. Just before closing and sealing the cap, the Lily would be checked for balanced flotation and an appropriate number of trimming pieces would be placed in the four trimming holes to balance the module. Right below the transparent cap, four LEDs signaling the board status as well as an ambient light sensor are placed. The cap is also a 3D printed model ordered



(a)



(b)

Figure 5.4 – (a) Placing the bent PCB inside the 3D-printed shell. The EPMs snap into the sockets devised on the inner walls. (b) Sealing the Lily by injecting Silicone paste in the sealing gap.

to the Shapeways company. The model is printed in acrylic plastic which is watertight and translucent. The material is also heatproof up to 48 °C.

A crucial aspect of the design was proper sealing of the electronics inside while maintaining the ability to quickly dis-assemble, repair, and re-assemble the device. The design includes a narrow gap of 1mm depth and width between the shell and the cap which is sealed using silicone paste (see also Figure 5.4(b)). The E41 Elastosil Silicone paste from Wacker was selected due to the softness of the cured paste which allows for easy and quick removal of the paste. To open the module one only needs to cut the paste using a sharp blade and lift the cap. It is easy to scrape off the dried paste and re-apply fresh silicone. Several tests conducted using humidity indicator paper slips inside the sealed shells submerged in water for several

hours proved the reliability of the design before experiments were attempted.

5.2 Printed Circuit Board

The Lily Printed Circuit Board (PCB) is a flexible design comprising two layers with a total thickness of 260 μm. Each copper layer is about 35 μm thick. The copper layer thickness and width of the tracks were critical design parameters due to the high current pulses required for reliable switching of the EPM latches. Compared to conventional rigid PCBs, flexible PCBs of the same size and layer count save up to 60% of weight. Additionally, the flexible PCB can be easily bent to conform with the Lily shell design and to be quickly placed inside the shell. While the first 20 boards were manually populated with components and soldered using a small infra-red oven, the final batch of boards were populated by the company fabricating the boards. The EPMs were later soldered on the boards using a 3D printed template for the boards to ensure precise placement as shown in Figure 5.3(a). If the EPMs are not well-aligned when soldered on the flexible PCB, the placement of the board inside the shell such that all four EPMs snap into the specific windows devised on the shell walls can be very difficult. After soldering the four EPMs, the board is bent around five bending lines using a small 3D-printed piece as shown in Figure 5.3(b). The bent board is then placed inside the shell on top of the battery sitting at the bottom and the module is closed and sealed as shown in Figure 5.4.

5.3 Electro-Permanent Magnetic Latches

Due to the high energy dynamics and stochastic nature of fluidic flows, in the realm of fluid-mediated self-assembly it is crucial that the latching faces can get aligned automatically. In order to meet a long-term energetic autonomy and the specific density requirements, a low-power and small-size latching mechanism was a necessity. We selected EPMs as they come with several advantages. These controllable magnets consume power only during the transient switching time. They are also efficiently down-scalable; while the required energy for switching is proportional to the latch volume, the force is proportional to its area [80]. The alignment of latching faces is also realized automatically through the interaction of the magnetic fields. Additionally, for low switching frequencies, electro-permanent magnets do not require coils with lower density than that of electro-magnets [80].

An EPM consists of two different types of permanent magnet rods, both having almost the same remnant magnetization but very different coercivities (see Figure 5.5 and Table 5.2). The rods are wrapped with a copper coil and have a small iron pole at each end for directing the magnetic flux. A high enough current pulse through the coil turns the latch “on" or “off" by setting the polarization of the soft magnet similar or opposite to that of the hard one, respectively. This high current peak is obtained by discharging a capacitor on the EPM coil. When the latch is “on", the magnetic flux reaches out and attracts magnetic materials; when it is “off" the iron poles provide a low resistance path for the flux to close within the two magnetic rods.

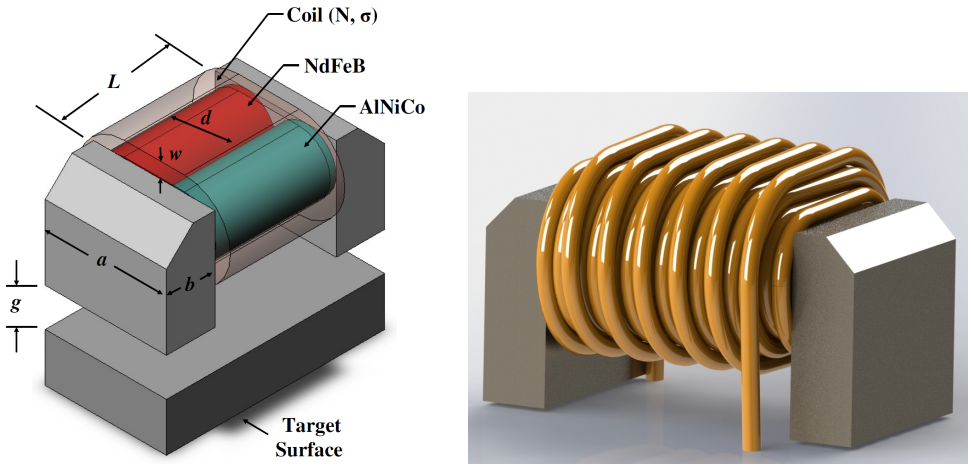


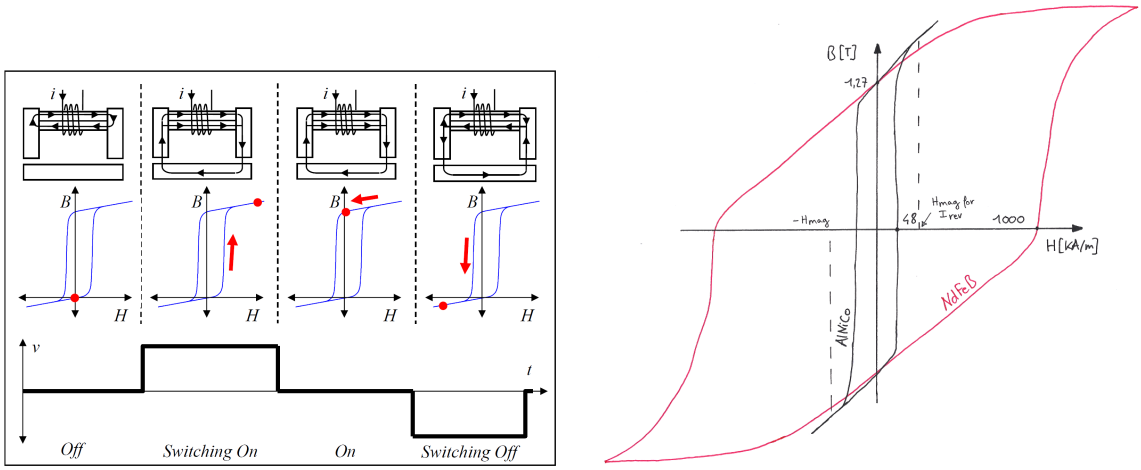
Figure 5.5 – EPM construction schematic [80] (a), CAD design of an EPM for the Lily robotic modules with three layers of winding (b).

5.3.1 Designing EPM Latches

In order to design customized EPM latches for our Lily robotic modules we started from the work described in [80]. We chose to use the same magnetic material, similar to the EPMs described in [80], we consider EPM devices made from the parallel set of a Neodymium-Iron-Boron (NIB or NdFeB) magnet rod with a very high coercivity, and an Aluminum-Nickel-Cobalt (Alnico) magnet rod with a relatively lower coercivity. Both magnetic material have roughly the same residual flux density as indicated in Table 5.2. Due to the high coercivity of the NIB magnet, the flux through it remains in the same direction under the operation conditions of the EPM device that we consider (see Figure 5.6). When the EPM device is in the off-state, the NIB and Alnico magnet rods are magnetized in opposite directions. Figure 5.6(a) shows the operation cycle of the EPM device. A positive current pulse through the EPM device coil results in a clockwise flux through the magnet and target surface, magnetizing the Alnico magnet towards right, turning the device to the on-state. A negative current pulse through the EPM device results in a counterclockwise flux through the magnet and the target surface, magnetizing the Alnico magnet towards left, turning the device to the off-state. Both magnet

Magnet types	Coercivity	Residual flux density
Grade N40 NdFeB	1000 kA/m	1.28 T
LNG40 Alnico	48 kA/m	1.26 T

Table 5.2 – Magnetic properties of the utilized magnet rods.



(a) **(b)**

Figure 5.6 – (a) The EPM operation scheme. In the off state, the two magnetic materials are oppositely polarized, so magnetic flux circulates inside the device, and there is no force on the target. In the on state, the two magnetic materials are polarized in the same direction, so magnetic flux travels outside the device and through the target, attracting it to the magnet. A current pulse in the coil of proper magnitude and sufficient duration switches the device between the on and off states, by switching the magnetization of only the Alnico magnet, which has a lower coercivity than the NIB magnet [80]. (b) Hysteresis curves of NdFeB and AlNiCo, the one of AlNiCo is much smaller. The coercitivity of NdFeB is 1000kA/m and the one for AlNiCo is 48kA/m.

rods see the same magnetic field created by the current through the device coil. However, as it can be seen in Figure 5.6(b), on the scale of the Alnico B/H curve, the NIB B/H curve appears as a line and is far from its saturation region because of the NIB magnet's much higher coercivity. As a result of having roughly equal residual flux density, at any of the two stable on-state or off-state of the EPM device the polarization of the NIB and Alnico magnet rods add up such they result in a residual flux density near zero on the lower part of the hysteresis loop but a positive residual flux density on the upper part of the hysteresis loop. A current pulse through the coil imposes a magnetic field H across the device, cycling it around the biased-up hysteresis loop shown in Figure 5.6(a).

In the following, we first summarize the mathematical descriptions, originally fully derived in [80], for several quantitative features of an EPM latch. We then use these descriptions for determining a set of appropriate parameters for the custom-design of the EPM deployed on Lilies and its corresponding switching circuitry using a simulation setup in Matlab. A list of parameters as well as their description and a reference to they show up is provided in table 5.3.

The magnetizing current, I_{mag}: Starting from the Ampere's law and considering the EPM schematic shown in Figure 5.5(a) one can follow a magnetic circuit through the magnet rods,

both air gaps and closing through the target surface, and write:

$$H_m L + 2H_g g = NI \tag{5.1}$$

We consider the ideal case where the pole-to-pole leakage flux $\phi_{leak} = 0$. Using Gauss's law for magnetic fields and the same magnetic circuit, we have:

$$\frac{\pi}{4} d^2 (B_{Alnico} + B_{NIB}) = (B_g ab) \tag{5.2}$$

For the NIB magnet, we consider a straight-line magnetization curve as in Figure 5.6(b).

$$B_{NIB} = B_r + \mu_0 H_m \tag{5.3}$$

Parameter	Description	Reference
B_{Alnico}	magnetic flux density in Alnico magnet	Eq. 5.2
B_{NIB}	magnetic flux density in NdFeB magnet	Eq. 5.2
B_r	residual magnetic flux density of NdFeB rod	Eq. 5.3
B_g	magnetic flux density in the air gap	Eq. 5.4
H_g	magnetic field intensity in the air gap	Eq. 5.4
H_m	magnetic field intensity in any of the magnet rods	Eq. 5.3
H_{mag}	required magnetic field intensity for Alnico to reach saturation	Eq. 5.6
B_{mag}	required magnetic flux density for Alnico to reach saturation	Eq. 5.6
I_{mag}	required current through the coil for Alnico to reach saturation	Eq. 5.6
μ_0	permeability of free space	Eq. 5.3
l_{wire}	coil wire length	Eq. 5.8
A_{wire}	coil wire cross section area	Eq. 5.9
L_{EPM}	inductance of the EPM device	Eq. 5.12
R_{EPM}	resistance of the EPM device	Eq. 5.10
N	number of turns in the coil winding	Eq. 5.1
L	length of magnet rods	Fig. 5.5
d	diameter of magnet rods	Fig. 5.5
a	width of iron pole	Fig. 5.5
b	thickness of iron pole	Fig. 5.5
g	width of air gap	Fig. 5.5
w	thickness of coil winding	Fig. 5.5

Table 5.3 – A summary of parameters concerning the formulations for an EPM device.

Through the air gap, the magnetic field and flux density are linearly related.

$$B_g = \mu_0 H_g \tag{5.4}$$

Unlike the air gap and due to the hysteresis effect, the Alnico magnet has a nonlinear B/H relationship, $B_{alnico}(H_m(t), t)$ as shown in 5.6(b). Combining the equations above, we have:

$$\frac{\pi}{4} d^2 (B_{alnico}(H_m(t), t) + B_r + \mu_0 H_m) = (\frac{\mu_0 ab}{2g})(NI(t) - H_m(t)L) \tag{5.5}$$

To switch the EPM device to the on-state, we will apply a pulse of voltage V to the coil for a time T, until the coil current rises to I_{mag}. The magnetic field intensity to saturate the Alnico is $H_m = H_{mag}$, and the associated magnetic flux density is $B_{alnico} = B_{mag}$. Substituting these values into Equation 5.5 and solving for I, we have:

$$I_{mag} = \frac{1}{N}(H_{mag}L + \frac{\pi d^2 (B_{mag} + B_r + \mu_0 H_{mag})}{4(\frac{\mu_0 ab}{2g})}) \tag{5.6}$$

The EPM device resistance, R_{EPM}: The resistance of the final EPM device is equal to the resistance of the unrolled wire.

$$R = \rho \frac{l_{wire}}{A_{wire}} \tag{5.7}$$

The length of the unrolled wire is N times the length of an average-length turn.

$$l_{wire} = N[2d + \pi(d + w)] \tag{5.8}$$

The wire cross section area as a function of the wire diameter d_w and assuming square wire packing in the winding is:

$$A_{wire} = \frac{\pi}{4} d_w^2 = \frac{\pi}{4} \frac{wL}{N} \tag{5.9}$$

Combining Equations 5.7, 5.8, and 5.9, we have the resistance of the EPM device:

$$R_{EPM} = \frac{4\rho N^2}{L}[1 + \frac{d}{w}(1 + \frac{2}{\pi})] \tag{5.10}$$

The EPM device inductance, L_{EPM}: We can approximate the EPM device behavior as an average constant inductance $L = \frac{\Delta\lambda}{\Delta I}$, in order to estimate the switching time T.

$$L_{EPM} = \frac{\Delta\lambda}{\Delta I} = \frac{N(B_{mag} + B_r + \mu_0 H_{mag})\frac{\pi}{4}d^2}{\Delta I} \tag{5.11}$$

$$L_{EPM} = \frac{N^2}{\frac{2g}{\mu_0 ab} + \frac{4H_{mag}L}{\pi d^2(B_{mag}+B_r+\mu_0 H_{mag})}} \tag{5.12}$$

The switching voltage: The voltage drop across the EPM device is the sum of the induced voltage and the voltage across the series resistance.

$$V(t) = N(\frac{dB_{alnico}}{dt} + \frac{dB_{NIB}}{dt})\frac{\pi d^2}{4} + I(t)R \tag{5.13}$$

From the above equation, we can see that higher voltage results in faster switching. We can also see that there is a minimum voltage, V_{min}, below which a current I_{max} will not grow after the transient time.

$$V_{min} = I_{mag}R = 4H_{mag}\rho N[1 + \frac{d}{w}(1 + \frac{2}{\pi})] \tag{5.14}$$

5.3.2 Building EPM Latches

Fig. 5.7 depicts the structure of a custom-sized EPM for the Lily robots. EPMs are placed horizontally inside the Lily shell on four walls, i.e. with their North and South poles pointing left and right (as opposed to pointing up and down) resulting in a 4-way symmetric design. The force between two meeting Lilies is then always non-repulsive, thus in favor of getting latched. It is shown that in scenarios where the configuration form or pattern matters, gender-less latches are favorable [81].

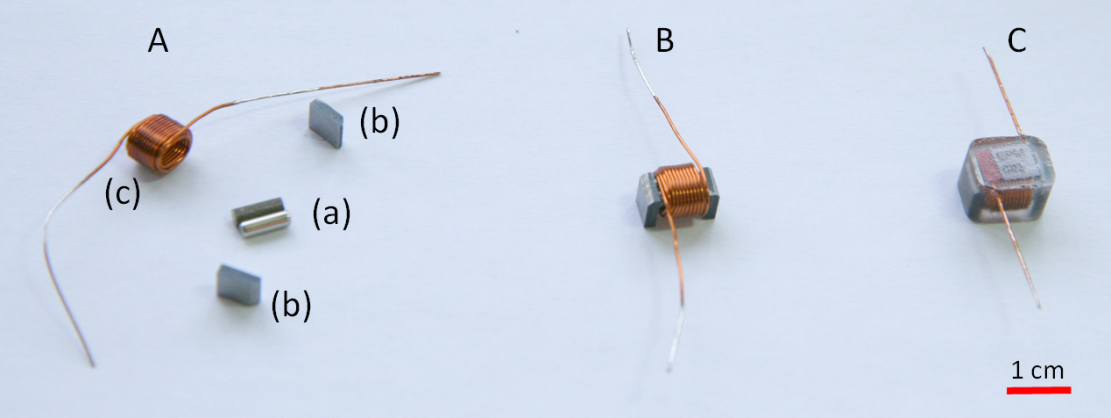


Figure 5.7 – A: An EPM is composed of two magnet rods (a), sandwiched between two iron pole pieces (b), and wrapped with 32 turns of grade 26 AWG wire (c). B: The pieces are held together using glue. C: The assembly is then put in a Polyurethan mold for protection against humidity and for increased sturdiness of the piece.

In order to build custom EPM devices for the Lily modules, we first determined the size of the magnet rods, i.e. the L and d parameters in Table 5.3, with the aim to achieve a certain attraction force. Using the equations derived above and by simulating an RLC circuit in Matlab, we then determined appropriate number of coil turns N as well as the capacitor size and supply voltage. The number of coil turns directly influences the overall coil inductance which in turn affects the choice of the capacitor used for producing the current pulse. The magnetic rods were chosen to be grade N40 NdFeB and LNG40 Alnico, both of 2 mm diameter and 6 mm length. The pole pieces were laser cut from an Iron sheet to a size of 5 mm by 4.5 mm by 1.5 mm. Simulating the equations governing the electrical characteristics of the EPM in MATLAB and using a systematic search in a coarsely discretized parameter space, the electrical parameters were determined: 32 turns for the coil, a required current peak of 22 A, a capacitor of 400 μF, and a voltage level of 12 V for charging the capacitor. The coils were made by wrapping 32 turns of thermal copper wire of #26 AWG around a mold. A current of 20 A was then passed through the coil for a few seconds to heat up the wire for the insulation to melt and the structure to be fixed. The magnet rods were glued to the pole pieces. We used a dedicated set-up for assembling the EPMs, allowing for a quick, precise and repeatable assembling procedure, resulting in EPM latches with almost identical characteristics.

Two types of tests were conducted to characterize the EPM latching strength, a pull test to determine the holding force and another test in which one EPM approached the other to sketch the force field. One EPM was always anchored to a 250 g block placed on a weighing scale with 0.1 g precision. For sketching the force field, the second EPM was mounted on a linear motion axis with a precision of 0.1 mm and approached the first one from a distance of 4 cm. Fig. 5.8 depicts the normal force for two different conditions, involving one or two latching pulses, averaged over 10 trials. It can be seen that the two curves are only slightly different. Therefore, it can be concluded that applying several pulses instead of one would not increase the latching strength considerably. For measuring the holding force, a pull test

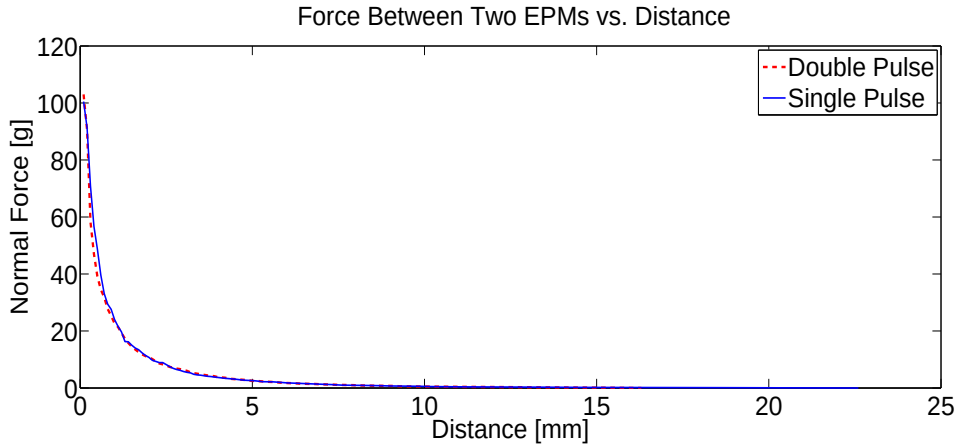


Figure 5.8 – Latching force between two EPMs as a function of the distance between them.

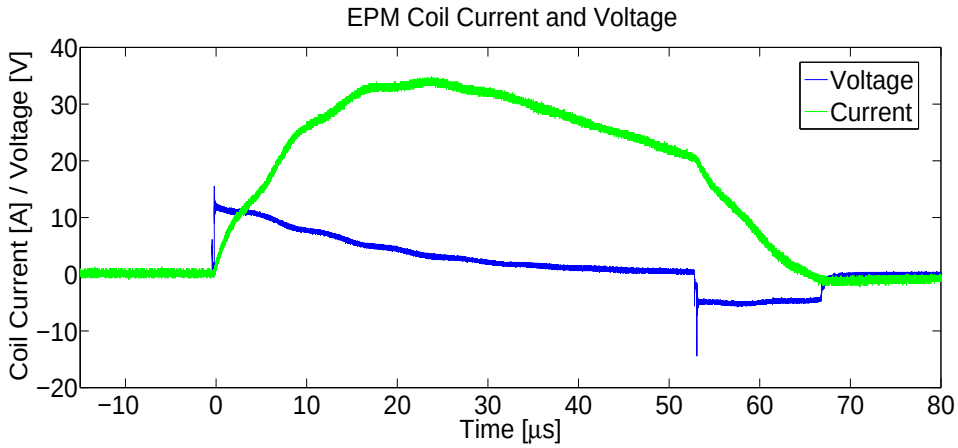


Figure 5.9 – Current through and voltage across an EPM coil during switching.

was performed in which the second magnet was fixed to one end of a spring of constant k = 6.7 N/m while the other end was pulled by the moving axis. The average holding force for the case where the EPMs were turned on using one switching pulse was 116 g. When the EPMs were turned on using two pulses the average holding force was 128 g. For the case where one EPM was turned on using one switching pulse and the other was off the average holding force was 32 g. The force between two magnets in the off state was measured to be 0 within the aforementioned precision.

Fig. 5.9 depicts the current through and the voltage across an EPM coil during a single switching pulse. As it can be seen the threshold current for switching the soft magnet is reached within 50 μs after which the current path is opened on the H-bridge. The voltage across the capacitor bank of 400 μF is depicted in Fig. 5.10. It can be seen that the back Electro-Motive Force (EMF) of the switching coil charges the capacitor back up by 0.7 V in less than 10 μs, the capacitor is then charged up to 12 V through the DC-DC converter in 17 ms.

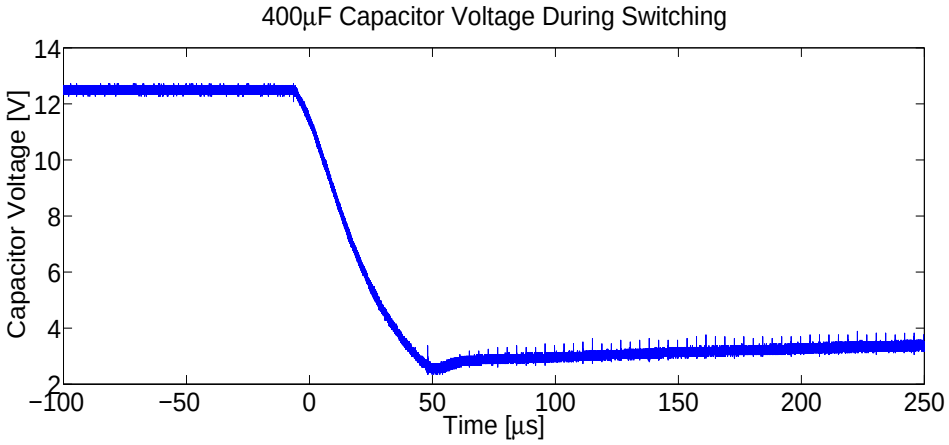


Figure 5.10 – Voltage across the 400 μF capacitor during switching.

5.3.3 EPM Switching Circuitry

Fig. 5.11 depicts the EPM switching circuitry. An H-bridge structure was necessary for driving current in the EPM coils in two directions. For the sake of saving space on the board, each EPM has one dedicated half-bridge while the four share one common half-bridge, similar to the design in [48]. The MOSFET switches are capable of handling a maximum pulsed current of 40 A, and are all N-Channel type, thus faster to switch, less bulky and less lossy than P-Channel types of similar current capacity. To make the switching fast and also to protect the microcontroller, gate driver ICs are used to turn the MOSFET switches on and off. We leveraged the intrinsic diodes in the MOSFETs, capable of standing a maximum forward current of 25 A, for passing the induced current to the capacitor while the switches are turning off. As shown in Fig. 5.10 this current charges the capacitor back up by 0.7 V.

5.3.4 EPM Communication Circuitry

The detection circuitry for EPM communication is shown in Fig. 5.11. The analog comparator is internal to the microcontroller. Due to the four EPMs sharing a common half-bridge and a single analog comparator, it is impossible for the controller to process messages on multiple EPM channels at the same time. The EPM channels are thus scanned sequentially. Similar to [48], to select a channel the dedicated side of the the H-bridge is grounded by turning on the low-side MOSFET, while the other end of the coil is connected to the positive input of the analog comparator through a high-pass filter with the DC level set to 1.5 V. The clipper diodes protect the input pins of the analog comparator against high voltage peaks. The negative input of the comparator is connected to a low-pass filtered square wave generated using the PWM timer channel of the microcontroller. This generates a DC level that can be adjusted for different detection sensitivities by modifying the duty cycle. If the processor does not detect an incoming message on a channel within 1ms, it will proceed to the next one.

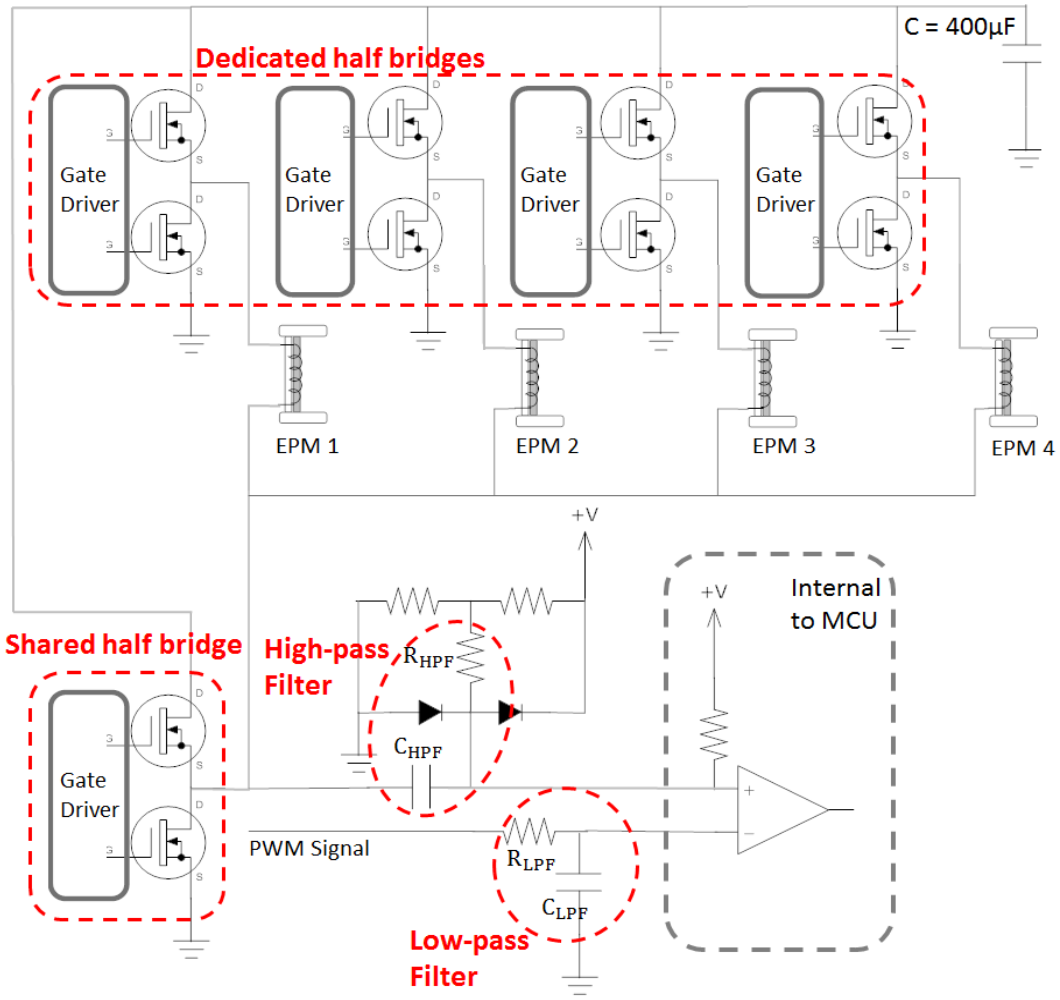


Figure 5.11 – The EPM switching and communication circuitry. In order to produce a current pulse in an EPM, the current path is closed through the dedicated half bridge and the shared one. Communication pulses are received on the positive input of the analog comparator, with the dedicated side grounded and the shared side floating.

5.4 Powering the Lily

Being endowed with an on-board battery, Lilies have a small push-button controller chip on board that eliminates the need for a physical switch for turning the robots on or off. A power-off input from the microcontroller to this chip allows for powering down the system. In order to turn off a swarm of Lilies the base station sends a multicast "turn off" command. Upon receiving the command, the Lily's microcontroller does some housekeeping operations and then activates the power-off input to the push-button controller which disconnects the battery from the rest of the circuit. In this state only the push-button controller is on, drawing a 6 μA current from the battery. Lilies can be safely stored in this state for more than two years on a single battery charge. In order to turn a Lily on, a short pulse of 5 V is applied on its charging input contacts. A swarm of Lilies can be turned on by lining them up and placing

the charger rail on top of them, the short pulse can then be applied to the charging rail. Upon turning on, Lilies will be running the bootloader code, waiting for a command to start the experiment.

