A Principled Design Methodology for Minimalist Multi-Robot System Controllers

by

Christopher Vernon Jones

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

August 2005

Contents

List Of Tables

List Of Figures

Abstract

To enable the successful deployment of task-achieving *multi-robot systems* (MRS), the interactions must be coordinated among the robots within the MRS and between the robots and the task environment. There have been a number of impressive experimentally demonstrated coordinated MRS; however, most have been designed through *ad hoc* procedures, typically providing task-specific, empirical insights with few contributions toward greatly needed general-purpose, principled design methods.

This dissertation presents a *principled MRS controller design methodology* applicable to *minimalist* MRS performing acyclic tasks. The methodology is formally grounded and provides precise definitions for the intertwined entities involved in any task-achieving MRS – the world, task definition, and the capabilities of the robots. Built from this formal foundation, the methodology includes a suite of systematic controller synthesis procedures. Through the execution of the synthesized controllers, system-level coordination is achieved through the use of a number of local control features: broadcast inter-robot communication, the maintenance of internal state, and both deterministic and probabilistic action selection. A probabilistic microscopic MRS modeling technique is integrated with the synthesis methods in order to provide system performance estimates during controller synthesis in order to optimize the resulting controllers. The presented controller design methodology is more than a pragmatic design tool. Based on its formal foundations, it provides a platform to formally characterize interesting relationships and dependencies among MRS task requirements, individual robot control and capabilities, and resulting task performance.

The design methodology is validated in a multi-robot construction task domain and a multi-foraging task domain through physically-realistic simulations and real-robot demonstrations. In these domains, the design methodology is used to synthesize a number of robot controllers, thereby demonstrating the utility of the controller synthesis methods as well as providing data for a comparative analysis on the use of different control features in coordinating minimalist MRS.

Chapter 1

Introduction

The study of multi-robot systems (MRS) has received increased attention since the mid-1990's. This is not surprising as continually improving technology and infrastructure have made the deployment of MRS consisting of increasingly larger numbers of robots possible. With the growing interest in MRS comes the expectation that, at least in some important respects, multiple robots will be superior to a single robot in achieving a given task. To date, the design of MRS has remained *ad hoc* and, as such, few formal methodologies have been devised. This lack of design procedures has limited the usefulness of MRS and has prohibited the development of MRS solutions to many potential domains.

This dissertation presents a principled controller design methodology for *minimalist* MRS. The methodology is based on a formal foundation and is composed of a suite of systematic controller synthesis methods integrated with a microscopic MRS modeling approach. The aim of this work is to place the design process on a formal foundation, thereby enabling the ability to harness the amazing potential and power of multiple coordinated robots and advance their application into an expanding set of domains.

1.1 From Single to Multi-Robot Systems

Potential advantages of MRS over a single robot system (SRS) are frequently discussed in the literature [58, 5]. For example, total system cost, it is frequently claimed, may be reduced by utilizing multiple simple and cheap robots as opposed to a single complex and expensive robot. Some tasks maybe accomplished more efficiently by a group by decomposing the task into subtasks and performing each in parallel. The inherent complexity of some task environments may require the use of multiple robots as the necessary capabilities are too substantial to be met by a single robot. Finally, multiple robots are often assumed to increase system robustness by taking advantage of inherent parallelism and redundancy. Therefore, negative effects on task performance caused by individual robot failure or the dynamic addition or removal of individual robots can be minimized.

However, the utilization of MRS poses potential disadvantages and additional challenges that must be addressed if MRS are to present a viable and effective alternative to SRS in an important subset of domains. A poorly designed MRS, with individual robots working toward opposing goals, can be less effective than a carefully designed SRS. Many have come to realize that the design of *multi*-robot systems is, in many critical respects, a very different paradigm from the design of *single* robot systems [6, 53, 54, 31, 46]. In most cases just taking a suitable SRS design and scaling it up to multiple robots is not adequate. A paramount challenge in the design of effective MRS is managing the complexity of group control introduced by multiple, interacting robots.

1.2 Minimalist Multi-Robot Systems

This dissertation is specifically concerned with the design of *minimalist* MRS. The term "minimalist" is vague and, depending on the context, can be used to represent a variety of meanings. In this section, the meaning of this term is discussed with respect to this dissertation and rationales regarding the utility of minimalist MRS are given.

1.2.1 Defining Minimalist Multi-Robot Systems

A minimalist MRS is on composed of minimalist robots. A "minimalist" robot is so named based on its minimal computational and communication capabilities. For example, minimalist robots typically have very limited, and often no, capability by which to perform computationally extensive planning or optimization operations.

Minimalist robots are often limited to simple reactive controllers directly mapping current sensory information to immediate actions [55]. Their mechanisms of communication, if they have any, are frequently of very low bandwidth and distance and lacking the capability for direct, one-on-one forms of inter-robot communication. Due to these limitations, minimalist MRS are not able to perform complex forms of negotiation or task planning.

It must be emphasized that just because a MRS is *minimalist* does not imply that the capabilities of the MRS are *minimal*. Even given extremely limited individual robot capabilities, minimalist MRS have been shown to perform impressively complex forms of coordinated behavior in a variety of task domains, including foraging [8, 71, 30], construction of physical artifacts [56, 68], object clustering and sorting [11, 52, 34], task allocation [70, 44, 2], formation control [9, 25], and robot soccer [69]. Cleverly designed emergent coordination mechanisms can lead to impressively complex group behaviors, even when each individual robot is simple – the nature and power of the interactions makes the performance of the system as a whole more than just the sum of its minimalist individual parts.

An important question in the design of minimalist MRS is "How can system-level coordination be achieved in a MRS composed of robots that have very limited capabilities?". At this time, there exist only empirical demonstrations that minimalist MRS are capable of achieving impressive system-level coordination, some of which were discussed above. Impressive as these demonstrations are, they are only *empirical* proof of the power of effectively designed minimalist MRS. The purpose of this dissertation is not to provide another empirical demonstration – instead, the purpose is to provide a *principled design methodology* that allows the design of effective minimalist MRS across a number of task domains and system characteristics.

1.2.2 Rationale for Minimalist Multi-Robot Systems

One may reasonably ask, "With the availability of cheap and easily accessible computation and communication capabilities, why would one be interested in *minimalist* MRS at all?". There are many justifications for using a minimalist approach to the design of MRS.

First, autonomous mobile robots are limited by the amount of power they have available. A robot equipped with the most advanced computational and communication capabilities available is useless if the robot's batteries can only power such devices for a very short time. This concern is especially relevant in domains requiring long periods of autonomous operations (e.g., robots for planetary exploration and construction).

Second, robots that require limited computational and communication capabilities can be made cheaper and smaller, in part because lower-end equipment is less costly and in part because less power-hungry electronics require fewer batteries that reduce the weight of the robot and thus reduce the power requirements of the actuators. This concern is especially relevant in the emerging consumer robotics market.

Third, because there are fewer complex operations performed by each robot and fewer complex interactions (e.g., timely negotiations) performed between robots, minimalist MRS are often more robust and adaptable to dynamic changes in the task domain. This feature is especially relevant for MRS that must robustly operate in the real world in presence of unpredictable dynamics.

Fourth, since all communication mediums have a finite bandwidth and range, the concern remains that as the group size grows, the available communication bandwidth and range will become inadequate. Since minimalist MRS place less emphasis on explicit communication, this concern is ameliorated. Such concerns are especially relevant in underwater domains and in systems involving very large numbers of robots where communication is expensive and limited.

Fifth, the study of minimalist systems is critical to understanding the uses and limitations of different control and coordination mechanisms and their relationship to the robot-environment dynamics and the task at hand. For example, it is still an open question as to when the use of inter-robot communication is necessary, when it is beneficial, and when it may even be detrimental to overall system performance. By studying minimalist systems one gains a broader understanding of such issues and relationships and can then transfer the gained insights to non-minimalist systems when appropriate. Future MRS designs may not be *purely* minimalist; however, they may employ relevant components of the knowledge gained from the study of minimalist systems to, for example, reduce the system's overall communication complexity or minimize dependency on internal state.

A final pragmatic reason that spans all of the above is that the study of minimalist systems will be key to emerging application domains such as "smart dust" [23] and swarm robotics [14].

1.3 Coordination in Multi-Robot Systems

In order to maximize the effectiveness of a task-achieving MRS, the robots' actions must be spatiotemporally coordinated and directed towards the achievement of a given system-level task. From a few robots performing a manipulation task [13], to tens of robots exploring a large indoor area [35, 43], to thousands of ecosystem monitoring nano-robots [73], as the number of robots in the system increases, so does the necessity and importance of coordination. Coordination is defined as "the act of regulating and combining so as to produce harmonious results" [1]. In the context of MRS, coordination involves the appropriate spatiotemporal regulation of the robots' actions such that the probability of a given task or goal being successfully achieved is maximized.

1.3.1 Coordination Approaches

Depending on the characteristics of the robots, characteristics of the task, and the level of desired system performance, the mechanisms by which coordination can be achieved cover a wide spectrum. Historically, *intentional* coordination mechanisms have been the most studied. Intentional coordination approaches typically rely on inter-robot communication and/or computationally expensive planning either independently by some or all robots or by some outside centralized coordinator. This form of coordination has found great success in many domains; however, such approaches quickly become ineffective as the size of the system grows or the individual robot's capabilities diminish.

Alternative approaches can be found in so-called *emergent* coordination [60], which typically shun the use of complex computation and communication approaches. Instead, such approaches rely on local sensing and exploit information and structure in the environment and in the dynamics between the robots and the task environment. As described in [16], such approaches take advantage of the information gained by the fact that a robot is situated in an environment and remove complexity from robots by more fully utilizing complexity in the environment. In one of the often referenced elegant examples [34], a group of robots move initially dispersed objects in a bounded environment into a small number of dense clusters, without the use of planning or communication. Each robot is endowed with a small number of simple and creatively tailored rules that exploit complexity (i.e., information) in the environment to effectively perform the collective clustering task.

1.3.2 Coordination in Minimalist Multi-Robot Systems

The limited robot complexity required by emergent approaches to coordination has made them a popular approach in the design of *minimalist* MRS. There have been a number of empirically demonstrated minimalist MRS utilizing emergent approaches to coordination. In construction

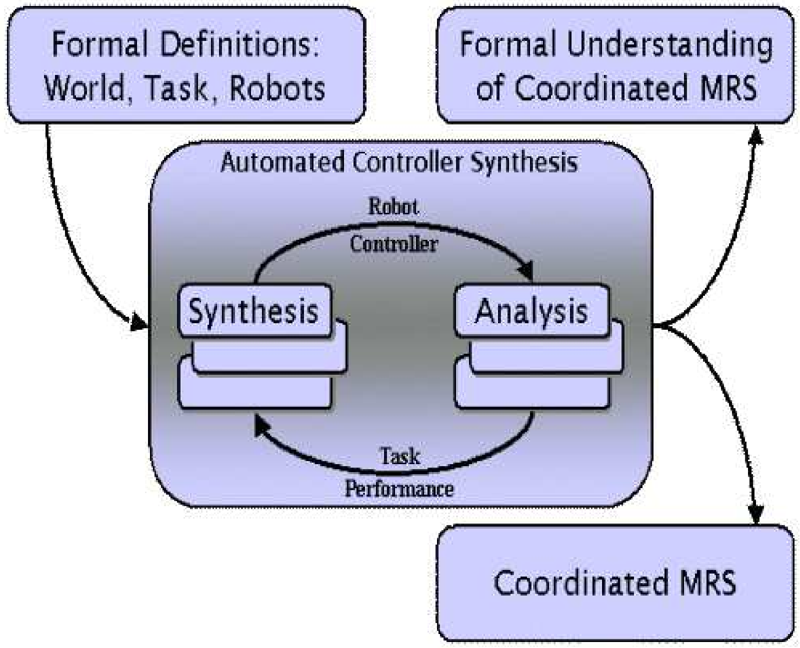

Figure 1.1: Principled minimalist multi-robot system controller design methodology presented in this dissertation.

[56, 68], collaborative stick pulling [36, 51], foraging [44], and many other domains, minimalist MRS have been shown to effectively achieve coordinated system-level behavior. Such coordination was achieved through the use of carefully exploited dynamics between the robots' mobility and sensory modalities and the dynamics of the task domain itself, using only limited computation and/or communication.

1.4 Design of Minimalist Multi-Robot Systems

Since the power and advantages of minimalist MRS is derived primarily through emergent forms of coordination which heavily rely on the interactions among the robots and between the robots and the task environment, robot controller design must pay special attention to the nature of those interactions.

To date, controller design for minimalist MRS has remained more of an art than a science. Current design methodologies are, in the best case, driven by informal and undocumented expert knowledge. In the worst case, they are driven by resource-intensive trial-and-error processes. Demonstrated systems are usually highly domain-specific and are infrequently accompanied by formal analysis of the expected system performance or bounds of applicability. Furthermore, formal explanations are rarely provided to justify the superiority of the given system relative to possible alternative designs. For these reasons, effective controller design is restricted by the lack of formal design tools and methodologies. The design of SRS controllers has greatly benefited from the principled methodologies provided by control theory, among other areas. To fully exploit their potential power and advantages, controller design of minimalist MRS is in need of analogous methodologies.

To address this need, this dissertation presents a principled minimalist MRS controller design methodology, aimed at moving the design of MRS closer to the realm of science. The methodology focuses on the interactions and mechanisms by which system-level coordination is achieved. Furthermore, the methodology is systematic and domain-independent, thereby at least partially removing the current necessity of expert knowledge and intensive trial-and-error design techniques.

As outlined in Figure 1.1, the design methodology presented in this dissertation is composed of three primary parts: a formal framework, controller analysis, and a suite of systematic controller synthesis methods.

The methodology provides a formal framework for the entire design process by precisely defining about the intertwined entities involved in any task-achieving MRS. Precise definitions of the dynamics of the world, the task to be performed, and the capabilities of individual robots is required if the design process is to be carried forward in a principled manner.

Included in the methodology is the ability to analytically evaluate the expected performance of a given controller. Furthermore, analysis capabilities are integrated with the controller synthesis procedures. Such a methodology allows for efficient evaluation of different design options, thereby leading to a process in which the design can be effectively optimized with respect to desired performance metrics.

The methodology contains a *suite of systematic controller synthesis methods* expressed in terms of the formal framework mentioned above. Each of these synthesis methods builds robot controllers utilizing a different combination of control features from the following set: deterministic or probabilistic action selection, the maintenance of internal state or stateless, and the use of broadcast inter-robot communication or non-communicative. The systematic nature of these methods provides a principled procedure for the synthesis of robot controllers.

1.5 Contributions

The primary contributions of this dissertation are a MRS formalism, a suite of systematic controller synthesis methods, and an integrated controller design methodology.

Multi-Robot System Formalism

This dissertation presents a formalism that provides a principled domain-independent framework for precisely defining and reasoning about the intertwined entities involved in any task-achieving MRS – the world, task definition, and the capabilities of the robots themselves, including sensing, maintenance of persistent internal state, inter-robot communication, and action selection. Furthermore, a MRS taxonomy based on the capabilities of the robots is provided. This taxonomy is used to classify and relate the controller synthesis methods described in Section 4.2.

Minimalist Multi-Robot System Controller Synthesis

A suite of systematic methods for synthesizing controllers for minimalist MRS is presented. Each of these methods synthesizes a controller that achieves system-level, task-directed coordination through the use of a unique combination of local control features, such as inter-robot communication, the maintenance of persistent internal state, and deterministic and probabilistic action selection.

Integrated Multi-Robot System Controller Design Methodology

Founded on the MRS formalism above, the synthesis methods are integrated into a complete end-to-end controller design methodology, as diagrammed in Figure 1.1. The methodology allows for complete controller design, and through close integration of controller synthesis and analysis methods, permits the designer to maximize the effectiveness of the designed MRS along pre-specified performance criteria. The methodology is validated in two different task domains: multi-robot construction and task allocation in multi-foraging. The construction task domain is relatively static with discrete actions performed by individual robots. In contrast, task allocation in the multi-foraging domain is dynamic and requires tight coordination among all robots to successfully perform the task.

The auxiliary contributions are controller designs for a multi-robot construction task domain and algorithms for task allocation in a multi-foraging task domain.

Multi-Robot Construction

The MRS controller design methodology presented in this dissertation provides more than a principled methodology for the study of MRS. Additionally, it provides a pragmatic tool for the synthesis of working robot controllers. The formalism providing the foundation for the design methodology is cast into a multi-robot construction task domain. Robot controllers are synthesized for this domain and validated in both physically-realistic simulations as well as a limited number of real-world trials. These controllers demonstrate novel coordination mechanisms for the multi-robot construction task as well as provide original experimental work on the application of minimalist MRS.

Multi-Foraging Task Allocation Algorithms

Furthermore, task allocation in a multi-foraging task domain is studied in the context of the design methodology. As a result, a number of robot controllers are synthesized and analyzed for this domain. These controllers are experimentally validated in physically-realistic simulations and demonstrate novel control and coordination mechanisms for task allocation in the multi-foraging domain.

1.6 Dissertation Outline

The remainder of this dissertation is organized as follows.

Chapter 2 presents work related to minimalist MRS controller design. Related work spanning the design spectrum, ranging from informally hand-designed systems which achieve impressive system-level coordinated behavior to formal design methodologies is provided.

Chapter 3 presents a MRS formalism which is used to define and reason about MRS and their task environment. This formalism provides a principled framework that is used throughout the rest of this dissertation.

Chapter 4 presents a MRS controller design methodology composed of controller analysis and synthesis. Section 4.1 describes a microscopic MRS modeling approach used in this dissertation to quantitatively predict the performance of a given MRS controller design. Analysis is used in an iterative fashion with controller synthesis methods to synthesize optimized controllers. Section 4.2 introduces a suite of four systematic controller synthesis methods, each of which utilizes a unique combination of local robot control characteristics to achieve system-level task-directed coordination. The control characteristics considered are the maintenance of internal state, inter-robot communication, and both probabilistic and deterministic action selection.

Chapter 5 applies the principled minimalist MRS controller design methodology to the synthesis and analysis of minimalist MRS in a multi-robot construction task domain and a multi-foraging domain.

Chapter 6 concludes this dissertation with a summary of contributions and directions for future work building on and utilizing the presented principled controller design methodology.

Chapter 2

Related Work

This dissertation is concerned with the development of a principled minimalist MRS controller design methodology, which includes integrated controller synthesis and analysis methods. As such, this chapter reviews related work in the following areas: empirically demonstrated minimalist MRS (Section 2.1), MRS design methodologies (Section 2.2), and the multi-agent systems community (Section 2.3).

2.1 Empirical Demonstrations of Minimalist Multi-Robot Systems

Empirical demonstrations of minimalist MRS are reviewed with the intention of defining, in practical terms, the adjective "minimalist" as well as illustrating the power and potential of appropriately designed minimalist MRS. Furthermore, empirical work is reviewed in order to motivate the need for a systematic, formally grounded, and domain-independent methodology to improve the design process.

As stated earlier, an MRS being *minimalist* does not imply that its capabilities are *minimal*. Minimalist MRS have been shown to be remarkably effective in a variety of task domains [54, 5]. Broad reviews of demonstrated MRS were performed by Dudek et al. [22] and Cao et al.

[17]. Dudek et al. [22] presented a taxonomy of MRS along dimensions of communication and computational capabilities of the individual robots and their application to a number of case studies including exploration, metric localization and mapping, and simple navigation. Cao et al. [17] presented a set of MRS research axes including dimensions of communication capabilities, reasoning abilities, on-line learning capabilities, and origins of coordination, and reviewed existing work in task domains such as multi-robot navigation, cooperative manipulation, and foraging.

Although the systems overviewed in [22] and [17] are not explicitly labeled as minimalist, they represent a sampling of empirically demonstrated minimalist MRS as well as some axes for a taxonomy of MRS, including the minimalist side of the spectrum. A MRS taxonomy was provided in [22] based on the size of the group and communication range, bandwidth, and topology. According to this taxonomy, the work presented in this dissertation focuses on large group sizes of *SIZE-LIM* and *SIZE-INF*, which are MRS composed of more than 2 robots and potentially a very large number of robots. The nature of communication studied in this dissertation, according to the taxonomy in [22], ranges from no communication (*COM-NONE*), to local communication (*COM-NEAR*), to broadcast communication (*COM-INF, TOP-BROAD*). Cases where direct, addressable communication is available between the robots (*TOP-ADD*) is not considered in this dissertation. Concerning communication bandwidth, cases of communication through the environment (*COM-MOTION*), low bandwidth (*BAND-LOW*), and no communication (*BAND-ZERO*) are all considered.

Beckers et al. [11] presented a homogeneous MRS consisting of robots with limited and local sensing, without the ability to maintain persistent internal state, and with no explicit inter-robot communication capabilities. Even with such limited individual robots, the MRS was carefully and elegantly designed, both in terms of control and physical design of the robots, such that it effectively clustered randomly scattered objects into increasingly larger clusters. Additional work in the clustering domain, carried out by Holland and Melhuish [34], further explored the capabilities of the approach presented in [11].

MRS with limited individual robot capabilities have also been shown to be effective in other task domains such as formations [25], foraging [32, 37], robot soccer [69], and distributed manipulation in a box-pushing task domain [45]. Vaughan et al. [67] demonstrated a MRS in a simulated foraging task domain that achieves robust coverage of the environment through the use of very low-bandwidth indirect communication between robots similar to pheromone communication used by ants. Similarly, Svennebring and Koenig [65] presented an approach to multi-robot terrain covering using trail laying analogous to ant pheromone trail laying. Their work was implemented on physical robots and verified that the trail laying approach resulted in robust performance in large-scale environments.

The above examples are of empirically demonstrated minimalist MRS that are primarily designed through *ad hoc* procedures and are targeted to specific task domains. This dissertation provides a MRS controller design methodology that is principled and not task specific.

2.2 Design of Multi-Robot System Controllers

Efficient and accurate methods of analysis are required if a design process is to intelligently evaluate a sufficiently large proportion of the design space. Furthermore, systematic controller synthesis methods are required if a design process is to remain principled. This section reviews work related to the mathematical analysis (Section 2.2.1) and systematic synthesis (Section 2.2.2) of minimalist MRS controllers.

2.2.1 Analysis of Multi-Robot Systems

MRS analysis approaches can be broadly classified into two categories: 1) those approaches utilizing macroscopic models, and 2) those utilizing microscopic models. A primary difference between the two approaches is in the granularity of the model.

Macroscopic models reason about system-level behavior without explicit consideration of each individual robot in the system. Since macroscopic operate at the system-level, the computational requirements remain largely unchanged whether an analyzed MRS is composed of a few robots or tens of robots. As such, macroscopic modeling approaches computationally scale well as the size of the system grows.

In contrast, microscopic modeling approaches directly consider each robot in the system and may model individual robot interactions with other robots and with the task environment in arbitrary detail, including simulating the exact behavior of each robot. However, most microscopic approaches abstract the individual robot behaviors to some degree, such as modeling each robot as a series of stochastic events where exact robot trajectories and interactions are not directly considered.

