Using Formal Modeling With an Automated Analysis Tool to Design and Parametrically Analyze a Multirobot Coordination Protocol: A Case Study

Joel M. Esposito and Moonzoo Kim

***Abstract*—Many robot systems employ logic-based or reactive controllers, making them hybrid systems (i.e., mixed discrete continuous). However, designing such control laws in a systematic manner remains a challenging task. In this paper, we apply the formal modeling paradigm to a team of mobile robots. The linear hybrid automata modeling framework is used to describe the high-level design, and the verification software HYTECH is used for symbolic analysis of the description. The goal is to symbolically quantify system-level performance as a function of the design parameters, for the purpose of optimizing and synthesizing design parameters, verifying safe operation, and quantitatively exploring tradeoff issues. In order to make the analysis tractable, a series of restrictive assumptions and simplifications must be made—some dictated by the linear hybrid automata model and others necessitated by computational cost. We comment on the restrictiveness of these assumptions and the overall utility of this automated analysis approach in designing complex robotic systems.**

***Index Terms*—Automata, design automation, formal languages, mobile robots.**

I. INTRODUCTION

DUE TO THE proliferation of small but powerful embedded microprocessors, most mobile robot systems rely on some form of logic-based or reactive control schemes. In addition, these processors often are used to fuse information from a variety of sensors, and communicate with other robots. As such, most robot systems can be considered hybrid systems, which typically consist of a collection of digital programs that interact with each other and with an analog environment. Other examples of hybrid systems include manufacturing controllers, automotive and flight controllers, medical equipment, and microelectromechanical systems. It is well known that the interaction between the discrete and continuous time dynamics of such system can produce rich and unexpected behavior. Unfortunately, designing reliable hybrid control systems is a challenging task. Control theoretic methods are quite limited and vary on a case-by-case basis.

In this paper, we explore the application of formal modeling and analysis to the design of a multirobot coordination and control protocol. This problem is inspired by our experience with our own experimental testbed of a system of autonomous mobile robots [14]. We consider a task that involves exploring a room with obstacles while navigating to a goal position. The task is motivated by military applications (scouting, reconnaissance, and surveillance). Typically, the sensory capabilities of each robot yield only imperfect information about the world, and in particular, each robot has only estimates about the positions of the obstacles. When there are multiple robots that can communicate with one another, they can share knowledge about the world. The challenge then is to design communication protocols, in conjunction with control strategies, so that the team of robots achieves its goal in a coordinated and optimal manner.

Inspired by the success of automated formal methods in discovering subtle errors in hardware designs (cf. [12]), a current trend is to investigate if these techniques can be generalized to obtain design aids for hybrid systems. The methodology advocated by formal approaches to system design requires construction of a high-level description or a (mathematical) model of the system. The model can then be subjected to a variety of mathematical analyses such as simulation, model checking, and performance evaluation. Such modeling and analysis can be performed in early stages of the design, and offers the promise of a more systematic approach and greater automation during the design phase. Unfortunately, the algorithmic analysis of hybrid systems is a challenging problem, and even the simplest analysis problems turn out to be undecidable. However, a useful analysis can be performed for a class of hybrid systems called linear hybrid automata. The analysis procedure involves symbolic fix-point computation over state sets that are represented by linear constraints over system variables, and can be implemented using routines to manipulate convex polyhedra. The procedure has been implemented in the tool HYTECH [3], [16], and has been applied to case studies such as an audio-control protocol [19] and a steam boiler [18].

In the case of our multirobot coordination problem, possible design parameters include the number of robots, the initial positions of the robots, the frequency of communication, the cost of communication (e.g., time required to process messages), and the positions of obstacles and target. Traditional simulation, using a tool like MATLAB (see www.mathworks.com), requires that all parameters remain fixed. The parameters could be sampled at discrete values in the range of interest and the simulation repeated to give the designer some intuition about their impact on performance. However, there is no guarantee or insight about how the system will behave at off-sample points. On the other

Manuscript received March 24, 2005; revised October 5, 2005. This paper was recommended by Associate Editor G. C. Calafiore.

J. M. Esposito is with the Department of Systems Engineering, U.S. Naval Academy, Annapolis, MD 21402 USA (e-mail: esposito@usna.edu).

M. Kim is with the Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: moonzoo@cs.kaist.ac.kr).

Digital Object Identifier 10.1109/TSMCA.2006.886378

hand, when using verification tools such as KRONOS [13], and UPPAAL [22], these parameters are set and the tools are used to detect logical errors by checking whether a high-level model satisfies a temporal logic requirement. In this paper, however, these parameters can be left unspecified, and the HYTECH tool performs an exhaustive symbolic search for all possible settings of the parameters. The information computed by the tool, then, can be used to understand the various tradeoffs and ultimately synthesize parameter values.

This paper contains contributions of interest to two parties: 1) potential users of automated analysis tools especially within the robotics application area; and 2) designers of the next generation of automated analysis and formal methods tools.

From a user's point of view, it illustrates the application of a relatively new technology to the area of robotics. The system considered attempts to model several nontrivial facets of robotics including sensor uncertainty and multirobot communication—explaining how certain restrictions can be met or worked around. More importantly, while previous case studies of formal methods in other application areas have focused on verifying safety properties, we are employing reachability analysis to compare and synthesize design alternatives in a novel way.

From the point of view of the designers of such tools, the negative outcomes of study might provide a useful feedback for the design of the next generation of analysis tools. It turns out that restrictions on the modeling approach, along with computational considerations, necessitated, making a variety of simplifying assumptions—the most restrictive of which was the lack of a Euclidian metric. Ultimately, we were only able to verify rather simplistic scenarios. Moreover, the extension of the results computed using the reduced model to the original problem proved to be quite difficult. Hopefully, the outlined scenario will serve as a challenge problem to guide the research in formal verification of hybrid systems, by illustrating the need for tools which can operate on more general types of mathematical models.

The outline of this paper is as follows. After reviewing the basics of formal verification and the definition of linear hybrid automata in Section III, we explain the multirobot scenario we wish to verify in Section IV. The main effort in this paper concerns modeling the application scenario using the linear hybrid automata. The modeling assumptions, required to fit the linear hybrid automata paradigm and to ensure the analysis is tractable, are discussed in detail in Section V. The results of the analysis experiments are reported in Section VI. Since the analysis is computationally expensive, we could successfully analyze only special cases of the original multirobot scenario. In particular, for two robots and one obstacle, HYTECH was able to synthesize the region of the possible positions of the target for which communication reduces the total distance traveled. While modest, this experiment does yield information that is computed automatically by a general-purpose tool. In Section VII, the limitations of this analysis and the impact of the modeling assumptions are discussed in some detail. Section VIII discusses the lessons learned and points to critical areas for improvement for the next generation of formal modeling and analysis tools.

II. RELATED WORKS

Several nice algorithmic approaches to verification exist [5], [10], [11], [25], but the set of actual automated parametric design tools is rather limited. While nonlinear switching controllers have been designed for systems with several modes of operation (see [7], [31], and [32]), the techniques are generally only applicable for simple systems with relatively few modes. Another approach is to carefully partition the state space into different regions, each with its own specialized control law, variations on this theme can be found in the literature on variable structure systems [31] and on multimodal systems [27]. By selecting the state-space partitions so that regions of interest overlap and by designing controllers with stable equilibrium points which lie in the overlap, it is possible to control the transition from mode to mode with predictable performance. However, requiring stable equilibria to lie in the given regions is difficult in all, but the simplest topological spaces. A game-theoretic approach to designing controllers for hybrid systems with a hierarchical structure is shown to be applicable to automated highway systems [24], [30]. Threaded petri nets [20] and backchaining [9] can be used to synthesize high-level controllers of systems when there is a palette of tunable controllers available. There is, however, little in the way of generally applicable automated design tools (the proceedings of the workshops on hybrid systems provide an excellent survey of various trends [6], [26]).

Here, we examine the most closely related work on applying formal methods to problems in robotics. In [15] and [33], the authors apply the linear temporal logic (LTL) formalism and some popular model checking tools for such systems to the problem of robot motion planning and control. They manually apply a rigorous discrete abstraction procedure; cast the robot's objective in the LTL framework; apply an LTL model checking tool to synthesize a sequence of discrete maneuvers to solve the resulting motion planning problem; and then devise a series of controllers that can implement these discrete maneuvers in continuous time. In [21] and also in [4] and [29], the authors apply a similar approach using timed automata and corresponding analysis tools (ORCCAD and UPPAAL, respectively) to synthesize a sequence of discrete transitions to solve a motion planning problem with moving obstacles.