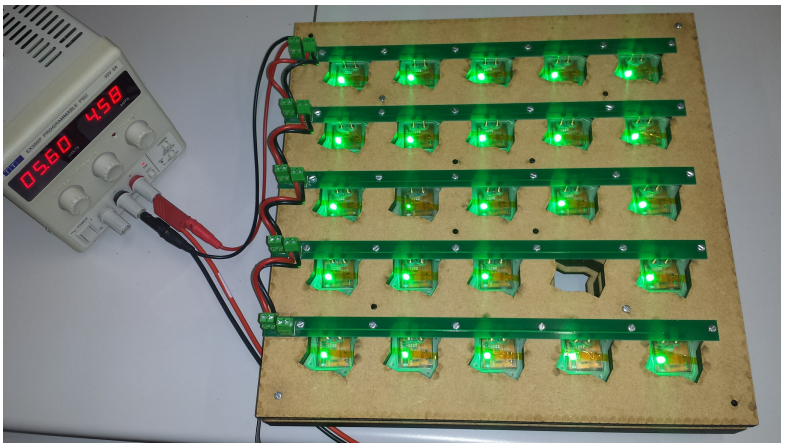
Table 5.4 shows the supply current drawn on different modes. As can be seen, power consumption when only the core is running and the radio is off, is considerably less than when the radio is on. It is thus efficient to keep the microcontroller in this mode most of the time, while the core needs to be running for EPM communications and switching, the radio can be turned on only when communication to the base station is necessary. When no EPM communication, switching, or radio communication task is pending the microcontroller enters the sleep mode. The microcontroller turns the radio on frequently and queries the base station for any commands that might have been issued and queued during the time the node was not listening. For a typical scenario with an estimated encountering rate of 2 per minute resulting in a communication over radio, the average current consumption will be 4.4 mA, resulting in 40 hours of power autonomy.

5.4.1 Power Circuitry

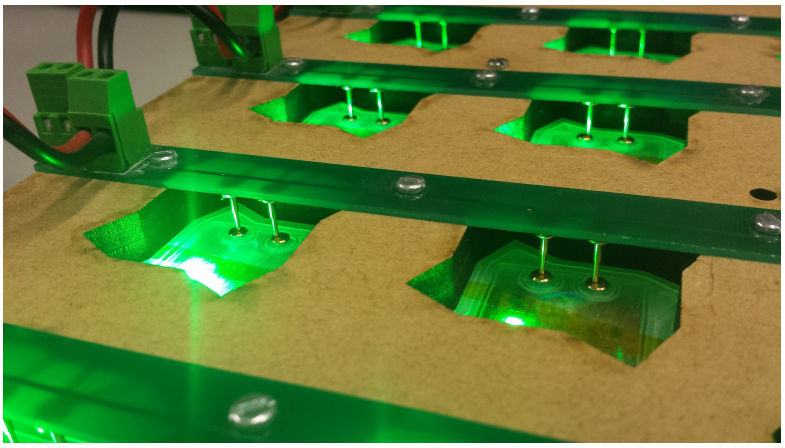
Lily robots are designed to be energetically autonomous deploying a 3.7 V 240 mAh Lithium-Polymer battery as shown in Fig. 5.2. Each module contains four surface-mount 100 μF ceramic capacitors charged through a DC-DC converter chip that raises the voltage level from 3.7 V to 12 V. Discharging these capacitors on an EPM coil for approximately 50 μs produces a current peak of around 30 A, well above the required amount of 20 A. The battery is protected against running on low voltage or high current. In addition, the controller monitors the voltage level and is capable of issuing a turn-off command. There is also a charger chip on the board. It is thus sufficient to connect the charging contacts (see Figure 5.2) to a power supply, and the charging scheme will be regulated automatically. By applying a short pulse on the same contacts, the robot can be turned on or off immediately.

Table 5.4 – Current consumption on different modes.

Item	Consumption
Sleep	700 μA
Core	4.1 mA
Radio Tx	18.6 mA
Radio Rx	16.6 mA
EPM Switching	30 A (50 μs)
EPM Tx	5 A (1 μs)



(a)



(b)

Figure 5.12 – Lily robotic modules in the charging station. The close-up shows the spring-loaded contact pins in touch with the Lilies' charging pins. The charger simply connects the Lilies to a power supply, while the LiPo batteries charging scheme is regulated by each module's on-board charger chip.

5.4.2 Charging

In order to charge the battery, the input contacts of the Lily's charger (see the pins in Figure 5.2) have to be connected to a 5.5 V power supply. These contacts are protected against wrong polarity and short circuit. The charger chip on the flexible board regulates the appropriate charging scheme for the LiPo battery. A green LED signals different charging states; it turns off when the battery is full. Deploying this charging chip eliminates the need for separate chargers. For charging a group of Lilies, the modules are placed in the charger base and a charging rail connected to a power supply is placed on top of them as shown in Figure 5.12. Each charger chip will draw the appropriate current to charge a battery and stops automatically when the battery is full.

5.5 Communication and Sensing

Lilies need to communicate with their neighbors to decide whether they should remain latched or unlatch in order to build-up the target structure collectively. They also need to communicate with the base station. For logging the assembly process, the nodes report the changes in their internal state back to the base station. The base is also capable of communicating individually or globally to the robots inside the arena to guide the assembly process or query a specific information, to make sure that the robots are running well. As another minimal means of sensing the environment, there is an ambient light sensor on board. Lilies can thus change their behavior according to the controllable light patterns generated through a dedicated overhead projector as in [82].

5.5.1 Radio Communication

Wireless communication between the base station and the Lily robotic modules is necessary for several purposes. Since the Lily modules need to be water resistant, the electronics is enclosed and sealed inside the plastic shell. While wired programming required the sealing to be opened to access a programming header, using an Over The Air Upgrade (OTAU) scheme only required a wireless communication link to the base station. Through wireless bootloading, the Lilies can be programmed easily without the need to open the sealing. The wireless communication is also used for logging the robotic modules' internal data during experiments. For instance, the robotic modules can send information about their internal state, battery level, or the measured luminosity level. This information can be later used to compare models with the ground truth experimental results and also to detect any faulty robotic modules during experiments. Lilies are also capable of receiving commands from the base station such as queries about specific information, or commands to change parameters in the modules' behavior.

For the wireless communication with the base station a command interface including 21 messages was designed. Table 5.5 lists the most important ones. When running the bootloader code the robotic module is constantly listening on the radio for incoming messages. In this state, being the default state after the module is turned on, the Lilies can reply to the base station's ping request, sending back their short address, and the version of the bootloader they are running. The wireless bootloader allows for programming the robotic module with a new application image. To allow for an update over radio, the program counter is redirected from the application code to the bootloader code upon receiving the corresponding command from the base. When running the application, the Lily's receiver is turned on for 100 ms every 1 second, after having sent a “command request" message to the base.

5.5.2 Inter-Robot Communication

Lilies communicate to their neighbors using the EPM latches. When two EPMs are in close contact, they couple magnetically through shared magnetic flux. Similar to a 1:1 isolation transformer, a current pulse through one of the coils induces a similar pulse on the second coil, the size of which is proportional to the size of the pulse on the first coil and also to the mutual inductance of the coupled EPM coils. This inductive communication channel is utilized to transfer data between robots. In order to avoid affecting the physical bonding quality, the communication pulses are sent in the same direction as the ones used to turn the latch on. The inter-robot communication takes place at 9600 bps using a series of 1 μs wide pulses. The data bits are encoded using at least one pulse of 1 μs length. Two such pulses less than 100 μs apart represent a logic 1 while the lack of a second pulse within the 100 μs time frame after a first pulse is considered as a logic 0. Compared to a synchronized approach where the high and low bits are encoded by the presence of a pulse within a time window, this scheme needs to send half a pulse more on average for transmitting the same information. However, the advantage is that there is no need for a synchronized communication clock between the two communicating units.
The detection circuitry for EPM communication is shown in Fig. 5.11. The analog comparator is internal to the microcontroller. Due to the four EPMs sharing a common half-bridge and a single analog comparator, it is impossible for the controller to process messages on multiple EPM channels at the same time. The EPM channels are thus scanned sequentially. Similar to [48], to select a channel the dedicated side of the the H-bridge is grounded by turning on the low-side MOSFET, while the other end of the coil is connected to the positive input of the analog comparator through a high-pass filter with the DC level set to 1.5 V. The clipper diodes protect the input pins of the analog comparator against high voltage peaks. The negative input of the comparator is connected to a low-pass filtered square wave generated using the PWM timer channel of the microcontroller. This generates a DC level that can be adjusted for different detection sensitivities by modifying the duty cycle.

As explained in Section 5.5.2, EPM channels are scanned sequentially for incoming messages. Independent from this scheme, the Lily robotic modules frequently send hello messages to actively discover their neighborhood. The exchange of hello message serves two purposes. First, it allows the robotic module to discover changes in its neighborhood by querying its neighbors to find out if a new module has been latched on a previously available face or if a previously latched neighbor has been detached due to high agitation in the environment. Second, the receiving Lily will use the handshaking scheme initiated with the hello message to detect the beginning of the neighboring transmitter data.

Hello messages are sent every 500 ms by default; a complete tour on all four EPM channels thus takes 2 seconds. The frequency of sending hello messages needs to correspond to the dynamics of the self-assembly process. In a highly stochastic environment where the rate of collision events is high, the Lilies need to check for neighborhood changes more frequently. While not latched to other robotic modules, a transmitting Lily receives no response to the

sent hello message. While latched to other modules, the query pulses from the transmitting Lily create an interrupt, waking up the receiver. Consisting of four bytes, the hello message is 4.2 ms long in total. If the hello message is correctly detected, the receiving Lily replies with an acknowledgment message (ACK) which consists of a single high bit, before the data communication starts. The Lilies will exchange their internal states and decide whether they should stay latched or to unlatch. Since all robotic modules are endowed with a given identical rule-set, their individual decisions will be the same. In order to unlatch, two pulses are sent to an EPM. The first pulse is in the direction that demagnetizes the EPM latch, while the second one, sent a few seconds after, magnetizes the EPM back to its default state.

5.6 Firmware

Several low-level routines have been implemented abstracting the low-level switching, communication, and sensing functionalities. The current firmware occupies 20 KB of the microcontroller's memory leaving more than 230 KB for the behavioral code.

Table 5.5 – Lily-base terminal commands. The wireless base station may issue the specified commands which will be received and processed accordingly by the addressed Lily module.

Command	Parameters	Description
Host commands		
crc	none	Get host data CRC
echo	<string to be echoed>	Echo a string
info	none	Information about myself
channel	<channel>	Set radio channel on host
?	none	Print commands
Comprehensible commands while Lily on bootloader		
ping	<short_addr>	Ping a node
finish	<short_addr>	Finish a node (force write)
feedhex	<short_addr> <line from hexfile>	Feed a line of hex file to a node
feedhexfile	<short_addr>	Feed hex file to a node (not used)
reset	none	Reset all nodes
exit	<short_addr>	Exit node to application
target	<F : Flash memory> or <E : EEPROM> or <L : Lock bits> or <X : No memory, dry run>	Set target memory
Comprehensible commands while Lily on application		
jbootl	<short_addr>	Jump to bootloader
unlatch_addr	<short_addr> <short_addr of lily to unlatch from>	Send unlatch command to node
unlatch_epm	<short_addr> <epm>	Send unlatch command to node
stoprx	<short_addr>	Stop Lilly from listening on radio
text	<short_addr> <message>	Send one line of text to node (radio echo)
sreset	<short_addr>	Soft reset application
poweroff	<short_addr>	Turn off node
battery	<short_addr>	Get battery voltage
light	<short_addr>	Get light measurement
pwm	<short_addr> <pwm> <true_delay> <valid_delay>	Set parameters

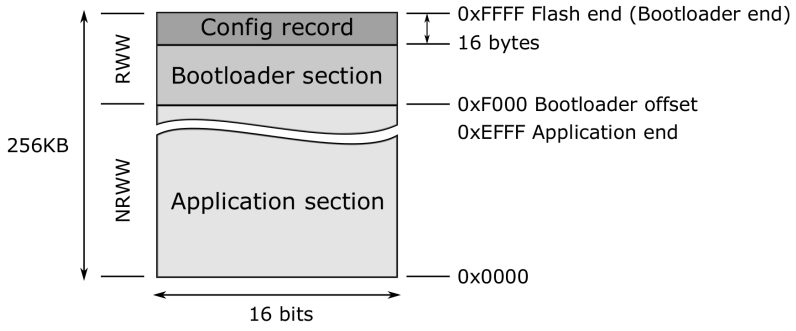


Figure 5.13 – Flash memory structure of the Lily's on-board microcontroller. The bootloader, residing in the Read While Write (RWW) section, receives and transfers new application data to the Non Read While Write (NRWW) section.

5.6.1 Wireless Programming

The flash memory division of the Lily module is shown in Figure 5.13. The bootloader, stored in the Read While Write (RWW) section, is capable of receiving and transferring data to the Non Read While Write (NRWW) section, where the application is stored. A config record that contains the information about the address of the module is appended to the bootloader section. While the size of the RWW and NRWW sections are fix, the bootloader offset can be changed to several different values if needed, this can be done via reprogramming the $BOOTSZ$ fuse bit. Figure 5.13 shows the actual configuration of the Lily flash memory. After initialization, the bootloader is constantly reading the radio receiving buffer and parsing the incoming frames with respect to the command set in the middle section of Table 5.5. It is clear that this symmetric radio communication is costly in terms of power consumption.

Summary

In this chapter, we describe in detail the design and development of hardware for our experimental self-assembly robotic platform built around the Lily robotic modules which serve as the self-assembly building blocks. In particular, different design choices for the Lily robotic module and the overall setup are explained and motivated. The functionality of the Lily robotic module as well as the full experimental setup is validated through several tests, assuring the achievement of the pre-established design objectives.

6 Experimental Setup

In this chapter we describe the setup built around the Lily robotic modules. Together with the Lily modules, this completes our experimental platform whose primary goal is to serve as a physical testbed for stochastic control strategies and corresponding modeling methods for programmable fluid-mediated self-assembly of robotic modules. Fig. 6.1 depicts the self-assembly arena. Forming a target structure by a swarm of Lilies involves several aspects. Given a target structure, an appropriate behavioral rule-set is deployed on all modules through wireless bootloading. The robotic modules' EPM latches are by default enabled, resulting in a default latching upon meeting another robot. Once two modules are latched, the EPM-to-EPM inductive communication channel is physically established. The blocks then exchange their internal states and according to their rule-set behavior, they will either decide to unlatch or remain latched and update their internal states accordingly. Each Lily then updates the base station with its new internal state over the radio. This information will then be part of the experimental logs and will also serve as, together with the trajectory tracks gathered through a dedicated overhead camera, the ground truth for validating models. In the case of an unfavorable interaction, the Lilies will both disable their involved EPM for a certain time during which the blocks will drift apart as a result of the agitation in the fluidic environment. In addition to event-based reporting of their internal state, Lilies periodically communicate to the base station to check for pending commands such as a query about the battery voltage level or the internal state, a command for pausing the experiment, or a command for turning the robot off. This scheme allows the robotic modules to spend most of their time in sleep mode or having the power hungry radio transceiver off, thus resulting in extended battery life. The commands from base station can be as well used to modify the robots' behavioral rule-set on the fly.

6.1 Setup Design

The experimental setup consists of a circular water-filled tank equipped with peripheral pumps, an overhead camera, an overhead projector, a wireless node communicating with

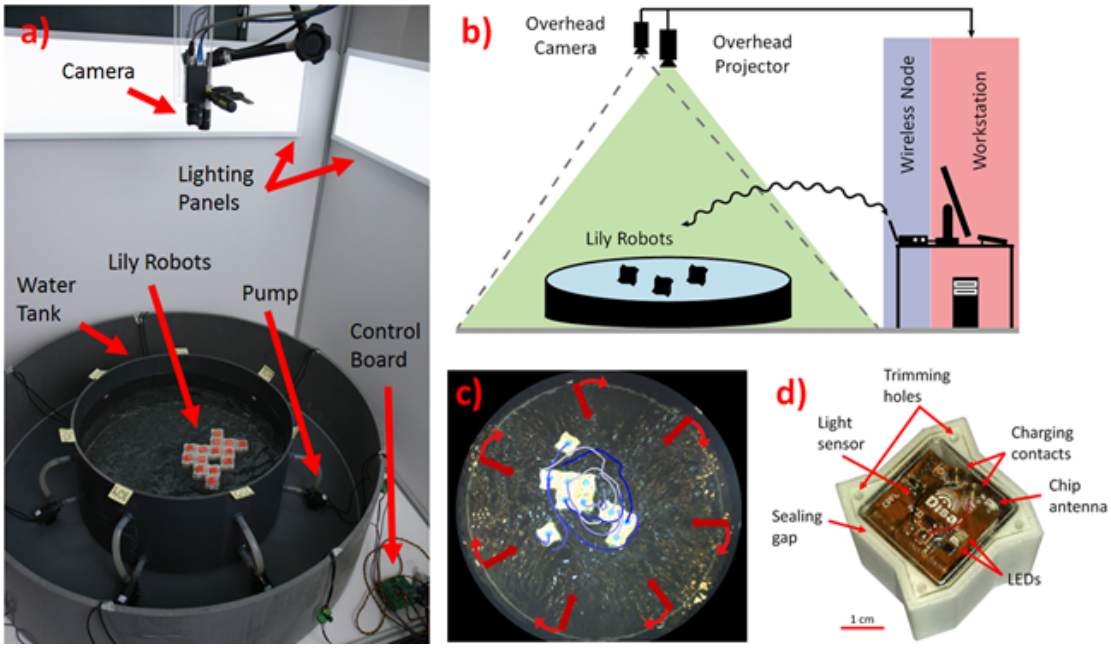


Figure 6.1 – Overall experimental setup. a) Image of the real experimental setup. b) Sketch of the experimental setup. c) Visual tracking of the Lily robots. The red arrows indicate the pump flow. The blue lines indicate a history of the trajectory of the robots. d) A Lily robotic module.

the robots, and a workstation. The Lily robots are not self-locomoted, they are instead stirred by the flow field produced by the pumps. The tank is approximately 0.6 m in diameter and 0.3 m in depth, and has seven inlets perpendicular to the wall which are endowed with a small insert piece to deviate the flow by about 15 degrees, creating a flow field with both radial and circular components. While the perpendicular flow components instigate irregular trajectories and induce collisions in the middle of the tank, they exhibit dead spots around the wall. The tangential components, however, generate a circular field, giving rise to regular closed trajectories which do not favor collisions but eliminate dead spots. To minimize any interference with the surface flow, the outlets are all placed at the bottom of the tank. Each pump's flow rate can be continuously controlled up to 9 l/min, allowing for a variety of flow fields and corresponding induced trajectories.

To monitor the evolution of the system, we use an overhead camera to track a passive marker located at the top of each robotic module using SwisTrack [83]. The positions of the markers are logged at a rate of 30 to 50 Hz. Complementary to the visual tracking data, is the data logged by the wireless node communicating with the robotic modules over radio. These data contain the evolution of the module internal states. The wireless node is also used to program the robots and send commands or feedback to them during the experiment, enabling the Lilies to adapt their behavior according to the feedback which is based on a global image of the self-assembly process rather than the modules' local perception.

The overhead projector allows for changing the ambient luminosity which is perceivable by

the robotic modules. Different ambient luminosity levels as well as patterns can be cast on the arena to define regions of different characteristics similar to [84]. As a result, while the information sent over radio can be used to adapt the behavior of uniquely identifiable robotic modules, the lighting system can be used to adapt the behavior of Lilies in specific regions of the arena.

6.2 Setup Characterization

In order to characterize the system and validate its functionalities, we conduct four main experiments. In the first experiment, the purpose is to assess the applicability of the standard chemical kinetics modeling to our system, and thus to show the relevance of our previous modeling efforts to the current experimental platform [17]. In the second experiment, we investigate the natural tendencies of the system to favor formation of certain structures depending on the latching forces and the flow field. The third experiment studies the dynamics of interactions among the modules as they decide to unlatch from one another. In the fourth experiment, we investigate adaptive formations based on a global signal perceived by the robotic modules.

All the experiments were conducted using 10 Lily robotic modules, and three main flow regimes were used. The flow regimes are referred to as high, medium, and low agitation modes hereafter, corresponding to 6.0 l/min, 4.4 l/min, and 2.6 l/min flow rate for all pumps, respectively. To gather statistics, each experiment was repeated 10 times. The error envelopes denote one standard deviation interval around the mean value.

Macroscopic Kinetics

Previous work has proposed a computational framework for automated modeling and centralized control of self-assembly [17]. The assumption of the framework was that the system was well-mixed and thus governed by reaction-diffusion dynamics, allowing us to apply canonical chemical kinetics models. Here we investigate the diffusion dynamics in the current system. For this experiment the robotic modules were programmed with an empty ruleset, thus immediately detaching upon binding events. Diffusing particles exhibit the Brownian motion. A characteristic of such particles is that they quickly forget their speeds [15]. Fig. 6.2 shows the speed autocorrelation. The speed of each Lily is essentially uncorrelated with its initial speed after approximately 5 s. Additionally, we can quantify the mixing in the system based on its diffusion. For the system to be well-mixed it is required that the modules diffuse faster than they react. This can be quantitatively stated as $D/k > A$, where $D = 0.0015\ m^2/s$ is the diffusion coefficient, $k = 0.01\ s^{-1}$ is the typical reaction rate measured experimentally in dimer formation experiments, and $A = 0.09\ m^2$ is the effective area in the system in presence of the high agitation mode measured experimentally through tracking the floating modules. The diffusion coefficient for a module is estimated as the expected value of $r^2(t)/4t$, where $r(t)$ is the displacement of the module extracted from the visual tracking data for any given t.

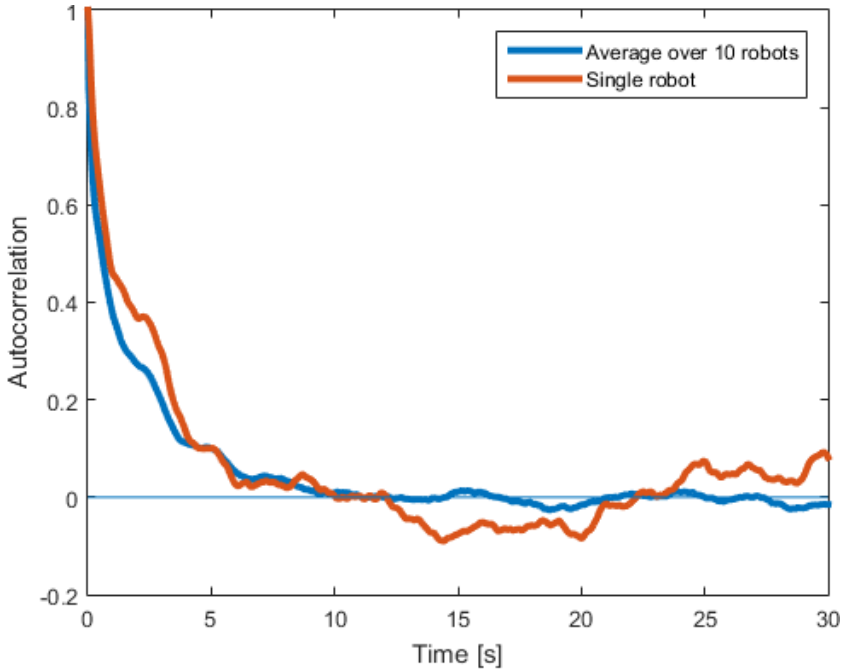


Figure 6.2 – Speed autocorrelation extracted from visual tracking of the robotic modules.

Natural Signature of the System

While the self-assembly process in our system is mainly guided by the programmed behavior of the Lily robotic modules, the stochastic fluidic field is leveraged for bringing the robotic modules into contact with one another. In initial experiments with the passive Lily modules, it was observed that each agitation mode in the system has its own signature distribution across the span of reachable assemblies. This was a result of the interplay of the magnetic latching force and the fluidic forces induced by the pumps' flow. The fact that the inter-module bonding was specifically designed to be reversible also allowed for a higher variability of observed assemblies for each agitation mode. In the current system with the Lily robotic modules, the inter-module bonds are designed to be more resilient, since the modules can always choose to turn their latches off if they wish to reject an interaction. In this experiment, we look at the growth rate of the size of the largest assembly in the system for the three different aforementioned agitation modes, under the two different states their EPMs can assume (on and off). While in the case where all of the robots have their EPMs on (see Figure 6.3), higher energy agitations only slightly accelerate the growth of the assembly, when all EPMs are off (see Figure 6.4), the final achievable size is strongly determined by the agitation mode. This shows the efficiency of the higher energy modes for taking the modules apart as a key factor in the unlatching mechanism. Furthermore, an interesting observation was made regarding the shape of the resulting structures when all robots have their EPM latches on: while for all studied agitation modes the system quickly ends up with one connected component, the resulting structures tend to have circular symmetry for the high agitation mode. On the other hand, chain-like structures are more common for the low agitation mode. This indicates that

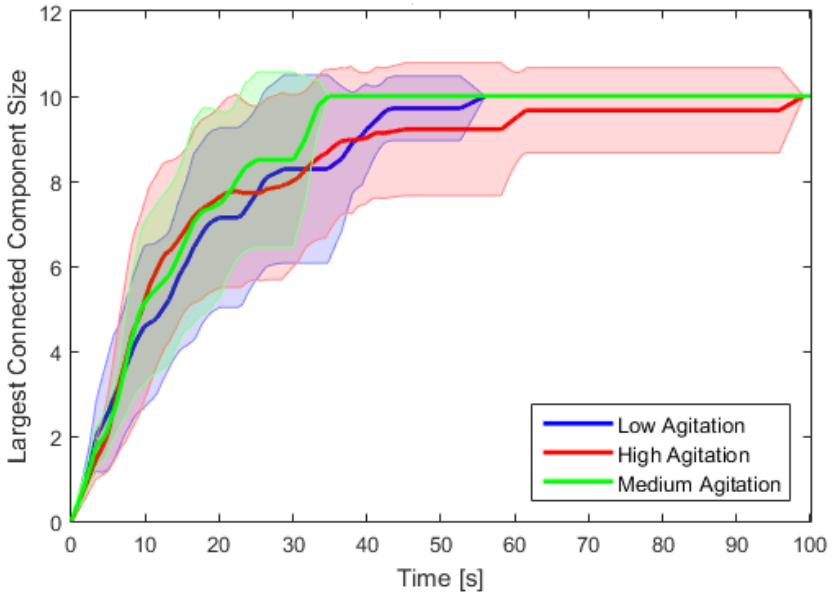


Figure 6.3 – Largest assembly size evolution, all robotic modules have their EPMs on.

utilizing the higher agitation mode facilitates forming compact structures while the lower agitation mode allows the robots to form structures which span wider. This finding highlights the crucial role of the agitation modes in guiding the self-assembly process. In other words, depending on whether the modules should be guided towards forming a compact structure or a widely spanning one, an appropriate agitation mode can be chosen similar to [17].

Unlatch Dynamics

Here, we studied how the coordination between two robotic modules turning their latched EPMs off affects dynamics of the unlatching interaction. The modules were programmed with an empty ruleset and two cases were studied. In the first case, the robotic modules performed their normal unlatch behavior, turning off their EPMs synchronously, while in the second case, only the module that initiated the communication turned off its EPM. As can be seen in Figure 6.5, the interaction time distribution tends towards significantly larger values for the second case, suggesting that the robots have more difficulty to detach. These results highlight the crucial role of the handshaking process between the two neighboring robotic modules involved in an unlatched dynamic. The handshaking process is crucial to reliably ensure that the modules are mutually aware of the decision and timing for turning their EPMs off.

Adaptive Formation

For these experiments, the robotic modules were programmed with appropriate rulesets derived using a dedicated framework discussed in Part IV. In addition to the static rulesets, the robotic modules constantly read the light signal values or radio messages, based on which

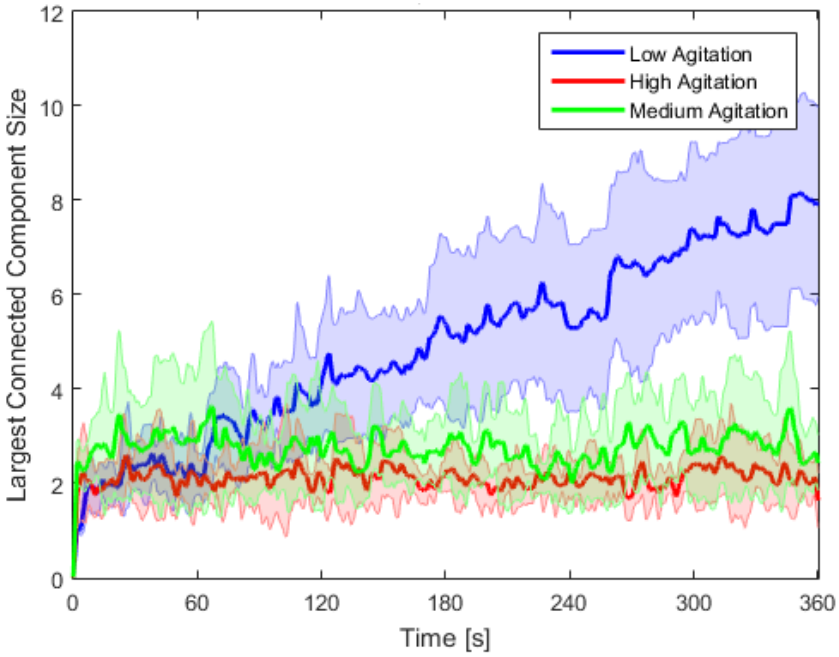


Figure 6.4 – Largest assembly size evolution, all robotic modules have their EPMs off.