Lerman et al. [49] provided a survey of methods for the macroscopic modeling of MRS and their application to collaborative stick pulling, foraging, and aggregation task domains. The methods reviewed are primarily limited to robots executing reactive controllers [15, 54], that is, robots that do not maintain persistent internal state. The models consider each robot to behave as a stochastic Markov process and this assumption is then used to describe the dynamics of the resulting system-level behavior. Lerman and Galstyan [46] and Sugawara and Sano [64] both presented work on a macroscopic mathematical model of MRS in a foraging task domain. The model was used to study the effects of interference between robots, the results of which could be used to modify individual robot control or determine the optimal density of robots in order to maximize task performance. Lerman et al. [47] presented a macroscopic analytical model of the dynamics of collective behavior in a collaborative stick-pulling domain using a series of coupled differential equations. Lerman and Galstyan [48] described a general macroscopic model for the study of dynamic multi-agent systems and apply it to the analysis of a multi-robot dynamic task allocation domain. In their work, the robots constituting the MRS maintained a limited amount of persistent internal state to represent a short history of past events but do not explicitly communicate with other robots. The model qualitatively, and sometimes quantitatively, predicts how the system-level task allocation evolves over time during task execution. Macroscopic models have proven to be a useful tool in predicting certain aspects of system-level behavior, for example how system behavior changes over time. However, since specifics of individual robots are rarely considered, macroscopic modeling approaches are not effective at predicting other forms of system behavior, such as time to complete a given task or how minor changes in robot control will affect overall system behavior.

Microscopic modeling is the other primary form of MRS modeling. Martinoli et al. [52] presented a probabilistic microscopic modeling approach for the study of collective robot behavior in a clustering task domain. The model is validated through a largely quantitative agreement in the prediction of the evolution of cluster sizes with embodied simulation experiments and with real-robot experiments. Martinoli and Easton [50] discussed the effectiveness and accuracy of microscopic and macroscopic modeling techniques compared to real robot experiments and embodied simulations. Martinoli et al. [51] presented a time-discrete, incremental methodology for modeling the dynamics of coordination in a distributed manipulation task domain. The approach presented provided analysis at both the microscopic and macroscopic levels and is suited for the modeling of large-scale MRS. However, the robots used in this model are simplistic as they do not have the capability to maintain persistent internal state or to explicitly communicate. Microscopic modeling approaches, in particular probabilistic approaches, have been shown to be effective in predicting aspects of system-level behavior in specific task domains, such as how the adjustment of individual robot parameters can influence resulting system-level behavior.

There have also been other, more specialized forms of analysis, used to aid in the design and understanding of MRS. Balch [7] presented hierarchic social entropy, an information theoretic measure of robot team diversity as a method to investigate the role of heterogeneity in MRS coordination. This measure is used in an off-line manner in order to evaluate the structure of a given MRS. Goldberg and Matarić [32] analyzed the relationship between MRS characteristics, such as controller differences among the robots, and resulting task performance in terms of the degree of inter-robot interference, time to complete a task, and overall energy expenditure.

Goldberg and Matarić [30] presented work on the on-line learning of augmented Markov models (AMM) which are used to model the robot and robot task environment interactions. The AMMs are defined in terms of the history of behaviors a robot executes while performing a task and can be built and maintained in real-time. AMMs were demonstrated in a multi-robot foraging task domain to model the interactions and improve control strategies in an on-line manner. Such a technique produces an adaptive MRS that is potentially able to improve task performance during task execution, even if the task dynamics change over time. Gerkey and Matarić [27] presented a principled framework and analysis methodology, based on theories from economics and operations research. This framework cast MRS as a task allocation problem and then draws on the literature from operations research and economics, among others, to address that well-studied problem. It also outlines a taxonomy of MRS problems, and then focuses on one subclass of it: single-robot tasks and single-tasks robots, the most commonly treated class of MRS.

Although powerful tools, each of the analysis approaches reviewed above is generally suited for a specific class of MRS. The design methodology presented in this thesis considers robots that may maintain a limited amount of persistent internal state and communicate with other robots. The macroscopic and microscopic approaches discussed above are almost solely limited to the study of stateless and non-communicative robots. Macroscopic modeling approaches are appealing because they can provide mathematically precise results in a timely manner; however, the complexity of such approaches can quickly become computationally intractable when considering communicating robots that may maintain state.

Although microscopic models provide probabilistically accurate solutions, they are much more easily adapted to the consideration of robots that may maintain state and communicate. As such, the design methodology presented in this dissertation employs a probabilistic microscopic modeling approach, derived from [52], that incorporates the capability to consider robots that may maintain state and communicate.

2.2.2 Synthesis of Multi-Robot Systems

In this section we review work related to minimalist MRS controller synthesis. Particular focus is placed on synthesis through formal methodologies; however, related work on synthesis through learning and evolutionary approaches is also briefly discussed.

A prominent contribution to the formal specification and synthesis of minimalist MRS is the work of Donald [21], who presented the derivation of information invariants aimed at defining the information requirements of a given task and ways in which those requirements can be satisfied in a robot controller. Their work put the design of SRS and MRS on a formal footing and began to identify how various robot sensors, actuators, and control strategies could be used to satisfy task requirements. Furthermore, their work attempted to show how these features were related and how one or more of these features could be formally described in terms of a set of other features. The concept of information invariants was experimentally studied in a distributed manipulation task domain [20]. Furthermore, Parker [59] extended the idea of information invariants by defining equivalence classes among task definitions and robot capabilities to assist in the choice of appropriate controller class in a given domain.

Klavins [39, 40] presented the Computation and Control Language interpreter that implements robotics modeling and programming software tool called the Computation and Control Language (CCL). Their work is related to the approach in this dissertation in that it provided a common formal foundation for the design of MRS. A distinguishing factor of the work in this dissertation is that it uses a suite of systematic procedures for the synthesis of a number of different types of controllers. Furthermore, it is capable of integrating the operation of synthesis and analysis in the design process in order to maximize system performance. Klavins also used the CCL formalism to study the scalability of algorithms for multi-robot control, in particular the communication complexity of a variety of communication schemes [38].

More empirical and task-specific approaches to the design of MRS include Balch and Arkin [8], who presented results on a largely empirical investigation of the role of communication in MRS control and its relationship to task performance. In a variety of foraging and coverage tasks, they found that communication is often unnecessary and even the simplest forms of implicit communication can be almost as effective as more complex forms of explicit, protocol-oriented communication. Zhang et al. [72] presented a task-specific theoretical framework for the design of control algorithms in a multi-robot object clustering task domain. Issues addressed include how to design control algorithms that result in a single final cluster, multiple clusters, and how to control the variance in cluster sizes. Spletzer and Taylor [63] presented a control-theoretic model of coordination in MRS and applied it to the synthesis of controllers in a multi-robot box-pushing task domain.

Alternative approaches to the synthesis of MRS controllers can be found in evolutionary methods [24] and learning methods [53, 58]. There also exist a number of MRS design environments, control architectures, and programming languages which assist in the design of task-achieving coordinated MRS [54, 4, 3]. Although each of these approaches is valuable, none provide formally grounded design approaches. For example, in evolutionary and learning methods, it is frequently not possible to understand the rationale for the resulting controller design as opposed to other possible designs. The value of the principled design methodology presented in this dissertation is in contributing a systematic process of design that allows for understanding *why* a particular design was chosen, and often how specific features in the world or task definitions influenced the resulting design.

2.3 Multi-Agent Systems

This section briefly surveys related work from the multi-agent systems (MAS) community. Although work in both MAS and MRS deals with multiple interacting entities, it is still an open question as to whether techniques developed in the MAS community are directly applicable to the embodied MRS community. In MRS, and in particular minimalist MRS, the interactions between the robots and the task environment heavily influence the performance of the system and, therefore, cannot be ignored. The MAS community traditionally studies non-embodied agents and does not consider the interactions resulting from embodiment that are crucial to understanding MRS. However, the formal nature of MAS work may help to provide direction to the largely informal techniques for the design of MRS.

Cassandra et al. [18] used a partially observable Markov decision process (POMDP) to model uncertainty in a robot navigation task domain. The formalism presented in this dissertation is similar to the POMDP model. Specifically, the environment is Markovian, and from the perspective of the individual robots, the task state is partially observable. The POMDP model has been extended to include individual robots which can maintain persistent internal state, communicate with other robots, and make use of probabilistic action selection. The POMDP model as presented in [18] is applicable for single robot systems. Extending it to consider MRS is not considered to be tractable and there is a focus on the development of heuristics to address the complexity issues. Therefore, the POMDP notations and definitions are used as much as possible; however, use of methods derived from the POMDP model remain largely intractable for the effective use in the study MRS and are not used in this dissertation.

There exists a number of generalized instances of POMDPs to situations in which there are multiple, distributed agents with each making independent control decisions based on local sensory information, including Decentralized POMDP (DEC-POMDP) [12] and Markov Team Decision Problem (MTDP) [61]. Pynadath and Tambe [61, 62] provided a unified framework called the

Communicative Multiagent Team Decision Problem (COM-MTDP) in order to evaluate the trade-offs between a theories complexity and resulting policy performance. This model is general and subsumes many existing models thereby allowing the formal comparison and analysis of different methodologies under different domain conditions. As an extension of [18], Koenig and Simmons [42] used a POMDP to model uncertainties in an office navigation task domain. The POMDP was compiled off-line and provided with robot sensor models and a topological map of the office environment to make the approach tractable. During task execution, a modified learning algorithm was used to appropriately adjust the model and improve task performance on-line.

Another coordination formalism popular in the multi-agents community is the belief-desire-intention (BDI) model [26]. The BDI formalism models agents that can reason and has been the core of several coordination frameworks, including SharedPlans [33] and Joint Intentions [19]. The Joint Intentions framework has been used in the STEAM [66] architecture and was demonstrated in a simulated helicopter coordination task domain. The BDI model, in particular, fails to capture and include complex forms of interaction present in embodied MRS, such as local spatial interference among robots, that are necessary to understand the system-level behavior.

2.4 Summary

This chapter has reviewed work relevant to the development of a principled controller design methodology for minimalist MRS. Although there has been ample demonstration of the promise of minimalist MRS, the design of such systems needs to be formalized to fully realize their potential. Ideally, the design process should include both the ability to analyze a given MRS in an efficient and accurate manner and a principled method to synthesize robot controllers.

Researchers are working toward providing tools and methodologies to formally analyze and synthesize MRS, but there is still much progress to be made. Perhaps most importantly, there often exists a disconnect between the descriptive nature of the analysis methodology and the prescriptive nature of the controller synthesis. This can make it difficult to efficiently translate the results of analysis into appropriate improvements in overall controller design. The design methodology presented in this dissertation is novel in that it provides integrated analysis and controller synthesis methods that share a common formal foundation. For this reason, it is possible to directly and systematically translate the results of analysis into meaningful modifications to the overall controller design. In the following chapter we introduce this formal framework that lays the common foundation between the integrated analysis and synthesis methods.

Chapter 3

Multi-Robot System Formalism Definitions and Notation

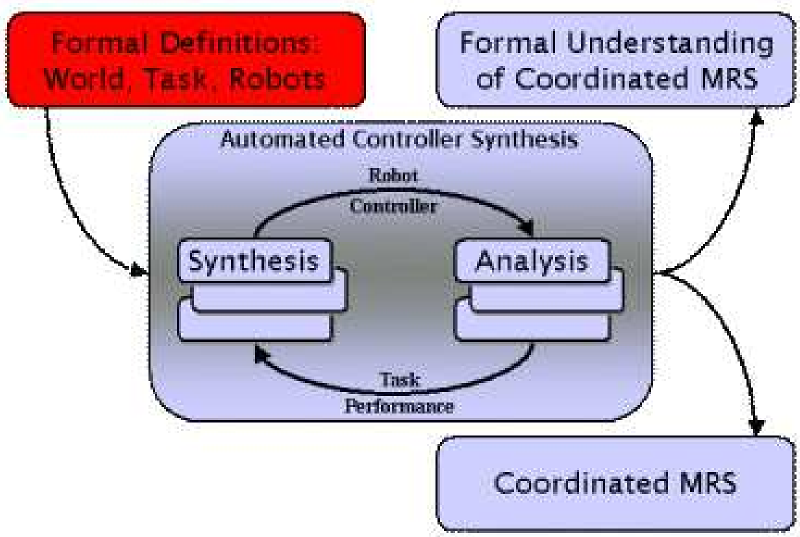

Figure 3.1: Diagram of principled minimalist multi-robot systems controller design methodology with the role of formal definitions highlighted.

This chapter introduces necessary definitions and notations that provide a formal framework for precisely defining and reasoning about the intertwined entities involved in any task-achieving MRS – the world, task definition, and the capabilities of the robots themselves. Figure 3.1 shows how this framework fits into the greater controller design methodology.

3.1 Role of Formalism in the Controller Design Process

Any MRS design methodology inherently requires knowledge about the dynamics of the world, the task to be performed, and the capabilities of individual robots. Without knowledge of each of these factors the design process is under-constrained and unprincipled. The world in which the task is to be performed must be understood in sufficient detail to be able to reasonably predict the mechanisms for and results of changing the state of the world. The task to be performed may be defined as specific goals the system much achieve, how these goals may be achieved, and perhaps some envelopes of required performance (e.g., time to complete task). The design methodology must also be informed as to the capabilities of individual robots, such action selection, sensing capabilities, maintenance of internal state, and inter-robot communication.

Depending on the design methodology being used, the above knowledge may be explicitly represented and reasoned about by a designer, or, as is more often the case, the knowledge may be represented internally by a designer who understands these issues and how they are related and employs this knowledge in the design process. In order to make the design process more principled and systematic, this knowledge needs to be represented in an explicit manner such that formal reasoning and analysis processes may build from it in a principled manner. The MRS controller analysis and synthesis methods presented in Chapters 4.1 and 4.2, respectively, comprise a formal and systematic controller design methodology that uses the formal framework presented in this chapter as a foundation.

The remainder of this chapter is organized as follows. Section 3.2 formally defines the world in which the MRS will be acting. Section 3.3 provides a formal definition of the class of tasks considered in the design methodology. Section 3.4 formally defines the relevant characteristics of the robots. Section 3.5 introduces a taxonomy of homogeneous MRS based on control characteristics of each robot. This taxonomy is used through this dissertation to discuss and compare different types of robot controllers.

3.2 World

The *world*, W, is the environment in which the MRS is expected to perform a defined task. It is assumed that the world is Markovian, populated by a finite set of robots R, and the state is an element of the finite set S of all possible states. The state of the world at time t is denoted as

$$S^t = \Xi_1 \times \cdots \times \Xi_m \tag{3.1}$$

where each $\Xi_i \in S$ represents the domain of an individual feature of the world. This set of features can contain many domain-dependent elements; including the physical positions of the robots, internal state values of each robot, or the location and proximate relationships of physical artifacts present in the world. The state of the world S represents the domain of all possible combinations of values over these individual features.

$\{A_r\}_{r \in R}$ is a set of actions that each robot r can execute. A_r^t represents the action a robot r makes at time t. $\{X_r\}_{r \in R}$ is a set of observations that each robot r can make. X_r^t represents the observation a robot r makes at time t. An observation consists of external information accessible to the robot and formally represents a subset of the world state. The probabilistic *observation function*

$$O(s, x) = Pr(X_r^t = x | S^t = s) \tag{3.2}$$

gives the probability that at some time t the observation x will be made in state s.

The world is defined by a probabilistic state transition function

$$P(s, a, s') = Pr(S^{t+1} = s' | S^t = s, A_r^t = a) \tag{3.3}$$

that states the probability the world state at time $t+1$ is s' given that the world state at time t was s and a robot r executed action a at time t.

3.3 Task

A *task* T is defined to be a set of world states and world state transitions represented as a directed acyclic graph (DAG). In the DAG for a task T, each vertex in the finite set of vertices V_T is a unique world state. For each vertex $u \in V_T$, all vertices v, such that $\{u, v\} \in E_T$, are called *children* of u. Each edge in the finite set of edges E_T between the vertices $E_T = \{(u, v)|u, v \in V_T\}$ represents an action a such that $P(u, a, v) > 0$.

There is a exactly one vertex in the graph denoted as $u\_start_T \in V_T$ that corresponds to the initial world state. A subset of the graph vertices are marked as terminal vertices and are denoted by the set $u\_term_T \subseteq V_T$. Furthermore, there must exist a path through the DAG from every vertex in V_T to at least one vertex in u_term_T.

If at any time t, $s_t \in u\_term_T$ then the task terminates. *Correct task execution* is defined as the case where the combined actions of the robots cause the trajectory of the world state to follow the DAG represented by T from the starting node to a terminal node. That is, for each encountered world state s_i during task execution, where s_i corresponds to vertex $v \in V_T$, the combined action actions of the robots must transition the world state to some state s_j, where s_j corresponds to $u \in V_T$, such that $\{u, v\} \in E_T$. If at any time the actions of the robots cause the world state to transition from a state s_i to some state s_j in which there does not exist an edge in E_T connecting the corresponding graph vertices, the task is *not* correctly executed and task execution immediately terminates.

It is important to note that the world states represented by the graph vertices are a subset of the set of all world states. Figure 3.2 graphically depicts a general task, T, and represents the relationship between world states in the task and those that are not part of the task. In this figure, to achieve correct task execution, the trajectory of the world state must proceed from the start state, $u\_term_T = s_0$, along edges between states in the task definition until reaching a terminal

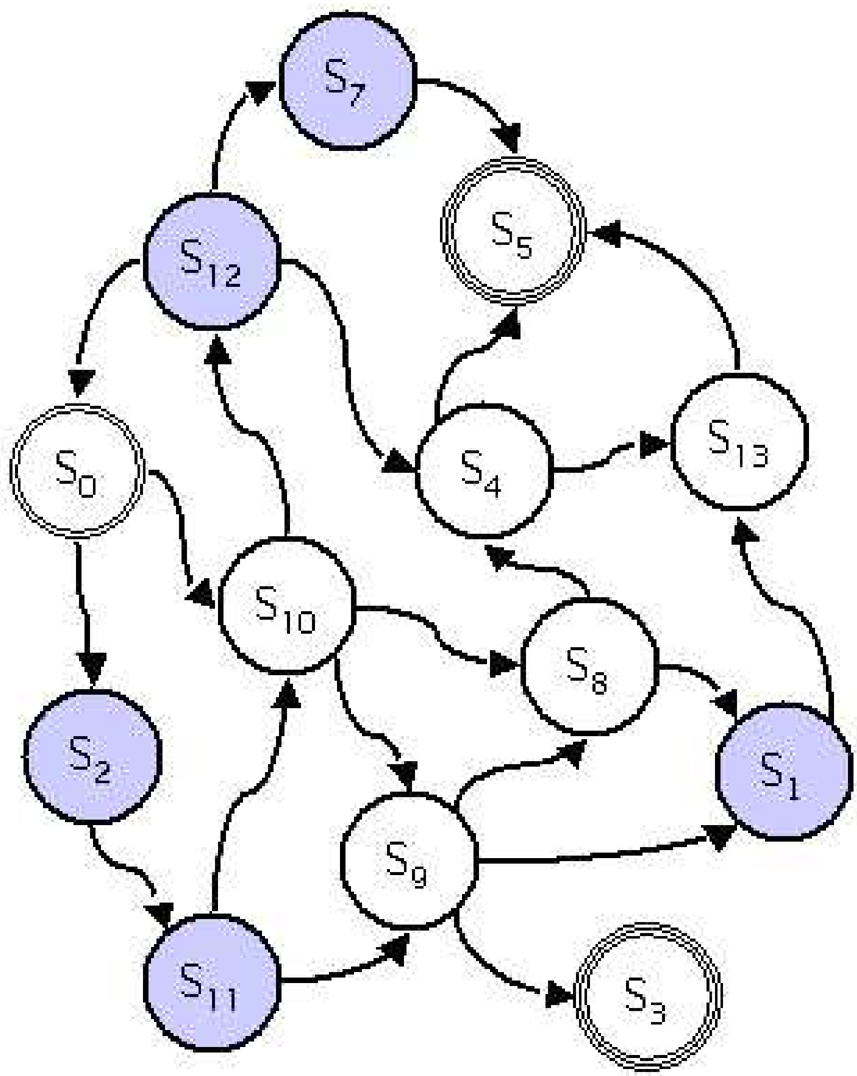

Figure 3.2: This figure provides a graphical depiction of the description of a task T. Each circle represents a unique world state. Unshaded circles represent states contained in the task and shaded circles represent world states that are *not* in the task definition. The start state, u_{start}, is s_0 and is represented by three concentric circles. The terminal states, u_{term}, are s_3 and s_5 and are represented by two concentric circles. Possible state transitions, as defined by the state transition function P, are represented as directed arrows connecting two states.

state, $u\_term_T = \{s_3, s_5\}$. The task will immediately terminate unsuccessfully if any state not in the task definition is reached.

3.4 Robot

This section provides definitions concerning the robot controller. Since this dissertation is limited to the study of homogeneous MRS, each robot in the system is considered functionally identical and all robots execute identical controllers.

3.4.1 Persistent Internal State

A robot is permitted to maintain a finite amount of persistent internal state. Internal state is considered to be temporally persistent memory that can be modified by the robot over time and is used in control decisions. Data that are stored for only one or a very few control cycles are not considered to be internal state since it is not persistent. Therefore, neither sensory data, motor control commands, nor the incoming inter-robot communication buffer, all of which are resident in memory for a short time, are considered to be internal state.

A robot r's internal state value m_r at any time is a member of the finite set $M_r = \{i_0, ..., i_p\}$, with i_0 being the initial internal state value for all robots. The actual content of each internal state value is not important, it is only required that each value be uniquely distinguishable. For example, in our implementation each value is just a unique integer. The method by which the internal state values are transitioned is provided in the controller definition given below in Section 3.4.3.

3.4.2 Inter-Robot Communication

A robot is permitted to communicate with all other robots in the MRS through the use of anonymous broadcast communication. A broadcast communication message is assumed to be instantaneously received by all robots and the identity of the sender is unknown to all receivers. Furthermore, it is assumed that each robot is capable of receiving multiple messages simultaneously.

The set of all possible communication messages a robot r may send and receive is denoted by the set $C_r = \{c_0, ..., c_q\}$. The actual content of each message is not important, it is only required that each message be uniquely distinguishable. For example, in our implementation each message is just a unique integer broadcast over the wireless network connecting all robots. A robot may only send one message at a time. The communication message a robot r is currently sending is denoted as Cs_r. The set of messages a robot r is currently receiving is denoted as Cr_r. The function that determines when a communication message is sent is provided in the controller definition given below in Section 3.4.3.

3.4.3 Controller

Three functions define a robot r's behavior in the world, known collectively as the robot's *controller*. The controller is comprised of an action function, an internal state transition function, and a communication function.

The *action function*

$$Act(x, m, c, a) = Pr(A_r^t = a | X_r^t = x, M_r^t = m, Cr_r^t = c) \tag{3.4}$$

gives the probability that a robot r will execute action a provided it is receiving observation x, has an internal state value of m, and is receiving communication messages Cr.

When executed, the actions in the action function will change the state of the world. Each robot also has another set of independent actions, called *competency* actions that are potentially executed in the event no actions in the action function are executed. Competency actions do not change the state of the world when executed and are primarily used for actions such as local navigation, obstacle avoidance, or servoing to a designated target. It is important to note that since competency actions cannot change the state of the world, they alone are not important in the consideration of correct or incorrect task execution. Any action that can change the state of the world is, by definition, not a competency action and should therefore be present in the action function. Abstractly, a robot will execute competency actions by default with actions from the action function superseding when appropriate.