Both of these sets of works are meritous and appear to have promising outcomes. However, it is important to distinguish them from the work presented here. First they examine the application of LTL and timed automata to area of robotics, while this paper examines a different modeling framework, linear hybrid automata. Second, through a series of abstractions, both works ultimately perform a synthesis on discrete systems; while this paper explicitly considers continuous dynamics (albeit simplified ones) and is able to synthesize optimal values of a continuous parameter. Finally, while the problems they consider are interesting applications of formal methods, many of them are essentially solved problems in robotics (e.g., motion planning on a grid in the plane). In contrast, we seek to address more complex systems, with no known unified solution framework, involving multiple robots, imperfect sensing, and communication. In doing so, we expose limitations of the framework and automated tool.

Regardless, the theme of this paper is still supported by those works: Automated formal method are promising, but currently of limited utility in robotics. For example, in [15] and [33], they are constrained to linear logic; and the discrete abstraction procedure hinges on a variety of embedded assumptions that may limit its application to more general problems in robotics. They also note that currently, model checking programs do not support design (augmentation was required). In [21], they comment on the computational complexity of verifying even modest robotic examples. These themes echo our observations that current frameworks are limited by expressiveness and tools are limited by computational complexity.

III. Modeling and Verification of Hybrid Systems

Before defining a linear hybrid automata, we begin with a more general description. A hybrid automation [1] is a formal model to describe reactive systems with discrete and continuous components. Formally, a hybrid automation H consists of the following components.

1) A finite directed multigraph (V, E). The vertices are called the control modes while the edges are called the control switches.
2) A finite set of real-valued variables X. A valuation ν is a function that assigns a real value $\nu(x)$ to each variable $x \in X$. The set of all valuations is denoted as Σ_X. A state q is a pair (v, ν) consisting of a mode v and a valuation ν. The set of all states is denoted as Σ. A region is a subset of Σ.
3) A function init, assigns to each mode v, a set $\text{init}(v) \subseteq \Sigma_X$ of valuations. This describes the initialization of the system: A state (v, ν) is initial if $\nu \in \text{init}(v)$. The region containing all initial states is denoted as σ^I.
4) A function flow, assigns to each mode v, a set $\text{flow}(v)$ of C^∞-functions from $R^+ \to \Sigma_X$ [i.e., a solution to a differential equation, $x(t)$]. This describes the way variables evolve in a mode.
5) A function inv, that assigns to each mode v, a set $\text{inv}(v) \subseteq \Sigma_X$ of valuations. The system can stay in mode v only as long as the state is within $\text{inv}(v)$, and a switch must be taken before the invariant gets violated.
6) A function jump, assigns to each switch e, a set $\text{jump}(e) \subseteq \Sigma_X \times \Sigma_X$. This describes the enabling condition for a switch, together with the discrete update of the variables as a result of the switch.
7) A function syn, assigns to each switch e, a label $\text{syn}(e)$ from a set of labels (names). When different components of a complex system are described individually by hybrid automata, the event labels on switches of different components are used for synchronization.

The hybrid automation H starts in an initial state. During its execution, its state can change in one of two ways. A discrete change causes the automation to change both its control mode and the values of its variables. Otherwise, a continuous activity causes only the values of variables to change according to the specified flows while maintaining the invariants.

The operational semantics of the hybrid automation are captured by defining transition relations over the state space Σ.

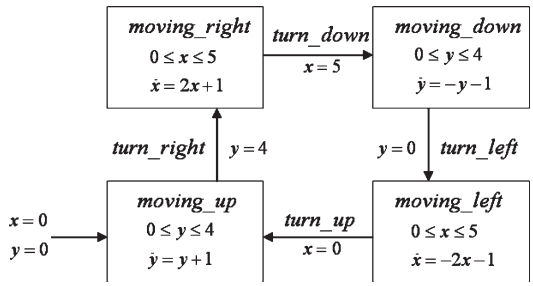

Fig. 1. Mobile robot automation which tracks the perimeter of a 4 × 5 room.

For a switch $e = (v, v')$, we write $(v, \nu) \to_e (v', \nu')$ if $(\nu, \nu') \in \text{jump}(e)$. For a mode v and a time increment $\delta \in R^+$, we write $(v, \nu) \to_\delta (v, \nu')$ if there exists a function $f \in \text{flow}(v)$ such that $f(0) = \nu$, $f(\delta) = \nu'$, and $f(\delta') \in \text{inv}(v)$ for all $0 \le \delta' \le \delta$. The transition relations $\to_e$ capture the discrete dynamics, while the transition relations $\to_\delta$ capture the continuous dynamics.

As an elementary example, consider Fig. 1 which shows a hybrid automation of a simple mobile robot which moves in four directions—up, down, right, and left. The robot is programmed to follow the wall. Correspondingly, the robot has four modes: moving_up, moving_down, moving_right, and moving_left. The mobile robot moves around a room whose depth is 4 units and whose width is 5 units. Initially, the robot is in the moving_up mode and located at (0,0). While the robot is in the moving_up mode, the robot moves up at the rate of $y + 1$ units per minute. When the robot reaches the end of the room (i.e., y becomes four), the robot turns right via the turn_right transition. Then, the robot changes its mode to the moving_right mode and moves right at the rate of $2x + 1$. Note that the transition condition asserts when a mode transition may occur. In order to force the mode transitions, we add invariants to modes: the system can remain in a mode only as long as the corresponding invariant condition is satisfied. Thus, the invariant condition $0 \le y \le 4$ of the moving_up mode prescribes that a mode transition must occur before the robot hits the wall of the room.

The central challenge in algorithmic formal verification of hybrid systems is to compute the set of reachable states of a given hybrid automation. In general, this is quite difficult, however, for a special class of automata, called linear hybrid automata, the analysis becomes tractable. A hybrid automation $H = (V, E, X, \text{init}, \text{flow}, \text{jump}, \text{syn})$ is called linear [1], [3].

1) For each mode v, the sets $\text{init}(v)$ and $\text{inv}(v)$ are described by Boolean combinations of linear inequalities over the variables X.
2) For each switch e, $\text{jump}(e)$ is described by a Boolean combination of linear inequalities over the variables $X \cup X'$, where the primed variables X' refer to the values of the variables in X after the switch.
3) For each mode v, allowed flows at a mode v are specified by a conjunction of linear inequalities over the set $\dot{X}$ of dotted variables representing the first derivatives of the corresponding variables in X. That is, a C^∞-function f belongs to $\text{flow}(v)$ iff the first derivative $\dot{f}$ of f with respect to time satisfies each linear inequality for all times $\delta \in R^+$.

Algorithms for symbolic reachability analysis of hybrid automata manipulate regions. Let σ be a region of H. The successor region of σ, denoted post(σ), contains states q' such that $q \rightarrow_e q'$ for some $q \in \sigma$ and some switch e, or $q \rightarrow_\delta q'$ for some $q \in \sigma$ and some $\delta \in R^+$. A state q is said to be reachable if $q \in \text{post}^i(\sigma^I)$ for some natural number i, where post^i denotes the post operator composed with itself i times. In other words, the set of all reachable states of a hybrid automation can be computed by repeatedly applying post to the initial region. The set of reachable states of a hybrid automation H is denoted as reach(H).

The above requirements for linear hybrid automata ensure that for each i, the set post$^i(\sigma^I)$ can be described by a Boolean combination of linear inequalities [1]. Furthermore, such sets can be computed effectively. Fig. 1 is not a linear hybrid automation because the differential equations contain x and y, and therefore, the resulting flows are not described by a set of Boolean inequalities.

The software HYTECH [3], [16][1] supports model checking of hybrid systems based on the above principles. The implementation is based on routines to manipulate convex polyhedra. The input of HYTECH consists of two parts: a system description section and an analysis section. The system description section has a textual representation of the linear hybrid automata. The user describes a system as the composition of a collection of components. The analysis section verifies the system against user-defined properties. Properties are checked by applying reachability tests to regions. For example, to verify a property that a robot never collides with obstacles, we define a region of collision states. Then, we show this region is not reachable from the initial region.

The input to HYTECH can include design parameters-symbolic constants with unknown, but fixed values. Such parameters are treated just like any other system variables. Given a correctness requirement, HYTECH uses the symbolic computation to determine necessary and sufficient constraints on the parameters under which violations of the requirement cannot occur. This feature of parametric analysis is central to our application as discussed later in Section VI.