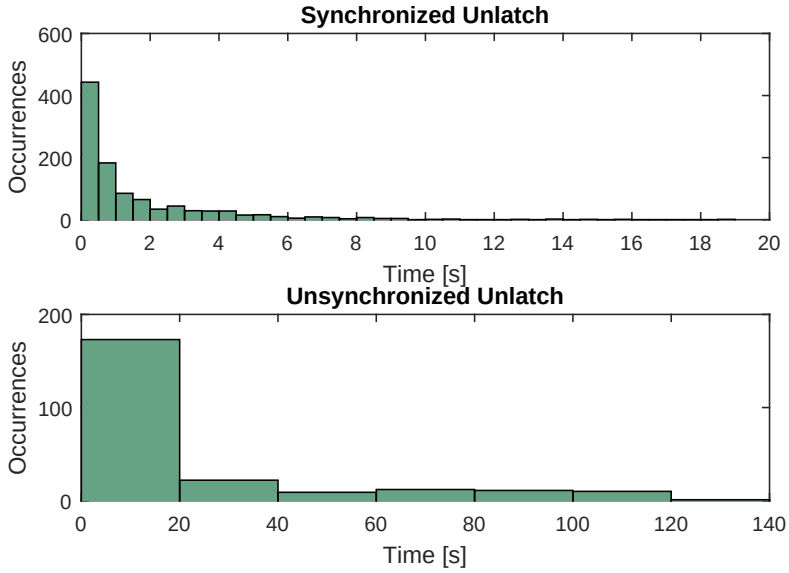


Figure 6.5 – Interaction time distribution: synchronized unlatching can be seen to allow for more efficient breaking of a bond.

they choose to pick certain rules among the available ones in their rule set. The fluidic arena is lit up using the overhead projector . The Lily robots perceive the green light using their on board light sensor. When the light is off, the environmental condition is considered to be undesirable for assembly. When the light is on, the condition is in favor of dipole formation. The light signal changes periodically, with each experiment having a different duty cycle of the

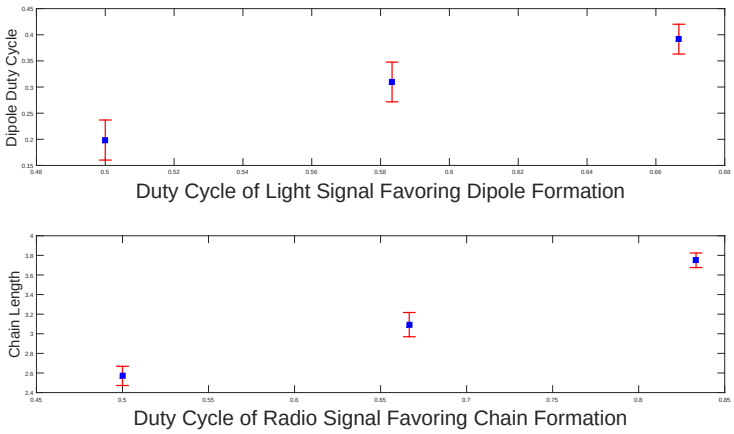


Figure 6.6 – Interaction time distribution: dynamic behavior changes upon perceiving environmental luminosity.

light being on. In a real-world application, this scenario may be associated with a medication delivery application where the particles enclose some medication while forming a dipole and release the substance only when they are located in the target region defined by a certain desirable condition. Additionally, in a similar scenario, the Lily robots periodically receive a message over the radio signaling them to switch their behavior from dipole formation to the formation of chains of unrestricted lengths and the other way around. We change the duty cycle of the message and measure the average length of assemblies in the system. Figure 6.6 depicts the dipole life time versus the duty cycle of the light signal, as well as the average length of the chains versus the radio signal duty cycle. In Figure 6.6, on the top, it can be seen that the time that the robots spend in a dipole formation increases accordingly with the duty cycle of the light signal, meaning that the system successfully detects the environmental condition and behaves accordingly. Figure 6.6, at the bottom, shows that the average length of the chains increases accordingly with the signal duty cycle, meaning that the system successfully behaves according to the feedback from the central node.

Summary

In this chapter, we describe the experimental setup built around the Lily robotic modules. Through a series of basic self-assembly experiments, we characterize the functionality of our experimental setup. Additionally, the conducted experiments validate and the achievement of the pre-established design objectives outlined in the Chapter 4.

7 Conclusion

In Part II, we have presented the first main contribution of this thesis which concerns the mechatronic design of the experimental self-assembly robotic platform. We first motivated the design objectives and then outlined the final design choices made in the process of developing the hardware. Moreover, we presented the validation and characterization results which verify the achievement of our pre-established design objectives.

We consider the following mechatronic design recipes to be the transferable methods explored in this part of the thesis:

- *Controllable environment, resource-constrained modules* - Employing a controllable environment allows for transferring some of the functionalities of the robotic modules that are required for guiding the self-assembly process to the environment. This in turn allows for further reduction of the complexity of the robotic modules' mechatronic design, ultimately fulfilling severe volume and weight constraints imposed by the targeted application area. An example of such design choice in our setup is the mobility of the robotic modules. The Lily robotic modules are not self-locomoted, instead they are driven around by the flow in the environment.

- *Low power, down-scalable latching, merging communication and latching mechanism* - The choice of the EPMs as the latching mechanism offers several advantages. In addition to being low-power due to a power consumption happening only during the latching state transitions, EPMs enable an inductive local communication channel, allowing for merging the communication and latching mechanism and thus eliminating the need for additional hardware. This design choice favors further miniaturization of the robotic modules. Particularly, an EPM latch can be efficiently down-scaled as the energy required to switch the latching state is proportional to the latch volume (i.e., l^3, with l being the characteristic length), while the obtained bonding force is proportional to the surface area of the latch (i.e., l^2).

- *Resource-constrained, down-scalable modules* - A design that natively takes into account

the resource constraints of the building blocks allows for efficiently transferring similar design ideas as well as core control concepts to smaller scales. For instance in our setup, the Lily robotic modules may be directly redesigned to fit in about a one-cubic-cm volume, provided that we relax the full energetic autonomy requirement based on an on-board battery. This will in turn trigger the use of the first point mentioned above for a design of the surrounding environment that provides an energy harvesting scheme to power the robotic modules. Such energy harvesting scheme could be for instance realized through endowing the modules floating on water with two power poles, one extruding the top side and connected to a pantograph and one submerged in water.

- *Scalable operation for large swarms* - Working with large swarms of robotic modules can be easily troubled by the number of individual operations required to be undertaken for each of the robotic modules. In our system design, all operations have been devised to scale well with the size of the swarm. Such operations include the broadcast wireless programming, the charging, the switching on/off of the robotic modules for starting/stopping an experiment, and the communication with the robotic modules during an experiment.

Summary

This chapter concludes the mechatronic design part of the thesis. The main outcome of the research and development carried out in this part is the realization of an experimental platform built around the Lily robotic modules. Our overall experimental setup is unique in its utilization of a fluidic environment for programmable self-assembly of robotic modules and allowing for the exploration of control strategies ranging from fully centralized to fully distributed. In this concluding chapter, we provided a summary of the contributions of Part II and highlighted the core methods and techniques which we believe one one may apply to the design of similar systems.

Modeling Self-Assembly Part III

8 Introduction

A key component in studying programmable stochastic self-assembling systems is developing models that accurately describe the assembly process dynamics. Such models could help with: (i) accurately predicting the performances (assembly rate and yield) of the self-assembling system, and (ii) evaluating as well as optimizing control strategies, whether distributed (e.g., ruleset controllers programmed on the modules) or centralized (e.g., modulating environmental features such as mixing forces deriving random interactions among modules), based on model predictions [17], [65], [85].

8.1 Related Work

One core problem in designing programmable self-assemblying systems is understanding the effect of individual robot characteristics, both in terms of hardware as well as software features, on the collective behavior [62]. Low level embodied simulators have been utilized in an attempt to faithfully recreate the environment, the robots' noisy actuation and sensing, and the interactions between the robots and the environment [86]. Abstract probabilistic models (see [62] for a general overview) have been developed for various aggregation and self-assembly experiments using mobile robots [87], [15], [88]. The choice of employing probabilistic modeling techniques for such systems is essentially motivated by the randomness lying at the core of these systems: random motion of the modules in the environment, explicit random decisions made by the modules' embedded controller, and random interactions among the modules [89]. Additionally, probabilistic models can be employed to provide a high-level macrostate description of the system state at each point in time by abstracting away low-level physical details of the system state such as positions, velocities, and internal states of all modules (i.e. the microstate description). A general methodology for developing accurate probabilistic models of the dynamics of programmable self-assembling systems is sought after to date.

8.2 Problem Statement

From a modeling standpoint, our focus will be on developing models at different abstraction levels, i.e. submicroscopic, microscopic, and macroscopic levels. Each model will be implemented using dedicated simulation tools which run the models to produce prediction results. We then employ these simulation tools to assess the performance of different control approaches for several case studies on our self-assembling robotic system. Our hope is that our efforts in this part will form the basis for a methodological framework that could be applied to a large variety of future self-assembling systems.

The motivation for employing multi-level models for self-assembling robotic systems is the great variety of time and length scales and possible design choices exhibited by such systems, which prevents single-level models to probe the dynamics of the whole system and explicitly represent all possible design choices. First, one needs very detailed models that capture low-level features of the individual modules such as body geometry, pose, and placement of various inter-module components. High-fidelity simulations using rigid-body physics engines belong to this category. However, such simulators can be computationally very expensive, and their computational cost grows at least linearly with the number of modules in the distributed system. Second, one could be interested in models that can yield accurate numerical results (in particular preserving exact quantities of building blocks) at a lower computational cost, in order to carry out systematic exploration of the design space, especially in terms of control. Agent-based models belong to this microscopic modeling level. Finally, at the highest level of abstraction, macroscopic probabilistic models allow one to obtain faithful predictions of collective metrics, investigate, possibly formally, macroscopic properties (e.g., the size, type and distribution of resulting populations of aggregates, stability, state reachability), and offer the best computational performance, enabling the simulation of arbitrarily large swarms of robotic modules. Below, we highlight the features and functionality of each modeling level in our studies. In particular, we shall emphasize that in our multi-level modeling approach the middle level, i.e. the microscopic level, serves a special purpose of verification of the synthesized ruleset behavioral controllers, assuring that the target structure is reachable provided that the ruleset's strategy is executed reliably. For this reason, we do not consider calibrating the microscopic model level with lower implementation levels. The submicroscopic model is calibrated based on the observations of the real system, while the macroscopic model is calibrated based on the observations of the submicroscopic simulated system.

The **submicroscopic model** provides a realistic replication of our real experimental setup, faithfully capturing the physics of the fluid-mediated self-assembly process in the system. Moreover, discrete intra-module details such as the EPM latches, EPM communication, radio communication, and light sensor are also faithfully reproduced 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 in our system 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 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.

The purpose of the **microscopic model** designed in this work 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 and an explicit representation of the spatiality in the system. This is particularly interesting considering that the rule synthesis algorithms, as detailed in Part IV, are agnostic about the spatial aspects of the system. More specifically, given a target structure, the relevant metric when comparing rulesets synthesized by rule synthesis algorithms is the number of concurrent steps in their assembly strategy as defined in Chapter 10. In reality, the realization of the conditions under which each assembly 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 excluded in the microscopic simulation framework.

The **macroscopic model** provides a high-level description of the system's state at each point in time by aggregating multiple building blocks together and capturing low-level physical details using probabilistic parameters. We focus on creating a general approach for developing a discrete-state Hidden Markov Model (HMM) of programmable stochastic self-assembly directly obtained from (1) a description of the robotic modules' embedded ruleset controller, and (2) an estimation of the rate constants defining the formation rates of different assemblies. We will show that assuming that the system is well-mixed and starting from different Markov models, the hidden states augmented through the automatic HMM refinement method improve the model prediction accuracy, compensating the imprecise modeling assumptions.

Summary

In this chapter, we reviewed the related work on developing models for self-assembling systems. In particular, we explained our objective and motivations for developing models at multiple abstraction levels, capturing the dynamics of the system at different computational costs. Moreover, we explained the features and functionalities of each modeling level in the context of the studies conducted in this dissertation.

9 Submicroscopic Model

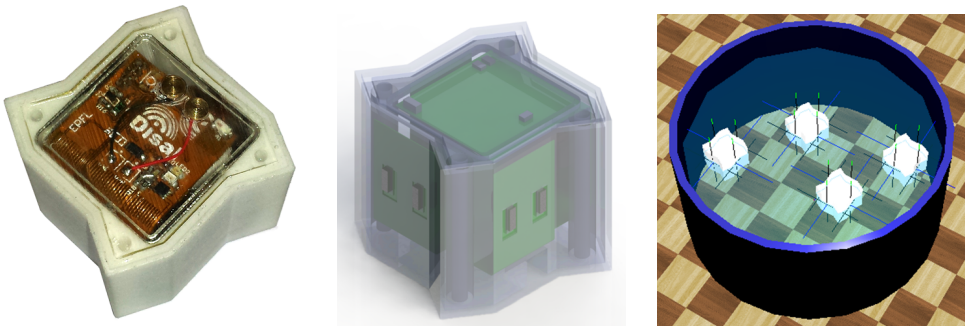
THE submicroscopic modeling level is the most detailed level. The word "submicroscopic" reflects the fact that the model provides a higher level of details than a canonical microscopic model, faithfully recreating intra-robot features such as the body shape and the functionality of individual sensors and actuators. With this level of details, a submicroscopic model and its corresponding simulator can keep track of a number of state variables such as the exact pose of a robotic module, the specific forces exerted by one of its actuators, or the signal perceived by one of its sensors. In this chapter, we explain how we employ the Webots robotic simulator as a platform where we develop a submicroscopic model of our fluid-mediated self-assembling robotic system.

9.1 Designing the Model

In order to realistically recreate our self-assembling robotic system in simulation, we use Webots [86], a physics-based robotics simulator which uses the Open Dynamics Engine (Open Dynamics Engine (ODE)) for simulating rigid body dynamics. Additionally, in order to simulate specific not natively supported physics such as complex fluid dynamics, it is possible to employ custom-designed physics plugins. Building our submicroscopic model within the Webots simulation framework comprised two main aspects. First, faithful recreation of the Lily robotic module's hardware and software features, and second, faithful recreation of the hydrodynamic forces acting on the floating robotic modules in the fluid-mediated self-assembly arena.

Recreating the Lily Robotic Module

We recreated the Lily robotic module within the simulated world of Webots in several steps. In the first step, we defined the physical entity of the module. A CAD design of the external shell of the module was designed in SolidWorks and directly exported to Webots in the VRML V2.0 format. This defines the bounding object (i.e. bounding volume) associated with a Lily and is the one referred to by the ODE engine for simulating the collisions among modules. In



(a) (b) (c)

Figure 9.1 – (a) A real Lily robotic module. (b) CAD design of the Lily robotic module exported from SolidWorks to Webots. (c) A sample world of simulated Lily robotic modules in Webots. The lines on the modules indicates the axes of the EPM connector nodes located inside the modules.

the second step, a Lily robotic module PROTO was created. Within Webots, a PROTO allows for capturing all feature of a certain object within one PROTO container. The Lily PROTO was then augmented with the physical features of the Lily robotic modules. In particular, its bounding object as exported from SolidWorks, mass, and center of mass. A physical object in Webots has its associated linear and angular damping coefficients which are used to slow down an object. The rotational and linear speed of each object is reduced by the specified percentage (between 0.0 and 1.0) every second allowing for coping with simulation instability. We initially left these parameters to their default values of 0.5 each. The Lily PROTO was then augmented with several *functionality nodes*, that is four connector nodes located on the sides to replicate the EPM latching mechanism, four emitter as well as four receiver nodes located on the sides with a range of 0.5 mm replicating the EPM inductive channel function, one light sensor node on the top, and an emitter as well as a receiver node located on the top with an infinite range to replicate the radio channel communication with the base station. Additionally, the Lily PROTO was declared as a *supervisor* rather than a *robot*. A *supervisor* has the base functionalities of a *robot* with the additional possibility to access other *robot*'s information fields. This was primarily done to facilitate further development and debugging of the simulated world. The supervisor's functionalities proved useful later on as we tried to implement the EPM communication and the hand-shaking established through exchanging *hello* and *acknowledgment* messages between the robotic modules. In some rare cases, due to simulation synchronization issues between the emitter-receiver nodes emulating the EPM inductive channel and the robotic modules' controllers, the state update of the two Lilies would not happen correctly despite the successful initial hand-shaking. For these cases, we employ the supervisor access to the state data of all modules, in order to make sure that the two Lilies involved in a communication manage to update their states accordingly, following a successful hand-shaking. To be precise, the Lily that initiates the communication utilizes its supervisor functionality to assure the consistency of the final state update with its neighboring Lily. The next step was then importing the Lily robotic modules' embedded controller software into the Webots simulated world. The low-level functionalities such as EPM communication through

sending current pulses needed to be abstracted away and replaced by similar functions from Webots. However, the adapted controller still maintained the same structure as the original one programmed on the real Lilies. The specific rulesets employed on the simulated robotic modules are identical to those embedded in the real robotic modules.

Recreating the Flow Field

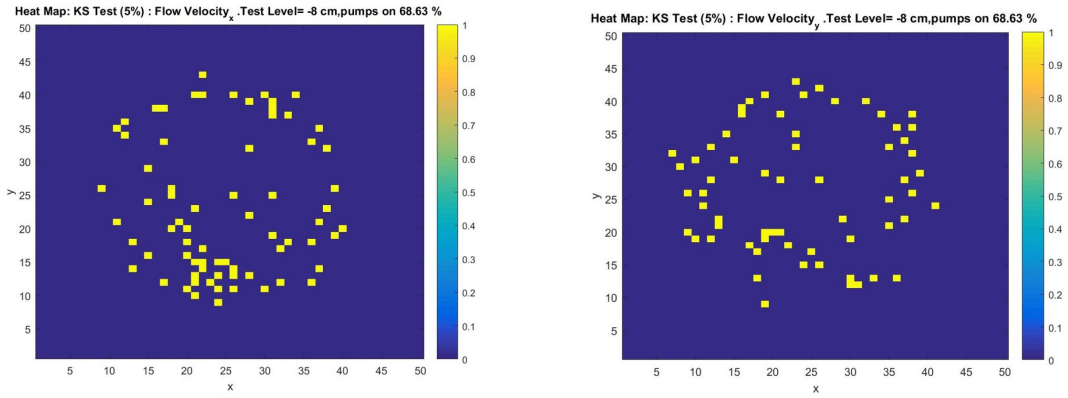
The latest version of Webots supports a basic fluid node which allows for a simple uniform stream velocity, but is not capable of simulating a complex fluidic field such as the one created in our experimental arena. In order to reproduce the complex flow field and the hydrodynamic forces acting on the Lilies in the real world, we use an approach inspired by the one in [90]. Our approach distinguishes from the one of [90] in two ways: (i) we capture trajectories of multiple floating blobs rather than a single one, with the aim of capturing the effects of interactions and collisions of the floating objects which disturb the flow field, (ii) rather than brute force optimization, we employ a Particle Swarm Optimization (PSO) algorithm to optimize for the model parameters.

A spherical object has an isotropic drag coefficient, i.e. a constant value in all directions, while submerged in a fluidic flow. We record the trajectory of floating spherical blocks (diameter of 3 cm), roughly the same size of a Lily robotic module, for three experiments with random starting positions and duration of 10 minutes each. For this we use ping pong balls whose weight is tuned such that the submersion level is similar to that of a Lily robotic module (25 mm below water level). The captured velocity fields acquired from different experiments are then augmented and discretized on a regular grid of 50 cells on each side, for our water tank of 60 cm in diameter. For each cell of the grid, the average and standard deviation of the observed velocity vectors are computed and assigned to that cell as expressed in Eq. 9.4. The fluid velocity field can be computed considering the drag force. The value of the Reynolds number R_e determines the flow regime and the form of the drag force. The Reynolds number is a dimensionless value that measures the ratio of inertial forces to viscous forces and descibes the degree of laminar or turbulent flow. Systems that operate at the same Reynolds number will have the same flow characteristics even if the fluid, speed and characteristic lengths vary. The Reynolds number is calculated as below:

$$Re = \frac{\rho V L}{\mu} \simeq 6700 \tag{9.1}$$

In the case of our system, this high Reynolds number indicates that the drag force takes a quadratic form:

$$|\vec{F}_{drag}| = \frac{1}{2}\rho AC|\vec{v}_{block} - \vec{v}_{flow}|^2 \tag{9.2}$$



(a) **(b)**

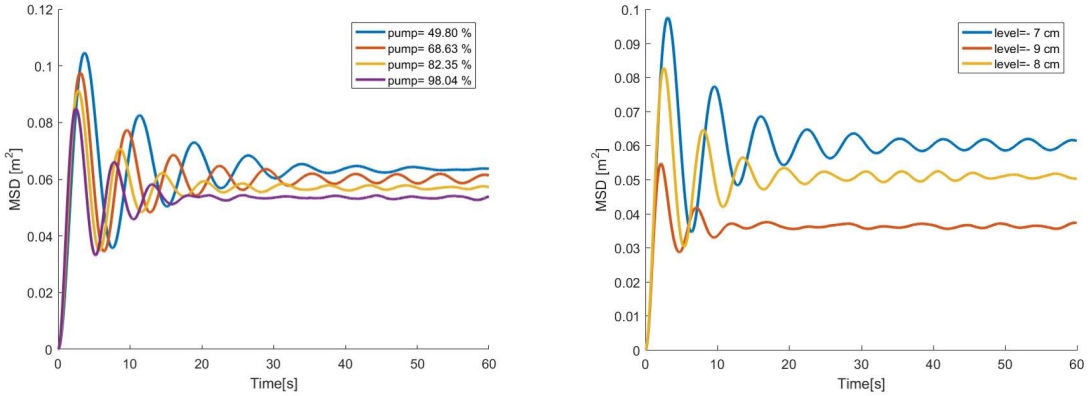
Figure 9.2 – Visualization of the Kolmogorov-Smirnov (KS) test results on the captured velocity field. The points of higher temperature (yellow color) indicate the grid cells for which the hypothesis that the data points within the corresponding cell have a Gaussian distribution $\mathcal{N}(0,\sigma^2)$ is rejected at a significance level of 5%.

where $\rho = 10^3\ kg/m^3$ is the density of water, $V \simeq 20\ cm/s$ the experimentally-measured mean velocity of a ball, $L = 3$ cm the characteristic dimension, and $\mu = 8.90\ .\ 10^{-4}$ Pa.s the dynamic viscosity of water. The submerged area of the globe is, $A = 7\ cm^2$ and the drag coefficient constant in all directions $C = 0.47$. The velocity and acceleration of the ping pong balls are computed using the captured trajectory data. Considering the mass of a ball m, the flow velocity is then computed from Eq. 9.2 as below, considering $\vec{F}_{drag} = m\vec{a}_{drag} = m\vec{a}_{drag}$:

$$\vec{v}_{flow} = \vec{v}_{block} + \frac{m\vec{a}}{\sqrt{\frac{1}{2}\rho ACm\sqrt{a_x^2 + a_y^2}}} \tag{9.3}$$

A customized physics plugin is then designed for Webots so that an appropriate drag force is applied to a simulated Lily module based on the velocity of the module and the flow velocity at its location at each time instant. In order to account for rotational effects, the drag force is integrated over each face of the module. Each face is divided into $N = 10$ sections, and the drag force is computed for each section using Eq. 9.2 with C being the estimated Lily robotic module's drag coefficient C_{Lily}.

For each cell j in the grid of a total of 2500 cells, we record the average μ_j and the standard deviation σ_j of the computed flow velocity vectors. We also test the normality of the distribution in each cell using the KS test. Results, shown in Figure 9.2, demonstrate that for the majority of the grid cells, the KS test failed to reject the hypothesis that the samples do not belong to a normal distribution with a confidence level of 95%. We can thus assume that the velocity at the location of each grid cell can be drawn from a normal distribution. Therefore, when a



(a) **(b)**

Figure 9.3 – (a) Effect of changing pump power on Mean Square Displacement (MSD). All experiments were done at the same water level, i.e. -7 cm . (b) Effect of the water level on MSD. The pump power is fixed at 68.63%.

block falls in a given cell j, the physics plugin applies the corresponding flow velocity as below, where K_v is a free optimization parameter.

$$v_{flow,j} = K_v \mathcal{N}(\mu_j, \sigma_j) \tag{9.4}$$

We consider the drag force as below, where K_F is a free optimization parameter.

$$|\vec{F}_{drag}| = K_F \frac{1}{2} \rho A C |\vec{v}_{block} - \vec{v}_{flow}|^2 \tag{9.5}$$

The physics plugin also adds a stochastic force $F_s \sim \mathcal{N}(0, \sigma^2_{stoch})$ to the center of mass of each block in order to take into account the stochasticity and non-modeled effects as in [90]. The standard deviation σ_{stoch} defines our third optimization parameter. Moreover, we have two additional free parameters in Webots which are the linear and angular damping of the block: $D_{linear}, D_{angular}$. In summary, we will have a total of five free optimization parameters to be calibrated: the constants $K_v, K_F, \sigma_{stoch}, D_{linear}$, and $D_{angular}$.

9.2 Calibrating the Model

In the previous section, we explained how we designed the model in order to faithfully capture the underlying physics present in our experimental platform. In this section, we focus on calibrating the free model parameters. By definition, model calibration is the process of

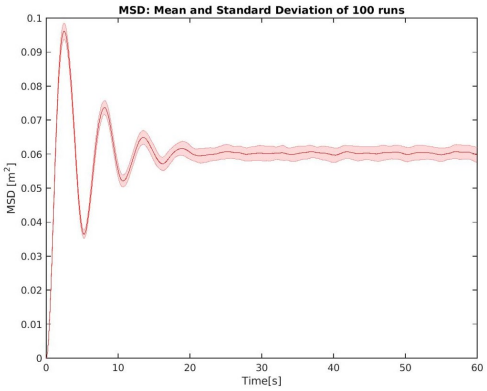


Figure 9.4 – Mean and standard deviation of MSD of simulated trajectories in Webots. The data is extracted from 100 runs.

adjustment of the model parameters to obtain a model representation of the processes of interest that satisfies pre-agreed criteria, typically expressed in the form of faithfulness metrics. We use the trajectories of the blocks floating on the simulated and real flow fields and refer to the MSD extracted from each data set. We then define our faithfulness metric to be optimized as the error between the real and simulated MSD functions. We run a PSO algorithm in order to optimize the free model parameters and encode our targeted metric as the fitness function of the PSO. In the following section we motivate the choice of the MSD metric.

9.2.1 Mean Squared Displacement Metric

In statistical mechanics, the MSD is a measure of the deviation of the position of a particle with respect to a reference position over time. It is the most common measure of the spatial extent of random motion, and can be thought of as measuring the portion of the space “explored” by the random walker. In the realm of biophysics and environmental engineering, the MSD is measured over time to determine if a particle is spreading solely due to diffusion, or if an advective force is also contributing. For instance, MSD analysis is a technique commonly used in colloidal studies and biophysics to determine the dynamics of displacement of particles over time. The MSD is expressed as below.

$$< \Delta r^2(t) > = \sum_{k=0}^{n} < [R_k(t) - R_k(t_0)]^2 > \tag{9.6}$$

Where R is the position vector, and n is the total number of particles. MSD is usually used to calculate the diffusion coefficient of a given system [91]. In order to verify that the MSD captures the change in the system dynamics, experiments of 10 minute length per water level and pump power configuration were conducted using 24 ping-pong balls. As indicated in the

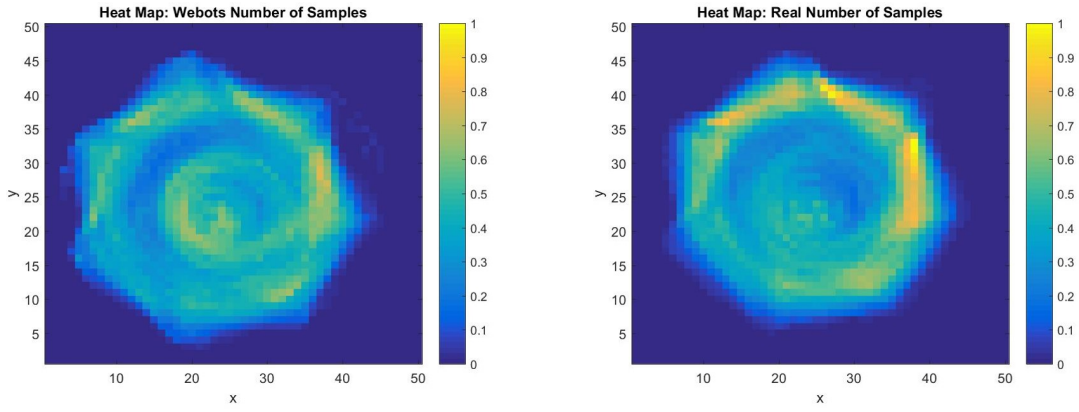


Figure 9.5 – Comparison of (a) real, and (b) simulated number of samples. The data is extracted from 10 minutes of experiment. The points of higher temperature (yellow color) indicate the grid cells for which the normalized presence of the floating blocks, i.e. the ping-pong balls, were more frequent.

previous section, we use ping-pong balls for simplicity as they are symmetric and have an isotropic drag coefficient. We used 3 water levels in the tank, measured from the upper border as -7, -8, and -9 cm, respectively.

Figure 9.3 illustrates the MSD curves for different pump power and water level settings. Each pump has a maximum flow of 9 l/min at 100% power. First, we can notice that the MSD has an oscillating pattern and a convergence plateau. The oscillations are related to a situation where the blocks are affected by the stirring flow generated by the pumps and the frequency of the oscillations is related to the speed of circulation. On the other hand, the plateau indicates a maximum effective displacement explored by the blocks. As we can see in Figure 9.3(a), the higher the pump power, the higher the frequency of the oscillations and the lower is the plateau. This can be explained considering that when the pumps power is increased, the force pushing the blocks towards the center is higher and therefore the effective radius of the area explored by the blocks is smaller. Furthermore, by keeping the same pump power, but reducing the water level, we can see a similar effect. Lowering the water level will change the relative alignment between the pumps' nozzles and the surface of the water, therefore increasing the fluidic force acting on the blocks' which in turn will decrease the effective exploration radius.