The *internal state transition function*

$$IState(x, m, c, m') = Pr(M_r^{t+1} = m' | X_r^t = x, M_r^t = m) \tag{3.5}$$

provides a robot r's internal state value at time $t + 1$ given that at time t it received observation x and its internal state value was m.

The *communication function*

$$Comm(x, c) = Pr(Cs_r^{t+1} = c | X_r^t = x) \tag{3.6}$$

provides the communication message transmitted by a robot r at time $t + 1$ given that at time t it received the observation x.

All the controller designs discussed in this dissertation share the same high-level design. Figure 3.3 provides the high-level controller design and shows the order in which the controller considers internal state transitions, sending and receiving of communication messages, and action selection.

The entries in the internal state, communication, and action functions are represented as an ordered set and are evaluated sequentially. The first activated entry is executed and no others are evaluated. In the event no entries in the action function are activated, the desired competency actions may be executed by default.

```
procedure Execute_Controller()
  m ← m_0
  repeat forever
    x ← current observation
    c_r ← communications being received
    if ∃m′(IState(x, m, c_r, m′)) > 0) then
      m ← m′ with prob. IState(x, m, c_r, m′)
    endif
    if ∃c(Comm(x, c) > 0) then
      send communication c with prob. Comm(x, c)
    endif
    if ∃a(Act(x, m, c_r, a) > 0) then
      execute action a with prob. Act(x, m, c_r, a)
    else
      execute desired competency actions
    endif
  endrepeat
end procedure Execute_Controller
```

Figure 3.3: High-level robot controller shared by all controllers presented in this dissertation, including the internal state transition function, communication function, and action function.

Not all controllers make use of all control features. For example, some do not use internal state, others do not use communication, and some do not use either. In such cases, the internal state transition function or communication function always return a value of 0 and are therefore irrelevant to the robot's control decisions.

3.5 Multi-Robot System Controller Taxonomy

Given the definition of a robot controller from Section 3.4.3, it is useful and convenient to taxonomize MRS based on characteristics of the controller each robot executes. Controllers are taxonomized based on the following three characteristics:

1) Deterministic or probabilistic action selection ($DAct$ or $PAct$)

2) Capable of maintaining persistent internal state or stateless (IS or $NoIS$)

3) Capable of inter-robot communication or non-communicative ($Comm$ or $NoComm$).

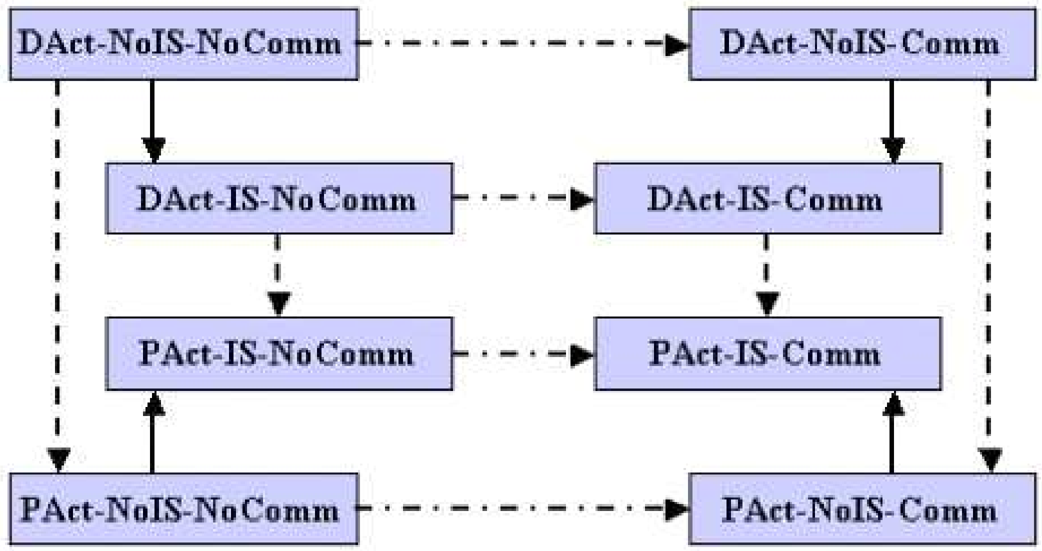

Figure 3.4: Diagram of the multi-robot systems controller taxonomy with the arrows representing the ways in which the synthesis methods presented in this dissertation transform one controller type into another.

Therefore, the label for a MRS composed of stateless, non-communicative robots utilizing deterministic action section would be DAct-NoIS-NoComm. The label for a MRS composed of non-communicative robots utilizing probabilistic action section which maintain persistent internal state would be PAct-IS-NoComm.

The taxonomy axes are in no way meant to be complete in the sense that they can precisely describe and characterize all unique MRS. Rather, the axes are defined in such a way as to provide a useful characterization of the controller classes of MRS discussed in the remainder of this dissertation. Chapter 4.2 presents systematic synthesis procedures to automatically construct MRS controllers sampled from this taxonomy.

3.5.1 Deterministic vs. Probabilistic Action Selection

The first axis of the taxonomy describes the nature of the individual robot's action function. Robots with deterministic action functions and robots with probabilistic action functions are considered. The action function *Act*, as defined in Section 3.4.3, is deterministic if

$$\forall x \forall m \forall c \forall a (Act(x, m, c, a) = 0 \lor Act(x, m, c, a) = 1)). \tag{3.7}$$

A probabilistic action function is one for which

$$\forall x \forall m \forall c \forall a (Act(x, m, c, a) \rightarrow [0, 1]). \tag{3.8}$$

3.5.2 Maintenance of Persistent Internal State vs. Stateless

The second axis of the taxonomy is whether the individual robots independently maintain persistent internal state. As described in Section 3.4.1, internal state is a temporally persistent memory that can be modified by the robot over time and is used in control decisions.

3.5.3 Communicative vs. Non-Communicative

The third axis of the taxonomy is whether the robots are able to explicitly communicate with each other. As described in Section 3.4.2, it is not required that communication be direct or one-to-one nor is it required for the receiver to be able to uniquely identify the sender.

3.6 Summary

This chapter has defined the framework which serves as the formal foundation for our MRS design methodology. Necessary definitions and notations that describe the world in which the

MRS is situated, the task the MRS is to perform, and the characteristics of the individual robots constituting the MRS have been provided.

Procedures can now be designed to systematically reason within the context of this formalism to analyze and synthesize MRS controllers in a principled fashion. A set of such principled controller analysis and synthesis methods represent the core of this dissertation and are presented in the next Chapter, in Sections 4.1 and 4.2, respectively.

Chapter 4

Principled Multi-Robot System Controller Design

This chapter presents the core of the principled MRS controller design methodology – the controller analysis and synthesis methods, which share a common formal foundation, as presented in Chapter 3. This shared foundation allows close integration of these methods and represents a novel controller design approach.

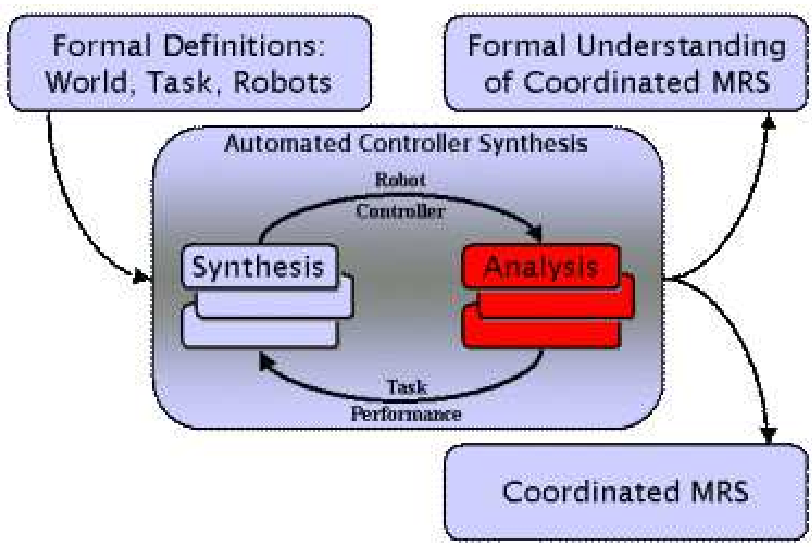

Figure 4.1: Diagram of principled multi-robot systems controller design methodology with the role of analysis highlighted.

This chapter is organized as follows. A probabilistic microscopic method of analysis is described in Section 4.1 and a suite of systematic controller synthesis methods are described in Section 4.2.

4.1 Controller Analysis

This section outlines the role of analysis in the controller design process and details an analysis method using a probabilistic microscopic modeling approach. Figure 4.1 shows how analysis fits into the principled controller design methodology. Analysis takes as input a controller, as defined in Section 3.4.3, provided by controller synthesis methods and outputs the predicted system performance of that controller design. As the figure demonstrates, due to the formal framework shared by analysis and synthesis, as well as their integrated nature, an iterated synthesis and analysis process is undertaken to optimize the final controller design.

4.1.1 Role of Analysis in the Controller Design Process

The design of an effective task-directed MRS is often difficult because typically there is a lack of an accurate understanding of the relationship between different design options and resulting task performance. In the common trial-and-error design process, the designer constructs a MRS and tries it out in either simulation or on physical robots. Either way, this process is resource-intensive as it requires much effort and time to evaluate many possible designs. Ideally, the designer should be equipped with an analytical tool for the principled evaluation of potential designs. Such an analytical tool would allow fast and efficient evaluation of different design options and thus facilitate more effective MRS through the principled design process that includes effective optimization with respect to desired performance metrics.

Used independently, analysis can be used as predictive tools to understand the expected system performance of a given design. It provides a means to determine where and why a given design fails or does not reach acceptable performance on a given task. This understanding can be used to provide formal reasons for the use and limitations of different possible designs. Used in an integrated manner with the controller synthesis methods presented in Section 4.2, analysis plays an important role in the synthesis of optimized MRS. Analysis shares a common formal framework with the synthesis methods, and, as such, they can be used to improve the design process by aiding in decisions related to the appropriate use of internal state or inter-robot communication and their trade-offs with resulting system task performance. A design methodology utilizing unified controller synthesis and analysis methods allows for a more systematic, efficient, and complete exploration of the design space, thereby leading to more effective design solutions.

4.1.2 Probabilistic Microscopic Model

This section describes a *probabilistic microscopic* modeling approach of analysis that is used in the controller design methodology. The approach is *probabilistic* in that the robots are modeled as parallel processes operating in a common environment that is modeled as a series of stochastic events. The approach is *microscopic*, as opposed to *macroscopic*, in that individual robots are modeled as separate processes and the outcome of the model is a result of the interactions among robots and between the robots and the world. In related work on probabilistic microscopic modeling, the work of Ijspeert et al. [36] and Martinoli et al. [52] provided definitions and examples on the use of probabilistic microscopic modeling approaches as applied to stick-pulling and aggregation domains, respectively, as discussed in Chapter 2.

Alternative modeling approaches include *macroscopic* models [49], which treat the group of robots as an indivisible whole without decomposing it into individual robots. In such approaches the system-level behavior is directly derived without consideration of specific robot interactions. When available, macroscopic modeling approaches are very useful; however, they are frequently computationally complex and fail to capture important local and dynamic interactions among the robots. As was discussed in Section 1.4, understanding the interactions among robots is key to understanding MRS design is related to resulting task performance. For this reason, a microscopic modeling approach is deemed most suitable and is utilized in this dissertation.

4.1.3 Model Definition

An overview of the probabilistic microscopic model used in this dissertation is detailed in Figure 4.2. This method, as implemented in the deterministic procedure *Probabilistic_Microscopic_Model*, takes as input a world W, a task T, and a set of robots R. All robots in R execute identical controllers consisting of an action function *Act*, an internal state transition function $IState$, and a communication function $Comm$, the operation of which was described in Section 3.

At a high-level, the procedure operates by performing independent Monte Carlo trials of the task T. A single task trial continues until the task is successfully completed or until an an incorrect action occurs and the task terminates unsuccessfully. Through repeated task trials, the *probability* of correct task execution is determined.

The procedure begins by initializing two counters, *count_correct* and *count_incorrect*, that represent the number of correctly and incorrectly executed task trials, respectively. The remainder of the procedure (lines 4-33) is then repeated with each iteration representing a single task trial.

Each individual task trial begins by initializing the internal state values of all robots and then initializes the world state s to be the starting state u_start_T of the task T to be executed. Next, a loop (lines 7-32) is entered that is repeated until either the world state becomes a terminal state of task T, u_term_T, or until an incorrect action is executed. In the former case, the task trial is terminated and a value of *correct* task execution is recorded. In the later case, the trial is terminated and a value of *incorrect* task execution is recorded.

This loop begins by providing the robots with a random observation set drawn from the weighted set of observations possible in the current world state. The weight of observation x is defined by the observation function $O(s, x)$. Next, in lines 9-13, using this observation the internal state transition function of each robot is independently evaluated and internal state transitions are made if necessary.

In lines 14-20, the communication function of each robot is then independently evaluated and each robot sends any appropriate communication message, as defined by the communication function *Comm*. Each message sent is received by all robots and placed in the set Cr_r for each robot r. This set of received communication messages is cleared during each cycle of this loop so the robots have no memory of past communication messages received.

Next, each robot's action function is evaluated and an action is selected and executed. Assuming the current world state is s, for each robot r's action a_r the world state is transitioned according to the world state transition function $P(s, a_r, s')$.

If the set of edges for task T, E_T, contains an edge from the current world state, s, to the new state, s', then the entire process repeats and all robots are provided with a new observation. If, however, the edge is *not* in E_T, then that means the actions executed by the robots resulted in incorrect task execution as this transition caused a deviation from the world state space trajectory as defined by the DAG for task T. In this case of incorrect task execution, this trial is terminated with a value of *incorrect task execution*. Alternatively, if the next world state is a member of the set of terminal states for T, u_term_T, then that means the task has been successfully executed. The current task trial is then terminated with a return value of *correct task execution*. Individual task trials are executed until the change in the estimated probability of correct task execution falls below some threshold value ϵ.

4.1.4 Summary

This section presented a probabilistic microscopic modeling approach for controller analysis. A procedure was presented that returns the probability of correct task execution of a set of robots R executing a task T in a world W. The synthesis methods presented in the following sections

```
procedure Probabilistic_Microscopic_Model(world W, task T, robots R)
  count_correct ← 0
  count_incorrect ← 0
  repeat until (Δ(count_correct (count_correct + count_incorrect) < ε)
      Initialize robots
    ∀r ∈ R(m_r ← i_0)
      Initialize world
    s ← u_start_T ∈ T
    repeat (until the task is successfully completed or until incorrect task execution occurs)
        Each robot receives an observation from the world
      X ← x, where x is a set of observations drawn from the weighted set defined by O(s, x)
        Each robot makes appropriate internal state transitions
      for all robots r ∈ R do
        if ∃m ∈ M(IState(m_r, x_r, m) = 1) then
          m_r ← m
        endif
      endfor
        Each robot sends any appropriate communication messages that
        are then received by all robots
      ∀r ∈ R(Cr_r ← {})
      for all robots r ∈ R do
        if ∃c ∈ C(Comm(x_r, c) = 1) then
          Cs_r ← c
          ∀r ∈ R(Cr_r ← Cr_r ⋃ c)
        endif
      endfor
        Each robot executes any appropriate actions
      A ← ⋃_{∀r∈R} a where a is drawn from the weighted set Act(X_r, M_r, Cr_r, a)
        The world state is transitioned appropriately
      new_s ← (s′ where s′ is drawn from the weighted set P(s, A, s′)

        Check to see if the world state transition resulted in incorrect task execution
      if {s, new_s} ∋ E_T then
        count_incorrect ← count_incorrect + 1
        exit inner repeat loop
      endif

        Transition the world state
      s ← new_s

        Check to see if a terminal state has been reached signifying correct task execution
      if s ∈ u_term_T then
        count_correct ← count_correct + 1
        exit inner repeat loop
      endif
    end repeat
  end repeat
  return(count_correct/(count_correct + count_incorrect))
end procedure Probabilistic_Microscopic_Model
```

Figure 4.2: Probabilistic microscopic model algorithm for a set of robots R executing a task T in a world W.

utilize the knowledge gained through the use of the probabilistic microscopic modeling approach to iteratively improve the controller design.

In this method of analysis, the required number of task trials is dependent on many factors, including the nature of the observation function O, the uncertainties in sensing and action, any probabilistic components of the robot controllers, and the desired accuracy of the results. Due to these domain specific features, it is not reasonable to determine the number of task trials to run *a priori*. Experimentally, task trials are performed until the change in the estimated probability of correct task execution stabilizes.

4.2 Controller Synthesis

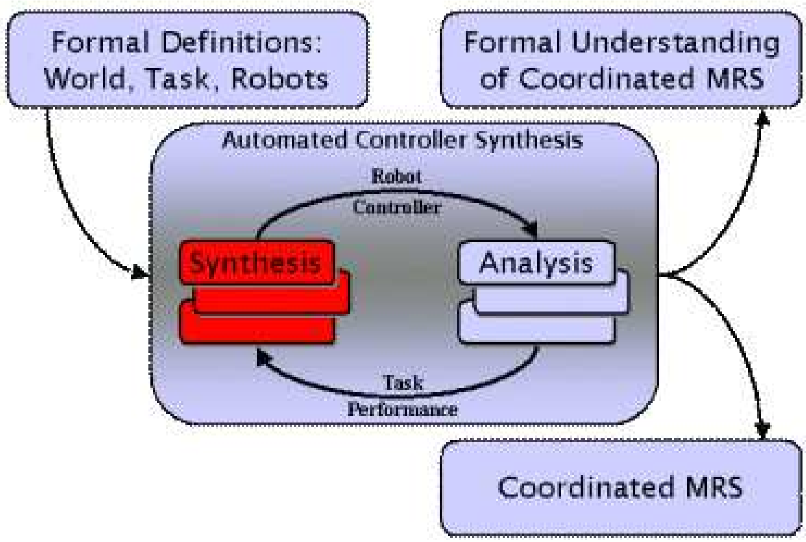

Figure 4.3: Diagram of principled multi-robot systems controller design methodology with the role of synthesis highlighted.

This section introduces the role of synthesis in the controller design process and details a suite of systematic controller synthesis methods that use the formalism presented in Chapter 3 in a *prescriptive* fashion. Synthesis is the process of constructing a specific instance of a controller that meets design requirements such as achieving the desired level of task performance while simultaneously satisfying any constraints imposed by limited robot capabilities.

Each of the presented synthesis methods builds a unique robot controller that utilizes a different combination of control features from the following set:

- deterministic or probabilistic action selection
- the maintenance of internal state or stateless
- the use of inter-robot communication or non-communicative

Figure 4.3 shows the role of synthesis within the principled controller design methodology.

4.2.1 Role of Synthesis in the Controller Design Process

Typically, robot controllers are synthesized by hand, relying on the expert knowledge of the designer and an intimate understanding of how the robots will interact with each other and with the world during task execution. The approach presented in this dissertation is novel in that it represents systematic and automated controller synthesis methods. The designer's responsibility is reduced to defining the task domain using the formal framework presented in Section 3, information a hand-designed system inherently requires as well. Using this formal definition of the task domain, the synthesis methods automatically produce a complete robot controller that may be executed by all robots in the MRS to achieve task-directed system-level coordination.

The synthesis methods presented in this section are specific to MRS, as defined in Section 3.4, performing acyclic tasks of the type defined in Section 3.3. Even given the constraints on the types of systems imposed by these definitions, there are many more unique synthesis methods than those presented in this dissertation. However, the presented methods represent a limited but important sampling of the space of possibilities and provide a means to study the formal synthesis of a range of MRS using a variety of common robot coordination and control features, such as probabilistic action selection, the maintenance of persistent internal state, and the use of inter-robot communication. It should be noted that this work does not consider controllers which use *both* internal state *and* communication. Controllers that could use both would indeed be more expressive and powerful; however, the study of controllers which use only one or neither of the two still provide much space for the formal investigation of the uses and limitations of internal state and communication in MRS.

Synthesis methods for four different controllers, as defined by the taxonomy described in Section 3.5, are presented in the following sections. A synthesis method for DAct-NoIS-NoComm controllers is presented in Section 4.2.2, a method for DAct-IS-NoComm controllers is presented in Section 4.2.3, a method for DAct-NoIS-Comm controllers is presented in Section 4.2.4, and a method for PAct-NoIS-NoComm controllers is presented in Section 4.2.5.

4.2.2 Synthesis Method for Deterministic Action Selection, No Internal State, No Communication Controllers (DAct-NoIS-NoComm)

The first, and simplest, synthesis method is for DAct-NoIS-NoComm controllers, which have deterministic action functions (DAct) and are stateless (NoIS) and non-communicative (NoComm). This section defines and discusses a systematic method for the synthesis of DAct-NoIS-NoComm controllers.

Although DAct-NoIS-NoComm controllers are quite simple, they represent a useful class of controllers to study. Through a formal understanding of the limitations of such controllers in achieving system-level coordinated behavior, one can more fully understand and appreciate the benefits of additional control features, such as internal state or communication, as will be discussed in later sections. Furthermore, the DAct-NoIS-NoComm controller serves as a foundation upon which the controller synthesis methods presented in the following sections are built.

Unencumbered by the need to define an action probability values, an internal state transition function, or a communication function, the DAct-NoIS-NoComm synthesis method only requires

(1) procedure Synthesize_DAct-NoIS-NoComm_Controller(world W, task T, robot r)
Initialization
(2) for all $x \in X_r, m, m' \in M_r$ do
(3) $IState(m, x, m') = 0$;
(4) endfor
(5) for all $x \in X_r, c \in C_r$ do
(6) $Comm(x, c) = 0$;
(7) endfor
(8) for all $x \in X_r, m \in M_r, c \in C_r, a \in A_r$ do
(9) $Act(x, m, c, a) = 0$;
(10) endfor

Construct the action function
(11) for all $\{u, v\} \in E_T, x \in X_r, a \in A_r(O(u, x) > 0 \wedge P(u, a, v) > 0)$ do
(12) $Act(x, *, \{*\}, a) = 1$;
(13) endfor
(14) end procedure Synthesize_DAct-NoIS-NoComm_Controller

Figure 4.4: Procedure for synthesizing a DAct-NoIS-NoComm controller for a robot r to execute a task T in a world W.

the straight-forward process of defining the deterministic action function *Act*. The action function definition is simply a direct mapping of observations to actions, information that can be extracted from the formal definitions of the world and task. The procedure *Synthesize_DAct-NoIS-NoComm_Controller* shown in Figure 4.4 provides a systematic procedure for the synthesis of DAct-NoIS-NoComm controllers.

There are two high-level steps to this procedure: **1)** initialize the controller as a blank slate, and **2)** construct the controller action function by appropriately mapping observations to actions.

4.2.2.1 Initializing a Blank Slate Controller

In lines 2-10 of Figure 4.4, the internal state transition function, communication function, and action function are initialized by setting the probability of each internal state transition, communication, and action to zero for all circumstances. After this stage, the controller is a blank slate with no chance of any internal state transitions, communications, or actions being executed.

4.2.2.2 Constructing the Action Function

In lines 11-13 in Figure 4.4, for each $\{u, v\} \in E_T$, the synthesis procedure adds a rule to the action function of the form $Act(x, *, \{*\}, a) = 1$ such that x is observable in u and $P(u, a, v) > 0$. As a DAct-NoIS-NoComm controller is stateless and non-communicative, the internal state values and received communication messages in the action function are set to "don't care", represented as "$*$".