IV. MULTIROBOT COORDINATION

The multirobot scenario we consider is motivated by a military search and rescue application. We consider a scenario with two robots, two static convex obstacles, and a goal target position for both robots as shown in Fig. 2. This scene could represent the floor plan of a typical indoor mission. Each of the obstacles can represent furniture for example. The target might be a door that the robots must reach and travel through. Each robot is autonomous, in the sense that each robot does its own sensing, planning, and control—there is no designated "leader." We assume the environment is two dimensional, and each mobile robot has the ability to determine its own position and orientation. This ability may come from a GPS sensor or from using a camera to determine landmarks in the environment. Each robot follows a continuous control law, which attempts to guide it to the goal based on its knowledge about its own location and the environment. There is only a single control mode.

[1]http://www-cad.eecs.berkeley.edu/~tah HYTECH.

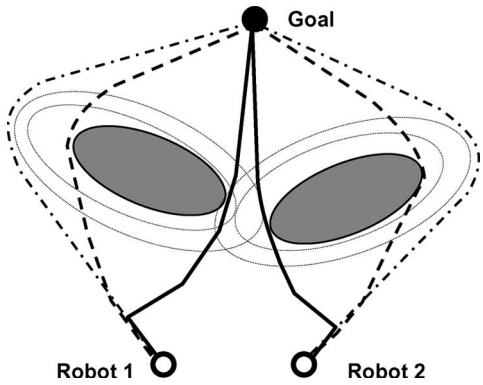

Fig. 2. Simple scenario that illustrates how cooperation between two robots can improve the performance of the team in locating and reaching a target in a partially known environment. Solid ovals are obstacles; dotted ovals are robot's successive perceptions. The dot-dash path is pure open loop based on initial perception. The dashed path is based on robots updating their information using sensors periodically. The solid path is generated using both sensors and communication between the robots.

Each robot is equipped with a camera that allows it to identify other robots, obstacles, and the target. The camera has errors in estimating the position of objects (obstacles, targets, and other robots) that decrease as the robot approaches the object. Referring to Fig. 2, the dark ovals indicate the actual obstacle location while the larger dotted ovals indicate the observed obstacle geometry, as seen by the robots at the starting configuration. Note that in the initial model, the two obstacles appear to overlap.

If an open-loop control was used for each robot, solely based on the initial estimate of the obstacles, without any communication or further sensing, the robot would follow the dot-dashed paths called the open loop. However, when each robot gets sensory information from its camera and refines its world model, we get discrete changes in the path as shown by the dashed lines. This is called sensor-based or closed-loop control. Now, the robot controllers are hybrid controllers. The performance, judged by the length of the path, has improved but not significantly. There is still no interaction between the robots.

In addition, our robots are able to communicate over a wireless local area network. However, because of the bandwidth limitations and the possible clandestine nature of the mission, the communication either may not be possible or may be limited to sporadic broadcast of a small volume of data. In this scenario, the two robots exchange information about their world models at discrete intervals. The corresponding paths followed by the robots are labeled solid black lines and are referred to as "sensor based with communication." Because the robots pool their information, the path followed is more efficient—they are able to take advantage of the narrow opening between the two obstacles while avoiding collisions.

Robots are in many ways true hybrid systems. In this scenario, each robot is driven by actuators that are intrinsically continuous. The dynamics are derived from laws of physics and are represented by continuous mathematics. Therefore, the robot motion is continuous. However, this behavior changes, possibly discontinuously, as new information becomes available

through sensing or communication. Furthermore, many of the aspects of robot operation are inherently algorithmic, such as path planning, sensing, and localization, and therefore evolve in a discrete time fashion.

V. Modeling

All formal methods require the system to be expressed in some standard high-level formalism. HyTech in particular requires the system to be described as a linear hybrid automata [1]—a finite automation augmented with a finite number of real-valued variables that change continuously, as specified by constant differential equations/inclusions and linear algebraic inequalities. A primary challenge in applying a tool such as HyTech, or any formal method, is to model a complex nonlinear and stochastic system such as a mobile robot in this rigid modeling framework. It is worth noting that in previous case studies in formal verification of hybrid systems, the challenge in modeling was approximating complex dynamics by rectangular inclusions. For us, the continuous dynamics can be reasonably simplified, but a significant approximation is required to make guard conditions and update rules linear. For instance, we model obstacles and their estimates as rectangles, approximate Euclidean distance by Manhattan distance, and require the robot to move only along horizontal or vertical directions.

In this section, we discuss various aspects of the modeling process. It is important to emphasize that the modeling effort was iterative. Frequently, a working model of the system would be developed, only to realize that for one reason or another, it was too complex. The final model is presented below. In particular, we examine the various simplifying assumptions that were made. They are categorized as follows:

1) simplifying assumptions which are typical in the robotics literature;
2) further simplifications which were dictated by the linear hybrid automata modeling framework, required by HyTech;
3) alterations to aspects of the model that, although permitted in the modeling framework, proved to be too complex in practice to be verified with limited computational resources.

A. Robots

First, as consistent with the scenario outlined earlier in Section IV, we restrict our attention to mobile robots operating in planar environments. The robots are referred to as R_1, R_2, and R_3. Due to computational costs, we were limited to a scenario with three robots.

Robots are modeled as points in $\Re^2$, and therefore have no size. We also ignore the orientation of the robot. Note that if the robot is symmetrical (e.g., cylindrically shaped as many mobile robots are), the point robot assumption is easily dealt with by simply expanding the size of the obstacles by an amount equal to the radius of the robot. This technique is quite standard in the robotics literature; the resulting system resides in what is called the "configuration space."

Most mobile robots travel on wheels or tanklike treads. Many such systems possess nonholonomic (differential) constraints that may limit the robots' direction of motion (e.g., no sideslip on wheeled vehicles). Such systems have notoriously nonlinear continuous dynamics, involving sine or cosine functions. Since such systems are not permitted in the linear hybrid automata framework, we assume that there are no such differential motion constraints in effect (holonomic robots).

Furthermore, we assume the continuous dynamics are first order, (i.e., kinematic) as opposed to the full Newtonian second-order dynamics. This limits the dimension of the continuous state space to two ($x-y$ positions) per robot as opposed to a four-dimensional state space (two positions and two velocities) per robot. This assumption makes the computations significantly simpler. While dictated by the modeling paradigm and computational considerations, assuming a system is kinematic and holonomic is a reasonable assumption commonly seen in the robotics literature. Therefore, we will model the dynamics by a set of first-order differential equations: $\dot{x}=u_x$ and $\dot{y}=u_y$, where (x, y) are the coordinates of a robot and (u_x, u_y) are the control inputs, in this case, speeds in the x and y directions, respectively.

B. Control

Note that the dynamics contain the undetermined input functions u_x and u_y. These functions must be assigned so that the robot tracks the desired path. In order to be a linear hybrid automata, the right-hand side of the differential equations must be a constant within each mode. To that end, the control law was designed to have four modes:

$$\begin{aligned} &\text{right}: && \dot{x} = v_{\max} && \dot{y} = 0 \\ &\text{left}: && \dot{x} = -v_{\max} && \dot{y} = 0 \\ &\text{forward}: && \dot{x} = 0 && \dot{y} = v_{\max} \\ &\text{back}: && \dot{x} = 0 && \dot{y} = -v_{\max} \end{aligned} \tag{1}$$

where $v_{\max}$ is the robot's maximum speed.

While control laws in robotics do often have several modes, this particular selection is not very realistic. Control laws are often quite complex and frequently are functions of the continuous state. These four modes represent the minimum number of control modes required for system to reach any point in the plane. However, arbitrary point-to-point straight-line paths are not possible and must be approximated by "stair case"-like motions consisting of an alternating series of left/right and forward/back steps. Note that the linear hybrid automata model does not prohibit the definition of successively finer directional discretizations (such as adding four diagonal modes); however, the addition of more modes increases the computational complexity of the verification problem. Note that since we are primarily concerned with optimal motions (least time or shortest path), the restriction of the speed to $v_{\max}$ does not create any limitations.

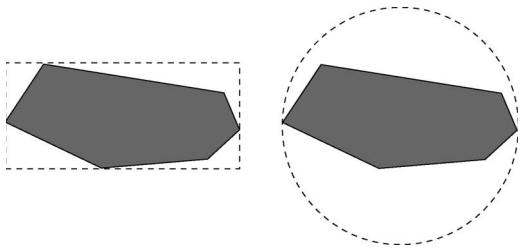

Fig. 3. Rectangle provides a reasonable approximation to most convex polygons, as compared to a circle.