9.2.2 Parameter Optimization

As discussed earlier, the basic principles of the calibration is to match the simulated and real MSD curves in an attempt to faithfully reproduce the dynamics of the real system in simulation. There are five model parameters to be tuned in the calibration process; K_v defining a scaling factor for the randomness in the velocity field as described in Eq. 9.4, K_F defining a scaling

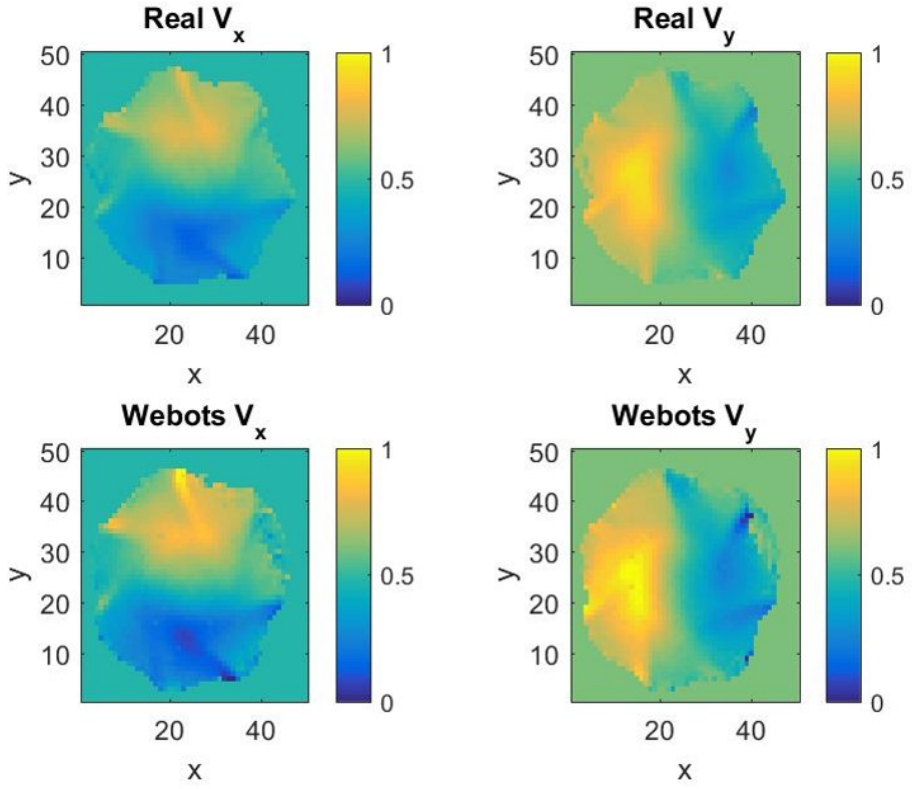
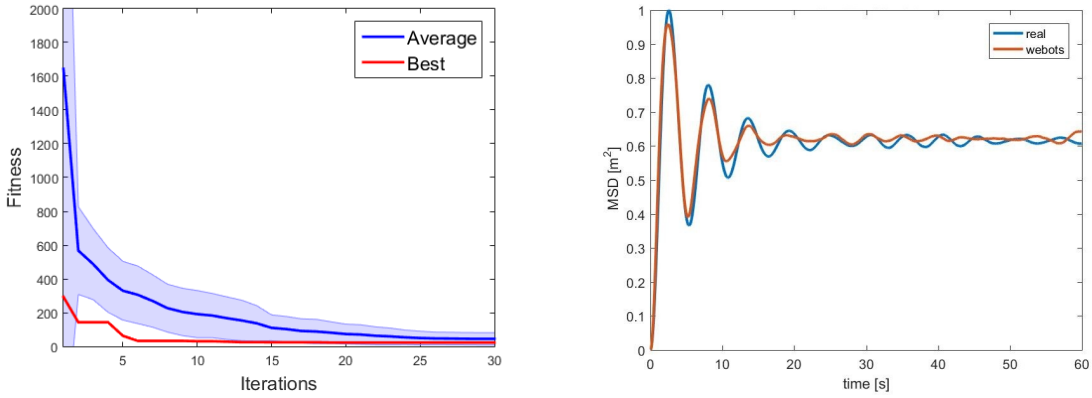


Figure 9.6 – Comparison of real and simulated mean velocity. The data is extracted from 10 minutes of experiment.



(a) **(b)**

Figure 9.7 – (a) Learning of the fitness function throughout the PSO algorithm iterations. (b) Comparison of simulated and real MSD data for the case of experiments with ping-pong balls. Simulated trajectories are obtained using the optimized submicroscopic model parameters expressed in Table 9.1. The pump power is at 68.6% and the water level is at -8 cm.

factor in the drag force as described in Eq. 9.5, σ_{stoch} defining the standard deviation of the stochastic force field, and the two linear and angular damping coefficients used by Webots for any body mass expressed as D_{linear} and $D_{angular}$, respectively. As mentioned before, we use a PSO algorithm to fine tune these parameters so that real and simulated MSD curves are as aligned as possible. The PSO parameters of self coefficient, personal best coefficient, and neighborhood best coefficient are set to 0.1832, 0.5287, and 3.1913, respectively. No particular attempt to optimize the PSO parameterization was carried out. To check the necessity to use a noise-resistant version in PSO, we have to verify the amount of noise characterizing the chosen fitness function described in Eq. 9.7. To do that, we carry out 100 runs of the same simulation setup with random initialization of the five parameters, and we calculate the mean and standard deviation. The result is illustrated in Figure 9.4. We notice that the standard deviation is small enough to assume that the noise-resistant PSO will not be necessary. The fitness function is therefore the difference between the resulting MSD from simulation and the mean MSD measured on our real set-up. When computing the MSD value the trajectories of each of the floating blocks are aggregated, rather than being considered separately, and the resulting MSD curves are averaged. Mathematically, we can formulate as below, where N_s is the number of time steps in one sample. Where $N_s = 2400$ in our case corresponds to 60 seconds emulated wall-clock time with a simulation time step of 25 ms.

$$Fitness = \sum_{i=1}^{N_s} |MSD_{webots} - MSD_{real}| \tag{9.7}$$

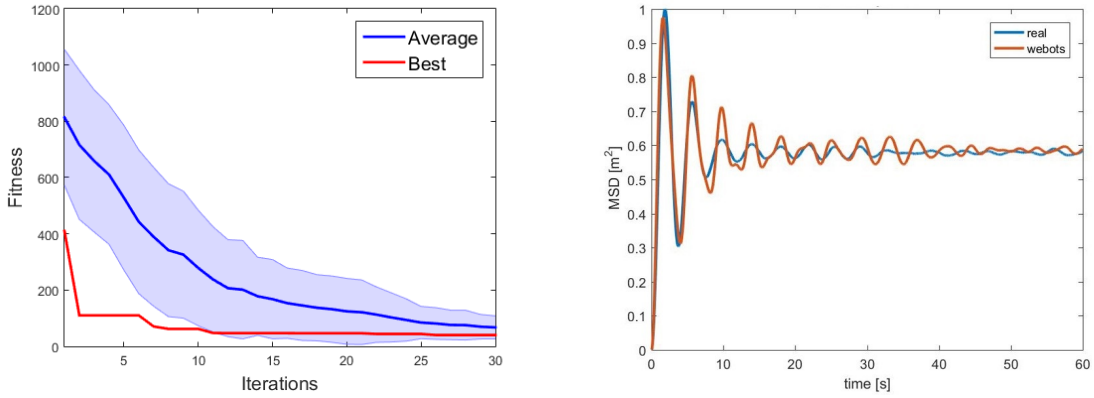
In what follows, we will consider two cases: we will first apply the optimization to the case of trajectories obtained from experiments with floating ping-pong balls for a specific pair of water level and pump power to validate the possibility of having matching MSD through our optimization process in this simpler case where the drag coefficient is isotropic. We will then apply the same method to match the MSD curves in the case of trajectories obtained from experiments with Lily robotic modules.

Experiments with Ping-Pong Balls

In this case, we have 24 ping pong balls, the water level is equal to 8 cm from the edge of the tank, the pump power is set to 68.6 %. The PSO results as well as the matching MSD are shown

Table 9.1 – PSO Algorithm Parameters and the optimized submicroscopic model parameters. Simulation with ping-pong balls.

Number of dimensions	Maximum number of iterations	Swarm size	K_v	K_F	σ_{stoch}	D_{linear}	$D_{angular}$
5	30	47	1.24	2.266	195.25	0.3483	0.2476



(a) (b)

Figure 9.8 – (a) Learning of the fitness function throughout the PSO algorithm iterations. (b) Comparison of simulated and real MSD data for the case of experiments with Lily robotic modules.

in Figure 9.7. The simulation as well as the optimized model parameters are listed in Table 9.1.

As a checking step, we compare the normalized number of samples in both Webots and real experiment, and we can see a matching between the two as illustrated in Figure 9.5. The number of samples in each cell gives an idea about the spacial distribution of the floating objects in the arena, i.e the cells with highest number of samples will indicates the most visited cells. Consequently, a visual inspection of the matching between the left and right plots of Figure 9.5 will indicate similar dynamics. In addition, we compare the mean velocities in x and y directions, and they are also matching as we can see in Figure 9.6. A matching mean velocities in both directions indicate similar system kinematics on average.

Experiments with Lilies

In this case, we have 15 Lilies, the water level is -8 cm and the pump power is set to 68.6 %. The PSO results as well as the resulting matching MSD are shown in Figure 9.8 and the used simulation parameters as well as the optimized model parameters are listed in Table 9.2. This combination of water level and pump power is the one that we will use as the default. However, for any other combination the same optimization procedure can be carried out in order to find the proper model parameters.

Table 9.2 – PSO Algorithm Parameters and the optimized submicroscopic model parameters. Simulation with Lily robotic modules.

Number of dimensions	Maximum number of iterations	Swarm size	K_v	K_F	σ_{stoch}	D_{linear}	$D_{angular}$
5	100	47	1.365	0.442	126.82	0.332	0.12

Summary

In this chapter, we introduced our submicroscopic model developed within Webots, a physics-based robotics simulator. We explained how we created a dedicated physics plugin in order to recreate the fluidic flow field observed in our real experimental setup within the simulated world. The designed model was then calibrated by finding optimized model parameters for matching real and simulated trajectory data. We used the matching between the MSD of the real and simulated trajectories as the metric assessing the simulation faithfulness and employed a PSO algorithm to optimize free model parameters influencing such metric.

10 Microscopic Model

THE microscopic modelling level sits between the submicroscopic level, at which the physical details of the real system are faithfully recreated, and the macroscopic level, at which only the aggregated collective behavior of the system is captured. In this chapter, we introduce our microscopic model and the corresponding simulation tool. At this modeling level, the system is still viewed as a swarm of individuals each having an internal state and executing a set of assembly rules upon interactions with one another. However, the spatial aspects of the dynamics of the system are not explicitly included. Our framework is based upon the abstract method for randomized interactions among bodiless modules originally introduced in [92]. We build upon this method in two ways. First, in order to model interactions between robotic modules we introduce and utilize the notion of "extended graphs" along with appropriate geometrical constraints. Second, we introduce a new shape recognition method which is an extension over a graph isomorphism check to track the progress of the self-assembly process in the modeled system.

Graph transformation or graph grammar formalisms concern techniques of creating a new graph out of an original graph algorithmically. It has numerous applications, ranging from software engineering to layout algorithms and picture generation. Graph transformations can in general be used as a computation abstraction. The basic idea is that the state of a computation can be represented as a graph, further steps in that computation can then be represented as transformation rules on that graph. Such rules consist of an original graph, which is to be matched to a subgraph in the complete state, and a replacing graph, which will replace the matched subgraph. Describing dynamics of self-assembling robotic systems by means of an appropriate graph grammar formalism offers several advantages. First, a graphical description inherently suits the self-assembly problem where the typically fixed number of constituting modules can be represented by the graph vertices and the bonds forming and severing throughout the self-assembly 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 [93]. Second, such a description enables the application of several formal rule synthesis algorithms, originally developed for self-assembly of abstract graphs,

to the case of self-assembly of robotic modules. Previous research has already demonstrated the relevance and application of the standard graph grammar formalism for formulating self-assembly of bodiless modules and successfully presented several automatic rule synthesis algorithms. However, to capture the morphology of the robotic modules and the assemblies that they form, it is necessary to extend the standard graph grammar formalism. In other words, 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. In the following sections, we first review the standard graph grammar formalism applied to the problem of self-assembly of bodiless modules and then present our extended graph grammar formalism for the case of self-assembly of robotic modules. In doing so, we consider the specific but widely common case of robotic modules endowed with genderless latching connectors arranged in a rotationally symmetric fashion.

10.1 Graph grammars for Self-Assembly of Bodiless Modules

In this section, we summarize the graph grammar formalism for formulating self-assembly of graphs as presented in [15], and [92]. 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 self-assembly 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 self-assembly 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 self-assembly 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 self-assembly 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 self-assembly 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 $\ell : V \to \Sigma$ 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 $h : I \rightarrow V_L$.

Definition - Ruleset: A ruleset ϕ is a set of rules $r_i = (L_i, R_i)$ which specifies the evolution of the self-assembly process towards a desired target assembly out of initially disconnected modules. The application of rules included in ϕ sequentially advances the self-assembly 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

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

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

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

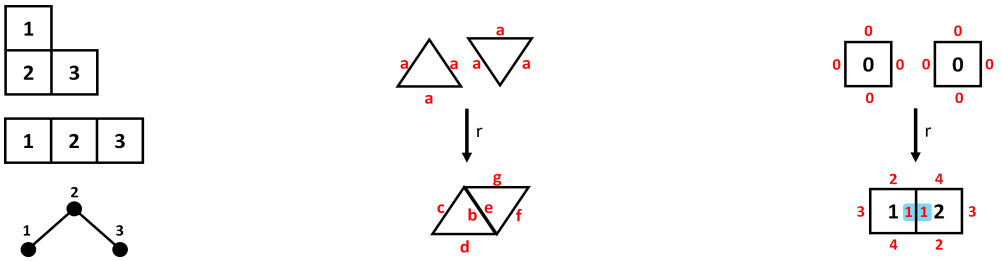


Figure 10.1 – Different structures represented by the same abstract graph not capturing the orientation of the formed links (left). Association of latching connectors with labels marked in red [15] (middle). Relative CCW convention for hop numbering, marked in red, starting at the most recently engaged latching connector, marked in blue (right).

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 $P : \phi \rightarrow (0,1]$, indicating the tendency for the corresponding event to take place provided that the conditions under which the rule is applicable are met. All formal rule synthesis methods proposed for programmable self-assembly of graphs automatically generate a ruleset ϕ for assembling a desired target by iteratively browsing and parsing the target graph [94], [15], [92]. Chapter 14 provides details on the functionality of such methods and how they can be extended to generate rules for self-assembly of rotationally symmetric robotic modules.

10.2 Graph Grammars for Self-Assembly of Robotic Modules

In this section, we explain how we extend the graph grammar formalism to formulate the problem of ruleset synthesis for programmable self-assembly of rotationally symmetric robotic modules [71]. As explained in Section 10.1, the self-assembly 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 10.1 (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 self-assembly 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 structure of the system graph, 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 [15]. 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 10.1 (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 [15] 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, [15] 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 with $O(N^2)$.

In what follows, we propose an alternative approach for applying a graph grammar formalism to the self-assembly 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 robotic modules in 2D, the assumptions we make are general enough to be directly applied to similar platforms. The method is also easily applicable to 3D self-assembly 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 self-assembly 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

Algorithm 1 Pseudo code for generating a sequence of graphs employing the random pairwise interactions dynamics..

1: Initialize with $t = 0$ and G_0.
2: Increment t.
3: $F(G_t)$ is sampled, giving a pair of vertices $\{x, y\}$.
4: Let $\phi_t = \{r \in \phi : \exists h \ \ (r, \{x, y\}, h)$ is an action on $G_{t-1}\}$.
5: If $\phi_t = \emptyset$ let $G_t = G_{t-1}$ and return to step 2.
6: Let $r \in \phi_t$ be chosen at random, uniformly.
7: Let $G_{t-1} \xrightarrow{\text{r, \{x,y\}, h}} G'$.
8: Let $G_t = \begin{cases} G' & \text{with probability } P(r) \\ G_{t-1} & \text{with probability } 1 - P(r) \end{cases}$
9: Return to step 2.

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 10.1, 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 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.

10.3 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 $P : \phi \to (0, 1]$. The set of pairs of disjoint vertices is 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 self-assembly process. More specifically, it is the dynamics of mixing in the physical system

Algorithm 2 Pseudo code of the shape recognition algorithm for the case of square-shaped Lily robotic modules.

```
1: procedure GROUPTOSHAPE(V_G, E_G, S_G)
2:     ∀ v_i : i = 1,2,...,|V_G|, pos(v_i) = (0,0)
3:     ∀ v_i : i = 1,2,...,|V_G|, prev(v_i) = 0
4:     ∀ v_i : i = 1,2,...,|V_G|, dir(v_i) = 1
5:     unvisited ← {v_1}
6:     visited ← {∅}
7:     while (unvisited ≠ {∅}) do
8:         v_i ← unvisited(1)
9:         s̄_i ← (dir(v_i) − 1 + 2) (mod 4) + 1
10:        s_i ← S_G(v_i, prev(v_i))
11:        d ← s̄_i − s_i
12:        if d < 0 then
13:            d ← d + 4
14:        end if
15:        {v_j : j = 1 : |n_{E_G}(k)|} ← n_{E_G}(k)
16:        for j = 1 to |n_{E_G}(k)| do
17:            if v_j ∉ visited then
18:                s_j ← S_G(v_j, v_i)
19:                s̄_j = (s_j − 1 + d (mod 4)) + 1
20:                for k = 1 : 4 do
21:                    if s̄_j == k then
22:                        Let R(θ) be the 2D
23:                        rotation matrix
24:                        pos(v_j) ← pos(v_i)+
25:                        [−1,0]R(k·π/2)
26:                    end if
27:                end for
28:                dir(v_j) ← s̄_j
29:                prev(v_j) ← v_i
30:                unvisited ← unvisited ∪ {v_j}
31:                visited ← visited ∪ {v_i}
32:            end if
33:        end for
34:        unvisited(1) ← {∅}
35:    end while
36:    (x_min, y_min) = {(x, y)|
37:    (x, y) = pos(v_i), ∀ j, pos(v_j) > pos(v_i)}
38:    ∀i   pos(v_i) ← pos(v_i) − (x_min, y_min)
39:    return (pos)
40: end procedure
```

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 [92]. Therefore, the interaction probabilities are kept uniform similar to [92]. Similar to the work in [92], a random sequence of graphs $\{G_t\}_{t=1}^{\infty}$ is generated as described in Algorithm 1. The asymptotic behavior of $\{G_t\}$ can be characterized for various choices of ϕ and P.

10.4 Shape Recognition

In order to keep track of the structures being formed at each step in the simulation, the created sub-assemblies must be identified. In the case of abstract graphs, i.e. the self-assembly of graphs, this would represent a graph isomorphism problem. However, in the self-assembly of modules case, the addition of link slots allows for a simplified identification to be achieved, since it endows the graph with means to define geometrical orientations and relative positions of the vertices engaged in a structure.

Some solutions optimizing the process of graph isomorphism, such as the one presented by Asadpour et al. in [95], were first considered. Indeed, the most promising part of their research lies in the capability to determine the closest shape we could obtain from the reorganizing of the parts thanks to the definition of a similarity metric. This additional information would probably be of some help in the exploration of greedy approaches and self-disassembly However, in the interest of time and considering the overall complexity of the algorithm, its computational cost, and the lack of information on its implementation, we chose to go a

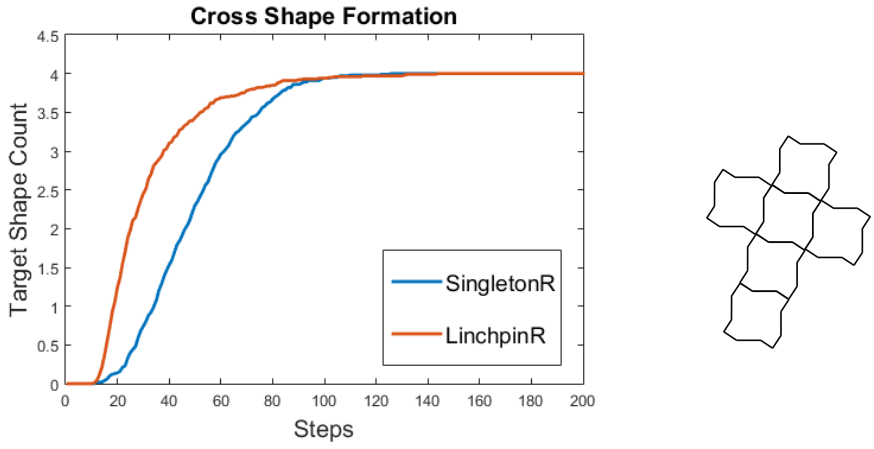
simpler solution.

Our method relies on 2D geometrical considerations and is based on the browsing of the enhanced graph and computing the relative locations of the vertices with respect to the starting node. In order to find the positions of the vertices and their orientation, the relative ordering of the slots of adjacent modules is used. Similar to the positioning of pixels on a screen, the method is applicable to modules performing structure formation with a wide variety of shapes provided that the edges of the shape are regular (latches of the same size, for instance).

Tracking the progress of the self-assembly 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 self-assembly of graphs, where the system is represented by an abstract graph at each time instant, this describes a problem of graph isomorphism [92]. 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 [95], [96]. The pseudo code of our proposed shape recognition algorithm for the case of Lily modules is shown in Algorithm 2.

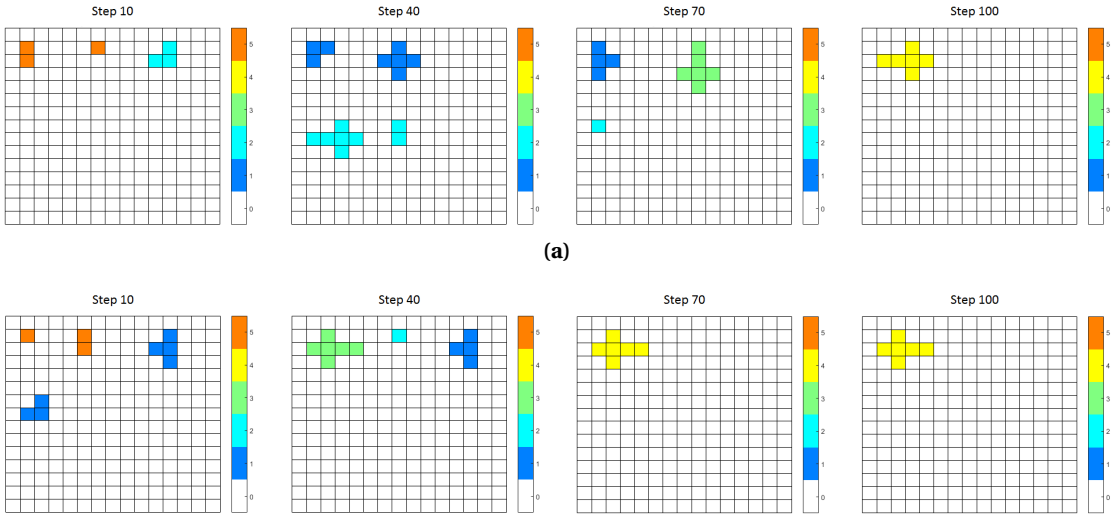
10.5 Running the Model

By definition, running a model is the process of running the appropriate simulation tool which contains a description of the model. In this section, we will illustrate using an example how the previous definitions and explanations are employed to realize an implementation of a microscopic model of self-assembling robotic modules in Matlab. In particular, consider a system of 24 initially isolated square-shaped modules endowed with four latching connectors, one on each side. Consider the modules to be Lily robotic modules as described in Chapter 5. The desired target structure is a cross shape comprising six robotic modules as shown in Figure 10.2(b). Employing our formalism detailed in Section 10.2, the system is modeled as an extended graph. We consider, two rulesets comprising extended rules for self-assembly of the cross shape target structure. Each ruleset is synthesized by a dedicated rule synthesis algorithm, namely the SingletonR and the LinchpinR algorithms, described in detail in Chapter 14. While the rules synthesized by SingletonR induce a serial assembly scheme by building the target adding one module at a time, LinchpinR rules allow for a more parallelized scheme in the self-assembly process by buidling dimer structures first and then joining them to build



(a) **(b)**

Figure 10.2 – Progress of the self-assembly process for a cross shape target in a system of 24 initially isolated square-shaped modules. Two different ruleset controllers are employed, a serial ruleset synthesized by the SingletonR algorithm described, and a parallel ruleset synthesized by the LinchpinR algorithm. Both algorithms are described in detail in Chapter 14.



(a)

(b)

Figure 10.3 – Visualization of the progress of the self-assembly process towards a cross shape target in a system of 24 initially isolated square-shaped modules. The color coding indicates the count number of each assembled structure. Two different ruleset controllers are employed, (a) a serial ruleset synthesized by the SingletonR algorithm described, and (b) a parallel ruleset synthesized by the LinchpinR algorithm. Both algorithms are described in detail in Chapter 14.

the target structure eventually. As a result of this parallelization, LinchpinR rules are expected to achieve the target structure faster. Figure 10.2(a) depicts the progress of the assembly process, in particular the number of assembled copies of the target structure, as a function of simulation steps. Figure 10.3 shows snapshots of the simulated self-assembly process.

From both figures it can be seen that the LinchpinR rules build the target structure in fewer steps, thus achieving a higher assembly rate. Both rulesets eventually achieve the maximum assembly yield of four.

Summary

In this chapter, we presented our microscopic model and corresponding simulation tool. The simulation tool is based upon the abstract model for randomized interactions among bodiless modules and employes the extended graph grammar formalism introduced in Chapter 14. In this dissertation, the microscopic model is specifically developed to evaluate the comparative intrinsic performance of the synthesized rulesets abstracting away the spatial aspects of the self-assembly process. Given a desired target structure and its associated appropriately synthesized ruleset, the microscopic simulation tool allows for verifying whether the ruleset and its corresponding assembly strategy attain the necessary complexity in order to build the target structure efficiently.

11 Macroscopic Model

At the highest abstraction level sit the macroscopic models. These models directly describe the collective behavior of the robotic swarm, abstracting away the often complex and partially predictable behavior of the constituting robotic modules. Because macroscopic models use fewer parameters to describe the dynamics of the swarm they offer higher computational efficiency and tractability compared to microscopic and submicroscopic level models. Calibration with real experimental data directly obtained from the physical system or from accurate submicroscopic or microscopic level models is crucial to ensure the accuracy of the macroscopic models. In this chapter, we focus on developing probabilistic models at the macroscopic level. We employ the Chemical Reaction Network (CRN) formalism and capture the system dynamics using Markovian stochastic processes.

11.1 Introduction

A key component in studying programmable stochastic self-assembling systems is developing models that accurately describe the assembly process dynamics. Such models would help in: (1) accurately predicting the performances (assembly rate and yield) of the distributed system, and (2) evaluating and optimizing control strategies, whether distributed (e.g., ruleset controllers programmed on the modules) or centralized (e.g., modulating environmental features such as mixing forces driving random interactions among modules), based on model predictions [65], [85].

Several works have addressed developing Markovian probabilistic models for stochastic self-assembling systems to date [93], [97]–[100]. The choice of employing probabilistic modeling techniques for such systems is essentially motivated by the randomness lying at the core of these systems: random motion of the modules in the environment, explicit random decisions made by the modules' embedded controller, and random interactions among the modules [89]. Additionally, probabilistic models can be employed to provide a high-level macrostate description of the system state at each point in time by abstracting away low-level physical

details of the system state such as positions, velocities, and internal states of all modules (i.e. microstate description). A general methodology for developing accurate probabilistic models of the dynamics of programmable self-assembling systems is sought after to date.

In this work, our focus is on creating a general approach for developing a discrete-state hidden Markov process model of programmable stochastic self-assembly directly obtained from (1) a description of the robotic modules' embedded ruleset controller, and (2) an estimation of the rate constants defining the formation rates of different assemblies. To demonstrate this approach, we consider the case of our floating self-assembling robotic system, where the self-assembly process is controlled in a fully distributed fashion through the programmable embedded ruleset controllers of the robotic modules. We use a high-fidelity calibrated simulation of the system as the ground truth and address different aspects of developing probabilistic macrostate models for the system. The contributions of this work are along three axes (1) a new rate estimation method for computing Markov model parameters is introduced and compared with two existing ones [93], [97], (2) a new method for estimating diffusion coefficient and evaluating the well-mixed condition is studied and compared with an existing one [93], and (3) a systematic method in the literature for refining Markov models using HMM formalism, presented in [98], is employed to develop an HMM through automatic refinement of an initial Markov model of the system.

11.2 Markovian Models for Programmable Self-Assembly

We employ similar formulations as in [98] and [99], and provide formal definitions of several concepts in this section.

Markov Models

A discrete-time stochastic process can be defined as a collection of random variables, $\{X_n\}_{n\in\mathbb{N}}$. Similar to realization of a single random variable, a realized trajectory $\omega \in \mathbb{N} \rightarrow X$ generated by the stochastic process defines an assignment of the random variables X_n to particular values x in state space X. A discrete-time discrete-state Markov process, i.e. a Markov chain, has the following property, where $x_i \in X$:

$$Pr\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, ..., X_0 = x_0\} = Pr\{X_{n+1} = x_{n+1} | X_n = x_n\} \quad (11.1)$$

The transition dynamics of a Markov chain is specified by the one-step transition probabilities, where matrix A is independent of n for a stationary process:

$$A_{ij}(n) = Pr\{X_{n+1} = j | X_n = i\} \quad for\ i, j \in X \quad (11.2)$$

Consider a self-assembling robotic system consisting of N_0 individual robotic modules that may be in N_s different control states $q_1, \ldots, q_{N_s}$ as specified in their embedded ruleset con-

troller. At the microscopic level, the state of the system can be described by the vector:

$$\vec{X}^{\text{micro}}(t) = \big[Q_1(t), Q_2(t), \ldots, Q_{N_0}(t)\big], \tag{11.3}$$

where $Q_i(t) = q_1, \ldots, q_{N_s}$ is the state of module i at time t. The embedded ruleset controller guides the modules to build the target structure through building specific intermediate assembly configurations. Assuming that the set of possible assembly configurations as prescribed by the embedded ruleset controller is $c_1, \ldots, c_{N_c}$. At the macroscopic level, one is interested in the number of copies of the specific assembly configurations. Therefore, the state of the system can be described as:

$$\vec{X}^{\text{macro}}(t) = \big[N_1(t), N_2(t), \ldots, N_{N_c}(t)\big], \tag{11.4}$$

where $N_i \in \mathbb{N}_{\geq 0}$ is the number of copies of assembly configuration c_i at time t. We consider discrete-time, with $\vec{X}_n^{\text{macro}}$ indicating the system macrostate at the n^{th} timestep. Assuming well-mixed conditions, the evolution of the system state may be expressed as a discrete-time discrete-state Markov process $\{\vec{X}^{\text{macro}}\}_{n\in\mathbb{N}}$. Note that in order to fully specify the model, one has to list all the feasible system macrostates as well as the transition probabilities between them.

Hidden Markov Models

An HMM can be defined as a stochastic process with state space Y whose trajectories are described by a Markov chain with state space X and an output function $f : X \to Y$. The function f can be many-to-one, therefore, the resulting process Y can be non-Markov while the process X is. In practice, HMMs correspond to the case where the system state is not fully observable and as a result several underlying Markov process states (hidden states) may correspond to the same observable state of a non-Markov process.