4.2.2.3 Discussion

Section 4.2.2 introduced a method for the synthesis of DAct-NoIS-NoComm controllers. The synthesis method has a complexity of $O(E_T AX)$, where E_T is the set of edges in the task T's DAG, A_r is the set of all actions executable by a robot r, and X_r is the set of observations that can be made by a robot r. It is important to note that this method scales with the number of states in the *task*, not the number of states in the world.

A primary point of failure for DAct-NoIS-NoComm controllers is in situations involving perceptual aliasing. Perceptual aliasing is the condition in which some observation x can be made in two different task states s_i and s_j. As long as there does not exist some action a in which x and a are correct for s_i but not for s_j, then there are no problems. If such a situation does exist, then the action a will be appropriately executed in s_i but may be incorrectly executed in s_j. The probability of correct task execution in such situations is dependent on the relative probabilities of observing x in s_j and observing an observation z in s_j, where z and some action b are correct for s_j. It is in such situations that persistent internal state or inter-robot communication can, in many cases, be used to increase the probability of correct task execution − a feature exploited in the synthesis methods presented in the following sections.

The DAct-NoIS-NoComm synthesis method does not guarantee an optimal probability of correct task execution. The method maps all observations and actions that have *any chance* of being correct for a given task state. Other options exists, for example, one may decide to only add to the action function the observation and action pairs that have a probability of being correct greater than some value α, or only the observation and action pair with the absolute highest probability of being correct. Such modifications that remove potentially correct observation and action pairs will likely result in increased times to task completion but could result in a higher overall probability of correct task execution as observations and actions prone to uncertainties could be ignored. This is primarily a trade-off between robustness and time efficiency that the designer will need to make with respect to the desired system performance.

4.2.3 Synthesis Method for Deterministic Action Selection, Internal State, No Communication Controllers (DAct-IS-NoComm)

This section presents a systematic method for the synthesis of DAct-IS-NoComm controllers, which have deterministic action functions (DAct), maintain persistent internal state (IS), and are non-communicative (NoComm). Using the DAct-NoIS-NoComm controller as a foundation, which Section 4.2.2 describes how to synthesize, the synthesis method in this section augments it with the use of internal state in order to improve task performance. A discussion on the DAct-IS-NoComm synthesis method in Section 4.2.3.5.

There are four high-level steps in the synthesis procedure: **1)** synthesize a baseline DAct-NoIS-NoComm controller in line 2, **2)** initialize relevant variables in line 3, **3)** introduce internal state to address instances of perceptual aliasing in lines 5-9, and **4)**, in lines 9-14, introduce internal state at task branches to allow robots to distinguish between different task trajectories. The full synthesis method is given by the procedure *Build_DAct-IS-NoComm_Controller* shown in Figure 4.5.

procedure Build_DAct-IS-NoComm_Controller()

Step 1: Synthesize a baseline DAct-NoIS-NoComm controller

Build_DAct-NoIS-NoComm_Controller();

Step 2: For each task state, identify the paired observations and actions used in the DAct-NoIS-NoComm action function

$X_a(s_i) = \{\{x_0, a_0\}, ..., \{x_n, a_n\}\}$ where for each $\{x_i, a_i\}$ $\exists m(Act(x, m, \{\}, a) > 0)$;

Consider all paths of correct task execution

for all paths p from u_start_T to a state in u_term_T do

Step 3: Identify uses of internal state to deal with observations involved in perceptual aliasing

for all $s_i, s_j \in p$ where $((i < j) \wedge \exists x \exists a(\{x, a\} \in X_a(s_j) \wedge O(s_i, x) > 0 \wedge \{x, a\} \ni X_a(s_i))$

$Z = \{z_0, z_1, ..., z_n\}$ where for all $z_i \in Z$,
$\exists s_k(i < k < (j+1) \wedge O(s_k, z_i) > 0 \wedge \exists s_a \in p(a \leq i \wedge O(s_a, z_i) > 0))$;

use $Probabilistic\_Microscopic\_Model$ to pick a $z_i \in Z$ to use to transition the internal state to a novel value that maximizes the probability of correct task execution up to state s_j;

project changes forward for $IState$ and Act functions to reflect this internal state transition;

endfor

Step 4: Identify branches in the task definition and insert an internal state transition to distinguish the internal state values across branches

for all $s_i \in p$ where the out degree of $s_i > 1$ do

$Z = \{z_0, z_1 ..., z_n\}$ where for all $z_i \in Z(O(s_{i+1}, z_i) > 0 \wedge \nexists s_a(a < s_{i+1} \wedge O(s_a, z_i) > 0))$;

use $Probabilistic\_Microscopic\_Model$ to pick a $z_i \in Z$ to use to transition the internal state to a novel value that maximizes the probability of correct task execution up to state s_j;

project changes forward for $IState$ and Act functions to reflect this internal state transition;

endfor

endfor

end procedure Build_DAct-IS-NoComm_Controller

Figure 4.5: Procedure for synthesizing a DAct-IS-NoComm controller for a robot r to execute a task T.

4.2.3.1 Synthesizing a Baseline DAct-NoIS-NoComm Controller

The DAct-IS-NoComm controller synthesis method uses a DAct-NoIS-NoComm controller as a starting point. The synthesis method for DAct-NoIS-NoComm controllers was presented in Chapter 4.2.2 by the function *Synthesize_DAct-NoIS-NoComm_Controller* shown in Figure 4.4. In the remaining three steps, we integrate an internal state transition function into the baseline DAct-NoIS-NoComm controller to improve system-level coordination.

4.2.3.2 Initializing Variables

In line 3, the variable $X_a(s_i)$ is initialized. This variable is a set of observations and actions pairs, $\{x, a\}$, for which there is a rule in the DAct-NoIS-NoComm controller action of the form $Act(x, m, \{\}, a) = 1$, for some internal state value m. The observation and action pairs constituting $X_a(s_i)$ are defined to be correct for task state s_i.

4.2.3.3 Internal State and Perceptual Aliasing

The remaining two steps of the synthesis procedure are repeated for all paths, p, through the task definition constituting correct task execution – all trajectories through the task DAG from u_start_T to all u_term_T.

Lines 5-9 of Figure 4.5 identify *troublesome* instances of perceptual aliasing that may occur in the states of path p. An instance of perceptual aliasing is troublesome if the observation x, given some task state s_j and some action a, $\{x, a\} \in X_a(s_j)$, but there also exists another task state, $s_i \in p$, where $\{x, a\} \ni X_a(s_i)$. This means that the observation x is mapped to the execution of an action that is meant to be performed in task state s_j, but this observation can also occur in task state s_i that should *not* be mapped to the same action.

There may also be other instances of perceptual aliasing that are *not* troublesome. In such cases, an observation can be made in multiple task states, but this observation is not mapped to an action in the action function and is therefore of no impact on the performance of the task.

In order to distinguish between different occurrences of an observation x that is the cause of a troublesome case of perceptual aliasing, an internal state transition is introduced between the instances of its observation. Therefore, each observation of x will occur when the robot has a *different* internal state value and the action function can be arranged to perform different actions upon the observations of x.

An internal state transition is introduced through the identification of an observation that, during task execution, will occur between the multiple observations of x. A new rule is then added to the internal state transition function that uses this selected observation to cause a change in the robot's internal state value. In many cases, there is more than one such observation that can be used to transition the internal state value. To select the most appropriate observation for each candidate observation, the action and internal state transition functions can temporarily be modified. These modified functions constituting the robot controller can then be evaluated using the probabilistic microscopic modeling approach presented in Section 4.1.2. The internal state transition that results in the controller giving the highest probability of correct task execution is then permanently introduced to the robot controller.

4.2.3.4 Internal State and Task Branches

Lines 10-14 in Figure 4.5 identify task states for which the task trajectory could possibly branch to a task state *not* in p. In order to maintain consistency in the controller across these branches in the task, it is necessary to introduce an internal state transition immediately after the branch. To do this, an observation is identified in the task state immediately after the branch. The internal state transition function is updated in order to use this observation to transition the internal state value to a new value.

The resulting controller will have an internal state transition immediately after each branch in the task that is inherently used to identify which branch was taken during task execution. This prevents incorrect task execution from occurring through perceptual aliasing across different task branches since each branch is uniquely identified through this internal state value.

4.2.3.5 Discussion

It should be clarified that although reasoning with specific task states is used in the synthesis procedure, the resulting action and internal state transition functions do not perform any explicit reasoning of underlying specific states. The robots are not directly provided with information on the current task state, which is the cause of perceptual aliasing and a reason why the use of internal state is useful to increase the probability of correct task execution.

The DAct-IS-NoComm synthesis method represents only one way in which internal state can be used in a robot controller. There are certainly other ways, including ones which maintain more of a direct representation of the world state or task progress. However, the method by which internal state is used in this DAct-IS-NoComm synthesis method can improve task performance in many situations and provides a concrete technique for the use of internal state that allows for principled MRS synthesis and analysis.

This section described how the use of internal state can be used to improve task performance. The next section introduces the use of inter-robot communication to improve task performance.

4.2.4 Synthesis Method for Deterministic Action Selection, No Internal State, Communication Controllers (DAct-NoIS-Comm)

This section presents a systematic method for the synthesis of DAct-NoIS-Comm controllers, which have deterministic action selection (DAct), are stateless (NoIS), and have the capability of inter-robot communication (Comm). Using the DAct- NoIS-NoComm controller as a foundation, which Section 4.2.2 describes how to synthesize, the DAct-NoIS-Comm synthesis method in this section will augment it with the use of inter-robot communication in order to improve task performance.

The synthesis method also includes a graph coloring approach used to minimize the number of unique communication messages required.

There are four high-level steps in the synthesis process: **1)** synthesize a baseline DAct-NoIS-NoComm controller, **2)** identify situations in which communication can be used to better facilitate coordination, **3)** assign specific communication messages to each of these situations using a graph coloring approach, and **4)** appropriately augment the action and communication functions provided by the baseline DAct-NoIS-NoComm controller. The full synthesis process is given by the procedure *Build_DAct-NoIS-Comm_Controller* shown in Figure 4.6.

4.2.4.1 Synthesizing a Baseline DAct-NoIS-NoComm Controller

A DA-NoIS-NoComm controller, which is simply a stateless, non-communicative controller, serves as a baseline controller that is augmented with communication to synthesize a DAct-NoIS-Comm controller. The process of synthesizing a DAct-NoIS-NoComm controller is given by the procedure *Build_DAct-NoIS-NoComm_Controller*, shown at the top of Figure 4.6.

Such a baseline DAct-NoIS-NoComm controller leaves room for error if x and a are correct for some task state s_i but there exists another task state s_j where x and a are *not* correct and $O(s_j, x) > 0$. In such situations, an MRS composed of robots with DAct-NoIS-NoComm controllers cannot enforce the action sequence necessary for correct task execution. This is a common problem with purely reactive controllers [15] in task domains with temporal constraints. In the DAct-NoIS-Comm synthesis steps that follow, the use of communication is incorporated to improve coordination in these situations. Due to sensing and action uncertainty, the addition of communication cannot guarantee correct task execution, but it can be used to *increase the probability* of correct task execution.

procedure Build_DAct-NoIS-Comm_Controller(world W, task T, robot r)
  Step 1: Synthesize a baseline DAct-NoIS-NoComm controller
  Build_DAct-NoIS-NoComm_Controller();

  Step 2: Identify situations in which communication can be used to improve task performance
  for all $s_i \in V_T$ do
    $X_c(s_i) = \{\}$;
    for all x s.t. $O(s_i, x) > 0 \wedge \nexists s_k(s_k \neq s_i \wedge O(s_k, x) > 0)$ do
      $X_c(s_i) = X_c(s_i) \bigcup x$;
    endfor
    $X_a(s_i) = \{\}$;
    for all x s.t. $O(s_i, x) > 0 \wedge \exists a(Act(x, *, \{\}, a) > 0 \wedge P(s_i, a, s_{i+1}) > 0)$ do
      $X_a(s_i) = X_a(s_i) \bigcup x$;
    endfor
    if $X_a(s_i) \subseteq X_c(s_i)$ then
      $X_m(s_i) = \{\}$;
    else
      $X_m(s_i) = X_c(s_i)$;
    endif
  endfor

  Step 3: Assign specific communication messages to each of these situations using a graph coloring approach
  $Assigned\_Comm \leftarrow$ Graph_Color($\bigcup_{\forall s_i \in V_T} \{pickx \in X_m(s_i)\}$);

  Step 4: Appropriately augment the action and communication functions provided by the baseline DAct-NoIS-NoComm controller
  for all $s_i \in V_T$ do
    $c = \{\}$;
    for all $x \in X_m(s_i)$ do
      $Comm(x, Assigned\_Comm(x)) = 1$;
    endfor
    for all $x \in X_a(s_i), a \in A$ s.t. $(Act(x, *, \{\}, a) = 1)$ do
      $Act(x, *, \{\}, a) = 0$;
      $Act(x, *, \{c\}, a) = 1$;
    endfor
  endfor
end procedure Build_DAct-NoIS-Comm_Controller

Figure 4.6: Procedure for synthesizing a DAct-NoIS-Comm controller for a robot r to execute a task T in a world W.

4.2.4.2 Identifying Uses of Communication

Next, a set of observations is defined that will serve as the basis of the DAct-NoIS-Comm controller's communication function. For each task state $s_i \in V_T$, a set of observations $X_c(s_i)$ is defined (Figure 4.6, lines 3-6), each of which can *only* occur in state s_i. Also defined is $X_a(s_i)$, a set of observations such that, for each $x \in X_a(s_i)$, there exists an action a where the action function maps x to a and the result of that action is a task state that is a child of s_i.

If all observations in $X_a(s_i)$ are also in $X_c(s_i)$, then a controller directly mapping observations to actions required for this task state is sufficient and improvements to the DAct-NoIS-NoComm controller regarding this task state are not necessary. If, however, there exists an observation in $X_a(s_i)$ that is not in$X_c(s_i)$, then perceptual aliasing with regards to this observation is possible and could result in incorrect task execution. In this case, the observations in the set $X_c(s_i)$ can be used to uniquely identify state s_i and overcome the perceptual aliasing issue. Lines 12-16 perform this evaluation and place the set of observations that can be used to uniquely identify task state s_i in the set $X_m(s_i)$, if any are required.

If $\exists s_i, s_j \in V_T$ such that $\{\forall x(O(s_i, x) > 0)\} \subseteq \{\forall x(O(s_j, x) > 0)\}$, then the state s_i is fundamentally unobservable. In such a situation, one cannot guarantee that a MRS composed of robots executing a DAct-NoIS-Comm controller will correctly execute the task, even in the absence of sensing and action uncertainty.

4.2.4.3 Graph-Coloring Approach to the Assignment of Specific Communication Messages

The previous step identified observations, contained in the set $X_m(s_i)$ for each task state s_i, that may be used to disambiguate the underlying task state when necessary. In this step, for each task state s_i the best observation is chosen from $X_m(s_i)$ and mapped to a communication message. The *best* observation from each set is chosen through constructing a temporary communication function mapping each observation individually to a communication message and updating the action function appropriately. This controller is then evaluated using the probabilistic microscopic model presented in Section 4.1.2. Each observation is evaluated in this manner and the one that results in the maximum probability of correct task execution is chosen. This process is repeated for each task state and resulting in a set of observations that are to be mapped to communication messages in the communication function.

The simplest mapping for all of these observations is to assign a specific, unique communication message to each observation. However, in many MRS communication bandwidth can be quite limited, such as in a MRS composed of underwater vehicles or in MRS in which communication is through through means such as the release of chemical or light signals. In such cases it is advantageous to minimize the number of bits transfered in each communication message. The more unique communication messages required, the more bits of information in each communication message must contain in order to uniquely identify each message.

Therefore, to minimize the actual number of *unique* communication messages needed, a *graph coloring approach* is used. Graph coloring is not used to minimize the *number of instances* in which communication is used, which is decided by the process in Step 2. Although graph coloring is NP-complete, there are a number of well-studied heuristics that provide understood bounds on resulting solution quality [57]. Furthermore, the graph coloring approach is desirable in many domains to reduce the number of unique messages required, but it is not absolutely required as the direct assignment of unique messages to each necessary observation is acceptable.

The problem of assigning unique communication messages to a given set of observations $O_c = \bigcup_{\forall s_i \in V_T} \{X_m(s_i)\}$ can be reduced to a graph coloring problem as follows. First, a graph G, consisting of a set of vertexes V_G and a set of edges E_G, both initially empty, is created. Next, a vertex is added to V_G for each observation in O_c. Then, edges are added to E_G between each pair of vertexes in V_G for which the associated observations interfere with each other. The test for interference between two observations x_i and x_j is given by the function $I(x_i, x_j)$ shown in Equation 4.1. Now a standard graph coloring algorithm [57] may be applied to G in which the color assigned to a vertex in V_G corresponds to a specific communication message assigned to the observation represented by that vertex. The function $Assigned\_Comm(x)$ as used in Figure 4.6 returns the communication message assigned to the observation x as a result of the graph coloring process.

$$I(x_i, x_j) = \begin{cases} 1, & \text{if } \exists s \in V_T(O(s,x_i) > 0 \wedge O(s,x_j) > 0), \\ 1, & \text{if } \exists s_u, s_v \in V_T \exists x \in X_a(s_v)(O(s_u,x) > 0 \wedge x_j \in X_m(s_v) \wedge O(s_u,x_i) > 0), \\ 1, & \text{if } \exists s_u, s_v \in V_T \exists x \in X_a(s_v)(O(s_u,x) > 0 \wedge x_i \in X_m(s_v) \wedge O(s_u,x_j) > 0), \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

4.2.4.4 Constructing the Action and Communication Functions

The DAct-NoIS-Comm controller is synthesized by augmenting the DAct-NoIS-NoComm controller synthesized in Step 1. This is accomplished by adding the communication function and appropriately modifying the action function such that an action is not executed unless all necessary communications are being simultaneously received.

Through the graph coloring approach presented in Step 3, a specific communication message was assigned to each observation in the set $X_m(s_i)$ for each $s_i \in V_T$. The controller communication function is constructed (Figure 4.6, lines 22-24) by adding a communication rule for all $x \in X_m(s_i)$ for all $s_i \in V_T$ of the form $Comm(x, Assigned\_Comm(x)) = 1$, where $Assigned\_Comm(x)$ is the specific communication message assigned to the observation x in Step 3. Such a communication rule will cause the robot to send the communication $Assigned\_Comm(x)$ every time the observation x is made. The action function is modified (Figure 4.6, lines 25-29) so that for each rule of the action function, $Act(x, *, \{\}, a) = 1$, where x and a are correct for a state s_i is modified to become $Act(x, *, \{c_r\}, a) = 1$, where c_r is the set of specific communication messages mapped in Step 3 to the observations in $X_m(s_i)$. All probabilities not explicitly declared for the controller are 0.

4.2.4.5 Discussion

The previous section introduced a method for the synthesis of DAct-NoIS-Comm controllers. A controller synthesized by this method is only one, and certainly not the only, way in which communication can be used to facilitate coordination. In fact, from a pragmatic standpoint, a MRS composed of robots executing such DAct-NoIS-Comm controllers has many disadvantages that other forms of communicative MRS may not exhibit. For example, the efficiency of the MRS in terms of time to task completion is likely to be quite poor, as several events must take place simultaneously before actions can be performed.

A part of this issue stems from the fact that DAct-NoIS-Comm controllers are stateless. Using communication alone to execute a task that has branches in its definition is not effective as the robots have no memory of *which* branch of the task was taken. If the actions that need to be performed are conditioned on some past action, then stateless robots will have no knowledge of this past action. Allowing the robots to retain some form of non-transient internal state or representation would likely improve the system performance in a number of respects.

However, from the perspective of identifying and understanding the fundamental requirements of coordination, MRS composed of stateless, communicative robots are very informative. By isolating the use of communication, analysis of such MRS provides a means to better understand when and why communication is able to facilitate coordination and when it is insufficient. Knowledge of the limitations of communication helps identify when and why the integration of other controller features, such as internal state, becomes necessary.

4.2.5 Synthesis Method for Probabilistic Action Selection, No Internal State, No Communication Controllers (PAct-NoIS-NoComm)

This section presents a genetic algorithm-based method for the synthesis of PAct-NoIS-NoComm controllers, which have probabilistic action selection(PAct), are stateless (NoIS), and lack the capability for inter-robot communication (NoComm). Using the DAct-NoIS-NoComm controller as a foundation, which Section 4.2.2 describes how to synthesize, the PAct-NoIS-NoComm synthesis method presented in this section modifies the DAct-NoIS-NoComm action function by transforming it from a deterministic to a probabilistic function.

Introducing a probabilistic action function can help to achieve a higher probability of correct task execution over the deterministic DAct-NoIS-NoComm controller. Let us assume that observation x_i and action a_i are correct for some task state s_i and that observation x_j and action a_j are correct for some other task state s_j. In this case, the action function of the DAct-NoIS-NoComm controller would contain $Act(x_i, *, \{\}, a_i) = 1$ and $Act(x_j, *, \{\}, a_j) = 1$. However, if $O(s_j, x_i) > 0$ then there is a perceptual aliasing issue and the controller may execute an incorrect action because the action a_i could potentially be incorrectly executed in task state s_j. However, by introducing a probabilistic action function one could reduce the impact of the perceptual aliasing problem and achieve a higher probability of correct task execution. Returning to the above example, let us assume that $O(s_j, x_i)$ is equal to $O(s_j, x_j)$. If this is the case, then if $Act(x_j, *, \{\}, a_j) >> Act(x_i, *, \{\}, a_i)$ then the probability that action a_j will be correctly executed in state s_j will be much greater than the probability action a_i will be incorrectly executed.

The PAct-NoIS-NoComm controller synthesis method presented in this Section utilizes a genetic algorithm (GA) approach. A GA approach was chosen because analytically determining the optimal probabilities for each element of the action function is a highly non-linear problem as many of the elements are not independent. An analytical solution is intractable and does not scale well with the size of the action function. The GA approach does not guarantee an optimal solution, but is capable of synthesizing a PAct-NoIS-NoComm controller with a higher probability of correct task execution over the baseline DAct-NoIS-NoComm controller.

4.2.5.1 Genetic Algorithm-Based Synthesis Method

There are three high-level steps in the synthesis procedure: **1)** synthesize a baseline DAct-NoIS-NoComm controller, **2)** initialize the population of chromosomes representing potential solutions, and **3)** iteratively evolve the population through various means of selection to search for an improved solution. The PAct-NoIS-NoComm synthesis method is given by the procedure *Build_PAct-NoIS-NoComm_Controller* shown in Figure 4.7.

The following Sections define the chromosomes making up the population, how a chromosome's fitness is evaluated, and how the population is selected from one generation to the next.

4.2.5.2 Chromosome Encoding and the Population

A chromosome holds values for each of the variable parameters of the problem domain. As such, each chromosome represents a potential problem solution. A chromosome's encoding is what each of these values represents. In the PAct-NoIS-NoComm synthesis problem domain, each chromosome has a set of real-valued numbers in the range $(0, 1]$. Each of these values represents the probability value for one element of the action function. The set of probabilities encoded by each chromosome is denoted by $E = \{e_0, e_1, ..., e_n\}$, where n is the number of non-zero elements in the DAct-NoIS-NoComm action function.