C. Obstacles and Workspace

We assume that the work space is a bounded subset of $\Re^2$, called $\mathcal{W}$ and that the environment is populated with multiple polygonal obstacles O_j, for $j = 1, \ldots, M$ which occupy closed sets in $\Re^2$. These obstacles are assumed to be in fixed positions. The collision free space through which the robot is permitted to move is $\mathcal{F} = \mathcal{W} - \cup_j O_j$. These assumptions again are very typical in the robotics literature in general and are certainly representative of our search and rescue application.

To reduce the required computation, the obstacles are assumed to be rectangles which are aligned with the coordinate axes rather than arbitrary polygons. Each rectangle can be completely described using only four parameters. In addition, certain geometric operations that we are concerned with, such as shrinking, growing, and intersection, can be performed on rectangles using strictly linear functions. The importance of these operations will be described later.

This assumption is not common in robotics; however, as shown in Fig. 3, most general polygons can be reasonably approximated by a rectangle. Note that nonconvex obstacles can be approximated using multiple overlapping rectangles.

D. Sensor Model

We assume that all sensing occurs at discrete intervals. Therefore, the robot only gets new information about its own position, and the world around them every δT seconds. Real sensors possess such sampling rates—for example, most cameras only update 30 times/s. Since the robot will not change the control modes without getting new information, this assumption has the effect of forcing the robots to essentially travel on a grid with spacing $\delta T \cdot v_{\max}$. Note that this is not explicit in the model, rather it follows from the selection of the control modes and sensor update rate. Also note that the vertices of the actual obstacles or estimates are not required to lie on a "grid point."

It is assumed that the information about a robot's own position is accurate. However, a robot's knowledge of the geometry of the obstacles is prone to error. Each robot is assumed to have some prior qualitatively correct knowledge of the workspace (e.g., provided by satellite imagery or a human user). The information is qualitatively correct, in that, it accurately reflects the number of obstacles in the environment and their general shape; however, their exact position, size, or geometry is uncertain. In other words, we assume that it is possible to parameterize the uncertainties, and the unknown information is limited to the value of certain parameters. Further, we assume that the robot sensor allows the estimation of these unknown parameters, and the estimates improve as the distance between the robot and the obstacle decreases.

Let Y_j^i be a map from the robot's position (x, y) to a closed set in the plane which represents the ith robot's estimate of the jth obstacle. In order to remove the type of stochastic uncertainty exhibited by the sensors, we make the assumption that the robot possesses an estimation algorithm which returns the worst case estimate of the obstacle's geometry, although we do not model the algorithms' operation. In other words, the uncertainty in a given estimate is bounded in such a way that $Y_j^i(x, y) \supseteq O_j, \forall x, y$. Although it is not known where O_j lies in $Y_j^i(x, y)$, it is certain that $O_j \cap (\neg Y_j^i(x, y)) = \emptyset$. As a consequence of the bounded uncertainty assumption, robots can always determine if a new estimate is better than a previous one by comparing the area of the two, the estimate enclosing the smaller area being superior.

The sensor also has the property that its estimation of the obstacles improves as the distance from the robot to the obstacles decreases. In the limit, as the robot touches the obstacle, $Y_j^i \longrightarrow O_j$.

Such uncertainty models, while idealized, are reasonable approximations of sensor systems where errors are primarily geometric in origin. Furthermore, many statistical algorithms are able to estimate worst case noise. For example, in vision applications in a two-dimensional world without occlusions and problems due to segmentation, the accuracy is limited by a charge-coupled device resolution, especially at long ranges, and the estimates improve as the distance to target decreases. The bounded uncertainty can be computed by finding the worst case error.

Allowing all four parameters of the rectangular obstacle to vary proved to be too computationally expensive, so the x coordinates of the right and left sides of the obstacle were taken to be the only information subject to uncertainty. This model was abstracted in HYTECH as

$$x_{\mathrm{l}}^{\mathrm{o}}(t) = x_{\mathrm{l}}^{\mathrm{a}} + \underbrace{(x_{\mathrm{l}}^{\mathrm{o}}(0) - x_{\mathrm{l}}^{\mathrm{a}})\,\frac{d}{d_0}}_{\text{error}} \tag{2}$$

$$x_{\mathrm{r}}^{\mathrm{o}}(t) = x_{\mathrm{r}}^{\mathrm{a}} + \underbrace{(x_{\mathrm{r}}^{\mathrm{o}}(0) - x_{\mathrm{r}}^{\mathrm{a}})\,\frac{d}{d_0}}_{\text{error}}. \tag{3}$$

Here, x_{l} and x_{r} denote the x coordinates of the left and right sides of the rectangle, superscripts o and a indicate observed and actual quantities, respectively. The distance at which the measurement is taken is d, and d_0 refers to the maximum possible distance from the robot to the obstacle within $\mathcal{W}$, resulting in the worst case estimate.

While the model does not capture the nonlinear behavior of most sensors, similar phenomenon occurs when using sonar sensors. In Fig. 4, a mobile robot (black circle) is shown with a sonar array with a rectangular obstacle (shown in gray). Here, we have a situation where the robot has reasonably accurate information about some of the obstacle's parameters, while information about the other parameters is subject to a considerable amount of uncertainty. The sonar readings only indicate

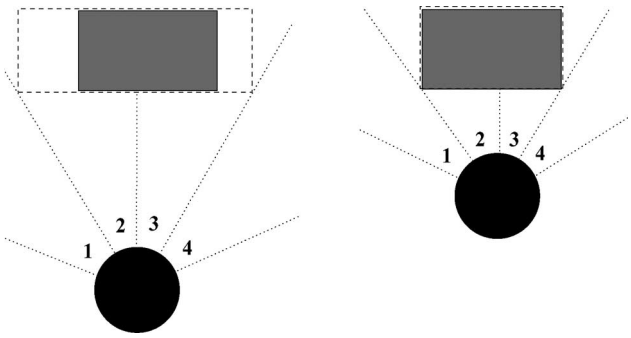

Fig. 4. Overhead view of a mobile robot equipped with sonar arrays detecting a rectangular obstacle. The sonars return the closest distance to an object which lies somewhere within the ensonification cone. At greater distances (left figure), the uncertainty can be rather large since the robot only knows that something lies within cones 2 and 3, while cones 1 and 4 are free. As the robot approaches the obstacle (right), however, its estimates get better.

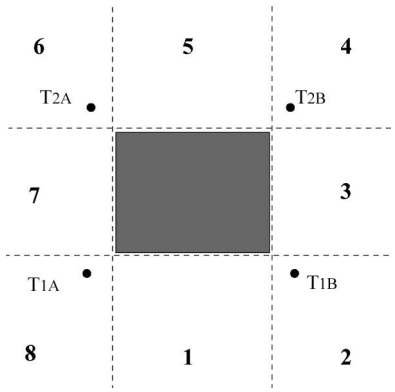

Fig. 5. Illustration of the exact cell decomposition planning method. The dark rectangle represents the obstacle while the numbered regions are free cells in the workspace.

that there is an object within ensonification cones 2 and 3 at a certain distance, while cones 1 and 4 are empty. The robot's worst case approximation is shown as a dashed rectangle. As the robot approaches the object and the distance between them decreases, the uncertainty in the measurement also decreases.

A more serious limitation is due to the lack of an acceptable distance function. Since robots must negotiate spatial environments, a critical quantity to be computed is the distance between the robot and an obstacle or its goal. Unfortunately, the classical Euclidian distance function is highly nonlinear $d = \sqrt{x^2 + y^2}$. Instead, we use the so-called Manhattan metric or the L_1 norm to measure the distance between two points. The Manhattan distance d_{m} from Point A to Point B is simply

$$d_{\mathrm{m}}(A, B) = |x_{\mathrm{a}} - x_{\mathrm{b}}| + |y_{\mathrm{a}} - y_{\mathrm{b}}| \tag{4}$$

which can be divided into four separate linear functions based on the signs of the two differences. Thus, even the distance computation is "hybrid," this addition of four "submodes" significantly complicates the model. An even more serious limitation of this assumption is that a sensor reading taken at a point whose true distance to the obstacle is small may be no different from a reading taken further away if the distances are deemed equal in the Manhattan sense. Thus, it is possible that the robot's estimate will not strictly improve as the robot approaches an obstacle along certain paths. This is viewed as the most serious and detrimental limitation imposed by the linear hybrid automata model.

E. Path Planning

The term path planning is used here in reference to a mapping from the currently available information to a collision-free kinematic trajectory. The planning algorithm used here is essentially an exact cell decomposition approach. A complete explanation of the algorithm can be found in [23]. For this scenario, the workspace decomposition used is shown in Fig. 5.