Chemical Reaction Networks

A CRN $\mathcal{N} = (\mathcal{R}, \mathcal{S})$ is a set of reactions $\mathcal{R} = \{R_1, \ldots, R_{N_R}\}$ acting on a set of species $\mathcal{S} = \{S_1, \ldots, S_{N_S}\}$. Each reaction R is then defined as two vectors of nonnegative integers specifying the stoichiometry of the reactants, $\vec{r}_R = [r_{R,1}, \ldots, r_{R,N_S}]$, and the products species, $\vec{p}_R = [p_{R,1}, \ldots, p_{R,N_S}]$, respectively. The stoichiometry determines the number of copies of a given reactant or product species that is required or produced when a reaction occurs. The CRN provides a population model, it thus keeps track of the number of copies of each species present in the system at each given time. Consider the case of the self-assembling robotic system. The CRN species correspond to the assembly configurations induced by the modules' embedded ruleset controllers. The CRN state is given by the vector $\vec{X}^{\text{macro}}(t) \in \mathbb{N}_{\geq 0}^{N_s}$ at each point in time, where the vector elements specify the number of individuals of each species. A reaction R may occur provided that the number of reactants is sufficient, that is, $\vec{X}^{\text{macro}} \geq \vec{r}_R$

element-wise. When reaction R occurs, the new state $\vec{X}_{new}^{macro}$ is given by:

$$\vec{X}_{new}^{macro} = \vec{X}^{macro} - \vec{r}_R + \vec{p}_R \tag{11.5}$$

A characterizing quantity for a reaction R is its propensity function a_R, defined such that $a_R(\vec{x},.)dt$ is the probability that one instance of reaction R will occur in the next time interval $[t, t+dt)$ as $dt \to 0$, assuming the current state of the system to be $\vec{X}(t) = \vec{x}$. When the propensity function a_R is determined only by the current state of the system the Markov property holds and the time t until the next firing of reaction R can be described by an exponential random variable with mean $1/a_R(\vec{x})$, that is, its probability density is given by:

$$f(t) = a_R(\vec{x})e^{-a_R(\vec{x})t} \tag{11.6}$$

where $\vec{x}$ is the state of the CRN (i.e., a population vector), and $a_R(.)$ is the propensity function of the reaction R. The specific form of $a_R(.)$, assuming that the system is in dynamic equilibrium, is determined by the law of mass-action [101], and can be thus expressed as below:

$$a_R(\vec{x}) = k_R \tilde{a}_R(\vec{x}) \tag{11.7}$$

where k_R is the rate of reaction R and $\tilde{a}_R(\vec{x})$ has the appropriate form according to the stoichiometry of R, and does not depend on k_R. As an example, for the reaction $R: 1+2 \to 3$ the propensity function is computed with $\tilde{a}_R(\vec{x}) = N_1.N_2$ where N_1 and N_2 denote the number of reactants of type 1 and 2 in the system.

11.3 Developing Markov Models

We use the CRN formalism as detailed in Section 11.2 to express a Markov model of our system and estimate the reaction rate constants of the network. The structure of the CRN model is fixed, with the species determined by the assembly configurations , the parameters can instead be estimated through different methods. It can be shown that for a given set of observations of the system, i.e. a sequence of events $(e_1, ..., e_n)$, with $e_i = (R_i, t_i, \vec{x}_i)$, the Maximum Likelihood (ML) estimator of the rate vector $\hat{\vec{k}} = [\hat{k}_1, ..., \hat{k}_{N_R}]$ of the underlying CRN is [98]:

$$\hat{k}_j = \frac{\Sigma_{i=1}^{n} \mathbf{1}_{R_i=R_j}}{\Sigma_{i=1}^{n}(t_i.\tilde{a}_{R_j}(\vec{x}_i)}) \qquad j = 1, ..., N_R \tag{11.8}$$

Given the high-fidelity simulation framework described in Chapter 9, one can evolve the system dynamics from any desired initial condition and for any desired duration of time. The essential idea for estimating the rate constants is to observe and record the different reaction events and the corresponding waiting times in between.

Different methods have been proposed for gathering the statistics necessary for the rate

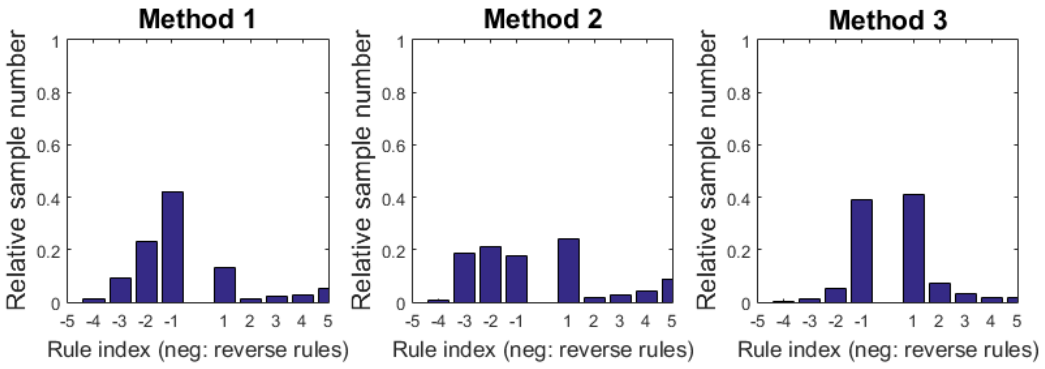


Figure 11.1 – Normalized number of samples per rule gathered through the three methods of Section 11.3.

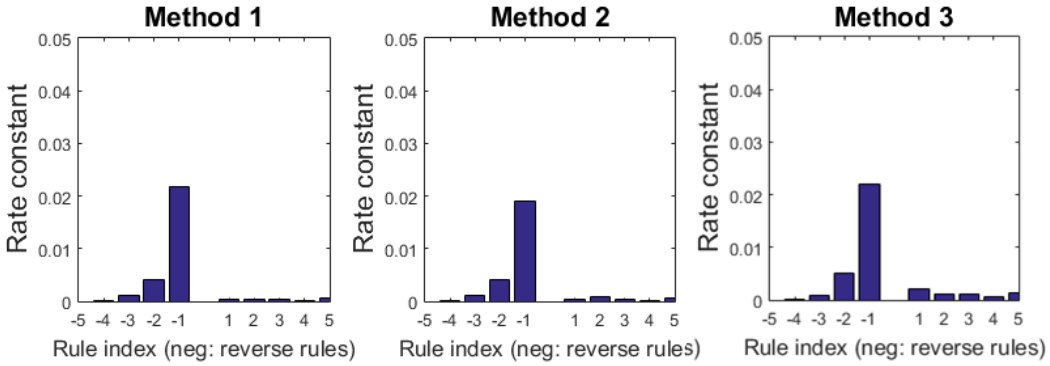


Figure 11.2 – Estimated rate constants using the three methods of Section 11.3.

estimation. We exploit the high-fidelity submicroscopic simulation framework described in Chapter 9 and study two existing methods and propose a third new method for gathering statistics. The main difference between these methods is the conditions under which they sample an event occurence and also the statistical composition of the dataset they gather. Methods 1 and 2 initialize and reinitialize the system to various macrostates upon occurrence of different events. Method 3, on the other hand, gathers observations from a full-length simulation of the system, initializing the simulation to fully isolated modules for each set of observations.

Method 1: This method has been originally proposed in [97]. The authors propose to run simulations for a wide variety of initial macrostates, covering all the states traversed by the system during the self-assembly process. Formally, the initial conditions are defined as follows:

$$s_M(t=0) = (S_1, S_2, \ldots, S_n)$$

for all feasible macrostates in the system. The simulation is initialized and run for each initial

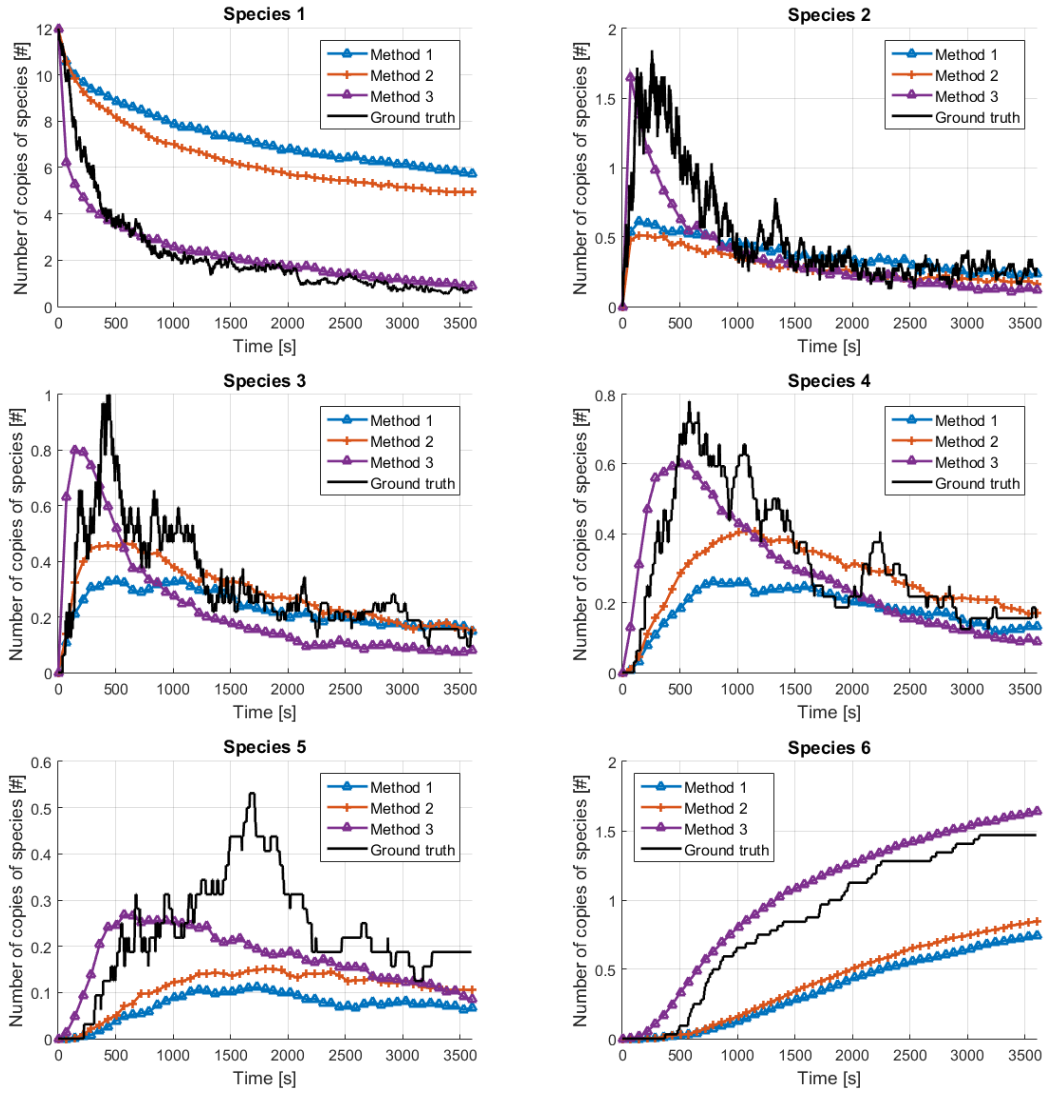


Figure 11.3 – Comparison of model prediction regarding the trajectory of each of the six species, averaged 1000 runs in Stochkit, formed by the ϕ_{chain} using the rates estimated by the three methods in Section 11.3. The ground truth is obtained by averaging 90 runs of the simulated world in Webots (see Chapter 9).

macrostate distribution, as soon as a reaction R occurs, the time of that reaction is noted and the simulation is reset.

Method 2: This method has been originally proposed in [93] and is similar to the Method 1 in that it also considers only a simulation period until a reaction occurs in the system. However, the number of different initial conditions is reduced. For computing the reaction rate for a specific reaction R, the initial macrostates used in the simulations are only the ones containing the reactants of reaction R. For instance, given the stochastic reaction equation $1+3 \rightarrow 4$, which describes the formation of substructure 4 from substructures 1 and 3, require

the following corresponding initial conditions:

$$s_M(t=0) = (S_1, 0, S_3, 0) \quad , \quad S_i = 0 \quad \forall\, i \neq \{1,3\}$$

The values of the number of copies of substructures S_i are randomly varied within the feasible macrostates. The simulation is thus initialized and run for each initial macrostate distribution, as soon as reaction R occurs, the time of that particular reaction is noted and the simulation is reset but the time is not reset.

Method 3: This is our proposed method. The idea is simply centered on the complete observation of simulations starting from different initial conditions (all the agents separated at start) until the target structures are ultimately built. This typically provides a wide variety of interactions between the agents distributed according to the natural tendency of the system, allowing therefore to collect enough relevant statistical information to determine rate constants. All initial conditions are defined as follows:

$$s_M(t=0) = (N_0, 0, \ldots, 0)$$

where the first element in the macrostate vector indicates the species corresponding to an isolated module, thus, $S_1 = N_0$.

For gathering the statistics, our simulation setup consists of 12 robotic modules that are stirred in the fluidic arena of 1.2 m diameter. The evaluation was performed for the ϕ_{chain} ruleset that builds the target structure of a chain using six robotic modules. All the forward rules are set to be executed with probability 1, for the reverse rules the probabilities were set to the following values 0.05, 0.01, 0.002, 0.0004, 0 for $\{\bar{r}_1, \bar{r}_2, \bar{r}_3, \bar{r}_4, \bar{r}_5\}$ respectively. Figure 11.1 shows the statistics of the number of sample points gathered while the estimated values of the rates are shown in Figure 11.2. It can be seen that the statistical composition of the datasets gathered by the three methods are very different. In addition to the different number of samples per reaction rule, the system state at which the reaction time has been sampled is also different for the three methods, resulting in different rate estimates. While the differences between the estimated rates as depicted in Figure 11.2 seem minor, they describe substantially different system evolution courses as depicted in Figure 11.3. We generate 1000 sample trajectories using the Gillespie method [101] with the Stochkit software for each CRN model. It is also noteworthy that Method 1 provides the most varied system state at the sampling time, while Method 3 samples the reaction events at system states through which the system has a natural tendency to traverse.

11.4 Evaluating Well-Mixed Condition

The mismatch between the Markov model predictions and the ground truth as depicted in Figure 11.3 can be ascribed to two factors. First, as previously mentioned, the underlying CRN

describing the states of the Markov model has its species determined by the ruleset. However, as a result of random interactions between the robotic modules, temporary bindings can also form, which are later severed as the modules switch their EPMs off and drift apart. These temporary bindings/structures are not explicitly modeled as the CRN species. Second, the physical dynamics of the system might be practically far from the ideal well-mixed conditions, thus voiding the Markov property assumption. In this section, we specifically look into validation of the well-mixed condition. The well-mixed condition is sufficient for guaranteeing that the underlying process is Markovian and that each possible combination of reactants for a particular reaction will be equi-probably involved in the next instance of the reaction [93].

We use the definition of well-mixedness from [101]: the rate at which new collisions occur should be greater than the rate at which reactions occur, in other words the modules should diffuse faster than they react. Similar to the work in [93], we rely on diffusion for module transportation. A measure of “well-mixed" as proposed in [93] is to require that $D/k_{av} > A$ where D is the diffusion coefficient, k_{av} is a nominal reaction rate in the system and A is the effective area of the fluidic arena occupied by the interacting modules. We study two methods for estimating the diffusion coefficient, one proposed in the previous literature [93] and a new method. In order to gather statistics, we exploit again our high-fidelity simulation framework described in Chapter 9 and conduct two simulated experiments with 12 robotic modules each for a duration of 10 min. The robotic modules are programmed with an empty ruleset controller and as a result unlatch right after they latch.

Method 1

This method has been originally proposed in [93]. The authors propose to estimate the diffusion coefficient for a robotic module as below:

$$D = \frac{E(r^2(t))}{4t} \tag{11.9}$$

where $r(t)$ denotes the random displacement of the robot as a function of time. We compute the expected value considering all the robotic modules and all the time steps. The final obtained estimated diffusion coefficient using this method is 0.0016 m^2/s. For the nominal reaction rate we consider the order of magnitude of the rate of the rule r_1 in the ruleset ϕ_{chain} as computed by Method 3, 0.001. The condition of $D/k_{av} > A$ implies that the arena size should be smaller than $A_{max} = 1.6m^2$ while the area of our arena is $1.13m^2$. We thus conclude that the system is well-mixed.

Method 2

Fick's law of diffusion gives a relation between the diffusion flux, the gradient in concentration and a diffusion coefficient under the assumption of the system being in steady state. It can be

expressed as:

$$J = -D\nabla_c \tag{11.10}$$

Where in the case of a 2D diffusion, J is the rate of module transfer per boundary length normal to the direction of transfer, expressed in $[modules/m.s]$, D is the diffusion coefficient, expressed in m^2/s, and ∇_c is the gradient in concentration, expressed in $[modules/m^3]$. In order to estimate the diffusion coefficient in the case of our 2D system, the arena is partitioned by a square grid of 16x16 with each cell i containing a concentration of modules c_i (where $c_i \in \mathbb{N}^{+0}$). For each simulation time step Δt, the modules' flow is computed across each cell boundary (e.g. from cell i to cell j) of a 4-connected neighborhood, using a discrete expression of the Fick's law:

$$F_{i\to j} = -D(c_j - c_i)\Delta t \tag{11.11}$$

To estimate the diffusion coefficient, we write:

$$D_{c_i\to c_j} = -\frac{F_{c_i\to c_j}}{(c_j - c_i)\Delta t_{c_i\to c_j}} \tag{11.12}$$

The diffusion coefficient is thus computed for every pair of observed concentrations (c_i, c_j). The result of this diffusion coefficient estimation is a 2D matrix. To investigate the well-mixed condition, individual values are then averaged to have a single value for the estimated D. The final estimation for the diffusion coefficient using this method is 0.00065 m^2/s. Referring to the well-mixed condition described as $D/k_{av} > A$, we should have an $A_{max} = 0.65m^2$. While the total area of the arena is $1.13m^2$, the effective area to which the motion of the modules is limited is smaller, about half of the total available arena size. We thus conclude that the system is border-line well-mixed.

11.5 Developing Hidden Markov Models

The goal here is to construct a CRN corresponding to an HMM. Based on an existing CRN $\mathcal{N}$ model of the system, we provide a method to automatically obtain a refined CRN $\widetilde{\mathcal{N}}$ model, where the original species are augmented with proper hidden species. We take the systematic approach described in [98] and employ a metric in order to provide for a fully automatic refinement method. The original approach starts from a given CRN $\mathcal{N} = (\mathcal{R}, \mathcal{S})$ and an associated set of events $(e_1, \ldots, e_n)$, and attempts to construct a refinement $\widetilde{\mathcal{N}} = (\widetilde{\mathcal{R}}, \widetilde{\mathcal{S}})$ of $\mathcal{N}$ such that the likelihood $\mathcal{L}(\vec{\tilde{k}}|e_1, \ldots, e_n) > \mathcal{L}(\vec{k}|e_1, \ldots, e_n)$, i.e., the sequence of events is better explained by the refined model than the original one. In order to refine a CRN, a species S_0 is selected. It is then split into two subspecies S_0^a and S_0^b. All reactions involving S_0 are then duplicated, with their corresponding reactants and products updated accordingly. Consider

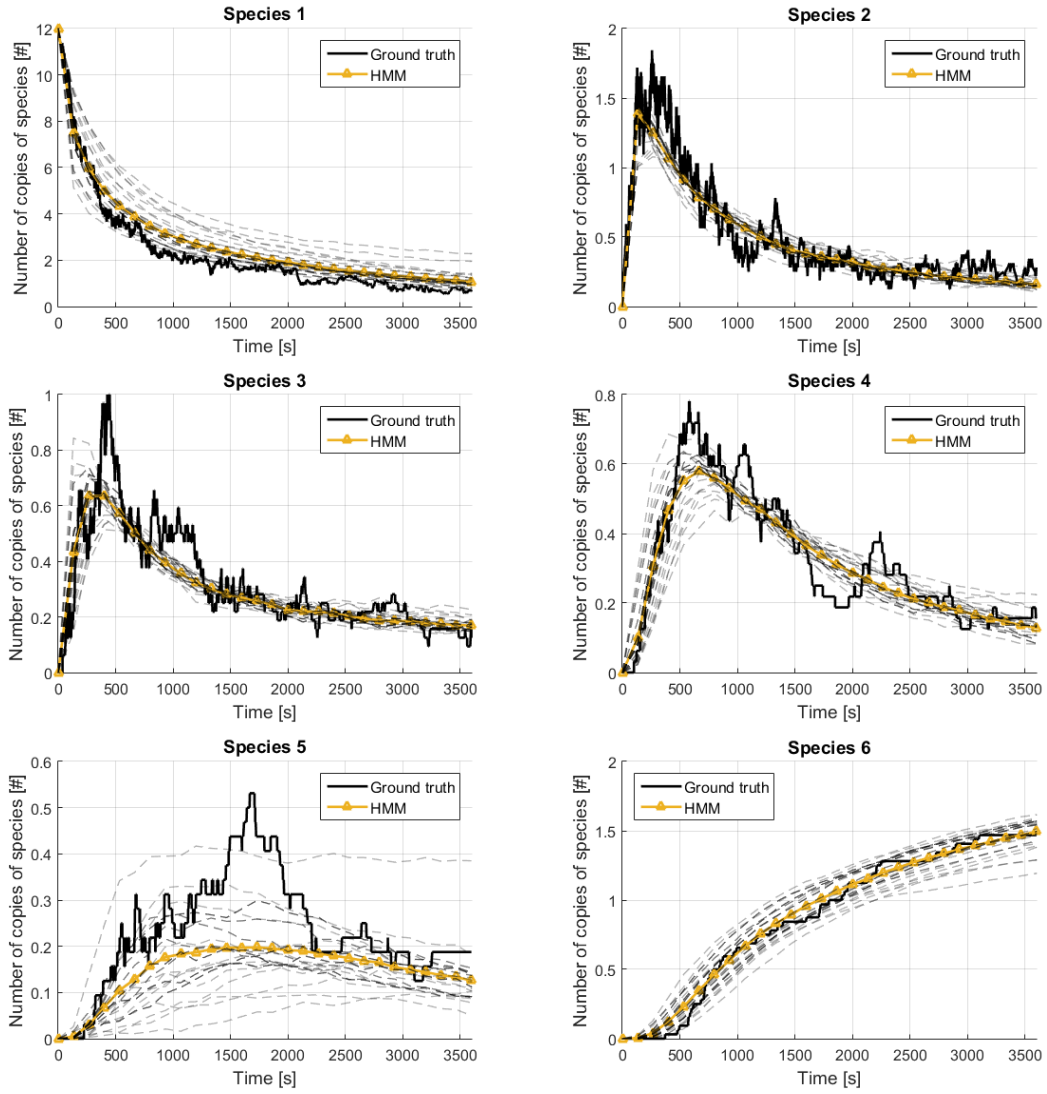


Figure 11.4 – Comparison of model prediction regarding the trajectory of each of the six original species formed by the ϕ_{chain} using the rates estimated by the Method 3 in Section 11.3 after the CRNs have been refined. The gray dashed curves are sample trajectories corresponding to different initial HMM states.

R_j:

$$R_j: \quad S_0 + 2\,A_1 \rightharpoonup 3\,A_2 \tag{11.13}$$

it will be duplicated as below:

$$R_j^a: \quad S_{0,a} + 2\,A_1 \rightharpoonup 3\,A_2 \tag{11.14}$$

$$R_j^b: \quad S_{0,b} + 2\,A_1 \rightharpoonup 3\,A_2. \tag{11.15}$$

In contrast to the approach proposed in [99], the resulting subspecies $S_{0,a}$ and $S_{0,b}$ are not specifically associated to the type of a previous interaction partner. More specifically, if an event $e_i = (R_i, t_i, \vec{x}_i)$ is such that R_i is duplicated, one needs to re-assign this event to either $R_i = R_i^a$ or $R_i = R_i^b$, and update accordingly the population vectors $\vec{x}_{j>i}$ of the upcoming events. To solve this problem, an expectation-maximization (EM) algorithm is used, an iterative method for finding maximum likelihood estimates of parameters in statistical models that depend on unobserved latent variables [98]. We introduce modifications in the original method in two ways: first, in order to automatically choose the species to be refined we propose a metric based on evidence from the observed set of events as below:

$$h = \frac{\sigma(\Delta t_i)}{m(\Delta t_i)} \tag{11.16}$$

Where for a reaction of interest, $\sigma(\Delta t_i)$ is the standard deviation of all the observed waiting times and $m(\Delta t_i)$ is the median of the observed waiting times. h is computed for all species in the CRN and the species with the maximum value of h will be refined next. Additionally, we introduce a stopping condition for the refinement procedure based on the same metric: the refinement process would thus be halted when the maximum h in the CRN is less than a predefined threshold t_h, set to 0.3 in our experiments. In order to keep the size of the CRN model tractable, here we perform one round of refinements starting from the CRN corresponding to the best model in Section 11.3, i.e. Method 3. In order to generate trajectories for the refined CRN, the original initial condition of fully isolated modules should be also refined to consider the initialization of the hidden species as well. Using the Stochkit software, we generate 1000 sample trajectories for each initial condition of the refined CRN corresponding to the HMM. This is a crucial part in evaluating the HMM accuracy which has not been addressed in previous works. The average trajectory of the HMM is then compared with the ground truth provided by high-fidelity simulations of the system as detailed in Chapter 9. The results are depicted in Figure 11.4. It can be seen that the model accuracy is significantly improved and the average trajectories match the ones of ground truth very well. Species 5 exhibits an interesting characteristics, the sample trajectories corresponding to different initial states differ largely compared to those of the other species, and the HMM does not manage to capture the ground truth trajectory peak. We speculate that a deeper refinement corresponding to an HMM with more hidden species in combination with a larger dataset should achieve better accuracy.

Summary

In this chapter, we considered the case of our fluid-mediated programmable self-assembling robotic system and investigated developing macroscopic probabilistic models of our system through multiple methods. A high-fidelity simulation of the system was used as the ground truth. We utilized the CRN formalism to express the self-assembly mechanism in the system. We show that assuming that the system is well-mixed and starting from different Markov models of the system, the hidden states augmented through our automatic HMM refinement method improve the model prediction accuracy, compensating the initial imprecise modeling assumptions.

12 Conclusion

THIS part of the dissertation covers the development of models and corresponding simulation tools for programmable self-assembling systems, forming the second main contribution of this thesis. In order to investigate the real system behavior, we develop models at different abstraction levels leveraging our experimental setup in several conducted studies. We estimated the model parameters for the submicroscopic level based on the observation of the real system and for the macroscopic level based on the realization of the submicroscopic model. Contrary to the convention in the literature, the aim of the microscopic model in this thesis was not to deliver quantitatively faithful predictions of the system behavior, but rather to provide a verification tool for the synthesized control strategies captured within the ruleset controllers, and therefore calibration based on one of the two underlying implementation levels was not considered.

The following transferable methods were developed in this part of the thesis:

- *Submicroscopic model, trajectory-based model calibration using PSO*- We presented a trajectory-based calibration method for capturing the dynamics of the fluidic field in our experimental setup that leverages a powerful metaheuristic method (PSO) for automatically optimizing model parameters. This method is particularly interesting as the resulting model, while being approximated, is computationally lightweight in comparison to models involving direct coupling between robotic and computational fluid dynamics simulators.

- *Microscopic model, extended graph grammar formalism*- While graph grammar formalism has been previously used for formulating models of programmable self-assembly, only bodiless modules were considered. We extended the graph grammar formalism such that the morphology of the robotic modules as well as the orientation of the bindings they form among each other are embedded in the extended graph representation. We then proposed a computationally light-weight shape recognition algorithm for efficiently identifying the type of sub-assemblies within an extended graph representation.

- *Macroscopic model, Markovian models based on ruleset structure-* We employed the structure of the ruleset controller embedded on the robotic modules s a blueprint for building Markovian models of the self-assembly process. This structure defines the ruleset strategy for building the desired target structure out of existing, initially isolated modules in the system. Consequently, the type of intermediate assembled structures is determined by the ruleset and their evolution over time can be directly observed, allowing for automatically extracting the model parameters. The choice of using a CRN formalism to formulate a Markovian model is well-aligned with the intrinsic nature of our programmable self-assembling system: the states of the CRN model correspond to the subassemblies in the system and its parameters, the reaction rates, correspond to their transformation times observed in the system. Markovian models can provide an accurate prediction of the system dynamics provided that the system is well mixed. To mitigate the influence of non-well-mixed conditions on the accuracy of the final model predictions, we employed an automatic technique to augment the model with additional refined states, eventually building an HMM. The overall automatic procedure that, starting from the ruleset controller is able to both generate an CRN-based Markov model and subsequently refine it to cope with hidden states resulting from non-well mixed conditions, is the final transferable method studied in this part of the dissertation.

Summary

This chapter concludes the modeling part of the thesis. The main outcome of the effort in this part is development of models at different abstraction levels for capturing the behavior of a programmable self-assembling system. Each model is coupled with a simulation tool, running the model for acquiring predictions. At the submicroscopic level, we present a novel method for calibrating computationally lightweight models capturing the dynamics of the fluid-mediated environment based on trajectory data of the building blocks in the system. At the microscopic level, we present an extended graph grammar formalism for capturing the dynamics of self-assembly of robotic modules and leverage this tool for evaluating the intrinsic performance of the deployed ruleset without taking into account the spatial effects mediated by the environment. At the macroscopic level, we investigate building Markovian models based on a description of the ruleset behavior of the robotic modules. In this concluding chapter, we briefly summarize the contributions of Part III and highlight the core methods which we believe one may apply to the development of models for programmable self-assembling systems of resource-constrained modules.

Part IV Controlling Self-Assembly

13 Introduction

DEPENDING on the capabilities of the self-assembly building blocks and the controlability of the environment, a range of fully distributed to fully centralized control approaches may be employed. The control approaches aim to guide the self-assembly process towards building the desired target structure, efficiently. The efficiency of a control approach can be measured considering two metrics, the rate and the yield of the controlled process. The assembly yield at each time is the number of copies of the target structure assembled at that time. The final yield is at a point in time when the experiment is halted. The assembly rate indicates the rate at which the process progresses towards building copies of the target structure. It can be measured as the assembly yield over time. In this chapter we briefly review the different control strategies that have been employed for guiding the process of self-assembly towards achieving the desired target structure, both in physical experimental systems and in abstract models of self-assembly. Moreover, we will outline our approach to controlling self-assembly.

