In *Proceedings of the Hawaii International Conference on Computer Sciences, pages 27--32, Waikiki, Hawaii, Jan 2004.*

The Use of Internal State in Multi-Robot Coordination

Chris Jones and Maja J Matarić
Computer Science Department
University of Southern California
941 West 37th Place, Mailcode 0781
Los Angeles, CA 90089-0781
{cvjones|maja}@robotics.usc.edu

***Abstract*— Coordination is an essential characteristic of any task-achieving multi-robot system (MRS), whether it is accomplished through an explicit or implicit coordination mechanism. There is currently little formal work addressing how various MRS coordination mechanisms are related, how appropriate they are for a given task, what capabilities they require of the robots, and what level of performance they can be expected to provide. Given a MRS composed of homogeneous robots, we present a method for automated controller construction such that the resulting controller makes use of internal state and no explicit inter-robot communication, yet is still capable of correctly executing a given task. Understanding the capabilities and limitations of a MRS composed of robots not capable of inter-robot communication contributes to the understanding of when and why inter-robot communication becomes necessary and when internal state alone is sufficient to achieve the desired coordination. We validate our method in a multi-robot construction domain.**

I. Introduction and Motivation

Coordination is an essential characteristic of any task-achieving multi-robot system (MRS). The nature of the coordination may take many forms, seemingly limited only by the creativity of the designer. *Explicit coordination* mechanisms make use of internal state maintained by individual robots and explicit inter-robot communication, often involving centralized or hierarchical control. Alternative approaches, involving *implicit coordination*, make use of fortuitous structure in the environment and its synergistic relationship to the task definition and the robots' sensing, control, and mobility characteristics to produce a different class of coordination techniques, often categorized by terms such as *emergent, self-organized, or stigmergic*.

From the perspective of the designer, the coordination mechanism employed in a given task domain is often heavily influenced by personal preference and less so by a formal understanding of why one is more appropriate than another or how the various coordination mechanisms are related. There is little formal work addressing the question of rationally choosing the most appropriate MRS coordination mechanism for a given task domain and performance requirements. Furthermore, there is little work on the more fundamental question of how various classes of coordination are related.

To provide insight into these questions, we are developing a coordination formalism which provides a framework for precisely defining and reasoning about the intertwined entities intrinsically involved in any task-achieving multi-robot system – the task environment, task definition, and the capabilities of the robots themselves. Our approach is novel in that it expresses a principled effort to understand the relationship between *explicit coordination mechanisms*, such as those primarily relying on the use internal state and direct communication, and more emergent *implicit coordination mechanisms*, which tend to make use of environment and task structure and more indirect forms of communication.

Our initial investigations have centered on multi-robot systems composed of robots equipped with internal state but lacking the capability for explicit, direct inter-robot communication. A formal understanding of the capabilities and limitations of such a system contributes to the understanding of when and why internal state alone is sufficient to achieve the desired coordination and when inter-robot communication becomes necessary. Furthermore, we hope this work will help elucidate the currently ongoing discussions regarding the meaning behind labels such as team robotics vs. swarm robotics, etc.

Toward this end, we present an automated multi-robot controller generation algorithm. The generated controller, when run by all robots in a homogeneous multi-robot system, will correctly execute a given task. We demonstrate our formalism in a multi-robot construction domain where we are able to provide specific task instances accompanied by formal explanations of the suitability of the use of internal state.

This paper is organized as follows. In Section II we provide the relevant related work. In Section III we provide formal definitions of the task environment, task definition, and robot characteristics used in the formalism. In Section IV we present an algorithm to construct the controller of each robot in a MRS such that a given task is correctly executed. In Section V we demonstrate and validate the formalism in a Multi-Robot Construction domain. In Section VI we draw conclusions and discuss future work.

II. Related Work

This section summarizes some of the most relevant related work involving characterization and analysis of coordination in multi-robot systems. The work of Parker [14] discusses the trade-offs of local versus global information for coordination in multi-robot systems. Beckers et al. [2] present a coordination mechanism in a multi-robot object clustering domain. Matarić [12] presents work on group coordination in multi-robot systems using a collection of simple basis behaviors. The information invariants work of Donald [5] addresses the problem of determining the information requirements to perform robot tasks and means in which this information may be acquired. Dudek et al. [6] present a taxonomy which classifies multi-robot systems based on communication and computational capabilities. In a clustering domain, Martinoli et al. [11] demonstrate how the collective behavior of a group of mobile robots can be accurately studied using a simple probabilistic model. They show how the results of the model are descriptive of the results obtained through experiments with real robots and in sensor-based simulations. Balch [1] presents hierarchical social entropy, an information theoretic method of analysis used to determine the extent of diversity among robots in multi-robot systems. Goldberg and Matarić [8] precisely define the foraging task for multi-robot systems and provide a collection of general distributed behavior-based coordination algorithms and their empirical evaluation. Gerkey and Matarić [7] present a formalism for the analysis of task allocation in multi-robot systems with an emphasis on explicit coordination mechanisms. Lerman and Galstyan [10] present a mathematical model of the dynamics of collective behavior in a multi-robot adaptive task allocation domain.