For the considerably simplified scenario of a point robot navigating amid rectangular obstacles, only two separate cases need to be considered. First, suppose the robot is currently in cell 1 (the cases for cells 3, 5, or 7 follow by symmetry). When the goal is in any adjacent cell (8, 1, or 2), no special planning is needed since adjacency guarantees that a collision-free path exists. If the goal lies in cells 7 or 6 (or 3 or 4), a temporary goal $T_{1\mathrm{A}}$ (or $T_{1\mathrm{B}}$) is set. From that point, a collision-free path to the target exists. However, if the goal resides in region 5, the robot first proceeds to $T_{1\mathrm{A}}$ (or $T_{1\mathrm{B}}$), then it sets a new temporary goal $T_{2\mathrm{A}}$ ($T_{2\mathrm{B}}$) based on which intermediate point will result in the shortest overall path. Once it reaches $T_{2\mathrm{A}}$ (or $T_{2\mathrm{B}}$), it can proceed to region 5 unobstructed.

The second case occurs when the robot is initially in a corner cell such as 8 (2, 4, or 6). In this case, collision-free paths exist when the goal lies in cells 6, 7, 1, or 2. Paths to regions 5 or 3 are determined, similar to the previous case, by setting temporary goals in cells 6 or 2, respectively. The degenerate case occurs when the goal lies in the corner cell opposite to the robot's starting position, cell 4 in this case. Due to the lack of Euclidean metric and the fact that the robot may only move in four directions, the clockwise and anticlockwise paths around the obstacle will always be of the same Manhattan distance. In this case, the robot chooses the path nondeterministically. This cell decomposition algorithm is optimal because it compares various choices of paths based on the length and selects the shortest one.[2]

Other than the ambiguity due to the Manhattan distance in the degenerate case mentioned above, this is essentially the same algorithm that might be used in a real robotic system. In addition, it is important to point out that the planning algorithm would be significantly more complex without the previous assumption that the obstacles are rectangular. Another ramification of our assumptions is that the robot can only reach goal points or temporary goal points that lie on the "grid" imposed by the sensor update rate. Therefore, care must be taken to select temporary goal points at the nearest grid point outside of the obstacle estimate.

[2]This cell decomposition algorithm is optimal in the Manhattan metric, not in the Euclidean metric.

F. Coordination and Communication

At discrete time intervals, robot R_i may send its current map of the environment to robot R_k.[3] Robot R_k must then fuse that information with its own representation of the obstacles. Again, as a consequence of the bounded uncertainty assumption, this fusion is accomplished by, for all obstacles j:

$$Y_j^k\text{new} = Y_j^i \cap Y_j^k. \tag{5}$$

R_k's resulting estimate of obstacle j, Y_j^knew, will naturally have an area less than or equal to R_k's previous estimate making it at least as accurate. This new estimate is also guaranteed to completely contain the obstacle. While this is certainly idealized, it follows from our sensor model. For the sake of simplicity in robot modeling and verification, we assume that there is no stochastic process involved in message transmission, such as noise or packet drop out. For similar reasons, we also assume that robots have unlimited communication ranges.

G. Cost Model

As mentioned in the previous section, the robots attempt to choose behaviors which minimize some type of cost function. In this case, the cost is the total time required to reach the goal. It is also assumed that communication is a potentially expensive operation, either due to the computational cost of processing the information, bandwidth limitations, or for security reasons. To reflect this, a time penalty ρ_{comm} is added to the overall cost function each time a message is sent over the network.

In this model, the cost function is the sum of the time taken to travel a path and the time taken to communicate. If f is the frequency of communication, the overall performance index J^i which indicates a total time for R^i to reach the goal can be expressed as

$$J^i = \frac{D_{\text{M}}^i}{v_{\max}} + \rho_{\text{comm}} \cdot f \cdot \frac{D_{\text{M}}^i}{v_{\max}} \tag{6}$$

where D_{M}^i is a total distance traveled by R^i in the sense of the Manhattan metric and $v_{\max}$ is the speed of R^i.

Again, the most serious limitation placed upon this cost estimate by the linearity requirement comes from the use of the Manhattan distance. Given a Manhattan distance D_{M}, upper and lower bounds, D_{u} and D_{l}, can be placed on the corresponding Euclidean distance. As shown in Fig. 6, these bounds can be expressed as

$$D_{\text{l}} = \frac{\sqrt{2}}{2} D_{\text{M}} \leq D_{\text{E}} \leq D_{\text{M}} = D_{\text{u}} \tag{7}$$

where D_{E} is the Euclidean distance. Note that $\sqrt{2}/2 \approx 0.707$, which implies that the Manhattan distance, at most, overestimates the actual distance by approximately 41%. This can be a large discrepancy which can result in rather severe overestimates of the true cost.

[3]In our experiments, R_i sends only estimates of obstacles to R_k [two values per obstacle—$x_{\text{l}}^{\text{o}}(t)$ and $x_{\text{r}}^{\text{o}}(t)$ defined in (2) and (3)].

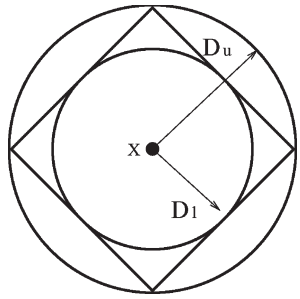

Fig. 6. Diamond-shaped line represents the set of points equidistant from X, in the Manhattan metric. The circles indicate the upper and lower bounds on the actual distance measured in the Euclidean sense.

It is also worth noting that the shortest path between two points is not unique when using the Manhattan distance even in the absence of obstacles. Recall that due to the fact that sensor and communication information are only updated at discrete intervals and that the robot's speed is constant, it turns out that the robot essentially travels on an equispaced grid. In this case, when traveling from grid point $(x_{\text{a}}, y_{\text{a}})$ to grid point $(x_{\text{b}}, y_{\text{b}})$, there are $((N+M)!)/(N! \cdot M!)$ distinct shortest paths, provided there are no obstacles in the region $[x_{\text{a}}, x_{\text{b}}] \times [y_{\text{a}}, y_{\text{b}}]$. Here, N and M are positive integers indicating the number of grid points, or steps, between point A and point B in the X and Y directions, respectively.

VI. Results

Recall that our goal is to answer questions about the role of communication in aiding the robots to reach their goal with a lower cost. In this section, we detail the specific scenario we explored, we then report the results of our experiences using HyTech. We use HyTech first as a design tool by performing a symbolic parametric analysis, letting the goal position and the communication frequency be symbolic unknowns. We then verify some safety properties of the control strategy.

A. Example

A high-level description of the robot's behavioral algorithm as a finite state machine appears in Fig. 7. The behavior can be sketched as follows.

```
while (reachedGoal == False) {
   1. Use sensors to update the map of the world
   2. Send or Process communication if appropriate
   3. Plan a path
   4. Travel for time period
}
```

The robot model description is around 1700 lines.[4] We verified this description using Sun Enterprize 3000 (4 × 250 Mhz UltraSPARC) with 1-GB physical memory.

Our scenario contains three identical robots (R_1, R_2, and R_3), one fixed obstacle and one fixed goal (see Fig. 8).

[4]Corresponding HyTech source codes and analysis results can be downloaded at http://www.postech.ac.kr/~moonzoo/robot.zip.

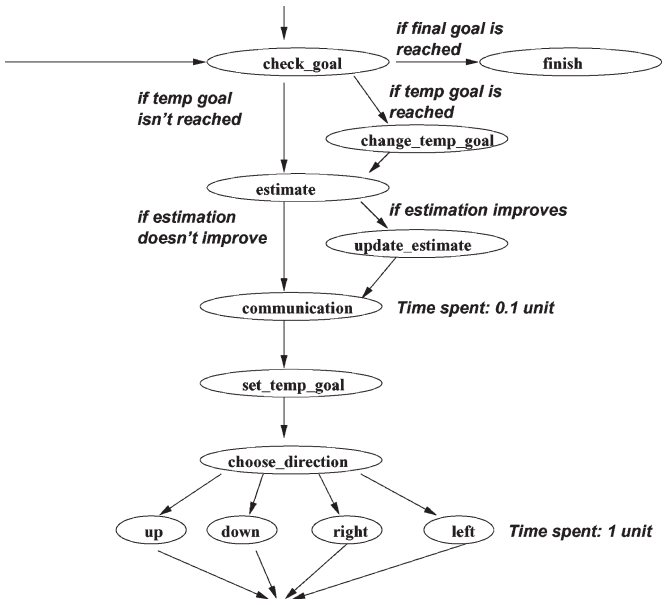

Fig. 7. Robot's behavioral algorithm as a finite state machine.