13.1 Related Work

Programmable self-assembly has been demonstrated in [26], and [15] where active modules self-assemble into predefined desired 2D structures following a set of assembly rules. In [26], the miniaturized self-locomoted Kilobot robots coordinate using a deterministic and quasi-serial approach in a large swarm of 1000 robots. Module transportation may also be achieved by taking advantage of the stochastic ambient dynamics, realizing stochastic self-assembly. This in turn can allow for simplifying the modules' internal design. In [15], 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 (where each node of the graph is a bodiless module) and automatically generates proper rules for self-assembly of bodiless modules. The resulting rules are then manually tuned to suit the specific morphology of the physical robotic modules. An alternative method employed for generating

self-assembly rules is using powerful metaheuristic methods. For instance, employing the abstract Tile Assembly Model (aTAM) [102], [103], evolutionary computing has been used to generate rules for self-assembly in a system of passive modules in 2D [104], and a system of real and simulated passive modules in 3D [105]. The resulting 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 self-assembly of simulated underwater robotic modules in 3D in [97], where the authors manually define ruleset controllers specifically tailored to their robotic modules and the target structures.

The problem of ruleset synthesis for programmable self-assembly of graphs is first addressed in [106] 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 self-assembly 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 [94], the formalism of graph grammar is formally applied to the self-assembly 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 [94], 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, the authors in [92] employ a graph grammar formalism and show that the self-assembly 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, the authors in [107] leverage weighted graphs in a case study to encode the geometric orientations of the edges. While several formal rule synthesis algorithms leveraging graph grammars have been proposed for programmable self-assembly of graphs, their synthesized ruleset controllers are not directly applicable to self-assembly of robotic modules where orientation of the forming links determines the resulting assembled structures.

13.2 Problem Statement

As we explained int Chapter 1, one means of controlling the process of self-assembly is through proper design of the building blocks. In a system of intelligent building blocks, this translates to proper design of the intelligence - or equivalently the behavioral controllers - embedded in these building blocks. As outlined in the previous section, several algorithms for automatic synthesis of self-assembly rules have been proposed in the literature. These algorithms take as input a description of the desired target structure in the form of a graph and produce assembly rules for bodiless modules. While this automatic and general scheme in designing the assembly rules is very desirable, the rules produced by these algorithms can not be readily applied and programmed on robotic modules, limiting their usefulness in the case of practical instances of engineered self-assembly in real systems where the self-assembly building blocks are robotic modules with a certain physical embodiment. Hence, we recognize the problem of automatically synthesizing rules directly applicable to robotic modules as the core problem we tackle in this part.

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 employ the extended graph grammar formalism introduced in Chapter 10 where the concept of abstract graphs is extended by augmenting vertices with link slots, introducing extended vertices, where a link between two extended vertices can only form through specific slots. Additionally, this extension is coupled with 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. Hence, we model the self-assembling robotic system 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 that of $O(N^2)$ obtained by assigning one vertex and label per connector as presented in [15]. 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.

In Chapter 14, we leverage the extended formalism of Chapter 10 and extend the Singleton and Linchpin rule synthesis algorithms originally introduced [92], 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 involving two specific target

structures (i.e., a chain and a cross shape). 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 (submicroscopic and microscopic). In Chapter 15, we leverage the extended formalism of Chapter 10 once more and propose a novel rule synthesis algorithm for robotic modules which allows for a further parallelized assembly strategy than the one of the LinchpinR algorithm introduced in Chapter 14.

Summary

In this chapter, we introduced the problem of controlling the self-assembly process towards building specific desired target structures and reviewed the approaches investigated, both in physical self-assembling systems and in abstract models of self-assembly. In particular, we explained that a variety of control approaches may be employed depending on the capabilities of the building blocks and the controlability of the environment. While with passive building blocks only centralized control approaches are feasible, with intelligent building blocks one may employ a range of distributed control approaches possibly still combined with centralized ones. In particular, we are interested in extending and employing a method based on graph grammars to automatically synthesize self-assembly rules for intelligent building blocks.

14 Synthesizing Self-Assembly Rules

In this chapter, we focus on the problem of rule synthesis for programmable self-assembly of robotic modules. In a broad sense, the engineering question that motivates this effort can be formulated as follows: given a desired target structure composed of multiple robotic modules, how can we design the proper ruleset controllers to be deployed on the individual robotic modules? Our focus here is thus on formulating automatic rule synthesis algorithms for the self-assembly of robotic modules. In particular, we consider the specific but widely common case of rotationally symmetric robotic modules endowed with genderless latching connectors.

Leveraging the extended graph grammar formalism introduced in Chapter 10, we extend the Singleton and Linchpin rule synthesis algorithms from [92], obtaining the SingletonR and LinchpinR algorithms. Unlike the original Singleton and Linchpin, the two new extended algorithms are able to synthesize 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.

14.1 Extended Rules for Self-Assembly of Robotic Modules

Following our explanations of the notion of the extended graph in Chapter 10, here we provide a detailed definition of the extended rules. The extended rules are 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 10.1, 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 $l_1 \quad l_2 \rightharpoondown l_3 - l_4$, 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.

Corollary 1. 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. ■

14.2 Singleton and Linchpin for Self-Assembly of Bodiless Modules

In this Section, we briefly review the Singleton and Linchpin algorithms originally presented in [92]. For a given acyclic target graph, these algorithms synthesize rulesets for self-assembly of a system of bodiless modules represented as an abstract graph. The rulesets generated by both algorithms include reverse rules executed probabilistically in order to avoid deadlock

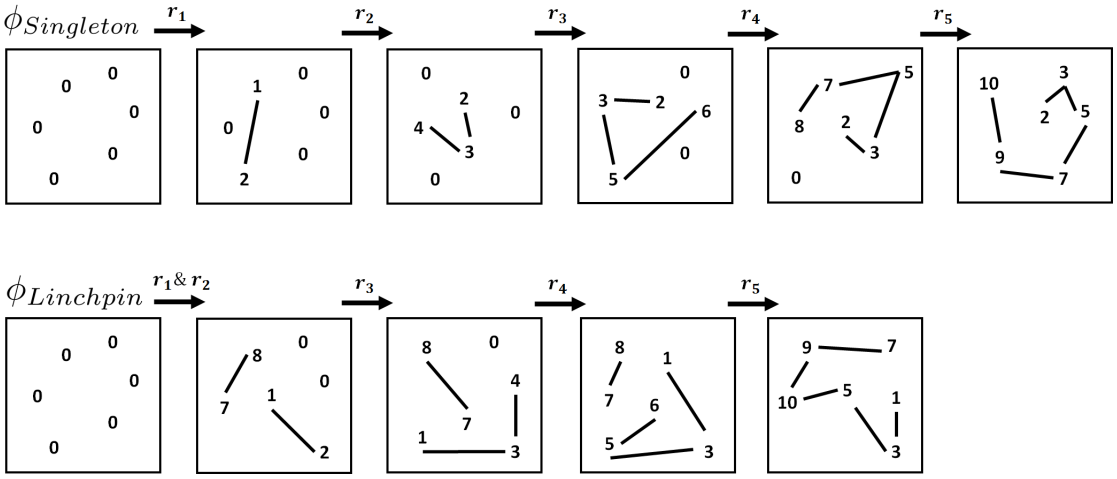


Figure 14.1 – Progress of the self-assembly process for a chain shape target graph as guided by the rulesets $\phi_{Singleton}$ and $\phi_{Linchpin}$.

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 self-assembly 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 [92]. 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 comprising six modules, $G = (V, E)$ where we have ($V = \{1,2,3,4,5,6\}$ and $E = \{12,23,34,45,56\}$), assuming vertex 2 as the root vertex fed to the algorithms in [92], 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:

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

Algorithm 3 Original Singleton algorithm for the self-assembly of bodiless modules as presented in [92].

```
1: Target graph: G = (V, E)
2: Root vertex: v_k
3: Initial label: l = 0
4: procedure SINGLETON(G, v_k, l)
5:     φ ← ∅
6:     if |n_E(k)| = 0 then
7:         return (l, φ)
8:     else
9:         {v_j : j = 1, 2, ..., |n_E(v_k)|} ← n_E(v_k)
10:        l̄ ← ł
11:        for j = 1 to |n_E(k)| do
12:            φ ← φ ∪ {l̄ 0 ⇌ (l+1) − (l+2)}
13:            l̄ ← ł + 1
14:            ł ← ł + 2
15:            Let (V^j, E^j) be the component
16:            of (V, E − {v_k v_j}) containing v_j
17:            G_j ← (V^j, E^j)
18:            (l_j, φ_j) ← SINGLETON(G_j, v_j, l)
19:            φ ← φ ∪ φ_j
20:            ł ← ł_j
21:        end for
22:    end if
23:    return (l, φ)
24: end procedure
```

Algorithm 4 SingletonR algorithm for the self-assembly of rotationally symmetric robotic modules obtained by applying our extended graph grammar formalism.

```
1: Target graph: G = (V, E, S, L)
2: Root vertex: v_k
3: procedure SINGLETONR(G, v_k)
4:     φ ← ∅
5:     if |n_E(k)| = 0 then
6:         return (l, φ)
7:     else
8:         {v_j : j = 1, 2, ..., |n_E(v_k)|} ← n_E(k)
9:         for j = 1 to |n_E(v_k)| do
10:            (s_k, s_j) ← S(v_k, v_j)
11:            l_k ← GVL(L, s_k, v_k)
12:            l_j ← GVL(L, s_j, v_j)
13:            l̄ ← INCREMENTSTATE(l, 1)
14:            l ← INCREMENTSTATE(l, 2)
15:            SVL(L, v_k, s_k, l̄)
16:            SVL(L, v_j, s_j, l)
17:            φ ← φ ∪ {l_k l_j ⇌ l̄ − l}
18:            Let (V^j, E^j) be the component
19:            of (V, E − {v_k v_j}) containing v_j
20:            G_j ← (V^j, E^j, S, L)
21:            (l, φ_j) ← SINGLETONR(G_j, v_j)
22:            φ ← φ ∪ φ_j
23:        end for
24:    end if
25:    return (l, φ)
26: end procedure

27: procedure GVL(L, s, v)
28:    (l_a, l_n) ← L(v)
29:    l_h ← (l_n − s + 1) (mod N)
30:    return (l_a, l_h)
31: end procedure

32: procedure SVL(L, v, s, l)
33:    (l_a, l_h) ← l
34:    l_n ← s
35:    L(v) ← (l_a, l_n)
36: end procedure

37: procedure INCREMENTSTATE(l, i)
38:    return (l_a + i, l_n)
39: end procedure
```

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 14.1. Note the difference in the natural course of the processes 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 achieve the target

Algorithm 5 Original Linchpin algorithm for the self-assembly of bodiless modules as presented in [92].

```
1: Target graph: G = (V, E)
2: Root vertex: v_k
3: Initial label: l = 0
4: procedure LINCHPIN(G, v_k, l)
5:     φ ← ∅
6:     for j = 1, 2, ..., |n_E(v_k)| do
7:         if |n_E(v_j)| ≥ 2 then
8:             Let (V^j, E^j) be the component
9:             of (V, E − {v_k v_j}) containing v_j
10:            (l_j, φ_j) ← LINCHPIN(G_j, v_k, l)
11:            l ← l_j
12:        else
13:            l_j ← 0
14:            φ_j ← ∅
15:        end if
16:    end for
17:    φ ← φ_1 ∪ {l_1 0 ⇌ (l+1) − (l+2)}
18:    l ← l + 2
19:    for j = 2, ..., |n_E(v_k)| do
20:        φ ← φ ∪ φ_j ∪ {l_j l ⇌ (l+1) − (l+2)}
21:        l ← l + 2
22:    end for
23:    return (l, φ)
24: end procedure
```

Algorithm 6 LinchpinR algorithm for the self-assembly of rotationally symmetric robotic modules obtained by applying our extended graph grammar formalism.

```
1: Target graph: G = (V, E, S, L)
2: Root vertex: v_k
3: procedure LINCHPINR(G, v_k)
4:     φ ← ∅
5:     for j = 1, 2, ..., |n_E(v_k)| do
6:         if |n_E(v_j)| ≥ 2 then
7:             Let (V^j, E^j) be the component
8:             of (V, E − {v_k v_j}) containing v_j
9:             (l_j, φ_j) ← LINCHPIN(G_j, v_k)
10:            l ← l_j
11:        else
12:            l_j ← 0
13:            φ_j ← ∅
14:        end if
15:    end for
16:    (s_k, s_1) ← S(v_k, v_1)
17:    l_k ← GVL(L, s_k, v_k)
18:    l_1 ← GVL(L, s_1, v_1)
19:    l̄ ← INCREMENTSTATE(l, 1)
20:    l ← INCREMENTSTATE(l, 2)
21:    SVL(L, v_k, s_k, l̄)
22:    SVL(L, v_1, s_1, l)
23:    φ ← φ_1 ∪ {l_1 0 ⇌ (l+1) − (l+2)}
24:    l ← l + 2
25:    for j = 2, ..., |n_E(v_k)| do
26:        (s_k, s_j) ← S(v_k, v_j)
27:        l_k ← GVL(L, s_k, v_k)
28:        l_j ← GVL(L, s_j, v_j)
29:        l̄ ← INCREMENTSTATE(l, 1)
30:        l ← INCREMENTSTATE(l, 2)
31:        SVL(L, v_k, s_k, l̄)
32:        SVL(L, v_j, s_j, l)
33:        φ ← φ ∪ φ_j ∪ {l_j l ⇌ (l+1) − (l+2)}
34:        l ← l + 2
35:    end for
36:    return (l, φ)
37: end procedure
```

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 self-assembly of both bodiless (see examples given in Section 14.2) and rotationally symmetric robotic modules (see examples given in Section 14.3).

14.3 SingletonR and LinchpinR for Self-Assembly of Robotic Modules

Algorithms 3 and 4 depict the pseudo codes of Singleton and SingletonR algorithms, while Algorithms 5 and 6 depict the pseudo codes of Linchpin and LinchpinR algorithms, respectively. SingletonR and LinchpinR essentially have the same structure as their abstract graph counterparts, Singleton and Linchpin, respectively. 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, that is 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 SINGLETON$((V_{\hat{G}}, E_{\hat{G}}, k, 0))$ for any $k \in V_{\hat{G}}$ generates a ruleset. The ruleset allows the self-assembly 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 link 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. It is the combination of these two values that provides a general description of the full internal state of a robotic module.

14.4 Synthesized Rulesets for 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 Lily robotic modules (see Chapter 5 for details), are reported below. The (l_a, l_h) notation is used for the relative extended labels expressed in the rules as explained in Section 10.2, and the reverse rules are separated.

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 self-assembly progress course using six Lily robotic modules is visualized in Figure 14.2. 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 14.2, 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-shape of Lily robotic modules. The reverse rules all have $l_h = 1$ at the LHS, indicating that the rule's

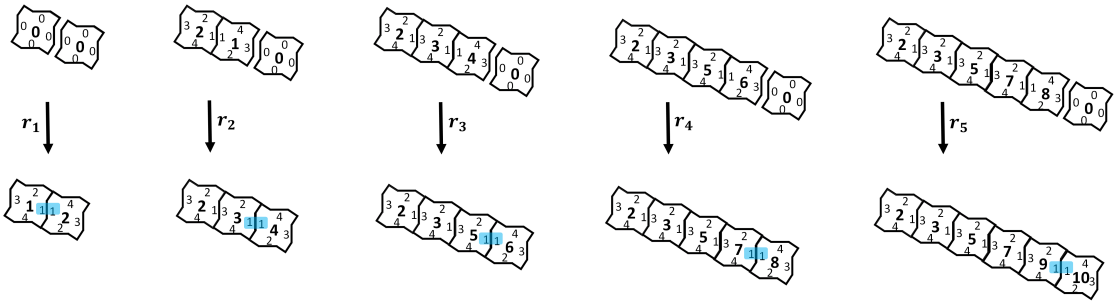


Figure 14.2 – Progress of the self-assembly process for the chain shape target structure employing ϕ^S_-. The latest engaged latching connectors on the modules are highlighted with a blue mark, while the relative hop numbering starting at the most recently engaged latching connector are shown on the sides of each module.

corresponding interaction happens at the link slot engaged the latest.

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

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

Consider the ϕ^L_+ whose self-assembly progress course using six Lily robotic modules is visualized in Figure 14.3. Each square represents a Lily, labeled with its internal state l_a value in the middle. The most recently 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 link slot and follows a CCW convention.

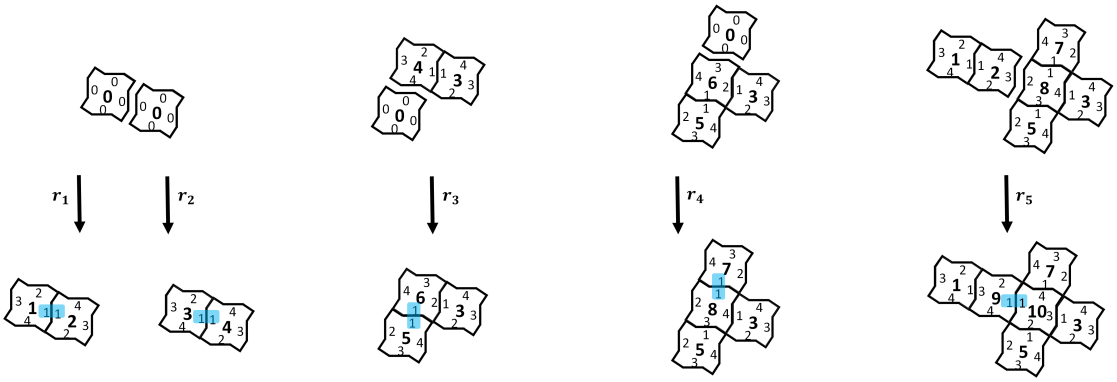


Figure 14.3 – Progress of the self-assembly process for the cross shape target structure employing ϕ_+^L. The latest engaged latching connectors on the modules are highlighted with a blue mark, while the relative hop numbering starting at the most recently engaged latching connector are shown on the sides of each module.

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.

14.5 Simulation Tools

In order to compare the performance of different rulesets synthesized by the SingletonR and LinchpinR algorithms for self-assembly of our Lily robotic modules and to study the transient system behavior corresponding to each of these ruleset controllers, we leverage two of the modeling levels and their corresponding simulation tools introduced in Part III: the microscopic model (Chapter 10) and the submicroscopic model (Chapter 9), 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 formally in Section 14.2 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 tool.

The submicroscopic modeling level and corresponding simulation tool (Webots), on the other

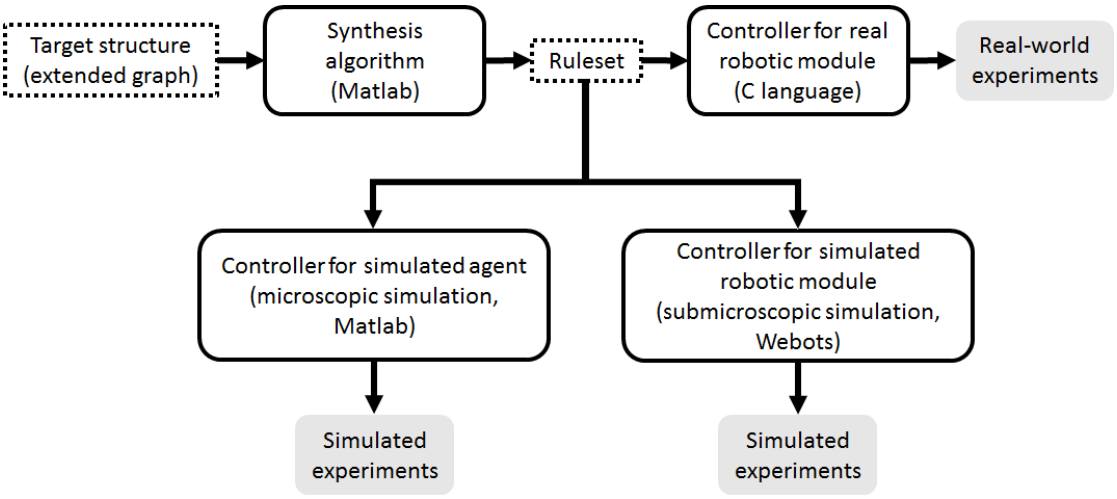


Figure 14.4 – Diagram of the overall software framework. The “synthesis algorithm" block may utilize different rule synthesis algorithms. In the current chapter SingletonR and LinchpinR synthesis algorithms are used.

hand, provides a realistic replication of the real experimental setup, faithfully capturing the physics of the self-assembly process in the system. This modeling level 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.

14.6 Experiments and results

We have conducted simulated experiments using the microscopic and submicroscopic simulation frameworks described in Section 14.5, as well as real-world experiments using the platform described in Part II. The performance of the rulesets synthesized by Singleton and Linchpin algorithms for self-assembly of bodiless modules have been comparatively studied in [92]. 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 self-assembly of Lily robotic modules. We expected to observe similar trends in the performance of the rulesets of SingletonR and LinchpinR compared to the ones

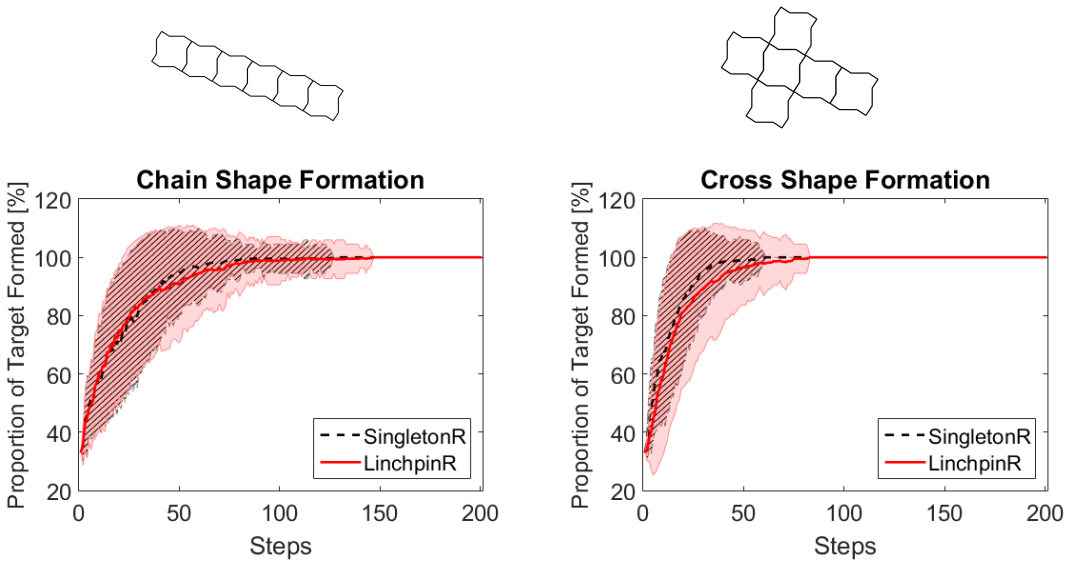


Figure 14.5 – Comparison of the rulesets synthesized by SingletonR and LinchpinR algorithms in the microscopic simulation using six Lily robotic modules for the two target shapes. The lines (dashed for SingletonR and continuous for LinchipR) and shaded regions (striped for SingletonR and uniform for LinchpinR) indicate the mean and standard deviation of 100 runs, respectively.

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 at the microscopic modeling level. In our second set of experiments, we employed the submicroscopic simulation tool to investigate the performance of the rulesets in more realistic conditions where the spatial aspects of the underlying self-assembly process are carefully modeled. Finally, we used physical Lily robotic modules to evaluate the rulesets performances in real world experiments.

Figure 14.4 depicts the structure of the overall software framework developed and employed in this work. Rulesets for self-assembly of Lily robotic modules are synthesized utilizing the SingletonR and LinchpinR algorithms in the synthesis algorithm block depicted in Figure 14.4 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 14.4, 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

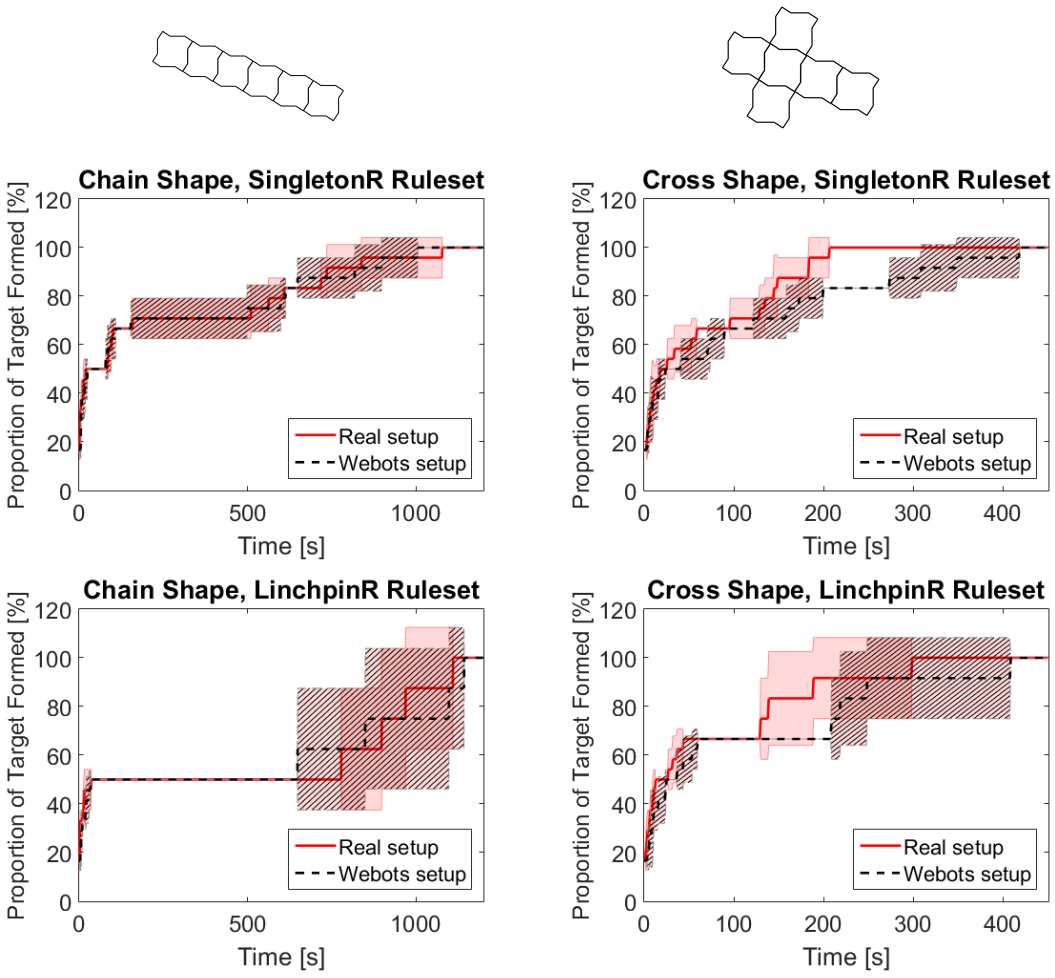


Figure 14.6 – Comparison of experimental results obtained with the real setup and the Webots setup using six Lily robotic modules for the two target shapes. The lines (dashed for the real and continuous for the Webots setup) and shaded regions (striped for the real and uniform for the Webots setup) indicate the mean and standard deviation of five runs, respectively.

on optimization of ruleset controllers for programmable self-assembling systems [108], [65]. However, with the focus of our current work being on automatizing rule 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 advanced 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. Regarding reverse rules probabilities, 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

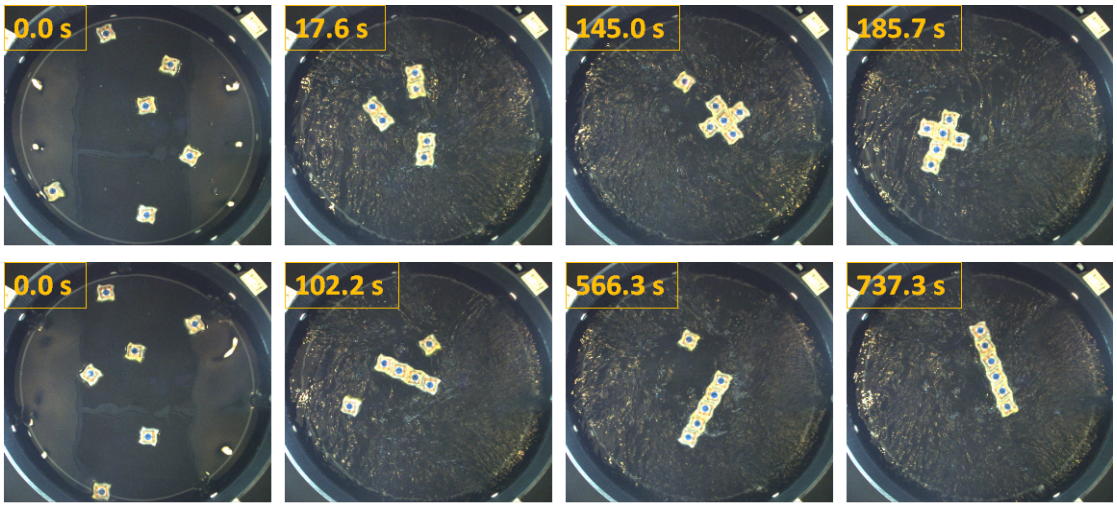


Figure 14.7 – Snapshots of the self-assembly process employing SingletonR rulesets for the cross shape target (arranged in the top row) and chain shape target (arranged in the bottom row) and using six Lily robotic modules.

comparison in [92]. We use the same statistical indicators due to roughly symmetric quasi-Gaussian distribution of our data for all the plots in this chapter. Figure 14.5 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 14.8 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 the self-assembly process is delayed until a dimer disassembles and reassembles with the required labeling.