The GA approach iteratively operates on a set of chromosomes, called the population, which is denoted as P. The number of elements in the population is called *population_size*.

4.2.5.3 Evaluating Chromosome Fitness

Since each chromosome encodes the probability values for each element of the controller action function, the appropriate metric of evaluating a chromosome's fitness is the expected probability

```
procedure Build_PAct-NoIS-NoComm_Controller(task T, robot r)
        Step 1: Synthesize a baseline DAct-NoIS-NoComm controller
  Build_DAct-NoIS-NoComm_Controller()

        Step 2: Initialize Population
  P = {}
  for i = 1topopulation_size do
    c ← new chromosome with random probabilities for each action function element;
    Calculate the fitness of c;
    P = P ⋃{c};
  endfor

        Step 3: Iteratively Evolve Population
  repeat number_iterations times
    Calculate the fitness of all elements of P;

        Step 3a: Selection through Elitism
    Elite ← set of top elite_percentage% of the most fit chromosomes in P;
    P ← {Elite};

        Step 3b: Selection through Mutation
    repeat (mutation_percentage ∗ population_size) times
      c ← random chromosome drawn from Elite;
      P ← P ⋃{Mutate(c)};
    end repeat mutation_percentage times

        Step 3c: Selection through Crossover
    repeat (crossover_percentage ∗ population_size) times
      c_a ← random chromosome drawn from Elite;
      c_b ← random chromosome drawn from Elite;
      P ← P ⋃{Crossover(c_a, c_b)};
    end repeat (crossover_percentage ∗ population_size) times
  end repeat number_iterations times
end procedure Build_PAct-NoIS-NoComm_Controller
```

Figure 4.7: Procedure for synthesizing a PAct-NoIS-NoComm controller for a robot r to execute a task T.

of correct task execution to be achieved by a MRS executing the controller with the probability values encoded by the chromosome. The higher the probability, the higher the fitness.

This synthesis method is used in an integrated manner with the probabilistic microscopic modeling approach presented in Section 4.1.2. When a chromosome's fitness needs to be evaluated, the action function of the baseline DAct-NoIS-NoComm controller is augmented with the probability values encoded by the chromosome. This PAct-NoIS-NoComm controller is then evaluated using the probabilistic microscopic model and the fitness of the chromosome is assigned as the probability of correct task execution as returned by the model.

4.2.5.4 Selection Through Elitism, Mutation, and Crossover

The population of chromosomes is continually evolved through the iterative execution of the *repeat* loop in lines 9-22 of Figure 4.7. In each iteration, the population from the previous generation is used to generate a new population of chromosomes. This new population is generated by three standard means: elitism, mutation, and crossover.

Elitism The first method of generating the new population is through an elitist process of selecting the chromosomes with the highest fitness from the previous generation and adding them to the next generation population. The chromosomes in the top *elite_percentage* percentile of the previous generation population are selected. This elitist method is useful in that it is a standard GA technique to prevent the loss of the best solution(s) from the previous generations.

Mutation The second method of generating the new population is through mutation, another standard GA selection technique. A percentage of *mutation_percentage* of the previously chosen elite chromosomes are randomly chosen to undergo mutation. A chromosome selected for mutation is first copied so that the original chromosome remains in the population. Then the copied chromosome will have one of its elements set to a random value. That is, a randomly chosen action probability value will be changed to a new random value.

Crossover The third method of selection is through crossover, by which *crossover_percentage* percent of the new population is generated. In crossover two chromosomes, c_a and c_b, are randomly selected from the elite chromosomes. Next, a random position in the set of elements making up the chromosomes is selected. A new chromosome is added to the population that as the elements from c_a from the beginning to the randomly selected point and has the elements from c_b from immediately after the randomly selected point to the end.

4.2.5.5 Discussion

This section has demonstrated a method for the synthesis of controllers with *probabilistic* action functions. The optimal assignment of probabilities to each action in the action function is a highly non-linear problem. Through the use of a genetic algorithm approach, this section has presented a tractable method for the assignment of action probabilities.

The use of probabilistic action selection to improve the probability of correct task execution can do so at the expense of the amount of time required to complete the task. By making some actions occur with decreased probability in order to decrease the likelihood of incorrect task execution in a given task state can cause the execution of that action in the correct task state to take longer. In some task domains with many instances of perceptual aliasing, the increase in time to task completion can be substantial as some necessary actions may have very small activation probabilities.

As the case studies in the following chapter demonstrate, with the simple introduction of probabilistic action functions, a MRS can achieve improved performance over robots with deterministic action functions.

4.2.6 Summary

This chapter has presented the core of this dissertation's principled controller design methodology: an integrated suite of systematic controller synthesis methods and a probabilistic microscopic MRS modeling approach.

The synthesis methods provide a means to study the formal synthesis of a range of MRS using a variety of common robot coordination and control features, such as probabilistic action selection, the maintenance of persistent internal state, the use of inter-robot communication. The probabilistic microscopic modeling approach provides an effective method to analytically predict the performance of a given controller to direct the synthesis methods toward a more effective design.

The next chapter applies the controller design methodology to two different task domains, construction and multi-foraging. Up to this point, the controller analysis and synthesis methods have intentionally not been specific to a particular task or task domain. The case studies presented next cast the formalism of the controller design methodology into specific task domains and demonstrate how the formalism can be applied to pragmatically construct a set of working robot controllers for these tasks.

Chapter 5

Case Studies: Applying the Design Methodology

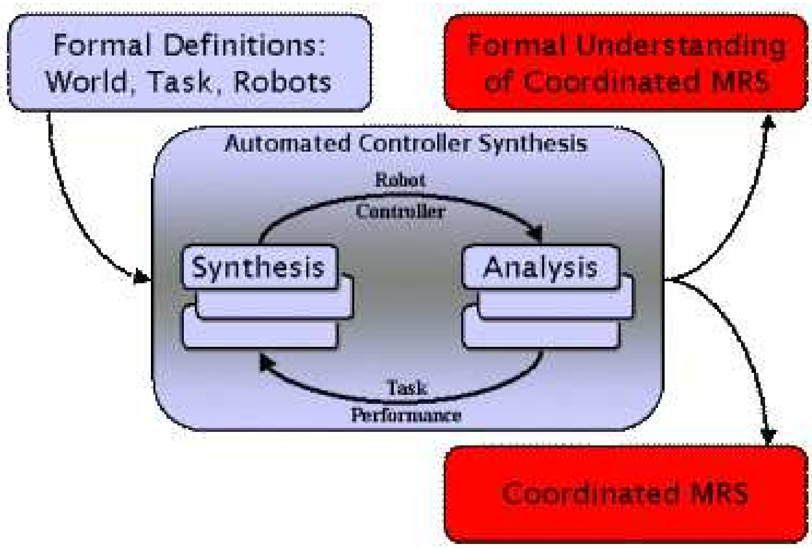

Figure 5.1: Diagram of principled multi-robot systems controller design methodology with the role of a successfully designed system highlighted.

This chapter applies the developed controller design methodology to two different task domains: 1) a multi-robot construction , and 2) task allocation in multi-foraging. Thus far, the formalism, controller analysis, and controller synthesis methods have been task-independent. This chapter demonstrates how to cast specific task domains into the formal definitions and apply the analysis and synthesis methods to the design of task-specific controllers.

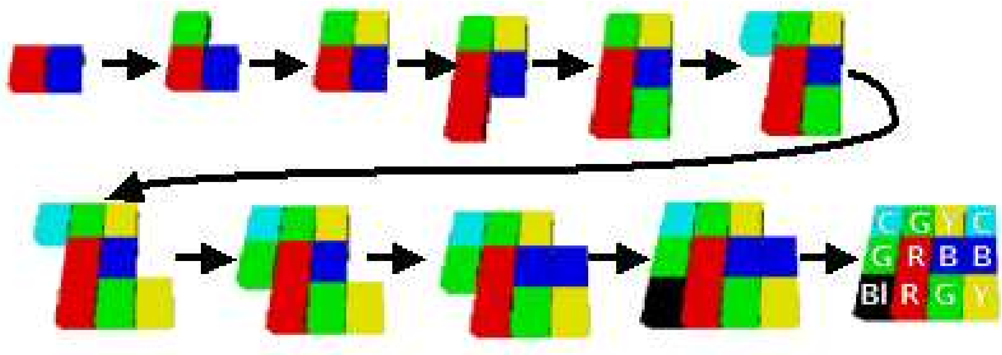

Figure 5.2: The sequence of brick configurations composing the task states for Construction Task 1. The task start state, u_start_T is on the far left and the terminal task state, u_term_T, is labeled with brick colors, denoted by the letter `R`, `G`, `B`, `Y`, `C`, or `Bl` that stand for `Red`, `Green`, `Blue`, `Yellow`, `Cyan`, and `Black`, respectively.

Experimental validation is provided through physically-realistic simulations and a limited number of physical robot demonstrations. For each task and controller design, results are provided from simulation experiments, analytical results from the probabilistic microscopic modeling approach presented in Section 4.1, and where available, empirical validation from physical robot demonstrations.

It should be noted that due to sensing and action uncertainty there can rarely be a guarantee of correct task execution; however, through the use of the presented controller design methodology one can aim to synthesize controllers that maximize the probability of correct task execution. This capability is demonstrated through the design of a number of controllers in this chapter. The controllers achieve different levels of performance, and it is through these differences that the advantages and disadvantages of controllers with different features, such as the use of internal state, communication, or probabilistic action selection, becomes apparent.

The multi-robot construction task domain and task allocation in the multi-foraging task domain were chosen because they represent two distinctly different types of tasks. The construction task domain is static in that it consists of discrete actions executed independently by individual

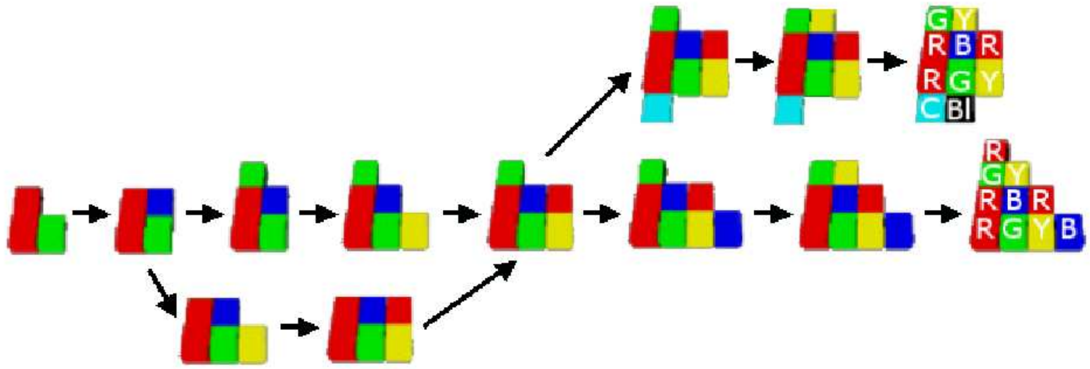

Figure 5.3: The brick configurations composing the directed acyclic graph defining Construction Task 2. The start state, u_start_T, is to the far left and the two terminal task states in u_term_T, labeled with brick colors, are to the far right. Brick colors are denoted by the letter `R`, `G`, `B`, `Y`, `C`, or `Bl` that stand for `Red`, `Green`, `Blue`, `Yellow`, `Cyan`, and `Black`, respectively.

robots. The tasks in this domain can, in principle, be executed by a single robot but with multiple robots the task efficiency can be greatly increased. The use of multiple robots, however, requires the consideration of spatiotemporal coordination among the robots, as discussed in Chapter 1. As opposed to the construction domain, task allocation in the multi-foraging domain is much more dynamic and requires tight coordination among all robots in order to successfully execute the task. The task-*independent* nature of the design methodology is demonstrated through its application to these two different domains.

The remainder of this chapter is organized as follows. Section 5.1 presents the application of the design methodology to the construction task domain. The application of the design methodology to task allocation in the multi-foraging task domain is presented in Section 5.2. Finally, Section 5.3 provides a summary with an overview of the application of the controller design methodology to both of these task domains.

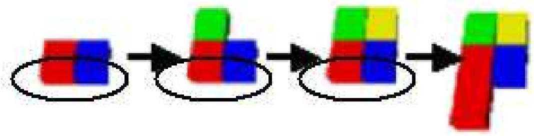

Figure 5.4: This figure highlights an perceptual aliasing example in the first part of Construction Task 1, as originally shown in Figure 5.2. The observation `<FLUSH B R>` is circled in the first three task states. It is only in the third task state that this observation should activate the action `<R FLUSH LEFT B R>`. To correctly execute the task, the occurrences of this observation in the first two task states should not activate a brick placement action. This issue can be resolved through the use of internal state or communication in the robot controller. The use of probabilistic action section alone can reduce the probability this issue will cause the task to be incorrectly executed.

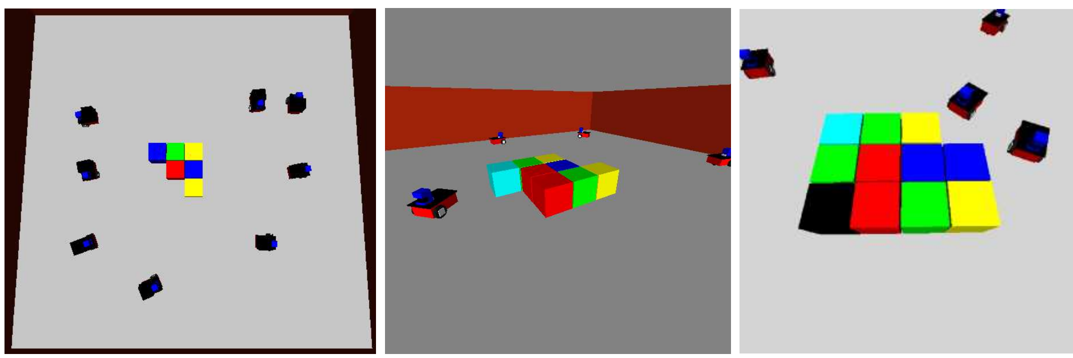

Figure 5.5: Snapshots of the multi-robot construction task domain simulation environment.

5.1 Multi-Robot Construction Task Domain

The construction task domain requires the spatiotemporally coordinated placement of a series of cubic colored bricks into a planar structure. A formal definition of the construction task domain is provided in Section 5.1.2. For all examples used in this section, a brick's color is denoted by the letters `R`, `G`, `B`, `Y`, `C`, and `Bl` which stand for `Red`, `Green`, `Blue`, `Yellow`, `Cyan`, and `Black`, respectively.

All construction tasks start with a seed structure, which consists of a small number of initially placed bricks forming the starting state of the construction process. The final structure is built through the addition of single bricks at a time placed individually by a robot. Bricks may not be

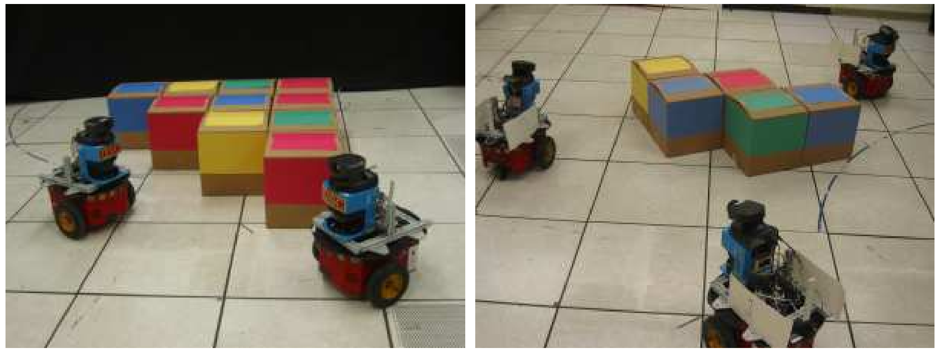

Figure 5.6: Snapshots of the physical robots used in the multi-robot construction task domain.

removed once added to the structure and sub-assemblies may be not independently constructed and then combined.

The controller design methodology was applied to the synthesis of controllers for the two construction tasks shown in Figures 5.2 and 5.3. These two construction tasks were specifically designed in order to demonstrate the strengths and weaknesses of the different controllers. Construction Task 1 is defined as a linear construction sequence and contains many occurrences of perceptual aliasing. This task was chosen to demonstrate the approach each controller synthesis method takes in dealing with instances of perceptual aliasing. Construction Task 2 also contains instances of perceptual aliasing, but it was specifically chosen to demonstrate the application of the controller synthesis methods to a task with branches. The synthesized controllers for these two tasks demonstrate the strengths and weaknesses of each controller synthesis method and the resulting controllers.

The remainder of this section is organized as follows. Section 5.1.1 describes the simulation and physical robot experimental setup. Section 5.1.2 formally casts the multi-robot construction task domain into the formalism presented in Chapter 3. The controller design methodology is applied to two tasks in the construction domain in Section 5.1.3 and synthesized DAct-NoIS-NoComm, DAct-IS-NoComm, DAct-NoIS-Comm, and PAct-NoIS-NoComm controllers are presented for each task.

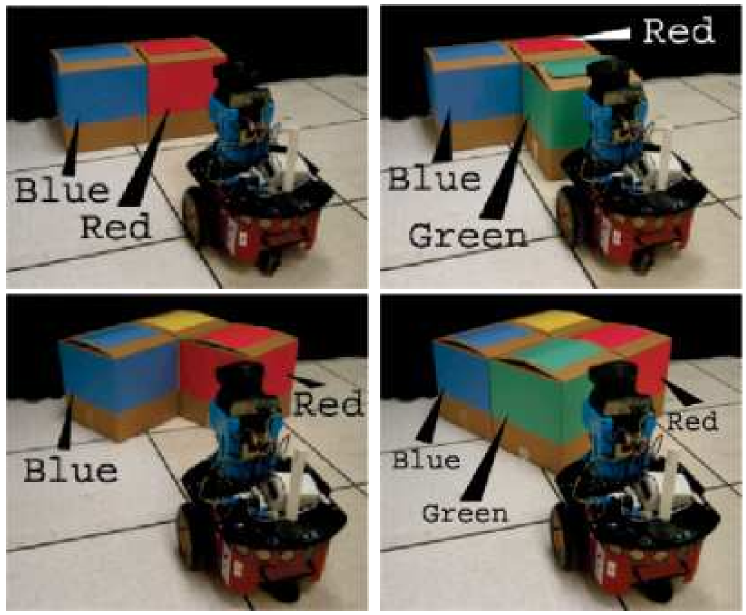

Figure 5.7: Example observations and actions in the construction domain. (top left) Robot in position to make observation `<FLUSH R B>`. (top right) Immediately after the robot performs action `<G RIGHT FLUSH R B>`. (bottom left) Robot in position to make observation `<CORNER R B>`. (bottom right) Immediately after the robot performs action `<G CORNER R B>`.

A discussion of the application of the design methodology to the multi-robot construction task domain is provided in Section 5.1.3.3.

5.1.1 Experimental Setup

This section describes the simulation and physical robot testbeds used in validation experiments for the construction task domain.

5.1.1.1 Robots

In all simulation experiments, 8 realistic models of ActivMedia Pioneer 2DX mobile robots were used. In all physical robot trials, 3 ActivMedia Pioneer 2DX mobile robots were used. Each robot, approximately 30 cm in diameter, is equipped with a differential drive, a forward-facing

Task 1 DAct-NoIS-NoComm Action Function
Act(<FLUSH R B>, *, {}, <G RIGHT FLUSH R B>) = 1
Act(<CORNER G B>, *, {}, <Y CORNER G B>) = 1
Act(<FLUSH B R>, *, {}, <R LEFT FLUSH B R>) = 1
Act(<CORNER B R>, *, {}, <G CORNER B R>) = 1
Act(<FLUSH R G>, *, {}, <C LEFT FLUSH R G>) = 1
Act(<FLUSH B G>, *, {}, <Y LEFT FLUSH B G>) = 1
Act(<CORNER R C>, *, {}, <G CORNER R C>) = 1
Act(<FLUSH Y B>, *, {}, <B LEFT FLUSH Y B>) = 1
Act(<CORNER R G>, *, {}, <Bl CORNER R G>) = 1
Act(<CORNER Y B>, *, {}, <C CORNER Y B>) = 1

Table 5.1: Deterministic action selection, stateless, and non-communicative controller (DAct-NoIS-NoComm) action function for Construction Task 1, as shown in Figure 5.2. All probabilities not shown are 0.

180-degree scanning laser rangefinder, and a pan-tilt-zoom color camera with a 100-degree field-of-view and a color blob detection system. The robots used in the simulation trials do not have the color camera and to replicate its function of identifying the colors of bricks, they use the laser rangefinder as a fiducial detector. In simulation, each brick color has a unique fiducial ID that is readable by the laser rangefinder.

It is noted that the bricks in the construction task domain are taller than the robot's sensors. This means the robots can only sense the local bricks on the periphery of the structure (i.e., robots do not have a birds-eye view of the entire structure).

The physical Pioneer robots used did not have the capability to independently manipulate bricks as require for the task. Consequently, their simulations did not either. To address this issue in simulation, when a robot wants to execute a brick placement action, it commands the simulator to place a brick of a given color at a given location relative to the robot's current pose. In the physical robot experiments, the appropriate brick is manually placed by the experimenter in response to the robot's dictated command (e.g., "Place yellow brick in the corner formed by the red and blue bricks directly in front of my position").

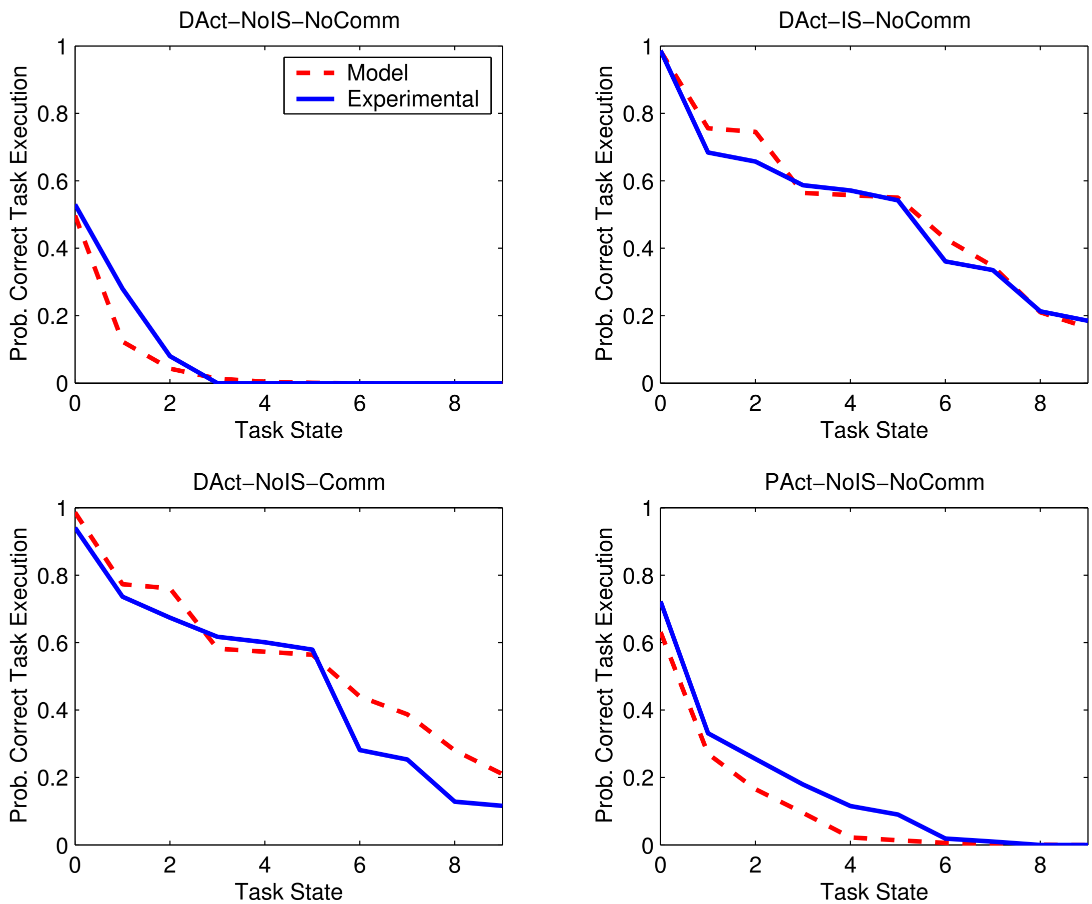

Figure 5.8: Construction Task 1 controller performance over 300 simulation trials and predicted performance provided by the probabilistic microscopic model. The probability of correct task execution given for each task state is the probability the task was correctly executed up to and including the action performed in that task state. The overall task performance is equivalent to the probability of correct task execution at the last task state.