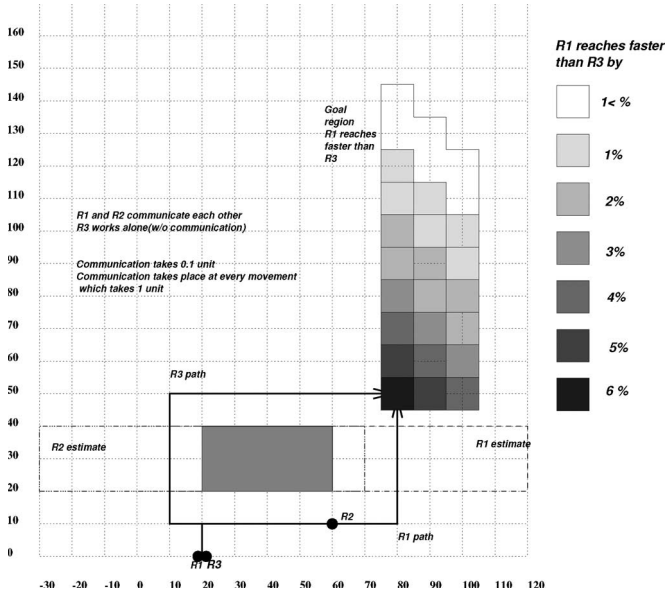

Fig. 8. Scenario analyzed with HYTECH.

R_1 and R_2 collaborate via communication, while R_3 works by itself. The initial positions of R_1 and R_3 are the same. Initially, R_1 and R_3 are located at (20,0). R_2 is located at (60,10). The obstacle is located somewhere within the region whose corner points are (20,20) and (60,40). Let us call the x position of left end of the obstacle x_1, and the x position of right end of the obstacle x_2. Similarly, y_1 is the y position of bottom end of the obstacle and y_2 is the y position of top end of the obstacle. R_1 and R_3 estimate x_1 as 10 and x_2 as 120 initially. R_2 estimates x_1 as -30 and x_2 as 70 initially. In other words, R_1 and R_3 estimate the obstacle to be far larger toward the right-hand side, but R_2 estimates the obstacle far larger toward the left-hand side. All robots estimate y_1 as 20 and y_2 as 40 initially (i.e., all robots have correct values for y_1 and y_2). The direction of movement is determined at the end of each iteration. A robot moves for one time unit once a direction is determined (see Fig. 7). Each robot's $v_{\max} = 10$. Communication has a cost of 0.1 time unit. Thus, unnecessarily frequent communication may increase the time to reach the goal. For verification purposes, the work space was restricted to a bounded rectangle with dimensions of 150 by 160 units. Since optimal motions are of primary concern, all paths can be expected to lie within the bounded region.

B. Parametric Analysis

Our first experiment attempted to determine if indeed communication helped R_1 reach the goal faster, with the help of R_2, than R_3; and if so, what goal positions could be reached with a lower cost if R_1 and R_2 communicate every time unit. To that end, setting x and y positions of the goal as parameters, we computed the geometric region which R_1 reaches faster than R_3 with the help of communication (see Fig. 8).

For example, referring to Fig. 8, the solid black robot paths help to illustrate one scenario when the target is located at (80,50). Initially, R_3 sets up a temporary goal as (0,10), because the estimated length of left path toward the goal is shorter than the length of right path. However, R_1 gets a good estimation of x_2 by communicating with R_2. It sets a temporary goal at (80,10), then chooses the right path. R_3 takes 13 time units to reach the goal (80,50), whereas R_1 takes 12.1 time units including communication overhead; it is verified that the collaboration between R_1 and R_2 helps R_1 to reach the goal faster than R_3 in this scenario.

Since the x and y positions of the goal are free symbolic parameters, we can answer such questions for any goal position within the free workspace. It turns out that for most regions in the workspace, no savings occur. The shaded region in the upper right part of Fig. 8 shows the set of goal regions for which communication is helpful. Not surprisingly, this is the region in the workspace for which it is necessary to circumnavigate the obstacle so the additional information R_2 supplies is quite helpful. Furthermore, we classify the region by how much R_1 reaches faster than R_3 as seen by the legend on the right of Fig. 8. Note that the region has a stairlike shape because, R_1 constantly communicates with R_2 and this communication overhead accumulates so that this overhead cancels out the saving after 20 movements.

Our second symbolic parametric analysis experiment was to determine the optimal frequency of communication for reaching the goal with minimal cost. In this experiment, the goal was fixed at (80,50) and we set the period of communication as a parameter. Note that the domain of period is finite because the period should be positive and be less than time for robots to reach the goal. The optimal scenario is when R_1 and R_2 communicate once in two time units, R_1 takes 11.5 time units to reach the goal.

In addition to the parametric studies, two safety properties for the robot controller are verified. First, a robot never collides with the obstacle while it navigates to reach any valid goal position, which is any position outside of the initially estimated obstacle. This was accomplished by adding a monitor automation to the description so that the monitor can check

whether a position of a robot overlapped with the estimate of the obstacle. Second, the verification establishes that a robot does reach any valid goal position in the work space. Together, these two criteria indicate that the control strategy is a valid one to complete the mission. These two verification results may seem obvious at first glance, because the path-planning algorithm is known to be correct. However, proving the correctness of the robot controller is useful for determining if the implementation of the algorithm is correct. Formal verification technique is very useful for detecting and debugging errors which are difficult to find manually.

VII. Limitations of the Analysis

A. Computational Limitations

The impact of the computational restrictions was that the complexity of the scenarios we could verify was significantly reduced. We had to make several simplifications in order to make the analysis tractable. Computational restrictions manifested themselves in two varieties: "memory overflow" errors and "library overflow" errors.

Memory overflow errors restricted the number of modes the model could posses. This for example limited the number of robots in our scenarios to three. It also limited the number of continuous variables. For example, we had to model only one obstacle in the scenario because when we modeled two obstacles, HyTech generated a memory overflow error. Also, only two parameters of the obstacle's geometry were estimated by the sensor in our model due to similar limitations. We limited the range of x between -30 and 120 and the range of y between 0 and 160. Furthermore, to make the analysis tractable, we divided this region into 21 partitions such as $P_1 = \{(x,y)| -30 \leq x < 0 \text{ and } 0 \leq y < 30\}$, $P_2 = \{(x,y)|0 \leq x < 30 \text{ and } 0 \leq y < 30\}$, and so on. Then, we set P_i as possible range of goal position and performed the parametric analysis; we performed 21 experiments on all 21 partitions $P_1, P_2, \ldots, P_{21}$. Most significantly, though, it constrained us to only examining the effect of two free parameters simultaneously. We were unable to examine the effect of communication frequency while allowing the goal position to vary. Hence, we performed the two experiments in sequence while fixing some of the parameters. This severely limited the generality of the conclusions we were able to draw from the model. Despite this myriad of simplifications, verifying each partition took up to 1.3-GB memory space and 1 h.

Another computational limitation is the internal arithmetic overflow error that can occur when HyTech manipulates reachable regions. A reachable region is defined by a set of linear constraints—effectively a polyhedra. At each iteration, these polyhedral regions are manipulated by growing, intersecting, and joining operations. Eventually, the number of vertices of these high-dimensional polyhedra can grow too large for HyTech's symbolic manipulation library to handle, generating a "library overflow error". Therefore, we have to be careful to make the linear equations as simple as possible. This for example motivated the modeling of the obstacles as rectangles. Note that a "library overflow" error can occur with plenty of free memory available.