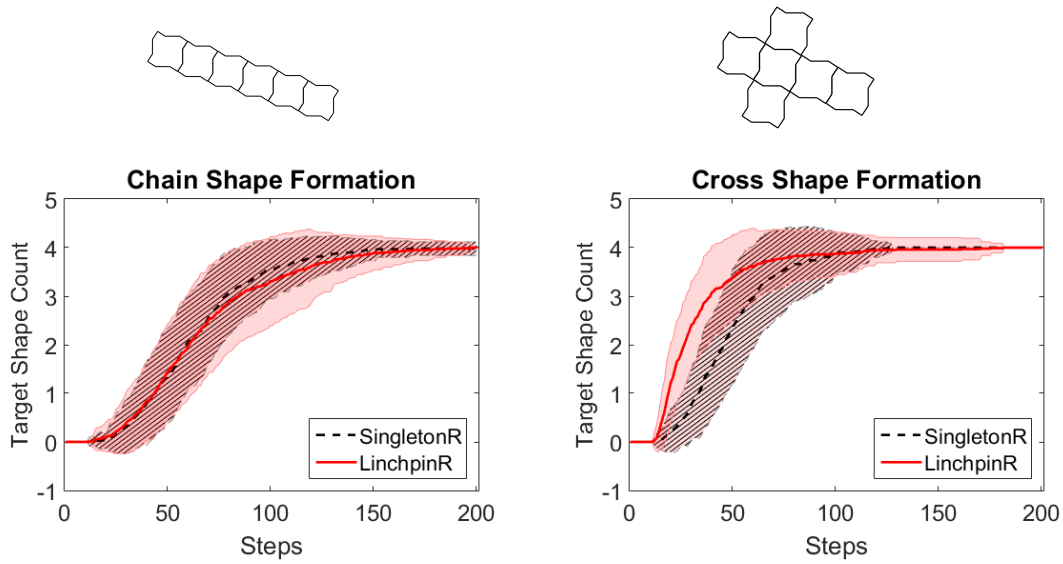


Figure 14.8 – Microscopic simulation results of rulesets derived by the two extended synthesis algorithms for the two target structures of chain and cross shape. The lines (dashed for SingletonR and continuous for LinchipR) and shaded regions (striped for SingletonR and uniform for LinchpinR) summarize the mean and standard deviation of 100 runs, respectively.

Real-world experiments were conducted by programming six Lily robotic modules with the four derived rulesets to build the two target structures. Each experiment was repeated five times. The same experiments were also conducted using the submicroscopic simulation tool, each repeated five times, to provide a direct comparison between the simulated and the real-world setups. Figure 14.6 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 14.6. These results roughly indicate a good matching between the two setups. Table 14.1 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 submicroscopic simulation, 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 14.1 results and horizontal width of the blue region in Figure 14.6, left column). LinchpinR 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

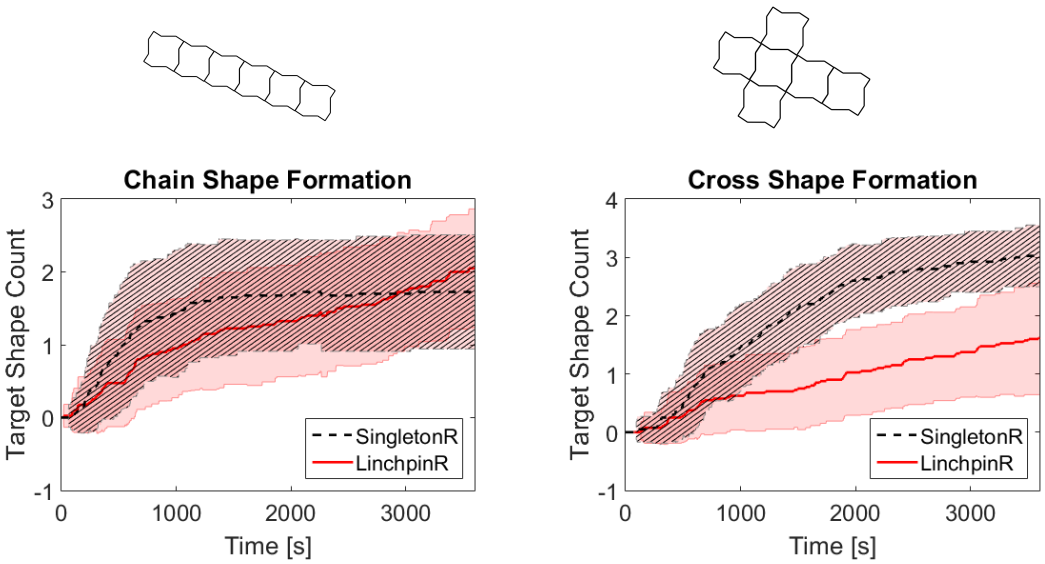


Figure 14.9 – Submicroscopic simulation results of the performance of rulesets derived by the two extended synthesis algorithms for the two target structures of chain and cross shape. The lines (dashed for SingletonR and continuous for LinchipR) and shaded regions (striped for SingletonR and uniform for LinchpinR) summarize the mean and standard deviation of 50 runs, respectively.

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 14.7 depicts the progress of the self-assembly process for the two target structures deploying the SingletonR ruleset on the Lily robotic modules. 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.

Table 14.1 – 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

Figure 14.8 shows the performance of the rulesets derived by the two extended synthesis algorithms for the two target assemblies in the 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 14.9 shows the submicroscopic simulation results of the derived rulesets using 24 available modules all initially isolated. These findings further confirm 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.

Summary

In this chapter, we addressed the problem of rule synthesis for programmable self-assembly of rotationally symmetric robotic modules endowed with genderless latching connectors. Exploiting the extended graph grammar formalism introduced in Chapter 10, we introduce extended versions of two automatic rule synthesis algorithms from the literature which are capable of synthesizing rules directly applicable on robotic modules. Moreover, we provide the proof that rule synthesis algorithms based on the extended graph grammar formalism achieve rulesets of lower complexities than the existing ones in the literature. We then focus on two case studies in our system concerned with the self-assembly of cross and chain shape targets. The self-assembly 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, which regulated the outcome of the random interactions between two robotic modules based on their internal states.

15 Synthesizing Parallel Rules

In this chapter, we take a step further along the line of synthesizing ruleset controllers for self-assembly of robotic modules and introduce an automatic rule synthesis algorithm which allows for a higher level of parallelization in the assembly strategy than that of the LinchpinR algorithm introduced in the previous chapter. The main idea is to break down the assembly process into a number of stages. At each each stage subassemblies of a certain size are built. These subassemblies eventually join and build the final desired target structure.

15.1 GS-RGS: A New Synthesis Algorithm

Given an acyclic target structure composed of rotationally symmetrical robotic modules with any number of connectors, our proposed synthesis algorithm derives rulesets based on two principles: 1) limiting the size of the concurrently built sub-assemblies to a user-defined value, and 2) unifying the rules which give rise to sub-assemblies with similar structures. The algorithm comprises two stages, each realizing one of the two principles. The first stage parses a graphical description of the target, and derives a ruleset which builds the target by merging sub-assemblies with sizes no more than the user-defined value and with distinct labelings, as a result of employing distinct rules. The second stage then processes this ruleset to identify the rules producing structures with identical morphology; such rules are then merged in a single one. As a result of the second principle, i.e. unifying the rules and consequently the labelings, the rulesets need to include update rules. Consider the case where the maximum user-defined size is two. With the rules unified properly, all dimers (sub-assemblies composed of two modules) are labeled similarly. As the dimers join to build the target, the labelings of both consisting modules need to be updated to reflect their placement in the forming target structure to allow for proper further reactions, completing the target eventually. In other words, the use of update rules is an alternative to building the target out of distinctly labeled sub-assemblies, as a result of being formed through distinct link rules, according to their intended placement in the target structure. Thus in general, introducing update

rules into the rulesets can reduce the number of link rules necessary to build the target, at the expense of possibly increasing the total ruleset size to include several update rules. However, this can offer a significant advantage in terms of assembly time. The occurrence of the link rules is probabilistic and is determined by the stochastic nature and dynamics of the system which is relied upon to provide proper interactions in order for the self-assembly process to progress. The occurrence rate for link rules is usually in the order of once in tens of seconds. On the other hand, update rules are purely communicational rules and do not depend on the system dynamics. Once a proper interaction has happened and two modules have bonded successfully, the occurrence of a proper update rule is solely determined by the modules communication rate, usually in the order of once in less than a second. Fewer link rules can thus significantly decrease the total assembly time. The first principle addresses the propagation delay concerns which can cause scalability issues. Limiting the size of the concurrently built sub-assemblies allows for restricting the extent by which the update rules need to propagate. Therefore, in a robotic system with measurable propagation delay and interaction rate, the update propagation depth can be set accordingly to allow for a parallel assembly scheme while minimizing possible propagation delay faults. In order to avoid deadlocks, we employ probabilistic dissociating rules. Appropriate reverse rules are generated at the end of the second stage. It should be noted that the algorithm only generates the ruleset. Appropriate probabilities should be assigned to the rules to reliably build the target while avoiding deadlocks.

15.1.1 Stage I: Grow Subtrees (GS)

Stage I allows for creating concurrently built sub-assemblies similar to the concurrency created by the LinchpinR algorithm [92], with the additional capability to control the maximum permitted size of such sub-assemblies. The GS algorithm employed in Stage I tries to build a given target structure using as many sub-assemblies of a defined size as possible built in parallel, before trying to join them to make a bigger sub-assembly and eventually form the target. In principle, the algorithm addresses the second issue with the LinchpinR algorithm (see Chapter 14). LinchpinR generates rules to build parallel substructures for every branch split in the target recursively in order to build the target using one final finishing rule. With one finishing rule in the ruleset, it is shown in [92] that the target can be built reliably while avoiding deadlocks by having probabilisitic dissociating rules for all rules except for the finishing rule. The GS algorithm on the other hand, permits a maximum size for the concurrently built sub-assemblies and as a result may end up building the target using several concurrent finishing rules. Stage II then processes this ruleset and results in one finishing rule. Algorithm 1 shows the pseudo code of the GS algorithm. The first call to GS is by $Size = 0$. The algorithm then recursively proceeds to create extended labels and corresponding rules, moving outwards from a starting vertex k. The ruleset returned by GS for a chain structure of size 6 and maximum sub-assembly size of 2, is depicted below along with the link rules generated by LinchpinR. In order to simplify the comparison, the rulesets have been designed for an abstract graph.

Algorithm 7 Pseudo code of the GS algorithm employed in Stage I.

```
1: C : (V, E, S, L, k, l, Size, S_max)
2: procedure GS(C)
3:     φ ← ∅
4:     φ̄ ← ∅
5:     Size ← Size + 1
6:     if |n_E(k)| = 0 then
7:         return (l, φ)
8:     else
9:         {v_j : j = 1, 2, ..., |n_E(k)|} ← n_E(k)
10:        for j = 1 to |n_E(k)| do
11:            s_k ← S(v_k, v_j)
12:            s_j ← S(v_j, v_k)
13:            l_k ← GVL(L, s_k, v_k)
14:            if Size < S_max then
15:                l_j ← GVL(L, s_j, v_j)
16:                l̄ ← INCREMENTSTATE(l, 1)
17:                l ← INCREMENTSTATE(l, 2)
18:                φ̄ ← φ ∪ {l_k l_j ⇌ l̄ − l}
19:                SVL(L, v_k, s_k, l̄)
20:                SVL(L, v_j, s_j, l)
21:                Size_j ← Size
22:            else
23:                φ̄ ← ∅
24:                Size_j ← 0
25:            end if
26:            Let (V^j, E^j, S^j) be the
       component of (V, E − {k v_j}) containing v_j
27:            C : (V^j, E^j, S, L, v_j, l, Size_j, S_max)
28:            (l, φ_j, Size_j) ← GS(C)
29:            φ ← φ ∪ φ̄ ∪ φ_j
30:            if Size == S_max then
31:                l_j ← GVL(L, s_j, v_j)
32:                l̄ ← INCREMENTSTATE(l, 1)
33:                l ← INCREMENTSTATE(l, 2)
34:                φ ← φ ∪ {l_k l_j ⇌ l̄ − l}
35:            else
36:                Size ← Size_j
37:            end if
38:        end for
39:    end if
40:    return (l, φ, Size)
41: end procedure

42: procedure GVL(L, s, v)
43:     (l_a, l_n) ← L(v)
44:     l_h ← (l_n − s + 1) (mod N)
45:     return (l_a, l_h)
46: end procedure

47: procedure SVL(L, v, s, l)
48:     (l_a, l_h) ← l(1 : 2)
49:     l_n ← s
50:     L(v) ← (l_a, l_n)
51: end procedure

52: procedure INCREMENTSTATE(l, k)
53:     return (l_a + k, l_n)
54: end procedure
```

$$\phi_{GS} = \begin{cases} 0\ 0 \rightarrow 1-2 & (r_1) \\ 0\ 0 \rightarrow 3-4 & (r_2) \\ 0\ 0 \rightarrow 5-6 & (r_3) \\ 4\ 5 \rightarrow 7-8 & (r_4) \\ 2\ 3 \rightarrow 9-10 & (r_5) \end{cases} \qquad \phi_{LinchpinR} = \begin{cases} 0\ 0 \rightarrow 1-2 & (r1) \\ 0\ 0 \rightarrow 3-4 & (r2) \\ 0\ 2 \rightarrow 5-6 & (r3) \\ 0\ 4 \rightarrow 7-8 & (r4) \\ 5\ 7 \rightarrow 9-10 & (r5) \end{cases}$$

Consider a set of initially isolated atomic agents, all labeled 0. The rules generated by GS allow for the target to be built in two concurrent steps, i.e. first (r_1, r_2, r_3) and then (r_4, r_5), while the rules synthesized by LinchpinR require three concurrent steps, i.e. first (r_1, r_2), then (r_3, r_4), and eventually (r_5).

15.1.2 Stage II: Re-Group Subtrees (RGS)

Stage II processes the ruleset generated by Stage I to unify the link rules which create up to the maximum size sub-assemblies and add proper update rules. The key idea of processing is to apply the rules synthesized by GS to two graphs with initially fully isolated vertices. The two graphs evolve identically in structure but differ in labeling, one graph is labeled according to the original ruleset, while in the other graph the forming sub-assemblies are processed to identify the rules with products of identical shapes. In order to identify structures with

identical shapes the shape recognition algorithm explained in Chapter 10 is utilized. The RGS algorithm can be explained in four phases:

Phase I: Forming dimers Unify the rules (of Stage I) which form dimers.

Phase II: Forming larger sub-assemblies Grow on the dimers. Recognize the shape of the resulting sub-assemblies. Unify the rules producing identical structures.

Phase III: Relabeling max-size sub-assemblies Create update rules for relabeling all the modules in sub-assemblies of up to the max-size (i.e. user-defined value) size.

Phase IV: Growing on max-size sub-assemblies Create necessary rules, both link and update, to form the target assembly out of the max-size sub-assemblies.

Considering the target shape of a chain of size six for an abstract graph, the rules generated by RGS are as below:

$$\phi_{RGS} = \begin{cases} 0 \quad 0 \quad \rightarrow \quad 1-2 \quad (r_1) \\ 1 \quad 2 \quad \rightarrow \quad 4-3 \quad (r_2) \\ 1 \quad 8 \quad \rightarrow \quad 10-9 \quad (r_3) \\ 1-3 \quad \rightarrow \quad 6-5 \quad (r_4^u) \\ 2-4 \quad \rightarrow \quad 8-7 \quad (r_5^u) \\ 7-9 \quad \rightarrow \quad 12-11 \quad (r_6^u) \\ 2-10 \quad \rightarrow \quad 14-13 \quad (r_7^u) \end{cases}$$

Two points are noteworthy here. First, the existence of the update rules in the rule-set (r_4^u to r_7^u). And second, the number of concurrent steps necessary for forming the target being equal to three. Assuming that the update events are instantaneous and that the number of available modules is limited, RGS can on average build the target faster than LinchpinR with the same number of necessary concurrent steps (see Section 15.3). This is due to the fact that RGS makes a better use of the available modules by limiting the number of distinctly labeled sub-assemblies with identical shapes. At the end of Stage II, the ruleset is augmented with proper reverse rules accounting for both the link and the update rules. The application of a reverse rule essentially takes the self-assembly process back in time by reversing the labeling and/or the bonding.

Corollary 2. The complete ruleset ϕ_{full} generated by our proposed method for assembling a target structure described as an extended graph $G = (V, E, S, l)$ will eventually achieve the maximum possible number of copies of the target structure (i.e. maximum yield) provided that the available assembly modules executing the ruleset interact often enough and that the corresponding execution probability is set to $p = 1$ for link and update rules and to $p < 1$ for reverse rules.

Proof. The ruleset ϕ_{full} contains an unlink rule for each link and update rule. Only the last link rule has no corresponding reversal rule. Therefore, while all partially formed structures dis-assemble with a non-zero probability, the finishing rule is reversed with zero probability, therefore leading to a stable target structure. ■

Corollary 3. The complete ruleset ϕ_{full} generated by our proposed method for assembling a target structure described as an extended graph $G = (V, E, S, l)$ will achieve an assembly rate at least as fast as that of a ruleset derived by the LinchpinR algorithm, assuming that the update rules are applied instantaneously.

Proof. Rulesets generated by Stage I have as many link rules as the ones generated by the LinchpinR algorithm, i.e. the number of edges in the target graph, as a result of forming the target out of uniquely labeled sub-assemblies. As a result of merging the link rules in Stage II, ϕ_{full} contains at most as many link rules as the ruleset ϕ_{GS} created in Stage I. Therefore, ϕ_{full} achieves the target structure in the same or fewer concurrent steps than LinchpinR rulesets. ■

15.2 Synthesized Rulesets for Lily Robotic Modules

We consider two targets, a chain and a cross structure, each composed of six Lily robots (see Fig. 15.1). The rulesets returned by our algorithm (with the maximum user-defined size set to 2) for the chain structure ϕ_-, and for the cross structure ϕ_+, are reported below. The (l_a, l_h) notation is used for the relative extended labels and the reverse rules are separated. Note that the reverse rules do not correspond to a single link or update rule, but rather have a time reversal effect, taking the labeling back in time.

$$\phi_- = \begin{cases} (0,0)\ (0,0) \xrightarrow{r_1} (1,1)-(2,1) \\ (1,3)\ (2,3) \xrightarrow{r_2} (4,1)-(3,1) \\ (1,3)\ (8,3) \xrightarrow{r_3} (10,1)-(9,1) \\ (1,1)-(3,3) \xrightarrow{r_4^u} (6,1)-(5,1) \\ (2,1)-(4,3) \xrightarrow{r_5^u} (8,1)-(7,1) \\ (7,1)-(9,3) \xrightarrow{r_6^u} (12,1)-(11,1) \\ (2,1)-(10,3) \xrightarrow{r_7^u} (14,1)-(13,1) \\ (1,1)-(2,1) \xrightarrow{\bar{r}_1} (0,0)\ (0,0) \\ (5,3)-(7,3) \xrightarrow{\bar{r}_{2/4/5}} (3,1)\ (4,1) \\ (3,3)-(6,1) \xrightarrow{\bar{r}_{2/4}} (2,1)-(1,1) \\ (4,3)-(8,1) \xrightarrow{\bar{r}_{2/5}} (1,1)-(2,1) \end{cases}$$

$$\phi_+ = \begin{cases} (0,0)\ (0,0) \xrightarrow{r_1} (1,1)-(2,1) \\ (0,0)\ (2,4) \xrightarrow{r_2} (4,1)-(3,1) \\ (0,0)\ (5,2) \xrightarrow{r_3} (8,1)-(7,1) \\ (2,3)\ (7,2) \xrightarrow{r_4} (10,1)-(9,1) \\ (1,1)-(3,2) \xrightarrow{r_5^u} (6,1)-(5,1) \\ (1,1)-(10,3) \xrightarrow{r_6^u} (12,1)-(11,1) \\ (1,1)-(2,1) \xrightarrow{\bar{r}_1} (0,0)\ (0,0) \\ (4,1)-(5,4) \xrightarrow{\bar{r}_2} (0,0)\ (3,1) \\ (6,1)-(3,2) \xrightarrow{\bar{r}_{2/5}} (1,1)-(2,1) \\ (7,1)-(8,1) \xrightarrow{\bar{r}_3} (5,3)\ (0,0) \end{cases}$$

15.3 Experiments and Results

We evaluate our algorithm leveraging the two modeling levels and corresponding simulation tools of Part III, studying the self-assembly process in a swarm of 24 initially isolated Lily robotic modules. Experiments using the real setup are additionally planned to be carried out. Two targets of chain and cross shapes, each composed of six robotic modules, are considered. A maximum of four copies of each target can be assembled thus. The microscopic model employs a random pairwise interaction dynamics as described in Chapter 10. All interactions among microscopic nodes are set to be equiprobable, i.e. we make the assumption that the

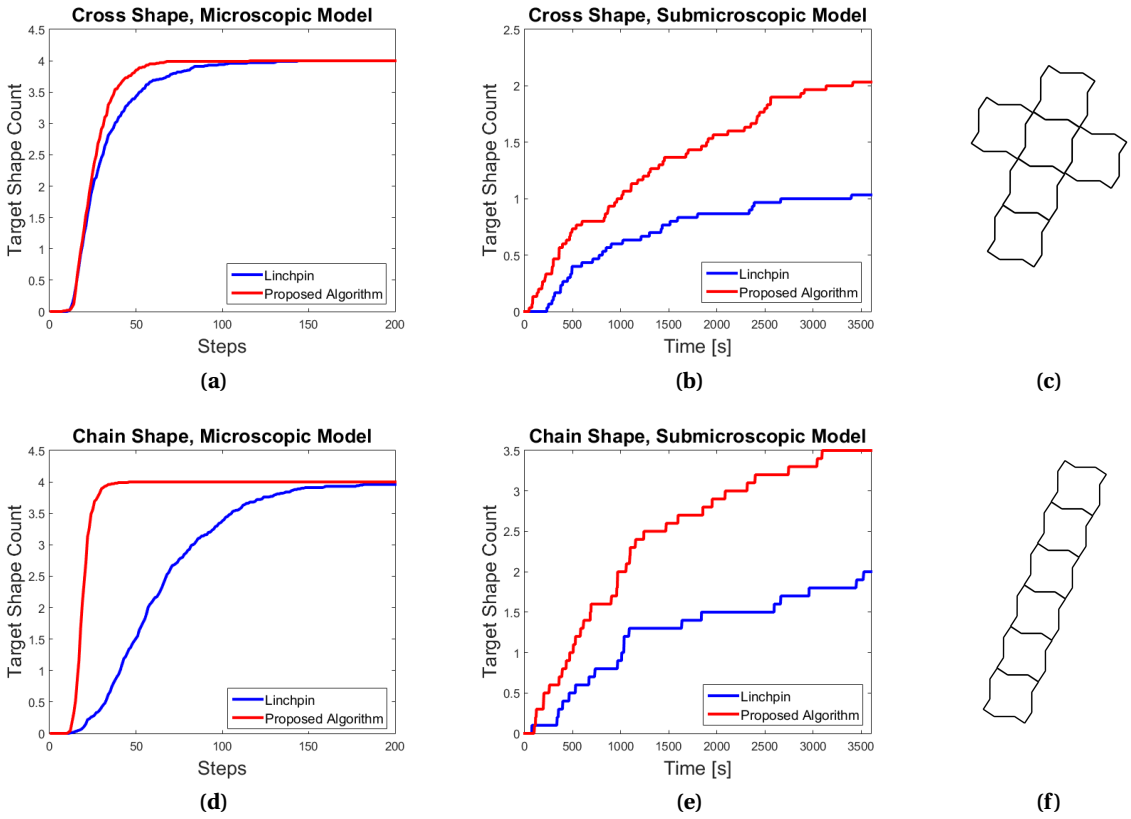


Figure 15.1 – Results of microscopic ((a) and (d), 100 runs averaged) and submicroscopic models ((b) and (e), 30 runs averaged) for two target shapes, cross (c) and chain (f).

system is perfectly mixed. We employ rulesets synthesized by our algorithm and the extended LinchpinR algorithm, for our simulated Lily robotic modules. For forward and update rules $P(.) = 1$ and for reverse rules $P(.) = 0.01$ is set. The finishing rule is set to be irreversible in all the rulesets, giving rise to stable target assemblies once they are formed. Figure 15.1 depicts the performance of the rulesets derived by the GS-RGS algorithm along with the ones of the extended LinchpinR for the two target shapes. While for the submicroscopic simulations the results are reported as a function of the experimental time (emulating the real time progress in a real experiment), the results of the microscopic simulations are reported as a function of steps, each step representing a formation event in the system. While such choice makes the results of the two modeling levels not directly comparable, the adopted progress unit is well suited for measuring the concurrency of the rulesets. It can be seen that the GS-RGS algorithm achieves higher assembly rates in all cases. Interestingly, the maximum yield of four is not obtained in the case of the submicroscopic simulations within the one hour simulated time. This can be ascribed to several reasons. First, it is observed in the submicroscopic simulation that larger structures with several corners trap other sub-assemblies and stall the self-assembly process. In addition, as the structures grow the conditions diverge from perfect mixing since the shape of the sub-assemblies affect their orientation in the fluidic field and certain interactions tend to be less probable.

Summary

In this chapter, we addressed the problem of synthesizing rules for programmable self-assembly of rotationally symmetric robotic modules endowed with genderless latching connectors, such that further parallelism in the assembly scheme is induced. More specifically, we focused on a case study involving our Lily robotic modules and employed the two submicroscopic and microscopic simulation frameworks introduced in Part III to compare the performance of the rulesets synthesized by the GS-RGS algorithm with those synthesized by LinchpinR. Results show that for both studied target shapes of chain and cross structures, rulesets synthesized by GS-RGS achieve higher assembly rates compared with those synthesized by LinchpinR.

16 Conclusion

In this part of the thesis, we investigated the synthesis of ruleset controllers for programmable self-assembly of robotic modules. In particular, we employed the extended graph grammar formalism introduced in Part III and developed automatic rule synthesis methods for programmable self-assembly of robotic modules. We first showed how the extended graph grammar formalism can be used to extend and transform existing rule synthesis algorithms for bodiless modules into algorithms which synthesize rules which are directly applicable to robotic modules. Moreover, we proposed the new rule synthesis algorithm GS-RGS which allows for further parallelization in the assembly scheme, resulting in increased assembly rates.

We highlight the following as the transferable methods developed in this part of the thesis:

- *Rules directly applicable to robotic modules* - Employing the extended graph grammar formalism, one can formulate automatic rule synthesis algorithms which consider the morphology of the robotic modules as they synthesize rules for the self-assembly process to build a predefined target structure comprising several robotic modules. This allows for eliminating the posterior tuning step which was previously necessary when the rules where actually synthesized for bodiless modules, as reported in the existing literature.

- *Lower ruleset complexity using the extended grammar* - We provided the proof that the rule synthesis algorithms employing the extended graph grammar formalism achieve a lower ruleset complexity, $O(N)$ as opposed to $O(N^2)$ reported in the literature. This lower complexity is due to the particular representation of the internal states of the robotic modules in the extended graph used as a model for deriving rules which control the course of the assembly process.

- *Parallel rulesets, higher assembly rate* - In general, one may take advantage of the symmetries in the target structure to break down the assembly process into a number of assembly stages. At each stage, several sub-assemblies can be built in parallel, eventu-

ally joining together to build the product of that stage. The succession of these stages eventually culminates in building the target structure. The proposed GS-RGS algorithm automatically generates rules able to leverage such decomposition in assembly stages.

Summary

This chapter concludes the control part of the thesis. The main outcome of the effort in this part are automatic rule synthesis algorithms which, based on a description of the desired target structure and the morphology of the robotic modules, synthesize self-assembly rules that can be directly applied to the robotic modules without the need for any further tuning or adjustment. This is achieved due to the fact that the model based on which the rules are synthesized explicitly captures the morphology of the robotic modules in the form of an extended graph, described in detail in Part III. Moreover, we introduced the GS-RGS rule synthesis algorithm that automatically creates assembly rules capable of obtaining higher assembly rates through allowing for higher concurrency in its assembly strategy. In this concluding chapter, we briefly summarize the contributions of Part IV and highlight the core methods and techniques which we believe one may apply to the development of control strategies for programmable self-assembling systems of resource constrained robotic modules with a rotationally symmetric morphology.

Conclusion Part V

17 Conclusion and Outlook

We have in this dissertation investigated the programmable self-assembly of resource-constrained robotic modules in 2D within a fluidic environment. The conducted research effort was guided by two main thrusts: (i) a technological research thrust, concerning the mechatronic development of the experimental self-assembly platform around a resource-constrained robotic module, and (ii) a methodological research thrust concerning building appropriate models for capturing the dynamics of programmable self-assembly as well as development and evaluation of different control approaches for the self-assembly process carried out by the robotic modules. Having a wide variety of applications, engineered programmable self-assembly is envisioned to be employed in numerous scientific domains and within systems with a variety of length-scales. For all the three aspects studied in this work, i.e. the mechatronic design, the modeling, and the control, we have intentionally made an effort to develop and employ techniques that extend beyond the experimental system used in our studies. It is our hope that the design, modeling, and control methods and principles explored in this work will provide a basis for researchers not only in the robotics community but also all the various fields that are currently investing in the development of engineered self-assembling systems.

17.1 Summary of Contributions

Our work in this dissertation provides core contributions along the lines of the two main research thrusts, with each main contribution explained in a dedicated part of the manuscript.

Part II of this manuscript describes the mechatronic design and development of our self-assembling robotic platform. We summarize our contributions as follows:

- We developed the 3-cm-sized Lily robotic modules as the building blocks in our programmable self-assembling robotic system. Lilies employ four custom-designed electropermanent magnetic latches which are also used as an inductive channel for local communication with their neighboring robotic modules. They employ a radio link for

communicating with a base station to receive new firmware or specific commands during an experiment. All the operations regarding charging, programming, and switching on/off the robotic modules have been designed to be scalable for swarms of tens of Lilies. As a result, the Lily robotic module captures a specific set of features which have been previously only partially demonstrated in various modules deployed in self-assembling systems, thus making it unique in the field.

- We developed the controllable fluidic experimental setup around the Lily robotic modules. The setup allows for controlling the ambient flow field driving the mixing in the fluidic arena as well as the ambient luminosity which can be perceived by the robotic modules. Through several experiments we fully characterized the system.
- We explicitly motivated the different design objectives and the final design choices in the process of developing the experimental robotic platform.