Task 1 DAct-IS-NoComm Action and Internal State Transition Functions
Act(<FLUSH R B>, m_0, {}, <G RIGHT FLUSH R B>) = 1
Act(<CORNER G B>, m_0, {}, <Y CORNER G B>) = 1
Act(<FLUSH B R>, m_1, {}, <R LEFT FLUSH B R>) = 1
Act(<CORNER B R>, m_1, {}, <G CORNER B R>) = 1
Act(<FLUSH R G>, m_2, {}, <C LEFT FLUSH R G>) = 1
Act(<FLUSH B G>, m_3, {}, <Y LEFT FLUSH B G>) = 1
Act(<CORNER R C>, m_4, {}, <G CORNER R C>) = 1
Act(<FLUSH Y B>, m_5, {}, <B LEFT FLUSH Y B>) = 1
Act(<CORNER R G>, m_6, {}, <Bl CORNER R G>) = 1
Act(<CORNER Y B>, m_7, {}, <C CORNER Y B>) = 1
IState(m_0, <FLUSH G Y>, m_1) = 1
IState(m_1, <FLUSH G R>, m_2) = 1
IState(m_2, <CORNER R C>, m_3) = 1
IState(m_3, <CORNER B Y>, m_4) = 1
IState(m_4, <FLUSH G C>, m_5) = 1
IState(m_5, <CORNER Y B>, m_6) = 1
IState(m_6, <FLUSH Bl G>, m_7) = 1

Table 5.2: Deterministic action selection, persistent internal state, non-communicative controller (DAct-IS-NoComm) action and internal state transition functions for Construction Task 1, as shown in Figure 5.2. $m_0, m_1, ..., m_6 \in M$. The robots' initial internal state value was m_0. All probabilities not shown are 0.

5.1.1.2 Simulation Trials

Extensive use of simulation trials was made to validate the synthesized controllers. The simulation trials were performed using Player and Gazebo. Player [29] is a server that connects robots, sensors, and control programs over a network. Gazebo [41] simulates a set of Player devices in a 3-D physically-realistic world with full dynamics. Together, the two represent a high-fidelity simulation tool for individual robots and teams that has been validated on a collection of physical robot experiments using Player control programs transferred directly to physical Pioneer 2DX mobile robots. Figure 5.5 shows some snapshots of the simulation experimental setup.

For all simulation experimental results in the construction task domain, 300 task trials were performed. All trails began with the task T seed structure, task state u_start_T, in place and the 8 robots randomly positioned around the environment. A simulation run terminated when either of the following conditions occurred: 1) a task terminal state (u_term_T) was achieved, which signifies the task was successfully completed, or 2) an incorrect action was performed. If

Task 1 DAct-NoIS-Comm Action and Communication Functions
Act(<FLUSH R B>, $*$, $\{\}$, <G RIGHT FLUSH R B>) = 1
Act(<CORNER G B>, $*$, $\{\}$, <Y CORNER G B>) = 1
Act(<FLUSH B R>, $*$, $\{c_0\}$, <R LEFT FLUSH B R>) = 1
Act(<CORNER B R>, $*$, $\{\}$, <G CORNER B R>) = 1
Act(<FLUSH R G>, $*$, $\{c_1\}$, <C LEFT FLUSH R G>) = 1
Act(<FLUSH B G>, $*$, $\{c_2\}$, <Y LEFT FLUSH B G>) = 1
Act(<CORNER R C>, $*$, $\{c_3\}$, <G CORNER R C>) = 1
Act(<FLUSH Y B>, $*$, $\{c_4\}$, <B LEFT FLUSH Y B>) = 1
Act(<CORNER R G>, $*$, $\{c_1\}$, <Bl CORNER R G>) = 1
Act(<CORNER Y B>, $*$, $\{c_5\}$, <C CORNER Y B>) = 1
Comm(<FLUSH Y B>, c_0) = 1
Comm(<FLUSH B G>, c_1) = 1
Comm(<FLUSH B Y>, c_1) = 1
Comm(<FLUSH C G>, c_2) = 1
Comm(<FLUSH Y G>, c_3) = 1
Comm(<FLUSH G C>, c_4) = 1
Comm(<FLUSH Bl G>, c_5) = 1

Table 5.3: Deterministic action selection, stateless, and communicative controller (DAct-NoIS-Comm) communication function for Construction Task 1, as shown in Figure 5.2. $c_0, c_1, ..., c_5 \in C$. All probabilities not shown are 0.

termination condition 1 occurs, the experiment was deemed successful as the task was correctly executed. If termination condition 2 occurs, the experiment was deemed a failure as the task was not successfully executed. The probability of overall correct task execution was calculated as the percentage of task trials resulting in correct task execution.

5.1.1.3 Real-Robot Trials

A limited number of physical robot trials were performed to verify the feasibility of the synthesized controllers on physical Pioneer 2DX mobile robots. No statistical data were collected of the probability of correct task execution. As in the simulation experiments, the initial state of the world in the physical robot trials consists of the seed structure and all robots were located in random starting positions. The termination conditions are the same as those described for simulation experiments in Section 5.1.1.2. Figure 5.6 shows some snapshots of the physical robot experimental setup.

Task 1 PAct-NoIS-NoComm Action Function
Act(<FLUSH R B>, ∗, {}, <G RIGHT FLUSH R B>) = 0.8529
Act(<CORNER G B>, ∗, {}, <Y CORNER G B>) = 0.8161
Act(<FLUSH B R>, ∗, {}, <R LEFT FLUSH B R>) = 0.4937
Act(<CORNER B R>, ∗, {}, <G CORNER B R>) = 0.9802
Act(<FLUSH R G>, ∗, {}, <C LEFT FLUSH R G>) = 0.1882
Act(<FLUSH B G>, ∗, {}, <Y LEFT FLUSH B G>) = 0.3733
Act(<CORNER R C>, ∗, {}, <G CORNER R C>) = 0.1076
Act(<FLUSH Y B>, ∗, {}, <B LEFT FLUSH Y B>) = 0.1152
Act(<CORNER R G>, ∗, {}, <B1 CORNER R G>) = 0.1312
Act(<CORNER Y B>, ∗, {}, <C CORNER Y B>) = 0.1075

Table 5.4: Probabilistic action selection, stateless, and non-communicative controller (PAct-NoIS-NoComm) action function for Construction Task 1, as shown in Figure 5.2. All probabilities not shown are 0.

Controller Type	Predicted Probability of Correct Task Execution	Experimental Probability of Correct Task Execution
DAct-NoIS-NoComm	0.000	0.000
DAct-IS-NoComm	0.413	0.351
DAct-NoIS-Comm	0.140	0.090
PAct-NoIS-NoComm	0.045	0.051

Table 5.5: Probability of correct task execution, probabilistic microscopic model prediction and results from simulation trials, for each of the controllers synthesized for Construction Task 2.

It is emphasized that the physical robot trials were performed in order to show that the controller design methodology is not merely an abstract concept but rather it successfully captures the difficult issues involved in real-world embodied MRS, thus providing a grounded and pragmatic tool for the description, synthesis, and analysis of task-directed MRS.

5.1.2 Formal Definitions

In order to ground the construction task in the formal framework presented in Section 3, the world, task definitions, and robot observations and actions for this domain are defined in the following sections.

Task 2 DAct-NoIS-NoComm Action Function
Act(<CORNER R G>, ∗, {}, <B CORNER R G>) = 1
Act(<FLUSH R B>, ∗, {}, <G RIGHT FLUSH R B>) = 1
Act(<FLUSH B G>, ∗, {}, <Y LEFT FLUSH B G>) = 1
Act(<CORNER B Y>, ∗, {}, <R CORNER B Y>) = 1
Act(<FLUSH R Y>, ∗, {}, <B LEFT FLUSH R Y>) = 1
Act(<FLUSH B R>, ∗, {}, <Y RIGHT FLUSH B R>) = 1
Act(<FLUSH G Y>, ∗, {}, <R RIGHT FLUSH G Y>) = 1
Act(<FLUSH G R>, ∗, {}, <C LEFT FLUSH G R>) = 1
Act(<CORNER G C>, ∗, {}, <Bl CORNER G C>) = 1

Table 5.6: Deterministic action selection, stateless, and non-communicative controller (DAct-NoIS-NoComm) action function for Construction Task 2, as shown in Figure 5.3. All probabilities not shown are 0.

5.1.2.1 World

The *world* state is composed as the specific relative spatial configurations and colors of all bricks in the world. The internal state values or absolute spatial positions of the robots are not part of the world state.

As with any task, the definition and information contained in the world state can take many forms. In the definition used for this domain, the absolute robot positions are not part of the world state because the structure to be built is not constrained to absolute spatial positions in the world but is defined as the *relative* spatial configuration of bricks. This notion of relative brick positions is captured in the definition of robot brick placement actions described below. If desired, one could define this task differently so as to include robot positions in the world state with modifications to the manner in which robot brick placement actions are defined.

5.1.2.2 Task

As described in Section 3.3, a task is defined as a set of world states and world state transitions represented by a directed acyclic graph (DAG) with the nodes representing world states and the edges representing actions to transition the world state in the appropriate way. Therefore, in the construction task domain, a task DAG consists of a set of brick configurations for the nodes

Task 2 DAct-IS-NoComm Action and Internal State Transition Functions
Act(<CORNER R G>, m_0, {}, <B CORNER R G>) = 1
Act(<FLUSH R B>, m_0, {}, <G RIGHT FLUSH R B>) = 1
Act(<FLUSH B G>, m_1, {}, <Y LEFT FLUSH B G>) = 1
Act(<CORNER B Y>, m_1, {}, <R CORNER B Y>) = 1
Act(<FLUSH R Y>, m_1, {}, <B LEFT FLUSH R Y>) = 1
Act(<FLUSH B R>, m_2, {}, <Y RIGHT FLUSH B R>) = 1
Act(<FLUSH G Y>, m_2, {}, <R RIGHT FLUSH G Y>) = 1
Act(<FLUSH G R>, m_3, {}, <C LEFT FLUSH G R>) = 1
Act(<FLUSH G B>, m_4, {}, <Y CORNER G B>) = 1
Act(<CORNER G C>, m_5, {}, <Bl CORNER G C>) = 1
Act(<FLUSH B G>, m_0, {}, <Y LEFT FLUSH B G>) = 1
Act(<CORNER B Y>, m_6, {}, <R CORNER B Y>) = 1
Act(<FLUSH R B>, m_7, {}, <G RIGHT FLUSH R B>) = 1
Act(<FLUSH R Y>, m_8, {}, <B LEFT FLUSH R Y>) = 1
Act(<FLUSH B R>, m_9, {}, <Y RIGHT FLUSH B R>) = 1
Act(<FLUSH G Y>, m_9, {}, <R RIGHT FLUSH G Y>) = 1
Act(<FLUSH G R>, m_8, {}, <C LEFT FLUSH G R>) = 1
Act(<FLUSH B R>, $m_1 0$, {}, <Y RIGHT FLUSH B R>) = 1
Act(<CORNER G C>, $m_1 1$, {}, <Bl CORNER G C>) = 1
IState(m_0, <CORNER G B>, m_1) = 1
IState(m_1, <CORNER R B>, m_2) = 1
IState(m_1, <FLUSH R Y>, m_3) = 1
IState(m_3, <FLUSH C R>, m_4) = 1
IState(m_4, <FLUSH G Y>, m_5) = 1
IState(m_0, <FLUSH Y G>, m_6) = 1
IState(m_6, <FLUSH B R>, m_7) = 1
IState(m_7, <CORNER G B>, m_8) = 1
IState(m_8, <CORNER R B>, m_9) = 1
IState(m_8, <FLUSH C R>, $m_1 0$) = 1
IState($m_1 0$, <FLUSH G Y>, $m_1 1$) = 1

Table 5.7: Deterministic action selection, persistent internal state, and non-communicative controller (DAct-IS-NoComm) action and internal state transition functions for Construction Task 2, as shown in Figure 5.3. $m_0, m_1, ..., m_1 1 \in M$. The robots' initial internal state value was m_0. All probabilities not shown are 0.

Task 2 DAct-NoIS-Comm Action and Communication Functions
Act(<CORNER R G>, $*$, $\{\}$, <B CORNER R G>) = 1
Act(<FLUSH R B>, $*$, $\{\}$, <G RIGHT FLUSH R B>) = 1
Act(<FLUSH R B>, $*$, $\{c_2\}$, <G RIGHT FLUSH R B>) = 1
Act(<FLUSH B G>, $*$, $\{c_0\}$, <Y LEFT FLUSH B G>) = 1
Act(<FLUSH B G>, $*$, $\{\}$, <Y LEFT FLUSH B G>) = 1
Act(<CORNER B Y>, $*$, $\{\}$, <R CORNER B Y>) = 1
Act(<FLUSH R Y>, $*$, $\{\}$, <B LEFT FLUSH R Y>) = 1
Act(<FLUSH B R>, $*$, $\{\}$, <Y RIGHT FLUSH B R>) = 1
Act(<FLUSH G Y>, $*$, $\{\}$, <R RIGHT FLUSH G Y>) = 1
Act(<FLUSH G R>, $*$, $\{c_0, c_2\}$, <C LEFT FLUSH G R>) = 1
Act(<FLUSH B R>, $*$, $\{c_0\}$, <Y RIGHT FLUSH B R>) = 1
Act(<CORNER G C>, $*$, $\{c_1\}$, <B1 CORNER G C>) = 1
Comm(<CORNER G B>, c_0) = 1
Comm(<CORNER R B>, c_1) = 1
Comm(<FLUSH B R>, c_2) = 1
Comm(<FLUSH C R>, c_0) = 1
Comm(<FLUSH G Y>, c_1) = 1

Table 5.8: Deterministic action selection, stateless, and communicative controller (DAct-NoIS-Comm) action and internal state transition functions for Construction Task 2, as shown in Figure 5.3. $c_0, c_1, ..., c_1 1 \in C$. All probabilities not shown are 0.

and a set of brick placement actions that will appropriately transition the world state (i.e., the configuration of bricks) to the next structure in the construction task.

Abstractly, a construction task is a set of acceptable construction sequences that encode required spatiotemporal building constraints. For example, a construction task with the goal of building a four-walled enclosure may be defined as "build the North wall, then the South wall, then the West wall, and finally the East wall" *or* "build the East wall, then the South wall, then the West wall, then the North wall".

5.1.2.3 Observations

Observations in the construction domain are made up of the spatial configuration and color of bricks in the field-of-view of the robot's laser rangefinder and color camera and within an appropriate range and bearing.

Task 2 PAct-NoIS-NoComm Action Function
Act(<CORNER R G>, ∗, {}, <B CORNER R G>) = 0.893
Act(<FLUSH R B>, ∗, {}, <G RIGHT FLUSH R B>) = 0.119
Act(<FLUSH B G>, ∗, {}, <Y LEFT FLUSH B G>) = 0.9221
Act(<CORNER B Y>, ∗, {}, <R CORNER B Y>) = 0.780
Act(<FLUSH R Y>, ∗, {}, <B LEFT FLUSH R Y>) = 0.033
Act(<FLUSH G Y>, ∗, {}, <R RIGHT FLUSH G Y>) = 0.889
Act(<FLUSH G R>, ∗, {}, <C LEFT FLUSH G R>) = 0.002
Act(<FLUSH B R>, ∗, {}, <Y RIGHT FLUSH B R>) = 0.003
Act(<CORNER G C>, ∗, {}, <Bl CORNER G C>) = 0.917

Table 5.9: Probabilistic action selection, stateless, and non-communicative controller (PAct-NoIS-NoComm) action function for Construction Task 2, as shown in Figure 5.3. All probabilities not shown are 0.

Two categories of observations can be made.

1) The first is two adjacent, aligned bricks. A situation in which such an observation is made is shown in Figure 5.7 and is denoted as <FLUSH R B>.

2) The second is two adjacent bricks forming a corner. A situation in which such an observation is made is shown in Figure 5.7 and is denoted as <CORNER R B>.

It is emphasized that the observations <FLUSH R B> and <FLUSH B R> constitute two different observations in which the spatial relationship between the Red and Blue bricks are switched. A similar point holds for the observations <CORNER R B> and <CORNER B R>.

In the simulation environment there is uncertainty in the robots' observations. The uncertainty is in interpreting the spatial relationship between the locally observed bricks. There is no uncertainty in determining the color of all local bricks. For all FLUSH observations there is some probability of seeing them as CORNER observations, and vice versa. The probabilities are: given a FLUSH observation, for example, <FLUSH R B>, the probability a robot will misinterpret it as a CORNER observation, in this case <CORNER R B>, is 11.5%. Likewise, given a CORNER observation, for example, <CORNER R B>, the probability a robot will misinterpret it as a FLUSH observation, in this case FLUSH R B>, is 1%. These probabilities of observation uncertainty were determined empirically through separate simulation trials. The probabilistic microscopic modeling approach presented in Chapter 4.1.2 uses these probabilities for observation uncertainties.

5.1.2.4 Actions

There are three categories of actions each robot can perform, each involving the placement of a brick.

1) The first is the placement of a brick on the right side (from the perspective of the acting robot) of a pair of adjacent, aligned bricks. The immediate result of such an action is demonstrated in Figure 5.7 and is denoted as `<G RIGHT FLUSH R B>`.

2) The second is identical to the first except that the brick is placed on the left side of a pair of adjacent, aligned bricks. This action is denoted as `<G LEFT FLUSH R B>`.

3) The third is the placement of a brick in the corner formed by two other bricks. The immediate result of such an action is demonstrated in Figure 5.7 and is denoted as `<G CORNER B R>`.

In the simulation environment there is uncertainty in brick placement actions. The uncertainty is characterized by the following probabilities. For all `LEFT FLUSH` and `RIGHT FLUSH` actions, the probability of the action succeeding is 98.5%. The probability of success for all `CORNER` actions is 78%. These probabilities of action uncertainty were empirically determined through separate simulation trials. As with the observation undertainties, these probability values for actions are used by the probabilistic microscopic modeling approach presented in Chapter 4.1.2.

The competency actions for the construction task domain include low-level actions such as performing local navigation and obstacle avoidance. Since the world state in this domain consists of the spatial locations and colors of bricks in the world, robot motion tasks such as these do not affect the world state. These competency actions allow the robots to safely move around the world and, when appropriate, an action in the action function will intervene and execute a brick placement action.

5.1.3 Results

This section details experimental results from the application of the controller design methodology to the tasks shown in Figures 5.2 and 5.3.

5.1.3.1 Construction Task 1

The first construction task considered is pictured in Figure 5.2. Each brick configuration in the figure represents a state in V_T and the arrows represent the edges in E_T. As the figure shows, the task is composed of a linear sequence of 11 states (i.e., brick configurations). The start state, u_start_T, is on the upper left and, following a linear sequence of states, the single terminal state, u_term_T, is shown on the lower right. This first task is a simple linear task; therefore, the robots must follow this linear trajectory from u_start_T to u_term_T through the world state space in order to correctly execute the task.

From the perspective of the robots, which can only sense the local periphery of the structure at any point in time, this construction task poses a number of difficulties involving perceptual aliasing. For example, as illustrated in Figure 5.4, the observation `<FLUSH B R>` occurs in the first three states of the task, but only in the third state should this observation activate a brick placement action by the robot, in this case `<R FLUSH LEFT B R>`. There are multiple other perceptual aliasing instances in this task definition, including the observation of `<FLUSH B G>` in the fifth and six task states where only the later instance of this observation should activate a `<Y FLUSH LEFT B G>` action. This form of perceptual aliasing can directly impact task performance because the same observation can be made in more than one task state, but only in a subset of those states should a brick placement action be performed.

Another instance of perceptual aliasing can be seen in the observation of `<FLUSH G Y>` in all but the first two task states. This particular instance does *not* impact the controller design because in no instance should this observation activate a brick placement action. More generally, an observation that should activate the *same* action in all of its occurrences is of little concern during controller design.

It is the situation where the same observation should activate *different* actions in its various occurrences that is a cause of concern. To appropriately deal with this later form of perceptual aliasing, controllers can intelligently use internal state, inter-robot communication, or probabilistic action selection, as demonstrated in the DAct-IS-NoComm, DAct-NoIS-Comm, and PAct-NoIS-NoComm controllers, respectively, presented below.

DAct-NoIS-NoComm Controller Design

The DAct-NoIS-NoComm controller synthesis method described in Section 4.2.2 is now applied to Construction Task 1. A DAct-NoIS-NoComm controller is one that has deterministic action selection and is stateless and non-communicative. The resulting controller is shown in Table 5.1. This table shows the mapping from observations to actions constituting the controller action function. The internal state transition function and communication function are not used and will always return a value of 0.

Given an observation, the controller checks the action function to see which action is activated. Each entry from the action function shown in Table 5.1 is evaluated in sequence until an action is activated. Once an active action is found, that action is executed.

This controller dictates, for example, whenever a robot makes the observation `<FLUSH B R>`, which is the observation of an adjacent blue and red brick, the robot will execute the action `<R LEFT FLUSH B R>`, which is the placement of a red brick adjacent to the red brick in the observation. As was previously stated, the observation of `<FLUSH B R>` occurs in each of the first three task states, however, only in the third state should the action `<R LEFT FLUSH B R>` be executed. With this DAct-NoIS-NoComm controller there is no guarantee that this action will not be executed in one of the first two task states resulting in incorrect task execution. It is purely a matter of chance, in the first task state for example, as to whether a robot will observe `<FLUSH R B>` *before* observing `<FLUSH B R>` and then therefore execute the correct action `<G RIGHT FLUSH B R>` before the incorrect action `<R LEFT FLUSH B R>` is executed. A similar situation occurs for all other instances of perceptual aliasing in this task when using the deterministic, stateless, and non-communicative DAct-NoIS-NoComm controller.