The domain in which we validate our approach is multi-robot construction. Related work in this area includes the work of Bonabeau et al. [4] which uses a rule-based model in the construction of biologically-plausible nest structures similar to those of some wasp species. Bonabeau et al. [3] investigate the use of genetic algorithms to generate such rules used in the construction of biologically-plausible structures and explores the relationship between the space of rules and resulting structures. In the area of construction by physical robots, Melhuish et al. [13] demonstrate how a group of minimalist robots can construct defensive walls using biologically-inspired templates. Wawerla et al. [15] present work on the comparison of different coordination strategies in the construction of simple 2D structures using a group of mobile robots. Jones and Matarić [9] present a method by which to automatically generate controllers for rule-based agents using local sensing and control in an intelligent self-assembly domain.

III. Definitions and Notation

In this section we formally define the intertwined entities intrinsically involved in any task-achieving multi-robot system – the task environment, task definition, and the capabilities of the robots themselves, including control, sensing, and maintenance of internal state.

A. Task Environment

The task environment is the world in which the multi-robot system is expected to perform a defined task. The environment state, s, at any given time is an element of the finite set S of all possible states. An action, a, performed in the environment by a robot is drawn from the finite set A of all possible actions. An environment is defined by a state transition function $s_j = F(s_i, a)$, which states that when action $a \in A$ is executed in state $s_i \in S$, the next state will be $s_j \in S$. In this work, we assume the state can transition only as the result of an action performed by a robot.

B. Task Definition

We define a task, T, assumed to be Markovian, as a set of n ordered environment states, $T_s = \{s_1, ..., s_n\}$ which must be progressed through in sequence. From here on, the use of the word *state* refers to task state. An action a is called a *task action* for state s_i, denoted by $A_t(s_i) \in A$, if $s_{i+1} = F(s_i, a)$. A task T is said to be executed correctly if and only if for each task state $s_i \in T_s$ any executed action a falls into one of the two following categories: $a = A_t(s_i)$ or $s_i = F(s_i, a)$. This means that all performed actions are either task actions or are actions which do not result in a task state transition.

C. Observations

An observation made by an individual robot consists of accessible information external to the robot and formally represents a subset of the task state. The content and properties of an observation are dependent on the specific sensing properties of the robot. The finite set of all possible observations is denoted as X. Since a given observation may occur in multiple states, for notational convenience we use x_s to mean the observation x as made in state s. An observation x with no sub-script, unless otherwise noted, refers to the observation x as made in any state.

In state s, the function $G(s) \subseteq X$ returns the set of all observations which can be made in s. An observation x is called *unique* if and only if there exists only one state s for which $x \in G(s)$. The observation at the physical location where the task action of state s is to be executed is denoted by $Y(s) \in G(s)$.

D. Robot Characterization

A robot's internal state is denoted by *m*. The finite set of all possible internal state values is denoted by *M*. A robot's observation, *x*, at any given time is an element of *X*. Two functions define a robot's action in the environment, known collectively as the robot's *controller*. The deterministic action function $a = B(x, m)$ specifies the robot's action, $a \in A$, given its current observation is *x* and its internal state is *m*. The internal state transition function $m' = L(x, m, a)$ is a deterministic function specifying the robot's next internal state value given its current observation *x*, its current internal state *m*, and the action it is executing *a*.

A value of "·" for any parameter in the *B* or *L* functions signifies the set of all possible values for that parameter (i.e., "don't care"). For example, the rule $a = B(x, \cdot)$ specifies that if a robot makes observation *x*, it will perform action *a* regardless of the current internal state value.

IV. Building a Satisficing Controller

We now describe a method by which a controller using internal state can be automatically constructed such that a given task is correctly executed. We call such a controller *satisficing*. Properly executing a task in a multi-robot system has additional challenges from doing so in a single robot system. It can never be assumed that a particular robot will or will not make a certain observation, as it could be the case that a robot is completely unaware of the progress of the task that is resulting from the actions of other robots. Formally, from the perspective of an individual robot, the task environment is highly non-stationary.