Part III of this manuscript describes the modeling effort in order to capture the dynamics of programmable self-assembly of robotic modules at multiple abstraction levels. We summarize our contributions as follows:

- At the submicroscopic modeling level, in order to faithfully recreate our self-assembling system in simulation, we used Webots, a physics-based robotics simulator, and developed a dedicated physics plugin software capable of recreating the fluidic force field of our experimental arena as well as a dedicated calibration method leveraging a PSO algorithm. The simulated world was then calibrated based on real experimental data. 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.
- At the microscopic modeling level, we captured the self-assembling system using an extended graph grammar description of the underlying process. For this, we first introduced the extended graph grammar formalism. This extended formalism allows for directly incorporating the morphology of the robotic modules in the graph structure by utilizing the notion of extended graphs that comprise extended vertices with ordered link slots representing the robotic modules' latching connectors. The microscopic model and its corresponding simulation tool developed in Matlab allow for evaluation and comparison of the intrinsic performance of different ruleset controllers, based on precise metrics such as 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.
- At the macroscopic modeling level, we employed the CRN formalism to express a Markovian model of our system by using the ruleset controllers structure as a blueprint for

determining the structure of the CRN and its corresponding Markov model. Moreover, we introduced a new rate estimation method for the CRN model and compared it with two existing ones in the literature, demonstrating its higher efficiency in capturing the model parameters. Finally, based on the initial computed Markov model and its corresponding EPM representation, an automatic method for creating an HMM was formulated, following upon a previously existing systematic method in the literature. Our effort in this part showed that starting from different Markov models assuming the well-mixed conditions, the hidden states augmented through our automatic HMM refinement method improve the model prediction accuracy, compensating for the inaccurate model assumptions.

Part IV of this manuscript describes the control effort in order to guide the self-assembly process carried out by the robotic modules towards specific pre-defined target structures. We summarize our contributions as follows:

- We addressed the problem of rule synthesis for programmable self-assembly of rotationally symmetric robotic modules endowed with genderless latching connectors. We showed that by utilizing the extended graph grammar formalism, one can formulate automatic rule synthesis algorithms whose direct output of synthesized rules can be directly programmed on the robotic modules without the need for any further adjustments, what was otherwise necessary as reported in the state of the art. Additionally, we provided the proof that employing the extended graph grammar formalism allows for synthesizing rulesets of $O(N)$ complexity, with N being the number of genderless connectors available on a robotic module, compared to rulesets of $O(N^2)$ in the literature.

- Using the extended graph grammar formalism and to illustrate automatic synthesis of rules directly applicable on robotic modules, we extended two synthesis algorithms originally introduced for the self-assembly of bodiless modules in the literature, namely Singleton and Linchpin. Doing so, we obtained their counterparts for the self-assembly 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. Following this verification, we presented the full framework which captures a close coupling between the ruleset controllers synthesis through employing graph grammatical models and the analysis of the performance of the corresponding self-assembly process using either the real system, the submicroscopic model, the microscopic model, or the macroscopic model of the real system.

- In a further step, we addressed the problem of synthesizing parallel rulesets for programmable self-assembly of robotic modules. Again, employing the extended graph grammar formalism, we proposed the automatic synthesis algorithm GS-RGS. Using the GS-RGS algorithm, we synthesized parallel rulesets for two target structures. Studies

on the synthesized rulesets in simulation, using both the microscopic and the submicroscopic models, demonstrated the superior performance of the GS-RGS algorithm compared to the LinchpinR algorithm from the literature. Additionally, we provided a formal proof for this superior performance.

17.2 Discussion and Future Work

We believe that, in the long run, engineered programmable self-assembling systems, particularly those systems comprising miniaturized robotic modules, have the potential of several applications. Notable applications will most likely lie in environmental, space, and medical domains, where robust structure formation out of miniaturized building blocks in a reversible and re-programmable fashion could be of significance.

This work constitutes a first step towards understanding and formalizing the principles of building fluid-mediated programmable self-assembling systems comprising resource-constrained robotic modules. Having addressed many aspects of the path to this goal, several problems remain to be tackled. Here, we will firstly address the aspects of this research along which further investigations and follow-up work may be conducted. We will then discuss promising research thrusts that can be undertaken leveraging the current experimental setup and developed methods. Finally, we will provide our outlook, reaching beyond the current state of our developments.

Following up on the technological research thrust, several aspects of the current self-assembling robotic platform may undergo further investigation and development in the future. Firstly, an investigation on the fluidic arena may be carried out to allow for creation of a variety of fluidic flow field patterns with distinct spatial characteristics, each giving rise to a substantially different mixing regime and thus assembly possibilities. To this goal, one needs to experimentally evaluate and to empirically find a set of configurations on the power level and perhaps time synchronization of the activation of the peripheral pumps installed in the fluidic arena. Secondly, the robotic modules may be endowed with two more latching connectors, one at the top and one at the bottom side of the cubic shell, and additionally tuned for neutral buoyancy in order to float and assemble in 3D. Finally, the mechatronic design of the robotic modules may be scaled down. This might require removing the on-board battery from the module and designing the setup such that the modules are capable of harvesting energy from the environment.

Following up on the methodological research thrust, one may consider developing accurate HMM models of the system for different combinations of mixing regimes and ruleset controllers, for a predefined target structure. Based on the observation of the modules configuration in the system and estimation of the only partially observable full state of the system, the set of accurate HMM models can then be employed to develop optimized control for choosing the environmental fluidic flow and ruleset parameters which can be communicated to the robotic modules over the existing radio link in real time in order to guide the self-assembly

process towards the predefined target structure. In order to be able to build a larger variety of target structures, one may extend the GS-RGS algorithm that only handles tree target structures by considering (i) general non-tree target structures for which a method needs to be developed to address formation of loops, and (ii) by considering further parallelization through, for instance, exploiting the symmetries in the target structure and breaking it down into sub-structures which can be assembled in parallel and then joined to form the target.

The current developed experimental setup can be utilized to conduct further experimental studies. One interesting approach would be to utilize the setup for performing experiments on a swarm of Lily robotic modules which are endowed with different ruleset controllers. Unlike the case where all modules are endowed with the same ruleset controllers, different modules may take up different roles in the assembly process. Allowing for non-homogenoity in the swarm could be realized in three ways. First, the modules may start with different initial internal states but identical set of rules and associated probabilities. Second, the modules may be endowed with the same set of rules but different associated probabilities, and third, both the structure and the associated probabilities of the ruleset controllers of the modules may differ. This non-homogenity can potentially allow for further parallelization in the process of assembling the target by allowing the modules to work on different parts of the target in parallel. Once built, these parts will eventually join to form a complete copy of the target structure.

We have in this work striven to explore and develop general and transferable methods for building programmable self-assembling robotic systems comprising resource-constrained robotic modules operating in a fluidic environment. While the main focus of our research has been on the design, modeling, and control of swarms of centimeter-sized robotic modules fabricated through standard mechatronic techniques, we believe that the flexibility and generality of our methods may pave the way to even smaller modules, down to the millimeter and sub-millimeter scales, possibly even realized through a different fabrication technique, as long as parametric behavioral rules can be encoded. In the near future, we wish to see the multi-level suite of platform-independent control and modeling tools hereby developed forming a reference for a range of applications in fields where control of distributed systems comprising resource-constrained modules is required. In a farther future, our hope is to see the techniques developed in this work to be employed as a knowledge base to deploy ever smaller robots. Miniaturized robots of a few centimeters in size have not attracted many application interests because of an unfavorable performance-size tradeoff. This will possibly not be the case with millimeter-sized robots, which due to their small size and light weight can be deployed in unprecedented environments such as living bodies or natural niches.

Glossary

CRN Chemical Reaction Network. 95, 97, 98, 101–106, 108, 144, 145

EPM Electro-Permanent Magnet. 33–48, 51, 52, 55, 58–60, 63, 68, 72, 102, 145

HMM Hidden Markov Model. 69, 96, 97, 103–106, 108, 145, 146

KS Kolmogorov-Smirnov. 74

MJM Multi-Jet Modeling. 34

MSD Mean Square Displacement. 75–81

NRWW Non Read While Write. 53

ODE Open Dynamics Engine. 71

PSO Particle Swarm Optimization. 73, 76, 78–81, 107

RWW Read While Write. 53

Bibliography

[1] J. A. Pelesko, *Self assembly: the science of things that put themselves together*. CRC Press, 2007.

[2] K. Hosokawa, I. Shimoyama, and H. Miura, "Two-dimensional micro-self-assembly using the surface tension of water", *Sensors and Actuators A: Physical*, vol. 57, no. 2, pp. 117–125, 1996.

[3] G. Aggarwal, Q. Cheng, M. H. Goldwasser, M.-Y. Kao, P. M. De Espanes, and R. T. Schweller, "Complexities for generalized models of self-assembly", *SIAM Journal on Computing*, vol. 34, no. 6, pp. 1493–1515, 2005.

[4] J. H. Reif, S. Sahu, and P. Yin, "Complexity of graph self-assembly in accretive systems and self-destructible systems", *Theoretical Computer Science*, vol. 412, no. 17, pp. 1592–1605, 2011.

[5] G. M. Whitesides and B. Grzybowski, "Self-assembly at all scales", *Science*, vol. 295, no. 5564, pp. 2418–2421, 2002.

[6] M. Fialkowski, K. J. Bishop, R. Klajn, S. K. Smoukov, C. J. Campbell, and B. A. Grzybowski, *Principles and implementations of dissipative (dynamic) self-assembly*, 2006.

[7] B. Haghighat, M. Mastrangeli, G. Mermoud, F. Schill, and A. Martinoli, "Fluid-mediated stochastic self-assembly at centimetric and sub-millimetric scales: design, modeling, and control", *Micromachines*, vol. 7, no. 8, p. 138, 2016.

[8] D. K. Eric, "Engines of creation. the coming era of nanotechnology", 1986.

[9] D. Kennedy and C. Norman, "What don't we know?", *Science*, vol. 309, no. 5731, pp. 75–75, 2005.

[10] G. M. Whitesides and M. Boncheva, "Beyond molecules: self-assembly of mesoscopic and macroscopic components", *Proceedings of the National Academy of Sciences*, vol. 99, no. 8, pp. 4769–4774, 2002.

[11] R. Groß and M. Dorigo, "Self-assembly at the macroscopic scale", *Proceedings of the IEEE*, vol. 96, no. 9, pp. 1490–1508, 2008.

[12] N. Bhalla, P. J. Bentley, and C. Jacob, "Mapping virtual self-assembly rules to physical systems", *Unconventional Computing*, vol. 167, 2007.

[13] A. L. Christensen, O. Rehan, M. Dorigo, *et al.*, "Morphology control in a multirobot system", *IEEE Robotics & Automation Magazine*, vol. 14, no. 4, pp. 18–25, 2007.

[14] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu, "Miche: modular shape formation by self-disassembly", *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 345–372, 2008.

[15] E. Klavins, "Programmable self-assembly", *IEEE Control Systems*, vol. 27, no. 4, pp. 43–56, 2007.

[16] S. Miyashita, M. Hadorn, and P. E. Hotz, "Water floating self-assembling agents", in *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, Springer, 2007, pp. 665–674.

[17] G. Mermoud, M. Mastrangeli, U. Upadhyay, and A. Martinoli, "Real-time automated modeling and control of self-assembling systems", in *IEEE International Conference on Robotics and Automation*, 2012, pp. 4266–4273.

[18] R. Groβ and M. Dorigo, "Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling", *Adaptive Behavior*, vol. 16, no. 5, pp. 285–305, 2008.

[19] R. O'Grady, A. L. Christensen, and M. Dorigo, "Self-sssembly and morphology control in a swarm-bot", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 2551–2552.

[20] R. O'Grady, R. Groß, A. L. Christensen, F. Mondada, M. Bonani, and M. Dorigo, "Performance benefits of self-assembly in a swarm-bot", in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, IEEE, 2007, pp. 2381–2387.

[21] S. Murata, K. Kakomura, and H. Kurokawa, "Toward a scalable modular robotic system", *IEEE Robotics & Automation Magazine*, vol. 14, no. 4, pp. 56–63, 2007.

[22] S. Murata and H. Kurokawa, "Self-reconfigurable robots", *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 71–78, 2007.

[23] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]", *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[24] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, and C. J. Taylor, "Towards robotic self-reassembly after explosion", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 2767–2772.

[25] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson, "Evolved and designed self-reproducing modular robotics", *IEEE Transactions on robotics*, vol. 23, no. 2, pp. 308–319, 2007.

[26] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm", *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[27] K. Gilpin, K. Koyanagi, and D. Rus, "Making self-disassembling objects with multiple components in the robot pebbles system", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3614–3621.

[28] J. W. Romanishin, K. Gilpin, S. Claici, and D. Rus, "3d m-blocks: self-reconfiguring robots capable of locomotion via pivoting in three dimensions", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1925–1932.

[29] M. Tolley and H. Lipson, "Programmable 3d stochastic fluidic assembly of cm-scale modules", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 4366–4371.

[30] M. Mastrangeli, F. Schill, J. Goldowsky, H. Knapp, J. Brugger, and A. Martinoli, "Automated real-time control of fluidic self-assembly of microparticles", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5860–5865.

[31] V. N. Manoharan, M. T. Elsesser, and D. J. Pine, "Dense packing and symmetry in small clusters of microspheres", *Science*, vol. 301, no. 5632, pp. 483–487, 2003.

[32] Z. Zhang and S. C. Glotzer, "Self-assembly of patchy particles", *Nano Letters*, vol. 4, no. 8, pp. 1407–1413, 2004.

[33] C. R. Iacovella, M. A. Horsch, Z. Zhang, and S. C. Glotzer, "Phase diagrams of self-assembled mono-tethered nanospheres from molecular simulation and comparison to surfactants", *Langmuir*, vol. 21, no. 21, pp. 9488–9494, 2005.

[34] K. Hosokawa, I. Shimoyama, and H. Miura, "Dynamics of self-assembling systems: analogy with chemical kinetics", *Artificial Life*, vol. 1, no. 4, pp. 413–427, 1994.

[35] K. Saitou, "Conformational switching in self-assembling mechanical systems", *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 510–520, 1999.

[36] P. W. Rothemund, N. Papadakis, and E. Winfree, "Algorithmic self-assembly of dna sierpinski triangles", *PLoS biology*, vol. 2, no. 12, e424, 2004.

[37] D. Doty, "Theory of algorithmic self-assembly", *Communications of the ACM*, vol. 55, no. 12, pp. 78–88, 2012.

[38] A. Martinoli, K. Easton, and W. Agassounon, "Modeling swarm robotic systems: a case study in collaborative distributed manipulation", *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 415–436, 2004.

[39] G. Mermoud, *Stochastic Reactive Distributed Robotic Systems: Design, Modeling and Optimization*. Springer, 2013, vol. 93.

[40] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in swarm-bots", *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1115–1130, 2006.

[41] T. Fukuda, T. Ueyama, and Y. Kawauchi, "Self-organization in cellular robotic system(cebot) for space application with knowledge allocation method", *i-SAIRAS'90*, pp. 101–104, 1990.

[42] M. Rubenstein, K. Payne, P. Will, and W.-M. Shen, "Docking among independent and autonomous conro self-reconfigurable robots", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, vol. 3, 2004, pp. 2877–2882.

[43] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw, "Connecting and disconnecting for chain self-reconfiguration with polybot", *IEEE/ASME Transactions on mechatronics*, vol. 7, no. 4, pp. 442–451, 2002.

[44] S. Murata, K. Kakomura, and H. Kurokawa, "Docking experiments of a modular robot by visual feedback", in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, 2006, pp. 625–630.

[45] S. Griffith, D. Goldwater, and J. M. Jacobson, "Robotics: self-replication from random parts", *Nature*, vol. 437, no. 7059, p. 636, 2005.

[46] P. White, K. Kopanski, and H. Lipson, "Stochastic self-reconfigurable cellular robotics", in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, 2004, pp. 2888–2893.

[47] P. White, V. Zykov, J. Bongard, and H. Lipson, "Three dimensional stochastic reconfiguration of modular robots.", in *Robotics: Science and Systems*, Cambridge, 2005, pp. 161–168.

[48] K. Gilpin, A. Knaian, and D. Rus, "Robot pebbles: one centimeter modules for programmable matter through self-disassembly", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2485–2492.

[49] L. S. Penrose and R. Penrose, "A self-reproducing analogue", *Nature*, vol. 179, no. 4571, p. 1183, 1957.

[50] N. Bhalla and P. J. Bentley, "Working towards self-assembling robots at all scales", in *Proc. of the 3rd Int. Conf. on Autonomous Robots and Agents*, 2006, pp. 617–622.

[51] J. Breivik, "Self-organization of template-replicating polymers and the spontaneous rise of genetic information", *Entropy*, vol. 3, no. 4, pp. 273–279, 2001.

[52] L. Jacot-Descombes, J. Brugger, and R. M. Gullo, "Fluid-mediated self-assembly of mems micro-capsules for liquid encapsulation and release", *EPFL PhD Thesis*, 2013.

[53] H. Jacobson, "On models of reproduction", *American Scientist*, vol. 46, no. 3, pp. 255–284, 1958.

[54] T. Fukuda, "Self organizing robots based on cell structures-cebot", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1988, pp. 145–150.

[55] T. Fukuda, T. Ueyama, and K. Sekiyama, "Distributed intelligent systems in cellular robotics", in *Artificial intelligence in industrial decision making, control and automation*, Springer, 1995, pp. 225–246.

[56] C. Bererton and P. K. Khosla, "Towards a team of robots with repair capabilities: a visual docking system", in *Experimental Robotics VII*, Springer, 2001, pp. 333–342.

[57] R. Groß, E. Tuci, M. Dorigo, M. Bonani, and F. Mondada, "Object transport by modular robots that self-assemble", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2558–2564.

[58] R. O'Grady, R. Groß, F. Mondada, M. Bonani, and M. Dorigo, "Self-assembly on demand in a group of physical autonomous mobile robots navigating rough terrain", in *European Conference on Artificial Life*, 2005, pp. 272–281.

[59] R. Groß, M. Dorigo, and M. Yamakita, "Self-assembly of mobile robots-from swarm-bot to super-mechano colony", in *International Conference of Intelligent Autonomous Systems*, IOS Press, 2006, pp. 487–496.

[60] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: a low cost scalable robot system for collective behaviors", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3293–3298.

[61] S. Griffith, D. Goldwater, and J. Jacobson, "Self-replication from random parts", *Nature*, vol. 437, no. 7059, p. 636, 2005.

[62] K. Lerman, A. Martinoli, and A. Galstyan, "A review of probabilistic macroscopic models for swarm robotic systems", in *International Workshop on Swarm Robotics*, 2004, pp. 143–152.

[63] S. Miyashita, M. Kessler, and M. Lungarella, "How morphology affects self-assembly in a stochastic modular robot", in *IEEE International Conference on Robotics and Automation*, 2008, pp. 3533–3538.

[64] T. Hogg, "Coordinating microscopic robots in viscous fluids", *Autonomous Agents and Multi-Agent Systems*, vol. 14, no. 3, pp. 271–305, 2007.

[65] L. Matthey, S. Berman, and V. Kumar, "Stochastic strategies for a swarm robotic assembly system", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 1953–1958.

[66] D. T. Gillespie, "Stochastic simulation of chemical kinetics", *Annu. Rev. Phys. Chem.*, vol. 58, pp. 35–55, 2007.

[67] B. Haghighat, E. Droz, and A. Martinoli, "Lily: a miniature floating robotic platform for programmable stochastic self-assembly", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1941–1948.

[68] B. Haghighat and A. Martinoli, "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)*, 2016, pp. 2778–2783.

[69] B. Haghighat, R. Thandiackal, M. Mordig, and A. Martinoli, "Probabilistic modeling of programmable stochastic self-assembly of robotic modules", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4656–4663.

[70] B. Haghighat and A. Martinoli, "Automatic synthesis of rulesets for programmable stochastic self-assembly of rotationally symmetric robotic modules", *Swarm Intelligence*, vol. 11, no. 3-4, pp. 243–270, 2017.

[71] B. Haghighat, B. Platerrier, L. Waegeli, and A. Martinoli, "Synthesizing rulesets for programmable robotic self-assembly: a case study using floating miniaturized robots", in *International Conference on Swarm Intelligence (ANTS)*, vol. 9882 of LNCS, 2016, pp. 197–209.

[72] B. Haghighat and A. Martinoli, "A rule synthesis algorithm for programmable stochastic self-assembly of robotic modules", in *Proceedings of the 13th Int. Symp. on Distributed Autonomous Robotic Systems (DARS)*, vol. 6, 2018, pp. 329–343.

[73] I. O'Hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, J. Seo, M. Turpin, V. Kumar, *et al.*, "Self-assembly of a swarm of autonomous boats into floating structures", in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 1234–1240.

[74] R. C. Merkle and R. A. Freitas Jr, *Kinematic self-replicating machines*. Landes Bioscience, 2004.

[75] T. Hogg, "Robust self-assembly using highly designable structures", *Nanotechnology*, vol. 10, no. 3, p. 300, 1999.

[76] V. Sariola, Q. Zhou, and H. N. Koivo, "Hybrid microhandling: a unified view of robotic handling and self-assembly", *Journal of Micro-Nano Mechatronics*, vol. 4, no. 1-2, p. 5, 2008.

[77] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms", *Robotica*, vol. 26, no. 5, pp. 691–701, 2008.

[78] K. Gilpin and D. Rus, "Modular robot systems", *IEEE robotics & automation magazine*, vol. 17, no. 3, pp. 38–55, 2010.

[79] M. Tolley and H. Lipson, "Fluidic manipulation for scalable stochastic 3d assembly of modular robots", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2473–2478.

[80] A. Knaian, "Electropermanent magnetic connectors and actuators: devices and their application in programmable matter", PhD thesis, Massachusetts Institute of Technology, 2010.

[81] K. Stoy and D. Brandt, "Efficient enumeration of modular robot configurations and shapes", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 4296–4301.

[82] G. Mermoud, L. Matthey, W. Evans, and A. Martinoli, "Aggregation-mediated collective perception and action in a group of miniature robots", in *The 9th International Conference on Autonomous Agents and Multiagent Systems*, vol. 2, 2010, pp. 599–606.

[83] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli, "Swistrack-a flexible open source tracking software for multi-agent systems", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 4004–4010.

[84] N. Napp, S. Burden, and E. Klavins, "Setpoint regulation for stochastically interacting robots", *Autonomous Robots*, vol. 30, no. 1, pp. 57–71, 2011.

[85] T. P. Pavlic, S. Wilson, G. P. Kumar, and S. Berman, "Control of stochastic boundary coverage by multirobot systems", *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 3, p. 034 504, 2015.

[86] O. Michel, "Webots: professional mobile robot simulation", *Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[87] W. Agassounon, A. Martinoli, and K. Easton, "Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes", *Autonomous Robots*, vol. 17, no. 2-3, pp. 163–192, 2004.

[88] N. Correll and A. Martinoli, "Modeling and designing self-organized aggregation in a swarm of miniature robots", *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 615–626, 2011.

[89] N. Correll and H. Hamann, "Probabilistic modeling of swarming systems", in *Springer Handbook of Computational Intelligence*, 2015, pp. 1423–1432.

[90] E. Di Mario, G. Mermoud, M. Mastrangeli, and A. Martinoli, "A trajectory-based calibration method for stochastic motion models", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 4341–4347.

[91] D. Frenkel and B. Smit, "Understanding molecular simulation: from algorithms to applications (academic, san diego, 2002)", pp. 63–107, 1997.

[92] M. Fox and J. Shamma, "Probabilistic performance guarantees for distributed self-assembly", *IEEE Transactions on Automatic Control*, vol. 60, no. 12, pp. 3180–3194, 2015.

[93] N. Napp, S. Burden, and E. Klavins, "The statistical dynamics of programmed self-assembly", in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2006, pp. 1469–1476.

[94] E. Klavins, R. Ghrist, and D. Lipsky, "A grammatical approach to self-organizing robotic systems", *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 949–962, 2006.

[95] M. Asadpour, M. H. Z. Ashtiani, A. Sproewitz, and A. Ijspeert, "Graph signature for self-reconfiguration planning of modules with symmetry", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2009, pp. 5295–5300.

[96] K. Golestan, M. Asadpour, and H. Moradi, "A new graph signature calculation method based on power centrality for modular robots", in *International Symposium Distributed Autonomous Robotic Systems (DARS)*, 2013, pp. 505–516.

[97] V. Ganesan and M. Chitre, "On stochastic self-assembly of underwater robots", *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 251–258, 2016.

[98] G. Mermoud, *Stochastic Reactive Distributed Robotic Systems*. Springer, 2014.

[99] N. Napp, D. Thorsley, and E. Klavins, "Hidden markov models for non-well-mixed reaction networks", in *2009 American Control Conference*, 2009, pp. 737–744.

[100] T. W. Mather and M. Ani Hsieh, "Macroscopic modeling of stochastic deployment policies with time delays for robot ensembles", *International Journal of Robotics Research*, vol. 30, no. 5, pp. 590–600, 2011.

[101] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions", *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[102] P. W. K. Rothemund, *Theory and experiments in algorithmic self-assembly*. University of Southern California, 2001.

[103] J. I. Lathrop, J. H. Lutz, and S. M. Summers, "Strict self-assembly of discrete sierpinski triangles", *Theoretical Computer Science*, vol. 410, no. 4-5, pp. 384–405, 2009.

[104] N. Bhalla, P. J. Bentley, and C. Jacob, "Evolving physical self-assembling systems in two-dimensions", in *International Conference on Evolvable Systems*, Springer, 2010, pp. 381–392.

[105] N. Bhalla, P. J. Bentley, P. D. Vize, and C. Jacob, "Programming and evolving physical self-assembling systems in three dimensions", *Natural Computing*, vol. 11, no. 3, pp. 475–498, 2012.

[106] E. Klavins, "Automatic synthesis of controllers for distributed assembly and formation forming", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2002, pp. 3296–3302.

[107] M. J. Fox and J. S. Shamma, "Communication, convergence, and stochastic stability in self-assembly", in *IEEE International Conference on Decision and Control*, 2010, pp. 7245–7250.

[108] E. Klavins, S. Burden, and N. Napp, "Optimal rules for programmed stochastic self-assembly", in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, 2006.

Curriculum Vitae

Bahar Haghighat

Education

2012-2018	Ph.D. in Robotics, Control, and Intelligent Systems *Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland*
2010-2012	M.Sc. in Electrical Engineering, Digital Electronics *Sharif University of Technology (SUT), Tehran, Iran*
2006-2010	B.Sc. in Electrical Engineering and Physics (Double-Major) *Sharif University of Technology (SUT), Tehran, Iran*

Honors and Awards

2017	Swiss National Science Foundation (SNSF) Early Postdoc Mobility fellowship
2017	Winner of the 3rd jury prize at the *Three Minute Thesis* competition at EPFL
2017	Winner of the public prize at the *Pitch Your Impact* competition at EPFL
2012	Ranked 1st among ~20 M.Sc. students of Digital Electronics
2012	Ranked 2nd among ~100 M.Sc. students of Electrical Engineering
2011	Sharif University of Technology Ph.D. studies scholarship
2009	Sharif University of Technology M.Sc. studies scholarship
2006	Iran National Elites Foundation B.Sc. studies scholarship
2006	Ranked 33rd among ~400,000 students in national university entrance exam

Publications

Journal Articles

1. **B. Haghighat**, A. Martinoli. "Automatic Synthesis of Rulesets for Programmable Stochastic Self-Assembly of Robotic Modules" In Swarm Intelligence Journal, 2017.
2. **B. Haghighat**, M. Masterangeli, G. Mermoud, F. Schill, A. Martinoli. "Fluid-Mediated Stochastic Self-Assembly at Centimetric and Sub-Millimetric Scales: Design, Modeling, and Control" In Micromachines Journal, Special Issue on Building by Self-Assembly, 2016.
3. I. Esmaili, S. Bagheri, **B. Haghighat**. "An Optimal Hardware Implementation for Active Learning Method Based on Memristor Crossbar Structures." In IEEE Systems Journal, 2014.

Refereed Conferences

1. **B. Haghighat**, A. Martinoli. "On Probabilistic Modeling of Programmable Stochastic Self-Assembly of Robotic Modules" In IEEE International Conference on Intelligent Robots and Systems (IROS), Vancouver, Canada, 2017.
2. **B. Haghighat**, A. Martinoli. "A Rule Synthesis Algorithm for Programmable Stochastic Self-Assembly of Robotic Modules" In International Symposium on Distributed Autonomous Robotic Systems (DARS), London, UK, 2016.
3. **B. Haghighat**, B. Platerrier, L. Waegeli, A. Martinoli. "Synthesizing Rulesets for Programmable Self-Assembly of Robots: A Case Study on Floating Miniaturized Robots" In International Conference on Swarm Intelligence (ANTS), Brussels, Belgium, 2016.
4. **B. Haghighat**, A. Martinoli. "Characterization and Validation of a Novel Robotic System for Fluid-Mediated Programmable Stochastic Self-Assembly." In IEEE International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea, 2016.
5. **B. Haghighat**, E. Droz and A. Martinoli. "Lily: A Miniature Floating Robotic Platform for Programmable Stochastic Self-Assembly." In IEEE International Conference on Robotics and Automation (ICRA), Seattle, Washington, USA, 2015.
6. **B. Haghighat**, S. Bagheri, and M. Firouzi. "A Novel Method for Function Approximation in Reinforcement Learning." In International Conference on Contemporary Issues in Computer and Information Sciences (CICIS), Zanjan, Iran, 2012.

Review Service

1. IEEE International Conference on Robotics and Automation (ICRA)
2. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
3. International Symposium on Distributed Autonomous Robotic Systems (DARS)
4. American Control Conference (ACC)
5. Swarm Intelligence Journal
6. Autonomous Robots Journal
7. IEEE Intelligent Systems Journal

Project Supervision

1. Hala Khodr, Internship Project (Summer 2017)
   Experimental Study of Self-Assembly with Lily Robots
2. Hala Khodr, Semester Project (Spring 2017)
   Optimization of Ruleset Controllers for Programmable Self-Assembly of Lily Robots
3. Maximilian Mordig, Semester Project (Fall 2016)
   Model-Based Control of Programmable Self-Assembly of Lily Robots
4. Matthias Ruegg, Semester Project (Fall 2016)
   Development and Experimental Evaluation of a Software Framework for the Lily Robots
5. Brice Platerrier, Semester Project (Fall 2015)
   Distributed Assembly Algorithm Design and Experimental Evaluation for the Lily Robots
6. Luca Brusatin, Master Thesis (Spring 2015), Co-supervised with Self-Organizing Systems Research Group at Harvard University
   Distributed Algorithms on a Thousand Robot Swarm
7. Loic Waegeli, Semester Project (Spring 2015)
   Distributed Assembly Algorithm Design and Experimental Evaluation for the Lily Robots
8. Beat Geissmann, Semester Project (Spring 2014)
   Communication and Computation Board Design for Self-Assembling Floating Miniature Robots
9. Alexandre Cherpillod, Semester Project (Fall 2013)
   Centimeter-Scale Water-Floating Robots for Studying Self-Assembly

Languages

Persian	native
English	fluent
French	proficient
German	proficient

Personal Details

Date of birth:	1 July 1988
Citizenship:	Iran
Email:	haghighat.bahar@gmail.com