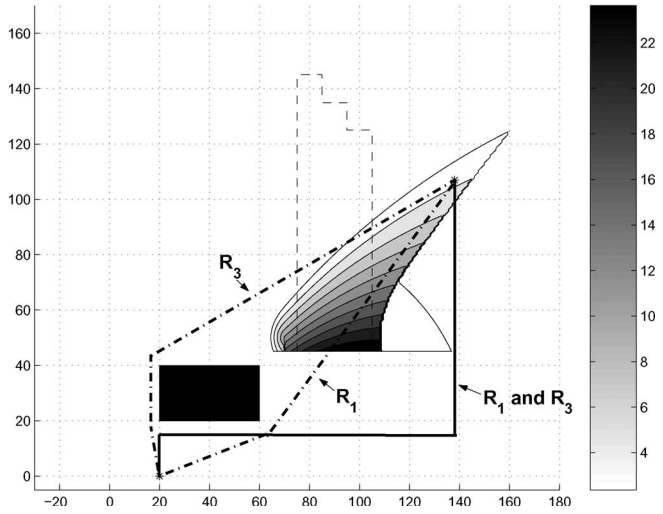

Fig. 9. Numerical simulation was used to compute the set of goal points which R_1 is able to reach with a lower cost than R_3. The shading scheme shows how much faster R_1 reaches the point as a percentage using the formula $(J_3 - J_1)/J_3$ where J_1 and J_3 are the cost functions of R_1 and R_3. The solid lines are an example of a path taken by R_1 and R_3 as computed by the approximate model (HyTech), and the dashed-dotted lines are the paths selected by the exact model.

B. Modeling Limitations

Throughout the experiment, various approximations were made to comply with HyTech's strict linearity requirements, in addition to the model simplifications introduced to reduce the computation time. Since no engineer would actually implement a robot system with these restrictions, the ultimate success or failure of the HyTech experiment hinges on determining exactly what the results of the verification for the approximate system imply about the performance of the original system. Due to the complex nature of the system, it is impossible to analytically determine the affect of approximation, hence numerical simulation was used to determine the performance of the original system. While simulation is never exact and therefore cannot be used to verify a model, it nevertheless remains the primary tool of system designers. Fig. 9 shows the simulation results. The figure was generated using the exact same scenario (starting conditions, obstacle and estimate geometries, cost function, etc.) as was used in the HyTech simulation. In addition, robots are simulated using similar behavioral algorithms as used in the verification procedure; the primary difference being that many of the approximations of the HyTech experiment have been removed, including: 1) Robots may move in arbitrary directions, instead of being constrained to left, right, forward, and back; 2) the Euclidean distance function, rather than the Manhattan distance, is used to compute the shortest path to the goal and for updating the sensor estimates; and 3) sensing occurs continuously. This results in a more realistic situation. Note that the communication protocol was not changed in any way. The simulation results were created by sampling 40 000 equispaced goal positions. For each goal position, the behavior of the robots was simulated and the difference in the cost functions for R_1 and R_3 was computed.

It is apparent from Fig. 9 that the region HyTech computed is neither a proper overapproximation nor underapproximation

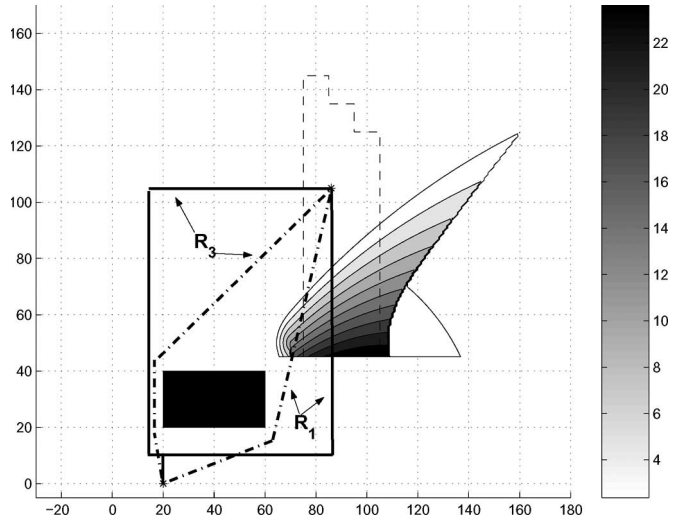

Fig. 10. Example of the approximate model falsely identifying a goal point where R_1 reaches the goal first. The solid lines are the path taken by R_1 and R_3 as computed by the approximate model (HYTECH), and the dashed-dotted lines are the paths selected by the exact model (numerical simulation).

of the true region where communication improves the team performance. The shading scheme indicates how much faster (expressed as a percentage) the team was able to reach a given goal configuration. In certain regions, the difference between the HYTECH and simulation results can be attributed to particular approximations. For example in the simulated region, the protrusion on the lower right part of the region, resembling a quarter circle, is the set of points where both R_1 and R_3 proceed around the right side of the obstacle; however, R_1 is able to reach the point faster since its additional information enables it to take a "straighter" path to the goal. HYTECH was unable to capture this behavior since the robots are not permitted to move in arbitrary (i.e., diagonal) directions. In this area, the difference in path costs for R_1 and R_3 was small (less than 4%). Also note that the remainder of the region generated by simulation, which is swept to the right and tapered to a point at the upper right extreme, looks quite different from the region computed by HYTECH. In this part of the figure, R_1 and R_3 take qualitatively different routes to the goal. R_3 travels around the left side of the obstacle while R_1 is able to recognize a shorter path on the right. The selection between the left/right handed path is based on length considerations and is naturally heavily dependent on the choice of metric. Hence, the main source of discrepancy between the simulation and HYTECH result in this region is the use of the Euclidean versus Manhattan distance function.

Figs. 9 and 10 illustrate the differences between the robot behavior when using the original model as compared with the model used in the verification for a few goal points. They help one get a feel for why the regions are shaped as they are. In Fig. 9, one can see a case where HYTECH failed to identify a point where communication helped. The solid lines (overlapping) indicate the paths the HYTECH model chose, where both R_1 and R_3 compute identical paths despite the additional information available to R_1. The dashed-dot line shows the paths result from simulating the original model. R_3 selects the path around the left side of the obstacle rather than the path around the right side. Here, R_3's shortest path is 203.5 units while R_1's path length is 181.9 units. Despite the communication penalty of 18.19 units, R_1 is still able to reach the goal faster. Fig. 10 illustrates a goal position which the HYTECH model falsely identifies as a position R_1 reaches first as a result of communication. The solid lines show the behavior of the robots in the approximated model. R_1's path is 175 units plus a communication penalty of 17.5 units resulting in a cost of 192.5. R_3 selects the path around the left side of the obstacle resulting a cost of 195 units. HYTECH indicates the goal point as one which R_1 is able to reach sooner. However, the dash dot shows that in the full model, R_1 and R_3 select paths of nearly equal length (130.8 and 133.2, respectively), once the communication cost is taken into account, R_1 is no longer able to reach the point sooner than R_3.

VIII. CONCLUSION

We have reported a case study in applying formal modeling and analysis aimed at exploring alternatives in the design of multirobot communication and coordination strategies. Simultaneous design of control strategies and coordination protocols for interacting dynamical systems is a significant challenge. The gist of our approach is to describe the system as interacting hybrid automata, and then employ a symbolic analysis to compute the constraints among various parameters for a given objective.

While earlier case studies focused on verifying safety properties, we use parametric analysis to explore design alternatives to enhance the performance (in addition to verifying safety properties).

In order to apply HYTECH to this problem, we had to make many simplifying assumptions. These simplifications fall into two categories: those that reduce computation time; and those that are required by the linear hybrid automata framework. In order to keep the computation time reasonable, we were forced to consider a very simple scenario with only one obstacle, limited uncertainty and three robots. While the results were interesting, most designers might be able to gather intuitively what might happen in such basic scenarios. The true benefit in applying formal methods would be in problems which are too complex for human judgment. The assumptions made to adhere to the linearity requirements made it difficult to extrapolate the results of the verification on the simplified model to the original problem. In comparing the results from HYTECH to those from repeated simulation, it was noted that they neither strictly over- nor underapproximated each other.

Even though we have reported only modest success in the goals of the exercise, we hope that it illustrates the possible potential of the approach. We were able both to vet our implementation of the algorithm and determine the optimal values of certain design parameters. Note the generality of this symbolic method compared to prevalent methods in simulation in which either the parameters need to be set to specific values and little can be said about off-sample points.

However, we feel that the most instructive aspect of this paper is to suggest the guidelines and focus areas for work

on the next generation of formal modeling and verification tools. In light of our experiences detailed in Section V, it should come as no surprise that significant advances in the formal verification technology are needed for it to be applicable to our problem in its full generality. Two specific obstacles were found. Computational requirements: All the parameters had to be scaled down to be able to get a feedback from HYTECH. Improving the efficiency of polyhedra-based analysis remains a significant challenge. Expressiveness: The linearity requirement forces us to apply a variety of approximations. Interestingly, the issue of approximating complex dynamics proved to be less of an issue than approximating transition rules. Robotics is an inherently geometric field and the lack of sine, cosine and, most importantly, Euclidian metric functions seem to pose the most significant obstacles. This problem suggests directions for further research and tool development for more general classes of problems.