Figure 5.8 shows the probability of correct task execution for a MRS with robots using this controller performing Construction Task 1. The figure also provides the prediction made by the probabilistic microscopic model. Since this controller does not have the ability to maintain persistent internal state, to communicate with other robots, or the capability of probabilistic action selection, errors introduced by perceptual aliasing are very prominent. As the figure shows, robots executing this DAct-NoIS-NoComm controller rarely successfully perform even half of the task before an incorrect action is executed, and the complete task is never correctly executed. The probabilistic microscopic model accurately predicted the performance of this controller.

DAct-IS-NoComm Controller Design

Applying the DAct-IS-NoComm synthesis method from Section 4.2.3 to Construction Task 1 results in the controller shown in Table 5.2. A DAct-IS-NoComm controller is one that has deterministic action selection, maintains a finite amount of persistent internal state, and is non-communicative.

Table 5.2 provides the controller action function, which is a mapping from observations and internal state values to actions, and the controller internal state transition function, which is a mapping from observations and internal state values to new internal state values.

The controller works as follows. Regarding the internal state transition function, when a robot with an internal state value of m_0, for example, makes the observation `<FLUSH G Y>`, the robot's internal state value is transitioned to m_1. Regarding the action function, when a robot with an internal state value of m_0 makes the observation `<FLUSH R B>` the action `<G RIGHT FLUSH R B>` is executed.

Recall that the DAct-NoIS-NoComm controller for Construction Task 1 above was not able to deal with instances of perceptual aliasing, such as the observation of `<FLUSH B R>` in the first three task states. The DAct-IS-NoComm controller, however, makes use of internal state to deal with this perceptual aliasing issue. As the action function shows, the action `<R LEFT FLUSH B R>` will not be executed unless the observation `<FLUSH B R>` is made *and* the robot's internal state value is m_1.

All robots begin task execution with an internal state value of m_0, so no robot will execute the action `<R LEFT FLUSH B R>` until after the transition from internal state value m_0 to m_1. The internal state transition function in Table 5.2 shows that this transition will not occur until the observation `<FLUSH G Y>` is made.

Consulting the task definition in Figure 5.3, this observation cannot be made until the third task state, exactly the state in which the action `<R LEFT FLUSH B R>` should be executed. Through this mechanism of preconditioning actions on observations and appropriately transitioned internal state values, the perceptual aliasing issues encountered in the DAct-NoIS-NoComm controller are overcome.

The performance of the DAct-IS-NoComm controller is shown in Figure 5.8 along with the prediction provided by the probabilistic microscopic model. As the figure shows, the addition of internal state substantially improves the probability of correct task execution over the stateless DAct-NoIS-NoComm controller. Although, due to inherent uncertainty in sensing and action, this controller does not always achieve correct task execution.

DAct-NoIS-Comm Controller Design

Applying the DAct-NoIS-Comm synthesis method from Section 4.2.4 to Construction Task 1 results in the controller shown in Table 5.3. A DAct-NoIS-Comm controller has deterministic action selection, is stateless, and has the capability of inter-robot communication.

Table 5.3 provides the controller action function, which is the mapping from observations and received communication messages to actions, and the controller communication function, which is the mapping from observations to transmitted communication messages. This controller will, for example, transmit the communication message c_0 upon receiving the observation `<FLUSH Y B>` and the action `<R LEFT FLUSH B R>` upon simultaneously receiving the observation `<FLUSH B R>` and receiving the communication message c_0.

Recall that the DAct-IS-NoComm controller presented above dealt with the perceptual aliasing issue caused by the observation of `<FLUSH B R>` through the use of internal state. The DAct-NoIS-Comm controller deals with this issue through the use of inter-robot communication. From the inspection of the task definition, the only task state in which the observation of both `<FLUSH B R>` and `<FLUSH Y B>` can be made is the third task state. By conditioning the execution of the correct action in this state, `<R LEFT FLUSH B R>` to the observation of `<FLUSH B R>` and the receipt of communication message c_0, which is transmitted only upon the observation of `<FLUSH Y B>`, this perceptual aliasing issue is resolved. Other such issues in this task are dealt with in a similar manner.

The performance of the DAct-NoIS-Comm controller is shown in Figure 5.8 next to the other synthesized controllers and the predictions made by the probabilistic microscopic model. As the figure shows, the use of inter-robot communication greatly improves the probability of correct task execution over the baseline DAct-NoIS-NoComm controller and achieves similar performance to the DAct-IS-NoComm controller.

PAct-NoIS-NoComm Controller Design

Applying the PAct-NoIS-NoComm synthesis method from Section 4.2.5 to Construction Task 1 produces the controller shown in Table 5.4. A PAct-NoIS-NoComm controller is one that utilizes probabilistic action selection and is both stateless and non-communicative.

Table 5.4 shows the controller action function, which is a mapping from observations to actions. Since the PAct-NoIS-NoComm controller uses a probabilistic action function, each entry in the action function is provided with a probability value. This means that, for example, a robot that receives the observation `<FLUSH R B>` will execute the action `<G RIGHT FLUSH R B>` 85.29% of the time. Furthermore, since this controller is stateless and non-communicative, the internal state transition function and the communication function are both empty and will always return values of 0.

The DAct-IS-NoComm and DAct-NoIS-Comm controllers dealt with the perceptual aliasing issues, for example the case of the observation of `<FLUSH B R>`, by the use of internal state and communication, respectively. The PAct-NoIS-NoComm controller uses probabilistic action selection to deal with perceptual aliasing issues. By using relative differences in probability values for different actions, the impact of perceptual aliasing is reduced.

For example, upon receipt of observation `<FLUSH B R>` the brick placement action `<R LEFT FLUSH B R>` will only be executed 49.37% of the time. However, the correct actions for the first two task states `<FLUSH R B>` and `<CORNER G B>` are executed over 80% of the time upon receipt of the appropriate observations. Even though `<FLUSH B R>` can be observed in the first three task states, the relative differences in action probability values will likely result in the correct action being performed in each of the first two task states instead of the incorrect action of `<G CORNER B R>`.

The performance of this DAct-NoIS-Comm controller is shown in Figure 5.8 along side data for the other synthesized controllers. The use of probabilistic action selection improves the probability of correct task execution over the baseline DAct-NoIS-NoComm controller but, in this task domain, does not reach the level of performance achieved by the DAct-IS-NoComm or the DAct-NoIS-Comm controllers.

5.1.3.2 Construction Task 2

The second construction task considered is shown in Figure 5.3. Each brick configuration in the figure represents a state in V_T and the arrows represent the edges in E_T. This second construction task has a number of branches in the task definition.

The start state, u_start_T, is on the far left and the two states to the far right are the terminal states comprising u_term_T. The arrows between task states represent the edges in E_T. Any path through this graph from u_start_T to either of the two states in u_term_T represents correct task execution.

The controllers resulting from the DAct-NoIS-NoComm and DAct-IS-NoComm synthesis methods are provided below. Table 5.5 summarizes the performance of the controllers. Because of the complicated nature of reporting state-by-state results for a task with many branches, the results for Construction Task 2 are given in the form of *overall* probability of correct task execution as determined through simulation trials and predicted performance provided by the probabilistic microscopic model.

DAct-NoIS-NoComm Controller Design

Applying the DAct-NoComm-NoIS synthesis method to Construction Task 2 results in the controller shown in Table 5.6. Since a DAct-NoIS-NoComm controller is stateless and non-communicative, the internal state transition function and communication function are empty and will always return a value of 0.

As the results from Table 5.5 show, the DAct-NoIS-NoComm controller has a predicted and actual probability of correct task execution of 0%. This is not surprising because, similar to the first construction task, there are many instances of perceptual aliasing that the DAct-NoIS-NoComm controller is not capable of handling.

Furthermore, this task contains branches in the task definition and correct actions are therefore dependent on past actions. The DAct-NoIS-NoComm controller is stateless and has no mechanism by which to remember past actions.

DAct-IS-NoComm Controller Design

Applying the DAct-IS-NoComm synthesis method from Section 4.2.3 to Construction Task 2 results in the controller shown in Table 5.7. This controller uses internal state to aid in determining the task state before executing an action involving an observation that occurs in multiple task states.

The task branches after the second task state and the branch taken depends on whether the action `<G RIGHT FLUSH R B>` was executed or the action `<Y LEFT FLUSH B G>` was executed. In the second task state, either of these actions is correct, but future actions will need to depend on which one was executed at this point.

In order to record which branch was taken, internal state is used to mark the entrance into a new task branch by using the presence of an observation that is unique to each of the task states immediately after the branching occurs. For example, the observations `<CORNER G B>` and `<FLUSH Y G>` are used to transition the internal state value in order to record the branch taken after the second task state. The value the internal state is transitioned to depends upon which of these observations is made. This internal state transition marks the branch in the task definition taken and therefore determines the future actions the robot will need to execute.

DAct-NoIS-Comm Controller Design

Applying the DAct-NoIS-Comm synthesis method from Section 4.2.4 to Construction Task 2 results in the controller shown in Table 5.8. As the results show, the DAct-NoIS-Comm controller performs better than the non-communicative DAct-NoIS-NoComm controller but not as well as the DAct-IS-NoComm controller. The reasoning behind this follows from the fact that the DAct-NoIS-Comm controller does not have the ability to remember past events, a capability necessary if the branch of the task is to be recorded.

The communication capability does improve performance over the baseline DAct-NoIS-NoComm controller because there are a number of instances of perceptual aliasing throughout the task that communication can help to disambiguate.

PAct-NoIS-NoComm Controller Design

Applying the PAct-NoIS-NoComm synthesis method from Section 4.2.5 to Construction Task 2 produces the controller shown in Table 5.9. As with Construction Task 1, the PAct-NoIS-NoComm controller for Construction Task 2 performs better than the baseline deterministic DAct-NoIS-NoComm controller but not as well as the controllers using internal state or communication.

This task is particularly difficult for a probabilistic controller because there are several instances in which the same action must be performed, but at different points in the task execution, depending on the branch of the task taken. Upon examination of the controller, it becomes clear that the genetic algorithm approach addressed this issue by converging on a controller that has action probability values that predispose it to take a particular trajectory through the task definition. For example, in the second task state, the task branches depending on whether the action `<Y LEFT FLUSH B G>` or `<G RIGHT FLUSH R B>` is executed. The probabilistic controller synthesized make it most likely that the former is executed by providing probability values to these two actions as 87.2% and 3.5%, respectively.

Although having a PAct-NoIS-NoComm controller that is predisposed to take a particular trajectory through the task definition helps in dealing with the issue of perceptual aliasing across different branches of the task, it also somewhat defeats the purpose of having tasks with branches as it essentially reduces the task to a single linear trajectory.

5.1.3.3 Discussion

This section has applied the controller design methodology to a multi-robot construction task domain. This domain requires the robots to spatiotemporally coordinate a series of brick placements in order to build a given structure. Through the synthesis of four different controllers, it was shown how the use of internal state, inter-robot communication, and probabilistic action selection can be used to facilitate improved MRS coordination.

The use of internal state in robot controllers provides improved task performance over stateless controllers as it provides the ability to record past task states that are required in order to correctly perform future actions. This ability is particularly relevant in tasks with branches, as they require some actions to be conditioned on past events.

The use of communication also improves task performance over the baseline DAct-NoIS-NoComm controller. However, the use of communication alone is less beneficial in tasks with branches, as the results for Construction Task 2 show. Communication can aid in dealing with perceptual aliasing issues, but it is not capable of incorporating past events into action decisions.

The use of probabilistic action selection also improves task performance over the baseline DAct-NoIS-NoComm controller, but is not as effective as the use of internal state or communication. Furthermore, the usefulness of probabilistic action selection declines as the number of perceptual aliasing instances that must be dealt with increase.

5.2 Task Allocation in Multi-foraging

This section applies the controller design methodology to the issue of task allocation in a multi-foraging task domain. Task allocation is the process of assigning individual robots to sub-tasks of a given system-level task [28]. A *traditional* foraging task is defined by having an individual robot or group of robots collect objects from an environment and either consume them on the spot or return them to a common location [32]. Multi-foraging, a variation on traditional foraging, consists of an arena populated by multiple types of objects to be concurrently foraged [10], each of which may be of varying importance or require different robot capabilities to forage.

Assuming the object types to be foraged are label as A, B, and C, and for each object of type A foraged a large reward is returned, for each object of type B foraged a moderate reward is returned, and for each object of type C foraged a small reward is returned. If there are finite objects of each type and a finite number of robots engaged in foraging, an important question arises: "How many robots should be foraging for each object type in order to maximize the reward?". This question is fundamental in the study of multi-robot task allocation [28] and has received considerable attention.

There is a second question that follows, "Assuming the optimal allocation of robots is known, how can this allocation be achieved in a distributed robot system?". This question has received much less attention and its understanding is fundamental to producing useful multi-robot systems capable of effectively allocation their resources to available tasks. It is this second important question that motivates the application of the controller design methodology presented in this dissertation to task allocation in a multi-foraging task domain.

The remainder of this section is organized as follows. Section 5.2.2 formally casts the multi-foraging task domain into the formalism presented in Section 3. Section 5.2.1 describes the simulation setup. The controller design methodology is then applied to this task domain and synthesized DAct-NoIS-NoComm controllers are presented in Section 5.2.3.1 and DAct-IS-NoComm controllers are presented in Section 5.2.3.2.

5.2.1 Experimental Setup

The multi-foraging task domain consists of a 40 robots foraging in an arena for cylindrical pucks classified into a finite number of discrete types. The arena is closed and is 729 square meters in size. Puck types are defined by the color of the puck and task domain considered in this section consists of the following puck types: P_{Red}, P_{Green}, P_{Blue}, P_{Cyan}, P_{Yellow}, and P_{Purple}.

This section describes the simulation testbed and the features of the simulated robots used in validation experiments for the study of task allocation in the multi-foraging task domain.

5.2.1.1 Robots

The robots used in the simulation trials are similar to those described in Section 5.1.1.1. Each puck to be foraged is marked with a fiducial that identifies the object type, and each robot is equipped with a fiducial that marks the active foraging state of the robot (e.g., what type of puck that robot is currently foraging). Note that the fiducials do not contain unique identities of the pucks or robots but only mark the type of a puck or the puck type a given robot is engaged in foraging. Each robot is also equipped with a 2-DOF gripper on the front, capable of picking up a single 8 cm diameter puck at a time. The range from which a robot can sense other robots is a field-of-view of 180-degrees out to a range of 12 meters.

The robots move in an enclosed arena and pick up encountered pucks. When a robot picks up a puck, the puck is consumed (i.e., it is immediately removed from the environment) and the robot carries on foraging for other pucks. Immediately after a puck is consumed, another puck of the same type is placed in the arena at a random location. This replacement is performed in order to maintain constant puck density in the arena and reduce the number of variables that affect the performance of the system. An interesting extension of this work includes the study of tasks in which the puck densities change with time.

Each robot is equally capable of foraging all puck types, but can only be allocated to foraging for one type at any given time. Additionally, all robots are engaged in foraging at all times; a robot cannot be idle. A robot may switch the puck type for which it is foraging according to its control policy, when it determines it is appropriate to do so. Robots engaged in foraging P_{Red} pucks are label as R_{Red} robots, those foraging for P_{Green} pucks are labeled R_{Green}, and similarly for the remainder of puck types.

It is noted that the limited sensing capabilities and lack of direct communication of the individual robots in the task domain implementation prohibits them from acquiring global information such as the size and shape of the foraging arena, the initial or current number of pucks to be

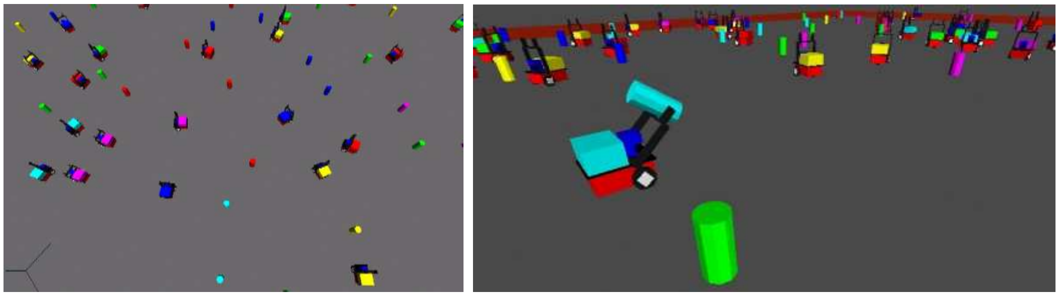

Figure 5.9: Snapshots of the simulation experimental setup used to validate the controller design methodology for task allocation in a multi-foraging task domain. The picture on the left shows the foraging arena as seen from overhead. The picture to the right shows a close-up of robots engaged in foraging.

foraged (total or by type), or the initial or current number of foraging robots (total or by foraging type).

5.2.1.2 Simulation Trials

Extensive use of simulation experiments was made to validate the synthesized controllers. The simulation environment was the same as that described in Section 5.1.1.2. Figure 5.9 shows some snapshots of the simulation experimental setup.

5.2.2 Formal Definitions

In order to ground this multi-foraging task allocation domain in the formal framework presented in Section 3, the world, task definition, and robot observations and actions for this domain are defined in the following sections.

5.2.2.1 World

The world state in the multi-foraging task domain is composed of the distribution of robot foraging states. A world state s,

$$s =< R_{Red} = 20\%, R_{Green} = 0\%, R_{Blue} = 80\%, R_{Cyan} = 0\%, R_{Yellow} = 0\%, R_{Purple} = 0\% > \tag{5.1}$$

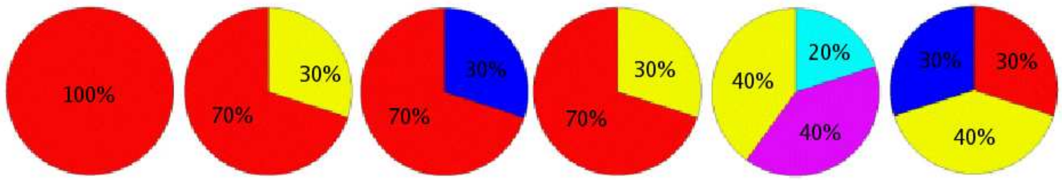

Figure 5.10: A task definition for the multi-foraging domain. Each pie chart represents a world state that is defined as the percentage of robots engaged in foraging for each puck type. Red denotes the percentage of robots foraging for pucks of type P_{Red}, green for pucks of type P_{Green}, and similarly for the other puck types. The task start state, u_start_T, is on the far left and the terminal state, u_term_T, is on the far right. Successful completion of the task entails having the MRS transition the distribution of robots foraging for each puck type through each world state from u_start_T to u_term_T. This representation shows six world states; however, the transition between each of these states is not discrete and includes a series of intermediate world states not shown that transition smoothly from one task state to the next.

indicates that 20% of the robots are foraging for P_{Red} pucks, 80% are foraging for P_{Blue} pucks, and 0% of the robots are foraging for the remainder of puck types.

5.2.2.2 Task

As described in the formalism in Section 3.3, a task is defined as a set of world states and world state transitions represented by a directed acyclic graph. Therefore, in a multi-foraging task domain, a task consists of a series of task allocation distributions.

A graphical representation of the specific task definition considered in this section is shown in Figure 5.10. In this task, all robots are initialized as foraging for P_{Red} pucks. Next, 30% of the robots must transition to foraging for P_{Yellow} pucks. Once this distribution is achieved, the robots must again change their distribution to have 70% foraging for P_{Red} pucks and 30% foraging for P_{Blue} pucks. This process is continued for the other world states in the task description until the terminal state is reached.

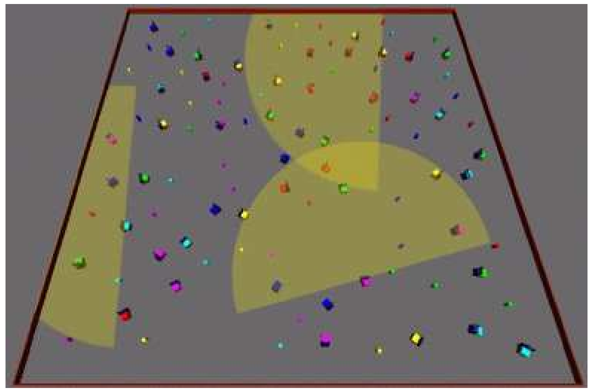

Figure 5.11: A highlighting snapshot the observation range and contents from three selected robots.

5.2.2.3 Observations

Observations in the multi-foraging domain consist of the proportion of robots foraging for each puck type in the robot's current local sensing field of view. Figure 5.11 shows a picture from the simulation with a robot's observation contents highlighted. The foraging state of all the robots in each highlighted region will be included in the composition of the respective robot's observation.

A robots observation, o, denoted as

$$o =< R_{Red} = 10\%, R_{Green} = 10\%, R_{Blue} = 0\%, R_{Cyan} = 20\%, R_{Yellow} = 70\%, R_{Purple} = 0\% > \quad (5.2)$$

corresponds to the situation where of all the robots observed in the robots current sensory field of view, 10% of them were foraging for P_{Red} pucks, 10% for P_{Green} pucks, 20% for P_{Cyan} pucks, and 70% for P_{Yellow} pucks.

5.2.2.4 Actions

Actions in the multi-foraging domain take the form of a robot changing the type of puck for which it is foraging. An action a, denoted as

$$a =< R_{Red} > \quad (5.3)$$

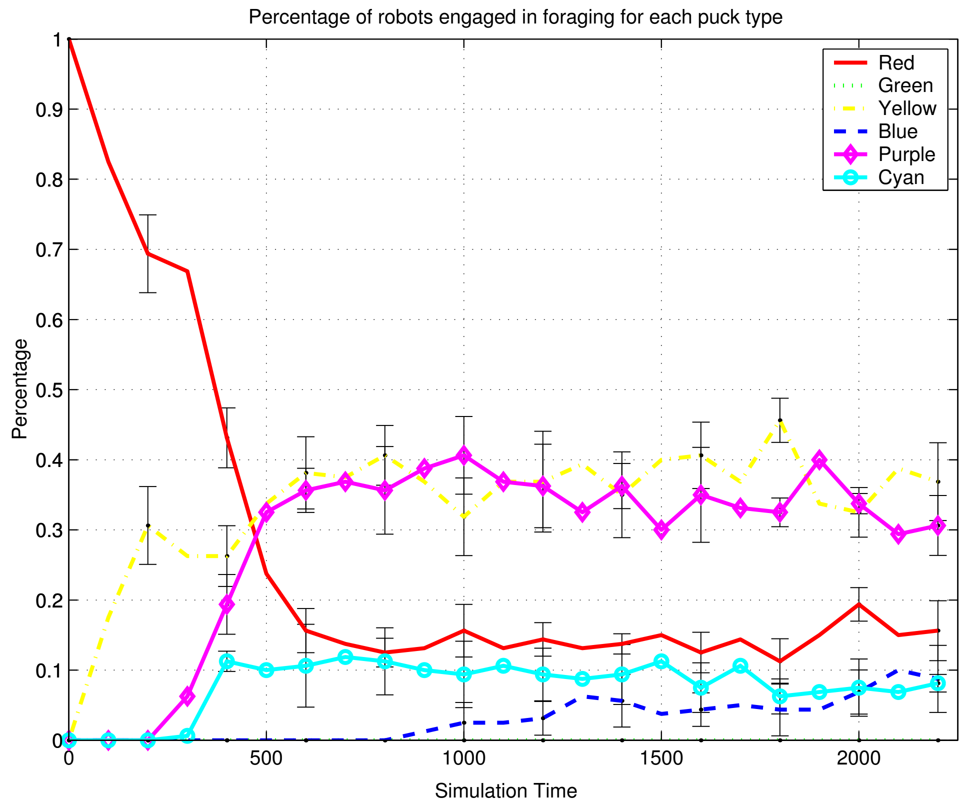

Figure 5.12: Task allocation results for the task shown in Figure 5.10 using the DAct-NoIS-NoComm controller. The plot shows the percentage of robots engaged in foraging the different puck types over time. The data are averaged over 20 simulation trials and the bars represent 1 standard deviation.

corresponds to the case where a robot engages in foraging for P_{Red} pucks. Similar actions exist for R_{Green}, R_{Blue}, R_{Yellow}, R_{Cyan}, and R_{Purple}.