A satisficing controller must satisfy two conditions. First, for all $s_i \in T_s$, the action function must specify a rule of the form $A_t(s_i) = B(Y(s_i), m)$ where $m \in M$. Second, if there exists an observation $x \in G(s_i)$ such that $x = Y(s_j)$ and $i < j$, the internal state value must be transitioned as the result of some observation which is guaranteed to be made after all observations of x_{s_i} and prior to the final observation of $Y(s_j)$. The procedure in Figure 1 presents an algorithm which constructs a satisficing controller based on the satisfaction of these two conditions.

If there exists a state $s_k \in T_s, k > j$ for which $x \in G(s_k)$ we note that since the environment is non-stationary from the perspective of the individual robot, internal state alone is not sufficient to distinguish x_{s_j} from x_{s_k} and is therefore not sufficient to guarantee correct task execution in cases where $x_{s_j} = Y(s_j)$. Assuming internal state is sufficient, the worst case in terms of necessary internal state values is $\|T_s\| - 1$. The best case is that no internal state is necessary, which occurs if for all $s_i \in T_s$ the observation $Y(s_i)$ is unique.

```
procedure Build_Controller()
  m = 0
  B ← {}
  L ← {}
  LastObsState = 0
  for i = 1 to ||T_s|| do
    if ∃s_j s.t. LastObsState < j < i and
         Y(s_i) ∈ G(s_j) then
      m′ = m + 1
      if ∃x ∈ (G(s_i) − Y(s_i)) s.t.
           ∄s ≥ LastObsState : x ∈ G(s) then
        LastObsState = i
        L ← L ⋃ {m′ = L(x,m,·)}
      else
        LastObsState = i - 1
        L ← L ⋃ {m′ = L(Y(s_{i−1}),m,A_t(s_{i−1}))}
        B ← B - {A_t(s_{i−1}) = B(Y(s_{i−1}),·)}
        B ← B ⋃ {A_t(s_{i−1}) = B(Y(s_{i−1}),m)}
      endif
      m = m′
    endif
    B ← B ⋃ {A_t(s_i) = B(Y(s_i),m)}
  endfor
end procedure Build_Controller
```

Fig. 1. Procedure for building a satisficing controller.

V. Validation: Coordination in Multi-Robot Construction

We experimentally demonstrate and validate our approach to the design of a satisficing controller in a multi-robot construction task. The construction task requires the placement of a series of square colored bricks, 0.5 meters on a side, into a desired 2D planar structure in a specified sequence. For all examples used in this section, a brick's color is denoted by the letters `R`, `G`, `B`, and `Y` which stand for `Red`, `Green`, `Blue`, and `Yellow`, respectively. The construction task starts with a seed structure, which is a small number of initially placed bricks forming a core structure.

Experimental demonstration was performed using Player and the Stage simulation environment. Player (Gerkey et al. 2001) is a server that connects robots, sensors, and control programs over the network. Stage (Vaughan 2000) simulates a set of Player devices. Together, the two represent a high-fidelity simulation tool for individual robots and robot teams which has been validated on a collection of real-world robot experiments using Player control programs transferred directly to physical Pioneer 2DX mobile robots.

Our construction task is conducted in a circular arena of approximately 315 square meters using 6 robots. The robots are realistic models of the ActivMedia Pioneer 2DX mobile robot. Each robot, approximately 30 cm in diameter, is equipped with a differential drive, a forward-facing 180 degree scanning laser rangefinder, and a forward-

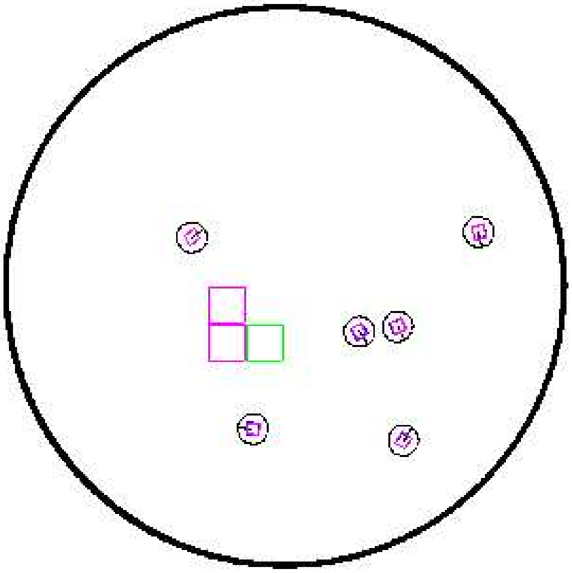

Fig. 2. A screenshot from the Stage simulation environment populated 3 bricks and 6 robots performing the construction task.

looking color camera with a 60-degree field-of-view and a color blob detection system. The bricks are taller than the robot's sensors, so the robots can only sense the bricks on the periphery of the structure. Figure 2 shows a screenshot of the simulation environment while the construction task is being performed.