In recent years, there has been significant progress in enhancing the scope of verification tools for hybrid systems. In particular, Hypertech [17] employs interval computations to improve robustness of computations with polyhedra, Checkmate [10], [11] allows specification of more complex dynamics and overapproximates the set of reachable states using polyhedral slices, d/dt [5] uses orthogonal polyhedra to analyze systems with complex dynamics, in [25], level set methods are employed and Charon [2] allows verification of hybrid systems with linear dynamics by combining the flow-pipe approximations with predicate abstraction. While these tools allow more general dynamics than linear hybrid automata, the guard conditions remain linear and scalability is still a problem. Thus, continued progress will be required to meet the challenges identified in this paper.

ACKNOWLEDGMENT

The authors would like to thank R. Alur, I. Lee, and V. Kumar for their valuable discussions and advice.

REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theor. Comput. Sci.*, vol. 138, no. 1, pp. 3–34, Feb. 1995.

[2] R. Alur, T. Dang, J. M. Esposito, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. J. Pappas, and O. Sokolsky, "Hierarchical modeling and analysis of embedded systems," *Proc. IEEE*, vol. 91, no. 1, pp. 11–28, Jan. 2003.

[3] R. Alur, T. A. Henzinger, and P.-H. Ho, "Automatic symbolic verification of embedded systems," *IEEE Trans. Softw. Eng.*, vol. 22, no. 3, pp. 181–201, Mar. 1996.

[4] M. S. Andersen, R. S. Jensen, T. Bak, and M. M. Quottrup, "Motion planning in multi-robot systems using timed automata," in *Proc. 5th IFAC/EURON Symp. Intell. Auton. Vehicles*, 2004, pp. 21–27.

[5] E. Asarin, T. Dang, and O. Maler, "The d/dt tool for verification of hybrid systems," in *Computer Aided Verification.* Lecture Notes in Computer Science, vol. 2404. New York: Springer-Verlag, 2002, pp. 365–370.

[6] R. Alur and G. Pappas, Eds., *Hybrid Systems: Computation and Control*. Lecture Notes in Computer Science, vol. 2994. New York: Springer-Verlag, 2004.

[7] M. S. Branicky, "Studies in hybrid systems: Modeling, analysis, and control," Ph.D. dissertation, MIT, Cambridge, MA, 1995.

[8] R. W. Brockett, "Hybrid models for motion control systems," in *Essays in Control: Perspectives in the Theory and Its Applications*, H. L. Trentelman and J. C. Willems, Eds. Cambridge, MA: Birkhäuser, 1993, pp. 29–53.

[9] R. Burridge, A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *Int. J. Rob. Res.*, vol. 18, no. 6, pp. 534–555, 1998.

[10] A. Chutinam and B. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," in *Hybrid Systems: Computation and Control.* Lecture Notes in Computer Science, vol. 1569. New York: Springer-Verlag, 1999.

[11] A. Chutinam and B. Krogh, "Verification of infinite state dynamic systems using approximate quotient transition systems," *IEEE Trans. Autom. Control*, vol. 46, no. 9, pp. 1401–1410, Sep. 2001.

[12] E. M. Clarke and R. P. Kurshan, "Computer-aided verification," *IEEE Spectr.*, vol. 33, no. 6, pp. 61–67, Jun. 1996.

[13] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, "The tool KRONOS," in *Hybrid Systems III: Verification and Control.* Lecture Notes in Computer Science, vol. 1066. New York: Springer-Verlag, 1996, pp. 208–219.

[14] R. Fierro, A. Das, J. Spletzer, J. M. Esposito, V. Kumar, J. P. Ostrowski, G. J. Pappas, C. J. Taylor, Y. Hur, R. Alur, I. Lee, B. Southall, and G. Grudic, "A framework and architecture for multi-robot coordination," *Int. J. Robot. Res.*, vol. 21, no. 10/11, pp. 977–995, Oct. 2002.

[15] G. E. Fainekos, H. K. Gazit, and G. J. Pappas, "Temporal logic planning for mobile robots," in *Proc. IEEE Conf. Robot. Autom.*, Apr. 2005, pp. 2032–2037.

[16] T. A. Henzinger, P. Ho, and H. Wong-Toi, "HYTECH: A model checker for hybrid systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 1, no. 1, pp. 110–122, 1997.

[17] T. A. Henzinger, B. Horowitz, R. Majumdar, P. Ho, and H. Wong-Toi, "Beyond HYTECH: Hybrid systems analysis using interval numerical methods," in *Proc. 3rd Int. Workshop HSCC*, pp. 130–144.

[18] T. Henzinger and H. Wong-Toi, "Using HYTECH to synthesize control parameters for a steam boiler," in *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control.* Lecture Notes in Computer Science, vol. 1165. New York: Springer-Verlag, 1996, pp. 265–282.

[19] P. H. Ho and H. Wong-Toi, "Automated analysis of an audio control protocol," in *Proc. 7th Conf. Comput.-Aided Verification.* Lecture Notes in Computer Science, vol. 939, 1995, pp. 381–394.

[20] E. Klavins and D. E. Koditschek, "A formalism for the composition of concurrent robotic behaviors," in *Proc. IEEE Conf. Robot. Autom.*, 2000, pp. 3395–3402.

[21] K. Kapellos, D. Simon, M. Jourdant, and B. Espiau, "Task level specification and formal verification of robotics control systems: State of the art and case study," *Int. J. Syst. Sci.*, vol. 30, no. 11, pp. 1227–1245, 1999.

[22] K. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a nutshell," *Int. J. Softw. Tools Technol. Transf.*, vol. 1, no. 1, pp. 134–152, 1997.

[23] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[24] J. Lygeros, D. N. Godbole, and S. Sastry, "A game-theoretic approach to hybrid system design," in *Hybrid Systems III. Verification and Control*. New York: Springer-Verlag, 1995, pp. 1–12.

[25] I. Mitchell and C. Tomlin, "Level set methods for computation in hybrid systems," in *Hybrid Systems: Computation and Control.* Lecture Notes in Computer Science, vol. 1790. New York: Springer-Verlag, 2000, pp. 310–323.

[26] M. Morari, L. Thiele, and F. Rossi, Eds., *Hybrid Systems: Computation and Control*. Lecture Notes in Computer Science, vol. 3414. New York: Springer-Verlag, 2005.

[27] K. S. Narendra, J. Balakrishnan, and K. Ciliz, "Adaptation and learning using multiple models, switching and tuning," *IEEE Control Syst. Mag.*, vol. 15, no. 3, pp. 37–51, Jun. 1995.

[28] R. Olifati and R. Murray, "Distributed cooperative control of multiple vehicle formations using structured potential functions," in *Proc. IFAC World Congr.*, Barcelona, Spain, Jul. 2002, pp. 232–238.

[29] M. M. Quottrup, T. Bak, and R. I. Zamanabadi, "Multi-robot planning: A timed automata approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 4417–4422.

[30] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multi-agent hybrid systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 509–521, Aug. 1997.

[31] V. I. Utkin, "Variable structure systems with sliding modes," *IEEE Trans. Autom. Control*, vol. AC-22, no. 2, pp. 212–222, Apr. 1977.

[32] M. Zefran and J. Burdick, "Stabilization of systems with changing dynamics," *Hybrid Systems*, 1998.

[33] P. Tabuada and G. J. Pappas, "Linear temporal logic control of discrete-time linear systems," *IEEE Trans. Autom. Control,* to be published.

[34] M. Zefran, J. Desai, and V. Kumar, "Continuous motion plans for robotic systems with changing dynamic behavior," in *Proc. 2nd Int. Workshop Algorithmic Found. Robot.*, 1996, pp. 113–128.

Joel M. Esposito received the B.S. degree from Rutgers University, New Brunswick, NJ, and the M.S. and Ph.D. degrees from the University of Pennsylvania, Philadelphia.

He is currently an Assistant Professor with the Department of Systems Engineering, U.S. Naval Academy, Annapolis, MD. His research interests include algorithmic and computational design tools for mobile robots.

Moonzoo Kim received the Ph.D. degree in computer and information science from the University of Pennsylvania, Philadelphia, in 2001.

After working as a Research Engineer at Samsung SECUi.COM and Pohang University of Science and Technology, he joined the faculty of the Korea Advanced Institute of Science and Technology, Daejeon, in 2006. He currently focuses on developing provable software in the domain of embedded systems by using model checking and runtime verification techniques. His research interests include the specification and analysis of embedded systems, automated software engineering tools, and formal methods.