Similar to the construction task domain, the competency actions for the multi-foraging task domain include low-level actions such as performing local navigation and obstacle avoidance. Since the world state in this domain consists of the type of puck each robot is foraging, robot motion tasks such as these do not affect the world state. These competency actions allow the robots to safely move around the world and, when appropriate, an action in the action function will intervene and change the type of puck for which the robot forages.

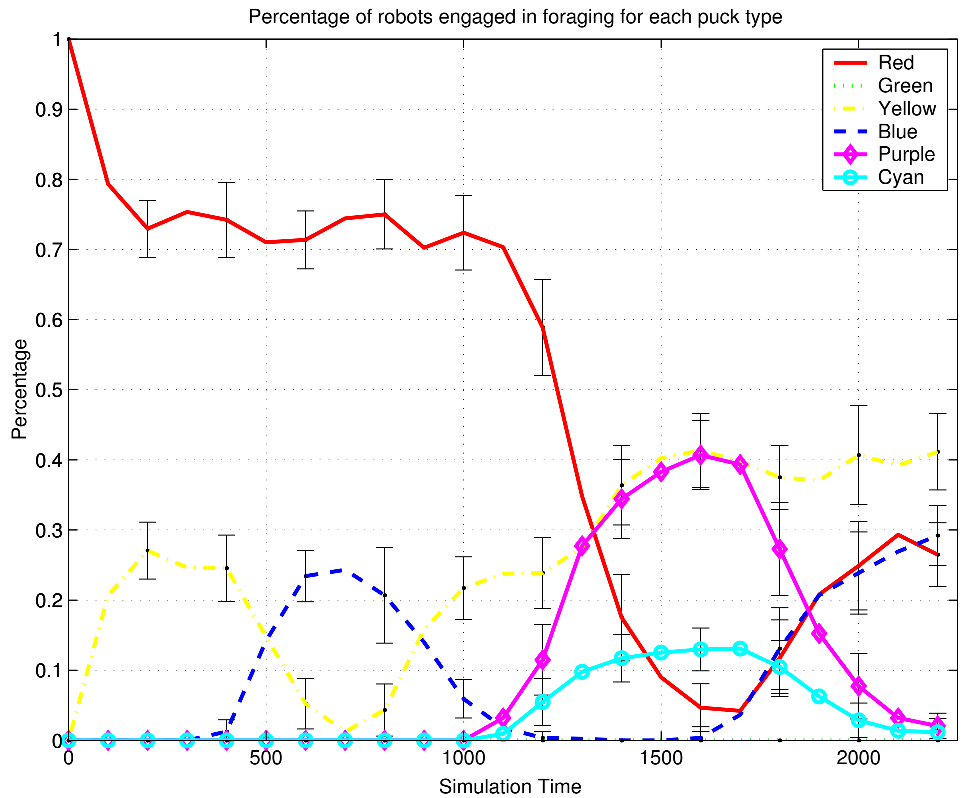

Figure 5.13: Task allocation results for the task shown in Figure 5.10 using the DAct-IS-NoComm controller. The plot shows the percentage of robots engaged in foraging the different puck types over time. The data are averaged over 20 simulation trials and the bars represent 1 standard deviation.

5.2.3 Results

This section presents results from the application of the controller design methodology to the task shown in Table 5.10.

5.2.3.1 DAct-NoIS-NoComm

Figure 5.12 shows the results from the synthesized DAct-NoIS-NoComm controller for the task shown in Figure 5.10. This controller has deterministic action selection and is both stateless and non-communicative. As the results indicate, this controller is not able to carry out this task effectively.

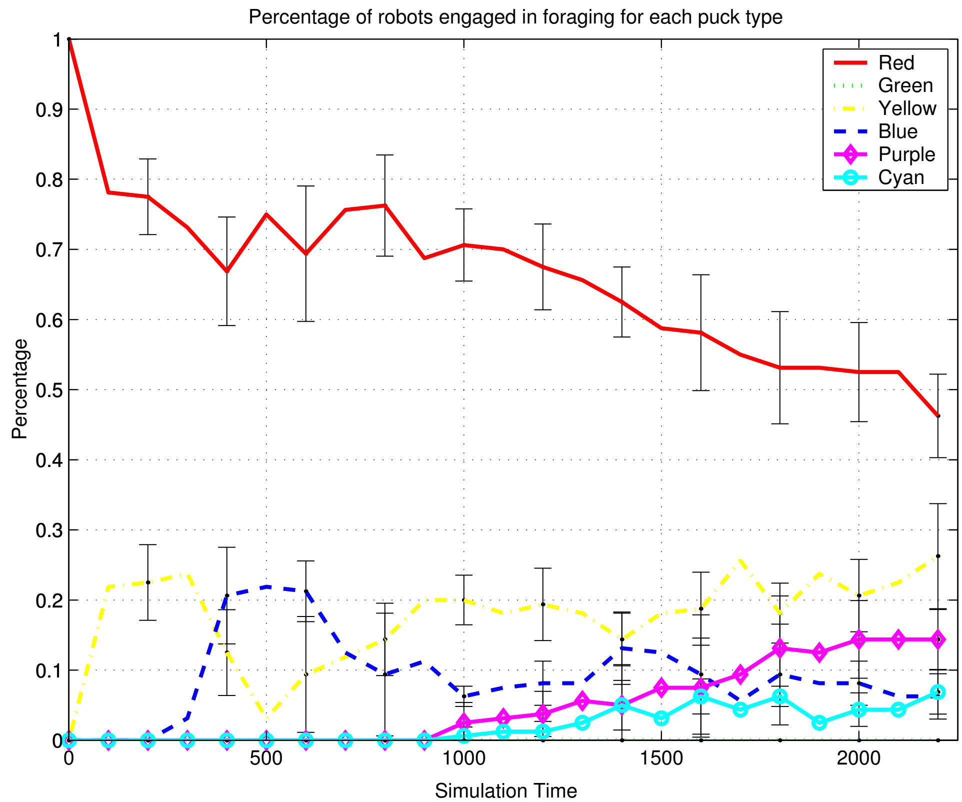

Figure 5.14: Task allocation results for the task shown in Figure 5.10 using the DAct-NoIS-Comm controller. The plot shows the percentage of robots engaged in foraging the different puck types over time. The data are averaged over 20 simulation trials and the bars represent 1 standard deviation.

The robots are able to carry out the first portion of the task, up to the task state $< R_{Red} = 70\%, R_{Green} = 0\%, R_{Blue} = 0\%, R_{Cyan} = 0\%, R_{Yellow} = 30\%, R_{Purple} = 0\% >$. Once this state is achieved, since the robots are stateless, there is no way for them to know whether the next allocation they should be heading toward is $< R_{Red} = 70\%, R_{Green} = 0\%, R_{Blue} = 30\%, R_{Cyan} = 0\%, R_{Yellow} = 0\%, R_{Purple} = 0\% >$ or $< R_{Red} = 0\%, R_{Green} = 0\%, R_{Blue} = 0\%, R_{Cyan} = 20\%, R_{Yellow} = 40\%, R_{Purple} = 40\% >$ (the third and fifth states from the left in Figure 5.10).

As was experienced in the construction task domain, stateless robots are not able to effectively deal with perceptual aliasing issues such as this.

5.2.3.2 DAct-IS-NoComm

The results from the synthesized DAct-IS-NoComm controller for the task shown in Figure 5.10 are shown in Figure 5.13. This controller, which maintains persistent internal state, is able to correctly execute this task. As opposed to the stateless controller mentioned above, this controller is able to use internal state to distinguish between the perceptual aliasing issues inherent in this task definition.

5.2.3.3 DAct-NoIS-Comm

Figure 5.14 shows the results from the synthesized DAct-NoIS-Comm controller for the task shown in Figure 5.10. This controller is stateless, but has inter-robot communication capabilities. Although this controller is able to leverage the use of communication to progress through the first third of the task, it is not able to correctly execute the remainder of the task.

The first third of the task is relatively simple, with the robots only needing to deal with a single class of perceptual aliasing issues related to the proportions of R_{Yellow} and R_{Blue} robots. However, after this first phase of the task, there become many more complicated issues to deal with. The use of communication is able to effectively steer the inertia of the robots' action during the first third of the task, but thereafter, the perceptual aliasing issues compound and the robots are no longer able to determine the current state of the task. At this point, the robots' actions are no longer coordinated and they are no longer able to correctly execute the remainder of the task.

5.2.4 Discussion

This section has presented the application of the controller design methodology to task allocation in a multi-foraging task domain. Controllers using deterministic action functions and with and without both internal state and communication were presented.

Due to the dynamic nature of this task domain, only the controller that is able to maintain persistent internal state was effective. The controller using communication had some measure of success, but is ultimately not effective. The controller without internal state and communication was not effective on the task in this domain.

5.3 Summary

This chapter has demonstrated the application of the controller design methodology presented in this dissertation to a multi-robot construction task domain and to task allocation in a multi-foraging task domain. Synthesized controllers and experimental results from their implementation were provided for the synthesis of controllers using internal state, communication, and probabilistic action selection.

Ideally, a controller design methodology should be general and applicable to a wide variety of task domains; however, it should not be so general as to not be able to capture the specifics of a particular task domain. Through the application of the design methodology to these two task domains representing different requirements and characteristics, the generality of the methodology was illustrated. Furthermore, it was shown that the design methodology is more than just a formalism, but rather that is has been applied to these domains and produced working robot controllers.

Chapter 6

Conclusions

The design of single robot systems (SRS) has greatly benefited from the formalisms provided by control theory – the design of MRS is in need of analogous formalisms. In response to this need, this dissertation presented a principled MRS controller design methodology. The successful deployment of a task-achieving MRS depends on the effective coordination of the interactions between the robots and between the robots and the task environment. The presented methodology is grounded by the use of a formalism that describes the entities involved in these interactions: the world, the task, and the robots themselves. Based on this formalism, a suite of systematic controller synthesis methods were presented. Each method produced a working robot controller using a different combination of control features, including the use of probabilistic action selection, the maintenance of persistent internal state, and the use of inter-robot communication. Furthermore, these synthesis methods were integrated with a probabilistic microscopic modeling approach of MRS analysis. These integrated synthesis and analysis methods produce a controller design methodology that is capable of synthesizing optimized robot controllers.

The controller design methodology was applied to the synthesis of controllers in two task domains, a multi-robot construction domain and task allocation in a multi-foraging task domain. These two task domains represent two different types of multi-robot tasks. The former is a relatively static domain in which robots perform discrete actions. The later is a highly dynamic domain requiring tight coordination among the robots to successfully perform the task. By applying the design methodology to these two domains, both the generality of the method to different task domains as well as its pragmatic benefits of of producing working robot controllers was demonstrated.

6.1 Directions for Further Research

The conception of formal MRS controller design methodologies is still in its infancy, and the approach presented in this dissertation only represents a first step toward a principled and general design methodology. However, the beginnings of a principled foundation have been formed and, with the usefulness of its currently simple incarnation demonstrated, further extensions within this methodology are now possible.

A direction that can be pursued is in the development of additional synthesis methods. MRS are capable of performing a wide variety of tasks, and the controllers that the presented design methodology is capable of synthesizing represent only a small subset of the space of possible controllers. There exists far more complex controllers that are potentially very useful; however, they remain out of reach of the design methodology presented in this dissertation. For example, there exist more potential ways in which internal state and communication can be used in robot control. This dissertation only considered simple uses of these important control features, leaving a large space of further design possibilities unexplored.

Another direction the presented methodology may be extended is in the inclusions of more facets of the overall MRS design space. This dissertation considered the principled design of controllers, assuming a world, task, and robot platforms are provided. A useful function of a general MRS design methodology would be to also include the design of the sensor system along with the robot controllers. Sensing capabilities of the robots have a great impact on the nature and requirements of the robot controllers. Simultaneously designing sensing and control systems would be a very interesting pursuit that would lead to a further understanding on the relationship between robot control, sensing capabilities, and resulting task performance. Similarly, such a process could be carried out on the design of the actuation capabilities of the physical robot hardware.

The principled design of effective coordinated MRS for a diverse array of tasks is a great challenge. This dissertation has provided a validated starting point that future work may now extent.

Bibliography

[1] *Webster's Revised Unabridged Dictionary.* C. & G. Merriam Co., 1996.

[2] W. Agassounon and A. Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *First International Joint Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1090–1097, Bologna, Italy, 2002. ACM Press.

[3] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specification of hybrid systems in charon. In *Proceedings of the 3rd International Workshop on Hybrid Systems: Computation and Control*, pages 6–19, Pittsburgh, PA, Mar 2000.

[4] R. Arkin and T. Balch. Aura: Principles and practive in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:175–189, 1997.

[5] R. Arkin and T. Balch. Cooperative multiagent robotic systems. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, pages 277–296. MIT/AAAI Press, Cambridge, MA, 1998.

[6] R. Arkin, T. Balch, and E. Nitz. Communication of beahvioral state in multi-agent retrieval tasks. In *IEEE International Conference on Robotics and Automation*, pages 588–594, Los Alamitos, CA, May 1993.

[7] T. Balch. Measuring robot group diversity. In T. Balch and L. E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*, pages 93–135. AK Peters, 2002.

[8] T. Balch and R. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1995.

[9] T Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[10] Tucker Balch. The impact of diversity on performance in multi-robot foraging. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 92–99, Seattle, WA, 1999. ACM Press.

[11] R. Beckers, O. Holland, and J. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems*, pages 181–189, Cambridge, MA, 1994.

[12] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[13] K. Bohringer, R. Brown, B. Donald, and J. Jennings. Distributed robotic manipulation: Experiments in minimalism. In O. Khatib, editor, *Experimental Robotics IV, Lecture Notes in Control and Information Sciences 223*, pages 11–25. Springer Verlag (Berlin), 1997.

[14] E. Bonabeau, M. Dorigo, and G. Théraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[15] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, Mar 1986.

[16] R. Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[17] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7–27, 1997.

[18] A. Cassandra, L. P. Kaelbling, and M. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1023–1028, Seattle, WA, 1994.

[19] P. R. Cohen and H. J. Levesque. Confirmation and joint action. In *International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 951–958, Sydney, Australia, Aug 1991.

[20] B. Donald, J. Jennings, and D. Rus. Information invariants for distributed manipulation. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *International Workshop on the Algorithmic Foundations of Robotics*, pages 431–459, Boston, MA, 1995.

[21] B. R. Donald. Information invariants in robotics. *Artifical Intelligence*, 72(1–2):217–304, 1995.

[22] Gregory Dudek, Michael Jenkin, and Evangelos Milios. A taxonomy of multirobot systems. In Tucker Balch and Lynne Parker, editors, *Robot Teams: From Diversity to Polymorphism*, pages 3–22. A.K. Peters, Natick, Massachusetts, Apr 2002.

[23] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*, pages 263–270, Seattle, Washington, USA, Aug 1999. ACM Press.

[24] D. Floreano. Patterns of interactions in shared environments. In *Proceedings of the Second European Conference on Artificial Life*, pages 347–366, Brussels, Belgium, May 1993.

[25] J. Fredslund and M. J. Matarić. A general, local algorithm for robot formations. *IEEE Transactions on Robotics and Automation, Special Issue on Multi-Robot Systems*, 18(5):837–846, 2002.

[26] M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 677–682, Seattle, Washington, 1987.

[27] B. Gerkey and M. Matarić. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3862–3867, Taipei, Taiwan, Sep 2003.

[28] B. Gerkey and M Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *International journal of robotics research*, 23(9):939–954, 2004.

[29] B. Gerkey, R. Vaughan, K. Stoey, A. Howard, G. Sukhatme, and M. Matarić. Most valuable player: A robot device server for distributed control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1226–1231, Maui, Hawaii, Oct 2001.

[30] D. Goldberg and M Matarić. Maximizing reward in a non-stationary mobile robot environment. *Best of Agents 2000, special issue of Autonomous Agents and Multi-Agent Systems Journal*, 6(3):287–316, 2003.

[31] Dani Goldberg and Maja J. Matarić. Interference as a tool for designing and evaluating multi-robot controllers. In *National Conference on Artificial Intelligence*, pages 637–642, Providence, Rhode Island,, Jul 1997.

[32] Dani Goldberg and Maja J. Matarić. Design and evaluation of robust behavior-based controllers for distributed multi-robot collection tasks. In Tucker Balch and Lynne E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*, pages 315–344. AK Peters, 2002.

[33] B. Grosz and S. Kraus. Collaborative plans for group activities. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 367–373, Chambery, France, 1993.

[34] O. Holland and C. Melhuish. Stigmergy, self-organization, and sorting in collective robotics. *Artificial Life*, 5(2):173–202, 2000.

[35] A. Howard, L. E. Parker, and G. Sukhatme. The SDR experience: Experiments with a large-scale heterogenous mobile robot team. In *9th International Symposium on Experimental Robotics*, Singapore, Jun 2004.

[36] Martinoli A Billard A Ijspeert, A and L Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171, 2001.

[37] C. Jones and M. Matarić. Adaptive division of labor in large-scale minimalist multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1969–1974, Las Vegas, Nevada, Oct 2003.

[38] E. Klavins. Communication complexity of multi-robot systems. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Algorithmic Foundations of Robotics V*, Springer Tracts in Advanced Robotics, pages 275–292. Springer, 2003.

[39] E. Klavins. A computation and control language for multi-vehicle systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 4133–4139, Hawaii, Dec 2003.

[40] E. Klavins. A language for modeling and programming cooperative control systems. In *Proceedings of the International Conference on Robotics and Automation*, pages 3403–3410, New Orleans, Louisiana, Mar 2004.

[41] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[42] S. Koenig and R. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 2301–2308, Minneapolis, Minnesota, Apr 1996.

[43] K. Konolige, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset, D. Schulz, B. Stewart, and R. Vincent. Centibots: Very large scale distributed robotic teams. In *9th International Symposium on Experimental Robotics*, Singapore, Jun 2004.

[44] M. B. Krieger and J.-B. Billeter. The call for duty: Self-organized task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1-2):65–84, 2000.

[45] C. Kube and H. Zhang. The use of perceptual cues in multi-robot box-pushing. In *IEEE International Conference on Robotics and Automation (ICRA-96)*, pages 2085–2090, Minneapolis, Minnesota, Apr 1996.

[46] K. Lerman and A. Galstyan. Mathematical model of foraging in a group of robots: Effects of interference. *Autonomous Robots*, 13(2):127–141, 2002.

[47] K. Lerman, A. Galstyan, A. Martinoli, and A.J. Ijspeert. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, 7(4):375–393, 2001.

[48] K. Lerman, C. Jones, A. Galstyan, and M. Matarić. Analysis of dynamic task allocation in multi-robot systems. Technical report, University of Southern California Center for Robotics and Embedded Systems, CRES-05-002, Apr 2005.

[49] K. Lerman, A. Martinoli, and A. Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. In *Proceedings of the International Conference on the Simulation of Adaptive Behavior*, pages 143–152, Los Angeles, CA, Aug 2004.

[50] A. Martinoli and K. Easton. Modeling swarm robotic systems. In B. Siciliano and P. Dario, editors, *Experimental Robotics VIII*, pages 297–306. Springer-Verlag, 2002.

[51] A. Martinoli, K. Easton, and W. Agassounon. Modeling swarm robotic systems: a case study in collaborative distributed manipulation. *Special issue on Experimental robotics, International Journal of Robotics Research*, 23(4):415–436, 2004.

[52] A. Martinoli, A. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: From probabilistic modeling to experiments with real robots. *Robotics and Autonomous Systems*, 29:51–63, 1999.

[53] M. Matarić. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4: 51–80, Dec 1995.

[54] M. Matarić. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2–4):321–331, Dec 1995.

[55] M. Matarić. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Theoretical and Experimental Artificial Intelligence, special issue on Software Architectures for Physical Agents*, 9(2-3):323–336, 1997.

[56] C. Melhuish, J. Welsby, and C. Edwards. Using templates for defensive wall building with autonomous mobile ant-like robots. Technical report, Towards Intelligent Mobile Robots, University of Manchester, Department of Computer Science, UMCS-99-3-1, Bristol, UK, Mar 1999.

[57] R. Nelson and R. Wilson. *Graph Colourings*. Longman Scientific and Technical, 1990.

[58] L. Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *Transactions on Robotics and Automation*, 14(2):220–240, Apr 1998.

[59] L. Parker. Toward the automated synthesis of cooperative mobile robot teams. In *Proceedings of SPIE Mobile Robots XIII*, volume 3525, pages 82–93, Boston, MA, Nov 1998.

[60] L. E. Parker. Designing control laws for cooperative agent teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 582–587, Atlanta, GA, May 1993.

[61] D. Pynadath and M. Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 873–880, Bologna, Italy, July 2002.

[62] D. V. Pynadath and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

[63] J. R. Spletzer and C. J. Taylor. A framework for sensor planning and control with applications to vision guided multi-robot systems. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 378–383, Kauai, Hawaii, 2001.

[64] L Sugawara and M. Sano. Cooperative acceleration of task performance: Foraging behavior of interacting multi-robot system. In *Physica D*, volume 100, pages 343–354, 1997.

[65] J. Svennebring and S. Koenig. Trail-laying robots for robust terrain coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 75–82, Taipei, Taiwan, Sep 2003.

[66] M. Tambe. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 22–28, Providence, Rhode Island, 1997.

[67] R. Vaughan, K. Stoy, G. Sukhatme, and M. Matarić. Blazing a trail: insect-inspired resource transportation by a robot team. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, pages 111–120, Knoxville, TN, Oct 2000.

[68] J. Wawerla, G. S. Sukhatme, and M. Matarić. Collective construction with multiple robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2696–2701, Lausanne, Switzerland, Sep 2002.

[69] B. Werger. Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams. *Artificial Intelligence*, 110:293–320, 1999.

[70] B. Werger and M. Matarić. From insect to internet: Situated control for networked robot teams. *Annals of Mathematics and Artificial Intelligence*, 31(1–4):173–198, 2000.

[71] Barry Werger and Maja J. Matarić. Robotic food chains: Externalization of state and program for minimal-agent foraging. In Jean-Arcady Meyer Jordan Pollack Pattie Maes, Maja Mataric and Stewart W. Wilson, editors, *Proceedings of the International Conference on Simulation of Adaptive Behavior*, pages 625–634. MIT Press/Bradford Books, 1996.

[72] A. Zhang, M. Chung, B. Lee, R. Cho, S. Kazadi, and R. Vishwanath. Variance in converging puck cluster sizes. In *The First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 209–217, Bologna, Italy, Jul 2002.

[73] B. Zhang, G. Sukhatme, and A. Requicha. Adaptive sampling for marine microorganism monitoring. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1115–1122, Sendai, Japan, Sep 2004.