A. Definition of the Construction Task

We define the environment state for the construction task as being a specific spatial configuration of bricks; therefore, a construction task is defined as a desired sequence of brick configurations – a specific construction sequence. The actions we are interested in are the placement of individual bricks to the growing structure; we do not consider construction tasks in which robots may remove bricks from the structure nor those in which sub-structures consisting of multiple bricks may be connected together. Other actions performed by the robots, such as moving through the environment, do not affect task state. Table I shows a set of environment states defining the example construction task that we will use throughout this section.

B. Observations in the Construction Task

Since the content of an observation is dependent on a robot's sensing capabilities, an observation in the construction domain is the spatial configuration and color of bricks in the field-of-view of the robot's laser rangefinder and color camera.

There are two general categories of observations that can be made. The first is two adjacent, aligned bricks.

Observations in $G(s_2)$
<FLUSH G B>
*<FLUSH B G>
<CORNER G B>
<FLUSH R G>
<FLUSH B R>

TABLE II

ALL OBSERVATIONS IN $G(s_2)$ FROM TABLE I. THE OBSERVATION $Y(s_2)$ IS MARKED BY A "*".

Such an observation would be made, for example, if in state s_1 in Table I, a robot were positioned above and oriented toward the surface of the structure made up by the Red and Blue bricks. Such an observation is denoted as <FLUSH R B>.

The second observation category consists of two bricks forming a corner. Such an observation would be made if in state s_0 in Table I a robot were positioned in the upper right-hand corner and oriented toward the corner formed by the Red and Green bricks. Such an observation is denoted as <CORNER R B>.

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>. Given the task state s_2 from Table I, Table II lists all observations in the set $G(s_2)$ with the observation $Y(s_2)$ highlighted.

C. Brick Placement Actions

The only actions in this construction domain that can transition the task state are brick placement actions. There are three such actions, with the first being the placement of a brick on the right side (from the perspective of the acting robot) of a pair of adjacent, aligned bricks. An example action of this type can be seen in the placement of the Green brick which transitions the state in the task in Table I from s_1 to s_2. Such an action is denoted as <G RIGHT FLUSH R B>.

The second action type is similar to the first except the brick is placed on the left side of a pair of adjacent, aligned bricks. An example of this action type can be found in Figure I in the placement of the Yellow brick which transitions the state from s_2 to s_3. Such an action is denoted as <Y RIGHT FLUSH B G>.

The third action type is the placement of a brick in the corner formed by two other bricks. An example of this action type can be seen found in Figure I in the placement of the Blue brick which transitions the state from s_0 to s_1. Such an action is denoted as <B CORNER R G>.

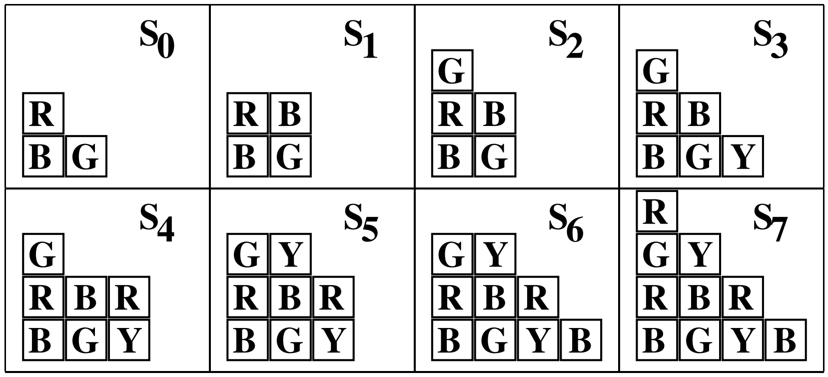

TABLE I

A SEQUENCE OF ENVIRONMENT STATES THAT DEFINE A CONSTRUCTION TASK. THE STRUCTURE IMAGES ARE TAKEN FROM A BIRDS-EYE-VIEW. EACH BRICK IS LABELED WITH ITS COLOR: R=RED, B=BLUE, G=GREEN, Y=YELLOW.

Observation	m	$->$	m′	Action
`<CORNER R G>`	00	$->$	00	`<B CORNER R G>`
`<FLUSH R B>`	00	$->$	00	`<G RIGHT FLUSH R B>`
`<CORNER G B>`	00	$->$	01	`<Y CORNER G B>`
`<FLUSH B G>`	01	$->$	01	`<Y LEFT FLUSH B G>`
`<CORNER B Y>`	01	$->$	01	`<R CORNER B Y>`
`<FLUSH G Y>`	01	$->$	10	`No Action`
`<FLUSH R Y>`	10	$->$	10	`<B LEFT FLUSH R Y>`
`<CORNER R B>`	10	$->$	11	`No Action`
`<FLUSH G Y>`	11	$->$	11	`<R RIGHT FLUSH G Y>`

TABLE III

RULES CONSTITUTING A SATISFICING CONTROLLER FOR THE CONSTRUCTION TASK IN TABLE I.

D. Satisficing Robot Controller

We now describe the satisficing controller for the construction task shown in Table I. The robot makes an observation, and if the current internal state value, observation, and action match one of the rules from the internal state transition function as shown in Table III, the internal state value is transitioned to the value designated by the matched rule. Next, if the current internal state value and observation matches a rule from the action function shown in Table III, the robot visual servos toward the location where the brick placement action is to be performed, as dictated by the matched rule. Once the robot is within range to perform the action, the brick of appropriate color is placed on the structure. If no rule in the function is matched, the robot performs a random walk, makes another observation, and the process repeats.

As can be seen, this satisficing robot controller for the construction task in Table I requires 4 unique internal state values. Our method does not guarantee to generate a satisficing controller using a minimal number of internal state values; however, for this particular construction task, 4 values is the minimal number required to correctly execute the task.

VI. CONCLUSIONS AND FUTURE WORK

We have presented a method for automated multi-robot controller generation for correct task execution. The individual robots in the system execute controllers using internal state but do not have the capability for direct, explicit inter-robot communication. Given these individual robot capabilities, we have shown characteristics the task must exhibit such that these capabilities are sufficient for correct task execution. Understanding the capabilities and limitations of a multi-robot system composed of robots equipped with internal state but lacking the capability for explicit, direct inter-robot communication contributes insight into the larger question of understanding the necessary characteristics of a coordination mechanism in a multi-robot system required to correctly execute a given task. Furthermore, such understanding can aid the designer in making modifications to the task environment or definition or the robot capabilities in order to transform a situation in which internal state alone is not sufficient to one in which it is sufficient to achieve correct task execution.

Our future work includes the development of an algorithm for constructing satisficing controllers using a *minimal* number of unique internal state values. Applying the formalism presented in this paper, we are also investigating the use of explicit inter-robot communication in multi-robot coordination. Specifically, we are studying in what circumstances such communication may replace or beneficially augment the use of internal state and when it may be required in order to correctly execute a MRS task.

VII. Acknowledgments

This work is supported in part by Defense Advanced Research Projects Agency (DARPA) Grant F30602-00-2-0573 and in part by National Science Foundation Grant EIA-0121141.

References

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

[2] R. Beckers, O. Holland, and J. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Proceedings of Artificial Life IV, Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 181–189, Cambridge, Massachusetts, 1994.

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

[4] E. Bonabeau, G. Théraulaz, E. Arpin, and E. Sardet. The building behavior of lattice swarms. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 307–312, 1994.

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

[6] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots 3*, pages 375–397, 1996.

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

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

[9] C. Jones and M. Matarić. From local to global behavior in intelligent self-assembly. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 721–726, Taipei, Taiwan, 2003.

[10] K. Lerman and A. Galstyan. Macroscopic analysis of adaptive task allocation in robots. In *Proc. of the IEEE International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, Oct 2003. To Appear.

[11] 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.

[12] M. Matarić. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:1:51–80, December 1995.

[13] C. Melhuish, J. Welsby, and C. Edwards. Using templates for defensive wall building with autonomous mobile ant-like robots. In *Towards Intelligent Mobile Robots (TIMR)*, Bristol, UK, 1999.

[14] L. E. Parker. Designing control laws for cooperative agent teams. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA-93)*, pages 582–587, 1993.

[15] J. Wawerla, G. S. Sukhatme, and Maja J Matarić. Collective construction with multiple robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-02)*, pages 2696–2701, Lausanne, Switzerland, Sep 2002.